Lauro César Araujo

Uma linguagem para formalização de discursos com base em ontologias

Brasília

2015

Lauro César Araujo

Uma linguagem para formalização de discursos com base em ontologias

Tese apresentada à Faculdade de Ciência da Informação da Universidade de Brasília como requisito parcial para obtenção do título de Doutor em Ciência da Informação, linha Organização da Informação, grupo de pesquisa de Arquitetura da Informação.

Universidade de Brasília (UnB)

Faculdade de Ciência da Informação (FCI)

Programa de Pós-Graduação em Ciência da Informação (PPGCInf)

Centro de Pesquisa em Arquitetura da Informação (CPAI/UnB)

Orientador: Prof. Dr. Mamede Lima-Marques

Brasília

2015

Araujo, Lauro César
AAR663l Uma linguagem para formalização de discursos com base em ontologias / Lauro César Araujo; orientador Mamede Lima-Marques. -- Brasília, 2015.
529 p.

Tese (Doutorado - Doutorado em Ciência da Informação) -- Universidade de Brasília, 2015.

1. Arquitetura da Informação. 2. Ontologia. 3. Programação em Lógica (Modal). 4. Unified Foundational Ontology. 5. Linguagem formal. I. Lima-Marques, Mamede, orient. II. Título.

Este trabalho foi realizado no âmbito do Laboratório de Lógica Aplicada e do Grupo de Pesquisa em Arquitetura da Informação do *Centro de Pesquisa em Arquitetura da Informação*[1] (CPAI) da *Faculdade de Ciência da Informação* (FCI) da *Universidade de Brasília* (UnB), com suporte financeiro do CPAI.

[1] <http://cpai.unb.br/>

FOLHA DE APROVAÇÃO

Título: **"Uma linguagem para formalização de discursos com base em ontologias"**.

Autor (a): Lauro César Araujo

Área de concentração: Gestão da informação

Linha de pesquisa: Organização da Informação

Tese submetida à Comissão Examinadora designada pelo Colegiado do Programa de Pós-graduação em Ciência da Informação da Faculdade em Ciência da Informação da Universidade de Brasília como requisito parcial para obtenção do título de **Doutor** em Ciência da Informação.

Tese aprovada em: 05 de novembro de 2015.

Prof. Dr. Mamede Lima-Marques
Presidente (UnB/FCI)

Prof. Dr. André Porto Ancona Lopez
Membro Interno (UnB/PPGCINF)

Prof. Dr. Walter Alexandre Carnielli
Membro Externo (UNICAMP)

Prof. Dr. Giancarlo Guizzardi
Membro Externo (UFES)

Prof. Dr. Antonio Eduardo Costa Pereira
Membro Externo (UFU)

Prof. Dr. Flavio Soares Cunha da Silva
Suplente (USP)

Este trabalho é dedicado às pessoas:
as cósmicas, as que choram,
as de vento, as que sonham,
as de barro, as que rezam,
as de fogo, as que amam,
as de gelo, as que acreditam,
as que vivem e as que ainda viverão.

Agradecimentos

Em primeiro lugar, agradeço à Deus pela saúde e por permitir tudo ser possível.

Agradeço à querida Joelma Araujo suas valiosas sugestões, o apoio e incentivo incondicional nesta incrível aventura. Sua companhia torna o caminho muito encantador.

À minha mãe, irmã, sobrinhas... agradeço a compreensão, o apoio e, principalmente, as orações. Sem elas, não chegaria nem pisar neste mundo. Especialmente, agradeço a dedicação da minha irmã preferida, por seus comentários e recomendações.

Ao amigo e companheiro de pesquisa Ismael Moura Costa, por me conduzir nos primeiros passos na busca do conhecimento sobre a Arquitetura da Informação, e por estar sempre ao meu lado em cada desafio que a informação apresenta, nas mais diversas arquiteturas de tempo e espaço.

Ao professor Dr. Samir Bezerra Gorsky, pelas aulas de Filosofia, de Lógica, pela parceria nas pesquisas e, especialmente, pelo grande exemplo de dedicação e amizade.

Aos amigos Dr. André Siqueira e Dr. Alfram Albuquerque agradeço o tempo e atenção sempre disponíveis. A paciência e a humildade de vocês sempre me servem de exemplo.

À amiga e companheira de pesquisa, Flávia Lacerda, por compartilhar algumas lágrimas e diversões desse processo de crescimento.

À colega de estudos Érica Carvalho, por apresentar-me o MProlog.

Ao amigo João Rafael Moraes Nicola, por estar sempre disponível para ouvir mais uma versão das ideias deste trabalho e ainda sugerir melhorias e apresentar novos pontos de vista.

Ao amigo Dr. João Alberto de Oliveira Lima, pela quantidade quase infindável de livros e referências oferecidas, além do apoio intelectual, mentoria e amizade.

Ao amigo Vládner Lima Barros Leal, pelo incentivo e apoio incondicional.

Ao amigo Flávio Heringer, pela agradabilíssima convivência, e pela disposição em praticar a modelagem conceitual.

Ao amigo Paulo Argolo, pela dedicação na leitura do texto e pelas valiosas contribuições ao Glossário.

Ao professor Dr. Alexandre Costa-Leite, especialmente por me conduzir ao incrível universo da Lógica e por me ajudar no passamento entre "o raciocínio primitivo e não-abstrato sobre o mundo atual para a riqueza e beleza do ilimitado universo de inferências

modais”[2]. Esse raciocínio foi crucial para as ideias desta tese.

Aos professores Dr. Walter Alexandre Carnielli, Dr[a]. Juliana Bueno-Soler e Dr. Giancarlo Guizzardi, pelas sugestões formidáveis e por representarem exemplos a serem seguidos.

Aos professores Dr. Flávio Soares Correa da Silva e Dr. Antonio Eduardo Costa Pereira, pela honra de contar com suas contribuições.

A Gunther Rademacher, pela gentileza de responder meus e-mails referentes ao software *Railroad Diagram Generator*, criado por ele, sem o qual não seria possível construir os diagramas *railroad* das especificações EBNF contidos no Capítulo 7.

Aos membros do grupo abnTEX2[3] e latex-br[4]: não apenas a afinidade com LATEX, mas a convivência e amizade criada com várias pessoas compensaram as centenas de horas usadas no desenvolvimento do abnTEX2. Não poderia deixar de salientar a ajuda personificada e a amizade criada com Youssef Cherem, Ole Peter Smith, ...

À Mara Karoline Lins Teotônio, bibliotecária da UnB, que magicamente encontrou todos os textos solicitados, por mais improváveis que fossem suas localizações.

Às secretárias Martha Araujo, Jucilene Moreira e Vivian Miatelo, da Faculdade de Ciência da Informação, por sempre manterem o sorriso alegre, as palavras de incentivo e a burocracia em ordem.

Aos estagiários, secretárias, recepcionistas, copeiras e servidores do Centro de Pesquisa em Arquitetura da Informação da Universidade de Brasília: obrigado por todos os dias de companheirismo, competência em todas as necessidades administrativas e, especialmente, pelas amizades formadas.

Ao professor Dr. Jarbas Tavares dos Santos, por ter sido um dos primeiros mentores a incentivar o investimento na formação stricto sensu.

A todas as pessoas que com palavras de apoio ou de crítica contribuíram para este trabalho.

Especialmente, ao pai e professor Dr. Mamede Lima-Marques, não há palavras suficientes para agradecer as incondicionais orientações que muito extrapolam os limites deste trabalho.

[2] Sim, fui um dos alunos mencionados em Costa-Leite (2013, p. 195).
[3] http://www.abntex.net.br
[4] http://groups.google.com/group/latex-br

Disse-lhe Pilatos: "Que é a verdade?..."
(Bíblia Ave Maria, João 18:38)

Resumo

Esta pesquisa propõe a arquitetura da informação de uma linguagem formal textual para representar discursos sobre entidades ontológicas e obter deduções a respeito de ontologias de domínio. Por meio do paradigma de metamodelagem, a linguagem permite tratamento de ontologias heterogêneas que podem ser descritas como instâncias de uma ou mais ontologias de fundamentação. A linguagem suporta comportamentos clássicos e modais sustentados por noções de prova baseadas no paradigma de Programação em Lógica (Modal). O arcabouço modal desenvolvido possibilita que diferentes interpretações modais sejam introduzidas às especificações das ontologias, e contempla especialmente sistemas baseados em lógicas de múltiplos agentes. Uma sistematização do fragmento endurante da *Unified Foundational Ontology* (UFO) é realizada com objetivo de compor parte do marco teórico que fundamenta a proposta e de servir de exemplo de instanciação do arcabouço desenvolvido. Como resultados complementares, destacam-se: uma sistematização de um conjunto ampliado de regras para produção de modelos conceituais e um glossário detalhado de termos e conceitos da UFO-A; protótipos funcionais que implementam os sistemas elaborados; traduções das teorias descritas no arcabouço proposto para linguagens visuais, como extensões da representação gráfica da OntoUML; e discussões a respeito da integração de Arquitetura da Informação, Modelagem Conceitual e Programação em Lógica (Modal) no contexto social aplicado.

Palavras-chave: Arquitetura da Informação. Ontologia. Programação em Lógica (Modal). *Unified Foundational Ontology*. Linguagem formal.

Abstract

This research proposes the information architecture of a textual formal language to represent and reason about ontological entities based on foundational ontologies. Through metamodeling, the language is able to deal with heterogeneous ontologies that can be described as instances of one or more foundational ontology. The language provides classic and modal inference mechanisms supported by proof notions based on the (Modal) Logic Programming paradigm. The modalities introduced by the modal framework allow a wide range of interpretations, including multi-agent systems. A systematization of the endurant fragment of the *Unified Foundational Ontology* (UFO) is produced in order to compose part of the theoretical framework underlying the proposal, and to serve as an example instantiating the developed framework. As complementary results we highlight: a systematization of an extended set of rules for conceptual modeling and a detailed glossary of terms and concepts of UFO-A; functional prototypes implementing the developed systems; translations of the theories described as instances of the framework to diagramatic representations, as extensions of the OntoUML visual language; and discussions regarding the integration of Information Architecture, Conceptual Modeling and Logic Programming within Applied Social Science.

Keywords: Information Architecture. Ontology. (Modal) Logic Programming. Unified Foundational Ontology. Formal language.

Lista de ilustrações

Lista de quadros

Lista de códigos

Lista de tabelas

Lista de abreviaturas e siglas

AI	Arquitetura da Informação
API	*Application Programming Interface*
BPMN	*Business Process Model and Notation*
CPAI	Centro de Pesquisa em Arquitetura da Informação
DC	*Description Logic*
DSL	Linguagem específica de domínio, do inglês *"Domain Specific Language"*
EBNF	*Extended Backus–Naur Form*
FOL	Lógica de Primeira Ordem, do inglês *First Order Logic*
LOP	*Language-oriented programming*
OCL	*Object Constraint Language*
OLED	*OntoUML Lightwight Editor*
OMG	*Object Management Group*
RNP	Regra negativa escrita de forma positiva (subseção 6.3.1)
UFO	*Unified Foundational Ontology*
UML	*Unified Modeling Language*
UnB	Universidade de Brasília

Lista de símbolos

$\neg$	*not*: negação lógica clássica
$\rightarrow$	implicação lógica clássica ("esquerda" implica "direita")
$\leftarrow$	implicação lógica clássica ("direita" implica "esquerda")
$\leftrightarrow$	bi-implicação
$=$	igualdade ou unificação (Programação em Lógica)
$\neq$	diferente ou não unificação (Programação em Lógica)
$\wedge$	*and*: conjunção lógica
$\vee$	*or*: disjunção lógica
$\Diamond$	*Diamond*: operador modal existencial
$\Box$	*Box*: operador modal universal
$\forall$	*for all*: operador de quantificação universal
$\exists$	*exists*: operador de quantificação existencial
$\vdash$	consequência lógica sintática
$\models$	consequência lógica semântica
$:=$	*def*: definição (esquerda é definido como)
$\in$	*in*: operador "pertence"
$\subset$	*subset*: operador "inclusão própria"
$\subseteq$	*subseteq*: operador "inclusão"
$\cup$	união de conjuntos
::	relação de instanciação
$\int$	*overlap*: relação de sobreposição
$[\![,]\!]$	"colchetes sintáticos". Usados para denotar funções semânticas sobre entidades sintáticas
$\mathbb{N}$	conjunto de números naturais

Lista de definições

Sumário

Introdução

A dedicatória desta obra, apresentada na página 7, é o rascunho de um poema despretensioso que dedica este trabalho "às pessoas". No entanto, ao explicar o que é pessoa, o autor usurpa ideias díspares, não comumente atribuíveis ao conceito de *pessoa*. Com base no conhecimento vulgar, no senso comum fecundado das experiências coletivas, como vislumbrar uma pessoa que é cósmica e chora, é vento e sonha, é de barro e reza, acredita mas é gelo? Caso o autor do poema se afirme como poeta, a licença poética inerente àquela dedicatória não incomoda, nem levanta suspeitas sobre a sanidade do pretendente a artista literário. No entanto, o que se deveria entender caso ele se posicionasse como um cientista ou como um profissional que tentasse descrever a realidade daquela forma imprecisa e ambígua? De fato, a poesia é aquela que primeiro revela a experiência com a realidade. Não apenas a realidade raciocinada, mediada, teorizada e conceitual dos filósofos e dos cientistas, mas a realidade vivida, convivida e sentida na imersão ôntica na qual todos os poetas estão subordinados. Além disso, e também por isso, a poesia é ela própria a filosofia primeira, a que proveu as primeiras formas de expressão sobre os entes que coocorrem conosco na realidade. Nas palavras de Maffei (1949, p. 1507):

> *La poesía, en efecto, surgió como una forma de concepción del mundo y de aprehensión de la realidad por la palabra, que adquiere de ese modo un valor trascendente. Sabido es que fué la poesía la precursora de la filosofía [...] Platón argumenta a menudo en base a la tradición mitológica, y Aristóteles reconoce expresamente que es digno de un filósofo citar en prueba de sus afirmaciones la palabra de algún poeta.*
> *Todo ello revela el alto valor de conocimiento asignado a la poesía en la antigüedad, y no a otra cosa que a este prestigio y valoración responde el hecho de que los primeros pensadores con intención filosófica adoptaran la forma poética como el medio más natural y apropiado de expresión de la realidad. Así, no es por obra de filósofos posteriores, que supieron desentrañar el sentido metafísico de algún poema, sistematizándolo en conceptos, por lo que la poesía tiene alcance cognoscitivo, sino por la naturaleza misma de la expresión poética y en razón de su legitimidad como forma de conocimiento.*

Porém, embora seja a poesia a precursora e grande mandante prócere do discursar sobre a Ontologia, a linguagem poética é tão dependente de sujeitos cognoscentes que a torna inadequada a "aplicações" compartilhadas entre humanos e máquinas, especialmente no que se refere ao alcance de objetivos específicos, que mantenham compromisso não apenas com a arte e com o conhecimento, mas também com finalidades pragmáticas, como construção de sistemas de informação, modelagem de processos de trabalho, desenho de arquiteturas da informação, entre tantas outras serventias que se possa dar às técnicas de discursar sobre o mundo. Como forma de contornar essa circunstância, busca-se minguar a intrínseca subjugação subjetiva inerente à própria experiência, de modo a tentar atingir

o objetivo, possivelmente inatingível, de se descrever a experiência em pura forma lógica. Não que se entenda possível arrancar toda a poesia dos atos de colocar a experiência com a realidade ôntica em formas de linguagem artificial. Mas é necessário que as experiências subjetivas sejam transformadas em objetos para que se tornem menos dependentes daqueles que as experimentam diretamente, e assim, possam ser comunicadas e compartilhadas.

Nesse sentido, o problema abstrato tratado nesta pesquisa é a questão clássica, e sempre em aberto, da relação que o homem conserva com a experiência do mundo, seja este um mundo externo, caracterizado como *realidade*, ou o mundo introspectivo, próprio e individual de cada sujeito. De modo concreto, este trabalho aborda o problema semiótico da expressão e registro, por meio de símbolos organizados em uma linguagem suportada por máquina, dos fenômenos que ocorrem na experiência fenomenológica do sujeito com os objetos. Com intuito de permitir prosperar sobre a questão, esta pesquisa propõe uma linguagem para formalização de discursos proferidos por sujeitos a respeito de entidades ontológicas. Esse objetivo visa ampliar a capacidade humana de raciocinar, e de racionalizar suas experiências. Esse propósito é alcançado com apoio nos fundamentos da Arquitetura da Informação, na *Unified Foundational Ontology* (UFO) de Guizzardi (2005), e sustentado na Lógica Clássica e em Lógicas Modais aplicadas com o paradigma de Programação em Lógica.

A Lógica Clássica é utilizada como primeiro fundamento que norteia a formalização da linguagem. As Lógicas Modais proveem complemento à camada clássica com introdução de lógicas com semânticas de múltiplos agentes, de modo que seja possível formalizar discursos de diferentes sujeitos referentes a um mesmo universo de entidades ontológicas. Já a UFO provê uma ontologia base, fundamental, que serve de alicerce para a construção de ontologias específicas de domínio – ou como preferimos: de discursos – sobre determinado universo. Desse modo, a UFO funciona como um sistema de conceitos fundamentais sobre a realidade, sobre os quais se constroem conceitos que mantém compromisso ontológico tanto com o que é observado, quanto com o sistema teórico e ontológico subjacente. Por conseguinte, a Programação em Lógica Clássica e em Lógicas Modais são utilizadas como arcabouço para a formalização e implementação do sistema de formalização desenvolvido. Esse sistema lógico, resultado desta pesquisa, recebe o nome de *Ontoprolog*.

De um ponto de vista geral, a proposta de Ontoprolog contempla o seguinte:

a) uma linguagem formal textual[5] base para descrição de discursos bem fundados sobre experiências do sujeito com entidades ontológicas. A linguagem é construída para que possa ser oralizada (lida em voz alta) de modo natural e que seus contructos mantenham compromisso ontológico com uma ou mais ontologias subjacentes. A proposta consiste em um arcabouço de metamodela-

[5] Linguagem *textual* é entendida em oposição à *linguagem diagramática*, baseadas em representações gráficas não textuais.

gem, em que teorias gerais são usadas como base a teorias específicas. Embora seja independente do arcabouço dedutivo baseado em Programação em Lógica, a linguagem de Ontoprolog é construída de modo que possa funcionar como uma linguagem de domínio específico (DSL) interna às implementações de Prolog. Dessa forma, a linguagem pode ser integrada potencialmente a qualquer programa Prolog novo ou existente que atenda as normas da série ISO/IEC 13211:1995 (ISO/IEC, 1995);

b) um corpo de regras sintáticas e semânticas dessa linguagem baseado no motor de provas do arcabouço da Programação em Lógica Clássica;

c) uma especificação do fragmento endurante da UFO, um conjunto de açúcares sintáticos[6] construídos sobre a linguagem base, e um conjunto estendido de regras que verificam ontologias de universais e de particulares daquela ontologia de fundamentação;

d) traduções das teorias especificadas na linguagem de Ontoprolog para linguagens visuais de representação de conceitos;

e) um conjunto de extensões à linguagem textual para incorporação de expressões modais, e a discussão de estratégias de incorporação de semântica modal aos discursos sobre entidades ontológicas baseado em semânticas de multi-agentes com interpretações epistêmicas/doxásticas, sustentados pelo arcabouço provido por uma extensão modal de Programação em Lógica, o MProlog, proposto por Nguyen (2006) (seção 3.2).

Por se apresentar como uma pesquisa desenvolvida no âmbito da Ciência da Informação, uma área naturalmente transdisciplinar, para o alcance dos objetivos do trabalho utiliza-se da epistemologia e de contribuições específicas de diferentes disciplinas. Dessa forma, pesquisadores de Ciência da Computação eventualmente se interessarão pela abordagem de Programação em Lógica utilizada para definição da sintaxe e da semântica de Ontoprolog. Pesquisadores de Lógica observarão a aplicação de lógicas clássicas e modais em problemas específicos de informação, especialmente de Modelagem Conceitual. Pesquisadores das áreas de Ontologia e de Modelagem Conceitual se interessarão pela harmonização dos fundamentos da Arquitetura da Informação, com a metáfora dos discursos, das estratégias da engenharia da linguagem proposta, e das aplicações possíveis. Por fim, cientistas da informação se interessarão pela abordagem como um todo, que visa integrar a engenharia lógica[7], formalização de discursos e ontologias para tratar problemas de arquitetura da informação.

[6] Ver nota sobre a expressão “açúcares sintáticos” na subseção 1.4.4.2.
[7] Abordada na seção 2.4.

O restante do fragmento textual[8] desta obra é estruturado em três partes, do seguinte modo:

a) Parte I (Preparação da pesquisa): a primeira parte da obra, que consiste no Capítulo 1 (Elementos de pesquisa), tem o encargo de apresentar o problema de pesquisa, os objetivos, a justificativa e a metodologia científica adotada no trabalho;

b) Parte II (Referenciais teóricos): a segunda parte do trabalho consiste na apresentação da propedêutica das disciplinas científicas abordadas. Adota-se uma abordagem sintética de exposição dos achados mais relevantes aos objetivos estabelecidos pelo Capítulo 1. O tratamento sintético é entendido como oposto a uma obra enciclopédica sobre os temas, de modo que foca-se em mencionar referências precisas sobre os textos consultados, para que possam ser encontrados e verificados. A parte é distribuída nos seguintes capítulos:

   – Capítulo 2 (Arquitetura da Informação): apresenta-se o contexto de Arquitetura da Informação em que esta tese está inserida, especialmente no que tange aos trabalhos do grupo de pesquisa do CPAI/UnB;

   – Capítulo 3 (Programação em Lógica): contém uma breve apresentação de Prolog, MProlog e de técnicas para especificação de linguagens formais, com foco na identificação das obras consultadas para obtenção dos resultados apresentados na Parte III;

   – Capítulo 4 (Ontologia de fundamentação UFO): trata-se do capítulo mais extenso de referencial teórico. Seu objetivo é apresentar uma sistematização das regras a serem observadas na construção de modelos conceituais com base na UFO. Essas regras são distribuídas em quadros, um para cada entidade UFO relevante para o uso da ontologia como linguagem de formalização de discursos. Uma larga pesquisa bibliográfica é realizada com foco nas obras publicadas pelo autor e pesquisadores que contribuem com a UFO;

c) Parte III (Resultados): a última parte do fragmento textual do texto consiste na apresentação dos resultados da pesquisa, distribuídos nestes capítulos:

   – Capítulo 5 (Ontoprolog: Visão geral): o primeiro capítulo dessa parte apresenta uma sumarização dos resultados da pesquisa referentes ao Ontoprolog. O capítulo introduz detalhes abordados pelos capítulos seguintes;

   – Capítulo 6 (Semântica formal base de Ontoprolog): nesse capítulo aborda-se aspectos técnicos referentes à semântica base de Ontoprolog. A semântica é

[8] Conforme a ABNT NBR 14724:2011 *Informação e documentação — trabalhos acadêmicos — apresentação* (ABNT, 2011), os trabalhos acadêmicos são divididos em três fragmentos: a parte pré-textual, que inicia-se na capa e encerra-se após o Sumário; a parte textual, que estende-se imediatamente após o Sumário até imediatamente antes das Referências e a parte pós-textual, que consiste das Referências até o Índice.

descrita por meio de uma *linguagem alvo* baseada no fragmento da Lógica de Primeira Ordem expressável por meio de cláusulas de Horn;

- Capítulo 7 (Sintaxe base de Ontoprolog): a sintaxe base proposta para a linguagem é abordada nesse capítulo por meio de definições em EBNF. A semântica de tradução da sintaxe é demonstrada com base em funções que traduzem a sintaxe base em relações semânticas na linguagem alvo;
- Capítulo 8 (Especificação da UFO em Ontoprolog): a sintaxe e a semântica bases de Ontoprolog são estendidas nesse capítulo, de modo a aprimorar a experiência de arquitetos da informação quando a UFO é a ontologia subjacente na formalização de determinado discurso;
- Capítulo 9 (Modalidades de MProlog com Ontoprolog): apresenta-se como as definições clássicas de Ontoprolog são adaptadas e integradas ao arcabouço lógico MProlog, que fornece expressividade modal com diferentes semânticas, inclusive semântica de múltiplos agentes por meio de lógicas epistêmicas/doxásticas;
- Capítulo 10 (Implementação de referência do arcabouço de Ontoprolog): destacam-se algumas características da implementação de referência e de uso de Ontoprolog tendo Prolog e MProlog como sistemas subjacentes hospedeiros. O capítulo detalha o código-fonte da linguagem proposta e aborda tradução das especificações Ontoprolog para a linguagens diagramáticas visuais, especialmente a OntoUML, proposta por Guizzardi (2005);
- Capítulo 11 (Considerações finais e conclusão): como encerramento da parte textual do texto, o capítulo apresenta as considerações finais, as contribuições mais relevantes da pesquisa e as sugestões a trabalhos futuros.

Após o fragmento textual, o fragmento pós-textual final contempla ainda as seguintes seções, além das Referências e do Índice:

a) Glossário da UFO: trata-se de um glossário com 186 entradas referentes ao fragmento endurante da UFO, criado com base nas publicações do autor da ontologia de fundamentação em foco realizadas nos últimos dez anos. O glossário funciona como extensão dos referenciais teóricos (Parte II), e é construído com a intenção de servir de base para a proposição de um tesauro (SIMÕES, 2008; DODEBEI, 2002; ISO, 2011) com foco na sistematização terminológica da UFO;

b) Apêndice A (Especificação da Metateoria de *Hypertypes*): contém especificação, realizada na linguagem proposta, da metateoria de referência utilizada para definição das demais ontologias. Trata-se das definições ontológicas e lógicas mais elementares apresentadas na forma de um arcabouço geral;

c) Apêndice B (Figura da metateoria UFO): apresenta uma representação estilizada

em UML da ontologia UFO conforme incorporada neste trabalho, adaptada aos objetivos específicos da pesquisa;

d) Apêndice C (Especificação da metateoria UFO em Ontoprolog): trata-se da especificação, na linguagem proposta, dos conceitos representados na figura contida no Apêndice B;

e) Apêndice D (Diagrama com visão da UFO-A original): apresenta um diagrama em UML de conceitos da UFO conforme encontrados na literatura;

f) Apêndice E (Operadores Prolog usados na definição da sintaxe de Ontoprolog): contém as definições dos operadores Prolog usados na implementação da linguagem proposta nesta pesquisa;

g) Apêndice F (Expansões das relações semânticas): consiste na implementação em Prolog de expansões realizadas no processo de tradução da linguagem Ontoprolog para relações simples na lógica subjacente de Prolog;

h) Apêndice G (Extensão das expansões das relações semânticas para UFO): trata-se da extensão das regras de expansão apresentados pelo Apêndice F específicas para a sintaxe proposta para a UFO.

Parte I

Preparação da pesquisa

1 Elementos de pesquisa

Neste primeiro capítulo apresenta-se elementos referentes à preparação do trabalho. A seção 1.1 aponta os problemas de pesquisa que motivam esta investigação. Com base na caracterização do problema, na seção 1.2 positiva-se o objetivo geral e objetivos específicos. Na seção 1.3 argumenta-se como o alcance dos objetivos colaboram cientificamente na solução dos problemas apresentados. Nessa seção apresenta-se também uma série de argumentos que visam justificar a adoção dos elementos teóricos envolvidos no trabalho. Por fim, na seção 1.4 apresenta-se a metodologia de pesquisa, que aborda a classificação, o modelo de referência, a Visão de Mundo e os aspectos técnicos do texto.

1.1 Problema de pesquisa

Para se comunicarem, pessoas devem exteriorizar a outros sujeitos a experiência individual que sofrem consigo mesmas e com a informação do mundo. Da mesma forma, máquinas enquanto imitadoras do intelecto humano devem ser capazes de exprimir ideias e de compreender, ou ao menos de simular compreender, conceitos humanos. Os conceitos fundamentam as ideias e possibilitam a comunicação. Eles são produtos da experiência dos sujeitos com a realidade. Ao compartilhar parte de uma mesma realidade, os conceitos comuns entre sujeitos funcionam como referência da comunicação, como substitutos de objetos do mundo. Porém, pelo fato de os conceitos também se referirem a ideias próprias, individuais de cada sujeito, quais conceitos fundamentais devem ser explicitados para que homens e máquinas possam colaborar entre si? Que tipos de problemas surgem da formulação de discursos a respeito de experiências do mundo? Como esses conceitos podem ser utilizados em aplicações e soluções concretas de problemas de inteligência (artificial)? Essas questões são relevantes não apenas na relação homem e máquina, mas também nas relações sociais humanas. Por exemplo, em um grupo de pessoas, como evidenciar e garantir consenso sobre acordos a respeito de uma realidade compartilhada? Quais seriam os conceitos cognitivos fundamentais a serem compartilhados para que seja possível reduzir a imprecisão da comunicação e evitar os falsos acordos? Como uma máquina poderia auxiliar na obtenção desses acordos conceituais formados por sujeitos humanos?

Nos últimos anos as áreas de Arquitetura da Informação, Inteligência Artificial e Modelagem Conceitual têm utilizado ontologias como instrumentos que visam responder algumas dessas questões. Elas são usadas como camada semântica intermediária da linguagem adotada pelos agentes na comunicação, seja essa linguagem natural, ou artificial. Nesse sentido, acordos referentes a conceitos do mundo podem ser obtidos na medida que podem ser expressos relativamente às entidades da ontologia adotada. Trata-se, então, de

um caso de uso de ontologias como fundamentação de discursos proferidos por agentes a respeito de suas experiências com os objetos da realidade.

Para que isso seja possível, é necessário que existam ontologias base que funcionem como teorias subjacentes aos discursos que sustentam as referências ao mundo, e de linguagens que permitam expressar e obter conclusões a respeito desses discursos, que são formalizados na forma de ontologias de domínio. Uma ontologia de fundamentação em evidência nos últimos anos é a *Unified Foundational Ontology* (UFO), proposta por Guizzardi (2005) como um sistema de fundamentação com categorias ontológicas que aprimoram a qualidade de modelos conceituais (GUIZZARDI; WAGNER, 2008) e, com isso, reduzem a imprecisão e os falsos acordos nas comunicações dos discursos. Como trabalho diretamente relacionado, a OntoUML é uma linguagem diagramática concebida como aprimoramentos da UML realizados com base nas categorias ontológicas da UFO. Ela permite que ontologias específicas de domínio[1] sejam representadas de modo fundamentado na ontologia de referência.

Embora linguagens de representação visual sejam úteis na comunicação de acordos, linguagens textuais dispõem de características complementares, como flexibilidade em relação a mecanismos de dedução e de instrumentos de integração com aplicações reais. Porém, nesse caso, a linguagem textual não pode ser demasiadamente complexa de ser escrita, caso em que ela não se prestaria a ser usada em tempo real de modelagem de discursos. Por outro lado, ela não pode ser demasiadamente incauta, ao ponto de não conter a expressividade, simplicidade, estabilidade semântica, laconicidade, correção e clareza necessárias a essa finalidade. Além disso, considerando que aspectos cognitivos humanos estão envolvidos nos discursos, para abarcar a complexidade inerentes aos sujeitos e às aplicações reais nas quais a formalização dos discursos estão inseridos, são desejáveis abordagens integradas de linguagens diagramáticas e textuais, de lógicas clássicas e não clássicas, de mecanismos de dedução e de inferência a respeito de entidades ontológicas, de checagem de instanciação de modelos, de suporte instrumental computacional e de interoperabilidade semântica. Especialmente, são desejáveis abordagens que supram a necessidade de formalização de discursos intelegíveis entre máquinas e homens, mediados por ontologias de referência, e de aplicação concreta de ontologias na solução de problemas reais de Inteligência Artificial. Porém, não encontra-se evidência na literatura especializada de abordagem integrada de linguagem formal textual que seja possível ser oralizada, de modo similar a um discurso oral, que permita o uso de diferentes ontologias de fundamentação e

[1] No contexto de linguagens de representação, uma ontologia é uma representação específica, "*an explicit specification of a conceptualization*" (GRUBER, 1993a, p. 907), referente à essência de "seres" existentes em uma realidade. Essa referência à essência do ser é o que diferencia uma ontologia, que possui compromisso ontológico com uma concepção de mundo, de uma teoria qualquer, que prescinde de evidenciar esse compromisso. Também é necessário observar a diferença de uso de "ontologias" enquanto especificações de conceitos, ou de modelos conceituais escritos em uma linguagem, da "Ontologia" enquanto área de investigação filosófica.

na qual seja possível formalizar o processo de modelagem de conceitos entre vários agentes simultâneos que compartilham uma mesma realiade. Na mesma forma, não encontra-se formalismo com essas características integrado com arcabouços de Inteligência Artificial que possibilite uso de diferentes tipos de lógica clássica e não clássica e que permita que especificações de ontologias sejam consumidas para raciocínio automatizado em bancos de dados clássicos e dedutivos, em problemas de satisfação de restrições, em sistemas de interpretação de linguagem natural, e em outras aplicações de Programação em Lógica.

Embora a existência desse tipo de arcabouço não solucione por completo as questões apresentadas no primeiro parágrafo, entende-se que avançar na direção de abordagens integradas de formalização de ontologias aprimoram a experiência de sujeitos com a informação, na medida que possibilitam tornar cada vez mais transparentes as diferentes formas de se comunicar e de se racionar com suporte de máquinas.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral desta tese é *propor uma linguagem formal textual para construção de discursos sobre entidades ontológicas.*

1.2.2 Objetivos Específicos

Como metas componentes do objetivo geral, esta pesquisa possui ainda os seguintes objetivos específicos:

a) definir uma especificação formal de uma linguagem textual de propósito específico para formalização de ontologias e implementar um interpretador dessa linguagem em Programação em Lógica Clássica;

b) construir um sistema de regras que valide ontologias de universais e de particulares especificadas com base no fragmento endurante da *Unified Foundational Ontology* (UFO);

c) estender o arcabouço clássico da linguagem textual para inclusão de operadores modais referentes a discursos de diferentes agentes.

O Capítulo 11 (Considerações finais e conclusão) indica como o objetivo geral e cada um dos objetivos específicos são alcançados.

1.3 Justificativa

O objetivo desta seção é apresentar porque é cientificamente relevante resolver os problemas apresentados na seção 1.1 a partir dos objetivos propostos na seção 1.2.

Embora muito já tenha sido obtido em termos de teorias e de instrumentos para suporte à Modelagem Conceitual, ainda diversas outras abordagens de representação de conceitos e de raciocínio sobre modelos conceituais baseados em ontologias de fundamentação podem ser exploradas, especialmente com base em resultados integrados de Inteligência Artificial, de Lógicas Clássicas e Não Clássicas, e de Programação em Lógica, que suporta as anteriores. Esses tipos de abordagens podem explorar diferentes estratégias de definição de semântica dos modelos, com diferentes características teóricas e pragmáticas de expressividade, clareza, noção de prova, consistência, e poder de computação. Para o caso das aplicações de Programação em Lógica, o rigor metodológico imposto pelas ontologias de fundamentação pode resultar em sistemas mais coerentes, interoperáveis e intelegíveis. Para o caso de linguagens de modelagem conceitual, os formalismos baseados em Programação em Lógica podem prover instrumentos de aprimoramento e de integração de ontologias de domínio com aplicações reais.

Dessa forma, o trabalho é justificado devido à ausência de linguagem textual, com expressividade adequada para a formalização de ontologias no paradigma de metamodelagem, que seja suficiente para a formalização de modelos conceituais, especialmente com base na UFO. Além disso, lacuna teórica está presente no âmbito da formalização dos discursos de diferentes agentes sobre uma mesma realidade compartilhada. Esse tipo de formalização de discursos em ambiente multiagentes pode ser sustentada por raciocínios automatizados, e utilizada como auxiliar no processo de obtenção de acordos a respeito da realidade. Porém, a ausência de linguagem que suporte e evidencie as distinções de agentes nos discursos é um fator limitante teórico ao avanço das práticas de Modelagem Conceitual. Posto desse modo, o alcance dos objetivos de pesquisa suprem ausência tanto teórica quanto instrumental na área. Portanto, trata-se de uma contribuição direta ao desenvolvimento de linguagens formais para a área de investigação de modelagem conceitual com base em ontologias de fundamentação, que vai além do paradigma de linguagens visuais comumente encontrado na bibliografia especializada.

Ademais, o desenvolvimento de uma linguagem de metamodelagem para formalização de ontologias em ambiente multiagentes, embora relevante a aplicações reais em que se deseja construir consenso ou acordo a respeito de determinada realidade, não pode ser realizado meramente no contexto de aplicação de técnicas. Diferentes sistemas de lógicas modais[2] podem ser empregados, dependendo do tipo de semântica que se deseja obter no processo de obtenção de acordos. Mesmo em contextos em que exista um único agente, pode

[2] Alguns sistemas são mencionados na seção 3.2.

ser o caso que se deseje prover semântica introspectiva à afirmações sobre a ontologia, como na situação em que se avalia graus de confiabilidade de determinadas afirmações. Esses diferentes tipos de semânticas modais devem ser analisados cientificamente, especialmente devido ao caráter epistêmico relacionado à concorrência de múltiplos agentes, de modo que se desenvolva um arcabouço teórico adequado para a realização do objetivo geral.

Trata-se de uma contribuição inédita no âmbito da Ciência da Informação, no que refere-se à investigação transdisciplinar de Lógica, com Lógica Clássica e Lógicas Modais, de Programação em Lógica, de Ontologia formal com Modelagem Conceitual, de Linguística com linguagens formais embutidas em programas baseados em Programação em Lógica e da própria Ciência da Informação, por meio de Arquitetura da Informação. A contribuição à Ciência da Informação é justificada pelo desenvolvimento de instrumentos baseados em ontologias que podem ser aplicados em contextos de representação de conteúdo, de definição de regras de conhecimento, e de explicação de ambiente social, atividades essas conduzidas pelos cientistas da informação (ALMEIDA, 2014, p. 253).

Como contribuição indireta e justificativa complementar, espera-se que os resultados desta pesquisa apontem novos caminhos para desenvolvimento da Modelagem Conceitual, especialmente quanto ao uso de lógicas não clássicas, à implementação de sistemas de lógicas modais, e ao uso de Programação em Lógica como intermédio entre teoria e prática de modelagem.

Adicionalmente a essas justificativas, apresenta-se na sequência perguntas cujas respostas reforçam a relevância desta pesquisa no âmbito social aplicado da Arquitetura da Informação, bem como argumentam quanto ao ineditismo do trabalho, especialmente no que tange à investigação entre práxis e teoria[3] que relaciona UFO, Programação em Lógica e Programação em Lógica Modal, desenvolvimento de linguagem específica interna à linguagem de propósito geral, e Arquitetura da Informação. Cada questão é apresentada e respondida em seção própria.

1.3.1 Por que o desenvolvimento de uma linguagem de propósito específico?

Language Driven Development (CLARK; SAMMUT; WILLANS, 2008), ou *Language-oriented programming* (LOP) (WARD, 1994; ROSENAN, 2010), é uma abordagem de desenvolvimento de programas de computador que, ao invés de desenvolver problemas diretamente em linguagens de programa de propósito geral – como Java, C, Lisp ou Haskell – desenvolve-se uma linguagem mais específica, orientada ao domínio em que o problema está concentrado e, a partir daí, resolve-se o problema com base nessa linguagem especializada.

A primeira característica dessa abordagem é o ganho com a abstração do problema

[3] Conforme metolodogia descrita na seção 1.4.

em relação às questões técnicas de tratamento e computação da informação. Isso permite atuar socialmente sobre o problema sem se preocupar demasiadamente com detalhes técnicos. Essa característica está ligada ao poder descritivo da linguagem, que de acordo com Israel e Brachman (1984, p. 120), deve permitir que intuições a respeito do domínio sejam expressas de forma natural:

> *A representational formalism should include constructs that allow a user to capture intuitions about the structure of the domain(s) of application – for example, intuitions about the appropriate conceptualization of the objects, properties, and relations of the domain.*

Aliás, uma linguagem específica de domínio (DSL), compartilhada por uma comunidade de profissionais, após o processo inicial de aprendizado da construção idiomática por seus usuários, deve, ao menos em tese, fortalecer uma cultura no círculo social em que ela está inserida.

Além desse aspecto social, há ganhos também do ponto de vista computacional. Dentre as características mais marcantes dessa abordagem, ressalta-se o reúso dos componentes de software, a facilidade de interoperabilidade dos programas, a simplicidade de manutenção, a clareza da codificação e, especialmente, a capacidade de aplicação de raciocínios lógicos mais elaborados, que vão além da abordagem clássica, uma vez que é possível incorporar diferentes noções lógicas às linguagens, como quantificadores, modalidades, esquemas, paraconsistência, etc. Ademais, uma vez separado o problema da solução final, na linguagem intermediária, que descreve o problema, é possível incorporar sofisticados arcabouços dedutivos, sem com isso necessariamente tornar mais complexa a especificação dos conceitos do domínio.

Essas diferentes concepções lógicas podem ser representadas tanto com linguagens visuais como com linguagens sentenciais textuais. Linguagens visuais possuem características interessantes, como a capacidade de incorporar aspectos semânticos à própria sintaxe, uma vez que os símbolos visuais são utilizados para informar o usuário sobre aspectos semânticos do domínio que aquela linguagem se presta a descrever[4]. Porém, linguagem textuais também possuem suas vantagens. Por exemplo, do ponto de vista pragmático, uma linguagem textual é útil para anotação rápida, muitas vezes sem necessidade de auxílio de computador ou de ferramenta de desenho. Se projetadas tendo como referência uma ontologia de fundamentação, como a UFO ou DOLCE (MASOLO et al., 2003), elas possibilitam que seus usuários sejam muito eficientes em comunicar suas ideias, pois permite incorporar aspectos semânticos na estrutura das sentenças, assim como ocorre com as linguagens visuais. Além disso, linguagens textuais podem ser rapidamente projetadas, implementadas, testadas e modificadas, e lançar mão de formalismos amplamente conhecidos, como EBNF (seção 7.1). Não obstante, diferentes aspectos lógicos podem ser

[4] O tema das linguagens visuais é abordado também pela subseção 3.4.2.

combinados e descritos sintaticamente, sem que com isso seja necessário lidar com detalhes de representação, de combinação de símbolos, definição de figuras ou formas, nem correr o risco de induzir interpretações errôneas sobre o domínio, devido ao uso ambíguo, divergente ou inadequado de representações visuais em relação ao domínio do problema. Nesse sentido, do ponto de vista de engenharia de linguagens formais, as linguagens textuais contam com largo suporte instrumental, técnico e teórico, que permitem não apenas a definição mas a integração de linguagens e incorporação em formalismos existentes. De toda forma, não se pode deixar de mencionar que os paradigmas visual (ou diagramática) e textual não são excludentes. Idealmente, para cada linguagem textual pode haver uma contrapartida visual, e vice e versa. Adicionalmente, linguagens visuais, em geral, possuem apelo sintético, de simplificação da comunicação, enquanto linguagens textuais têm potencial para descrever detalhes analíticos, de modo que também são adequadas para incorporação de aspectos de engenharia de domínio[5]. No caso, a OntoUML proposta por Guizzardi (2005) é uma linguagem diagramática baseada em UML. O desenvolvimento de uma representação textual da linguagem, otimizada para leitura e escrita, que contemple em sua sintaxe aspectos específicos da ontologia de fundamentação subjacente, pode ampliar a abrangência e as integrações de OntoUML com e em outras linguagens. Esses são alguns dos motivos que justificam a opção por definir uma linguagem textual neste trabalho.

Do ponto de vista da Arquitetura da Informação (AI), é natural que a práxis da área envolva um conjunto de pessoas e que, somado a própria natureza complexa da informação (ZHANG; YUEXIAO, 1988; CRNKOVIC; HOFKIRCHNER, 2011), a prática envolva tarefas multifacetadas. Apenas no contexto de websites, uma das área da AI, Ding e Lin (2010, p. 23) menciona que:

> *Creating complex websites requires an interdisciplinary team involving business sponsors, user researchers, visual designers, software developers, project managers, content writers and other experts. In order for everyone to collaborate effectively, a structured development process must be agreed upon. The process may vary from project to project depending on many factors, but the ultimate goal should be the same: Increase the business value of the design and meet the user needs.*

Desse modo, defende-se que uma abordagem orientada ao desenvolvimento de uma linguagem de propósito específico ao arquiteto da informação deve trazer clareza e formalização às atividades de compreensão das entidades da realidade.

[5] Por exemplo, Costa, Grings e Santos (2008, p. 440) apresentam que *"If one needs to design a really large circuit, one will end up using a textual programming language, like Verilog."*

1.3.2 Por que a aplicação de Programação em Lógica?

A aplicação de Lógica enquanto instrumento do arquiteto da informação, e como fundamento da Arquitetura da Informação, já foi objeto de discussão de Siqueira (2008)[6]. Por conseguinte, devido à intrínseca presença da Lógica nas produções referentes à Arquitetura da Informação, aproximar a Lógica do instrumento concreto a ser utilizado em atividades da práxis diminui barreiras e reduz a indireção do ferramental auxiliar disponível para as execução dessas atividades.

"Prolog" é um nome genérico dado às linguagens, ferramentas e sistemas lógicos e computacionais utilizados para Programação em Lógica. Embora Prolog seja uma linguagem criada nos anos 1970 (KOWALSKI, 1988), ela ainda é usada não só no desenvolvimento contemporâneos de soluções, como consta em diversos trabalhos mencionados no Capítulo 3[7], como também é alvo de pesquisas de ponta, como o caso do computador Watson da IBM (LALLY; FODOR, 2011), reconhecido pela façanha de vencer o sistema de perguntas e respostas do tipo "Homem X Máquina" do Jeopardy (IBM, 2011):

> ***6. What operating system does Watson use? What language is he written in?***
> *(RatherDashing) Watson is powered by 10 racks of IBM Power 750 servers running Linux, and uses 15 terabytes of RAM, 2,880 processor cores and is capable of operating at 80 teraflops. Watson was written in mostly Java but also significant chunks of code are written C++ and Prolog, all components are deployed and integrated using UIMA* (ZIMMERMAN, 2011).

Do ponto de vista técnico, a facilidade na definição de aspectos de metalinguagem dentro da própria linguagem (por meio de *operadores* e de predicados de introspecção da base de dados de *runtime*), o amplo arcabouço teórico disponível, a disponibilidade de soluções para tratamento da informação em diferentes tipos de lógica, tornam Prolog uma ferramenta claramente compatível com os objetivos deste trabalho. A exemplo do que defende Christiansen (2002) sobre meta-linguagem para tratar e, principalmente, ensinar conceitos de linguagem de programação, Prolog é uma escolha natural na prototipagem rápida e consistente de DSL e linguagens de programação de propósito geral. Gupta (1998), por exemplo, é ainda mais enfático e afirma que o uso de cláusulas de Horn em Prolog puro para especificar a semântica denotacional de linguagens e inferir propriedades de programas é uma escolha mais simples, clara e direta do que Cálculo Lambda. Nesse sentido, dentre as características de Prolog que nos faz optar pela ferramenta no âmbito deste trabalho, destacam-se[8]:

[6] Porém, mesmo se isso não tivesse sido o caso, é defensável que a Lógica é o instrumento mais natural para lidar com a *informação*.

[7] Além das obras mencionadas naquela seção, no âmbito de comunidades de prática, destaca-se que no sítio <http://stackoverflow.com/questions/130097/real-world-prolog-usage> encontra-se mais de uma centena de respostas ao tema: *"Real world Prolog usage"*.

[8] Parte dos itens salientados são baseados em Christiansen (2002). O autor, na obra citada, defende

a) no âmbito sintático, o recurso de definição de operadores provê representação e acesso direto às árvores sintáticas das sentenças, de modo que a análise léxica de uma linguagem construída internamente na linguagem Prolog é realiza de forma automática;

b) além dos operadores, predicados como `assert/1` e `retract/1` são recursos que possibilitam que características de metaprogramação sejam diretamente incorporados às teorias e programas descritos em Prolog, embora explicitem questões meta-lógicas referentes a não-monotonicidade das teorias[9];

c) definições indutivas estruturalmente são expressas em Prolog de forma direta por meio de regras e de unificação, por exemplo:

```
sentenca(while(C,S), ... ) :-
  condicao(C, ... ),
  sentenca(S, ... ), ... .
```

Isso permite definir semântica à operadores da linguagem de modo declarativo e também executável;

d) estrutura de dados, tabelas e ligação de variáveis estão disponíveis tanto no âmbito de programação como de metaprogramação de forma nativa;

e) Prolog provê um arcabouço leve, mas suficiente para definição de linguagens e de DSL, se comparado a Teoria de Conjuntos e Cálculo Lambda, uma vez que as especificações podem ser diretamente executadas e monitoradas em detalhes por ferramentas de depuração (*debug*) e pelo *tracer*. Desse modo, as linguagens podem ser programadas incrementalmente e testadas a medida que são desenvolvidas;

f) Prolog possui uma vasta gama de bibliotecas, compiladores, ferramentas e técnicas de integração com diversas arquiteturas. É o caso, por exemplo, das diversas bibliotecas de programação por restrições (FRÜHWIRTH, 1998; WALLACE et al., 2004; SNEYERS et al., 2010);

g) além dos recursos de desenvolvimento de linguagem interna com base em operadores, Prolog possui mecanismo para o desenvolvimento de DSL com base em gramáticas de cláusulas definidas (DCG) (PEREIRA; WARREN, 1980);

h) Prolog é adequado para desenvolvimento exploratório, conforme é defendido por Covington et al. (2012):

o uso de Prolog como uma ferramenta ideal para ensinar princípios de construção de linguagens de programação. Para isso, demonstra a implementação de uma máquina abstrata.

[9] Desse modo, Prolog não é exatamente um tipo de linguagem de Lógica Clássica, uma vez que a incorporação de controles como *cut* – simbolizado por ! – representam aspectos operacionais incorporados à semântica dos programas. Isso justifica a descrição comum de Prolog como "Lógica + Controle": ele engloba tanto aspectos lógicos denotacionais como procedimentos operacionais que extrapolam as características clássicas. Além disso, a "hipótese do mundo fechado" (subseção 3.1.4) implica em comportamentos não-monotônicos não presentes originalmente em lógicas com operações de consequência tarskianos, considerados os comportamentos clássicos.

> *Many Prolog programs are exploratory; when trying to figure out whether something can be done, it is reasonable to do each part of it in the easiest possible way, whether or not it is efficient* (COVINGTON et al., 2012, p. 31).

i) a linguagem base de Prolog é padronizada por uma série de normas da ISO/IEC, o que garante a interoperabilidade de programas entre diferentes implementações. A primeira parte das normas da série, a ISO/IEC 13211-1:1995 *Information technology – Programming languages – Prolog – Part 1 : General core* (ISO/IEC, 1995), normatiza o segmento central da linguagem. Ela possui duas correções, a primeira em 2007 (ISO/IEC, 2007), e a outra em 2012a (ISO/IEC, 2012a). A segunda parte trata da padronização dos módulos do sistema (ISO/IEC, 2000) *Information technology – Programming languages – Prolog – Part 2 : Modules*, que são usados para organizar as extensões do sistema;

j) há diferentes implementações que enriquecem aspectos teóricos específicos – como lógicas de ordem superior, banco de dados dedutivos, entre outras – sem, no entanto, abrir mão do suporte à especificação padrão ISO. Dentre essas implementações, destacam-se os softwares livres SWI-Prolog[10] e XSB[11,12], além do sistema com licença comercial e acadêmica SICStus Prolog[13]. Todos provêm, inclusive, integrações com linguagens usadas corporativamente, como Java;

k) diferentes tipos de lógicas e de abordagens da informação podem ser programadas em Prolog: além da própria Lógica Clássica e da programação por restrições, destacam-se Lógicas Não Monotônicas em geral, Lógicas Modais e Multimodais (FARIÑAS DEL CERRO, 1986; NGUYEN, 2006; NGUYEN, 2009), Lógicas de Ordem Superior (NAISH, 1996) e as Lógicas paraconsistentes (RODRIGUES, 2010), apenas para mencionar algumas. Especialmente, as Lógicas Modais são adequadas para raciocínios aléticos, temporiais, doxásticos e epistêmicos, entre outros;

l) por fim, de forma muito enfática, Kowalski (1974), um dos célebres criadores de Prolog, defende que a linguagem é adequada não apenas para representar outros tipos de lógicas, mas é ela em si uma lógica que permite que "conhecimentos" sejam expressos diretamente em forma clausal:

> *Methods for transforming arbitrary first-order sentences into clausal form are described in Nilsson's book*[14]*. It is our thesis, however, that clausal form defines a natural and useful language in its own right, that thoughts can conveniently be expressed directly in clausal form, and that literal translation*

[10] <http://www.swi-prolog.org/>
[11] <http://xsb.sourceforge.net/>
[12] Uma característica marcante do XSB é capacidade de lidar naturalmente com *tabling* e *memoization* de resultados (SWIFT; WARREN, 2012).
[13] <https://sicstus.sics.se/>
[14] A obra mencionada no artigo citado é Nilsson (1971).

> *from another language, such as full predicate logic, often distorts the original thought* (KOWALSKI, 1974, p. 569).

É devido a essa visão de Kowalski (1974) que no decorrer desta pesquisa usa-se preferencialmente a notação clausal adotada na Programação em Lógica em detrimento de outros tipos de notação, como a tradicional notação de Primeira Ordem ou mesmo a notação modal, conforme "tradicionalmente" encontrada nos livros clássicos de Lógica Modal, a exemplo de Fitting e Mendelsohn (1998).

Como encerramento e justificativa final do tópico em discussão, Prolog é utilizado devido à abordagem de Lógica Aplicada, com base na ideia de *Engenharia Lógica* (seção 2.4) introduzida por Lima-Marques (1992, p. 7-10), em contrapartida à investigação teórica da Lógica. Destaca-se, por isso, que este trabalho não é uma tese *em Programação em Lógica*, mas uma pesquisa *com Programação em Lógica*, no sentido que a Programação em Lógica é utilizada como meio capaz de oferecer características úteis ao alcance dos objetivos, como a capacidade de alinhar as abstrações adequadas aos problemas tratados com o *savoir-faire* de executar e validar as construções lógicas propostas.

1.3.3 Por que o uso de ontologias de fundamentação?

Cientistas da informação não são mais meros catalogadores de livros com base em padrões como MARC e AACR2. O modo de usar as classes tem mudado. Elas são mais gerais e com abordagem teórica mais consistente (RUPP; BURKE, 2004). Desse modo, a compreensão de questões metafísicas é importante não apenas para o entendimento do mundo, como é de extrema relevância para moldá-lo. No âmbito jurídico, por exemplo, questões sobre sistematização e classificação de conceitos, embora sejam tópicos de pesquisas recentes (PEPINO; GAVIORNO; FILGUEIRAS, 2003; COSTA, 2008), já são levantadas há séculos. No âmbito brasileiro, por exemplo, o eminente jurista doutrinador Augusto Teixeira de Freitas (1816–1883), no século XVIII, discutia essas questões no âmbito de seus projetos que englobavam o Código Civil brasileiro (FREITAS, 2003)[15]. Conforme Freitas (1859, p. 52-53)[16]:

> Classificar não é simplesmente dividir, não é sómente designar por uma denominação commum os individuos que se assemelhão á certos respeitos. A divisão é instrumento da analyse; mas, terminada esta, e conhecidas as differenças e sememelhanças dos entes ou factos observados; a classificação,instrumento da synthese, os distribue, não em series isoladas, mas em classes superiores e inferiores, subordinadas umas ás outras, e fumando um verdadeiro systema, que não é um simples *arranjamento* e *superposição*, mas um tecido, um aggregado de partes reciprocamente unidas

[15] O código mencionado foi vigente no país por mais de cinco décadas (1858-1917).

[16] Transcreve-se o texto exatamente como se lê na obra. Por isso, detalhes ortográficos não contemporâneos e de tipografia, como reticências com dois pontos no segundo parágrafo, ausência de ponto final no primeiro parágrafo, entre outros, são mantidos como encontrado no original consultado.

> Para haver essa união, bem se vê, que a classificação só pode ser o producto de uma idéa geral, de um principio dominante. Se a classificação não é fundada sobre um principio, não existe systema; porque as classes ja não dependem umas das outras.. A escolha desse principio é a grande difficuldade, e determina as classificações *naturaes* e as *artificiaes*.

Como um grande sistema de informação, o sistema de normas jurídicas é apenas um caso que exemplifica a necessidade de ter à disposição dos arquitetos da informação um sistema geral de classificações que permita que outros conceitos sejam especializados, compostos e derivados de conceitos mais abstratos, elementares ou gerais. Esse é um papel que, contemporaneamente, atribui-se às ontologias de fundamentação: o de servir de fundamento para o desenvolvimento de ontologias mais especializadas. Dessa forma, ontologias de fundamentação tratam-se de ontologias genéricas, usadas como referência para desenvolvimento de ontologias de domínio mais específicos[17]. Essas ontologias gerais são descritivistas, e têm objetivos múltiplos: servem como língua franca para sistematizar (e formalizar) conceitos de senso comum dos envolvidos no processo de modelagem e de referência para a generalização de aspectos da realidade, de modo que situações e entidades específicas possam ser explicadas a partir de algum tipo de relação com aqueles conceitos mais gerais (como a partir de relações de exemplificação, instância, subsunção, etc.). A organização de classes é a base para o desenvolvimento de ontologias em diferentes níveis de abstração. Conforme Benevides et al. (2010, p. 2.905):

> *In pace with Degen et al. (DEGEN et al., 2001), we argue that "every domain-specific ontology must use as framework some upper-level ontology". This claim for an upper-level (or foundational) ontology underlying a domain-specific ontology is based on the need for fundamental ontological structures, such as theory of parts, theory of wholes, types and instantiation, identity, dependence, unity, etc., in order to properly represent reality. From an ontology representation language perspective, this principle advocates that, in order for a modeling language to meet the requirements of expressiveness, clarity and truthfulness in representing the subject domain at hand, it must be an ontologically well-founded language in a strong ontological sense, i.e., it must be a language whose modeling primitives are derived from a proper foundational ontology* (GUIZZARDI; MASOLO; BORGO, 2006; GUIZZARDI, 2007b).

Nessa mesma linha de raciocínio, as ontologias têm papel fundamental no contexto em que múltiplos agentes estão envolvidos, caso que se coaduna com problemas apresentados na seção 1.1. Gruber (1993b), para mencionar um exemplo, corrobora essa posição:

> *In the context of multiple agents (including programs and knowledge bases), a* common ontology *can serve as a knowledge-level specification of the ontological commitments of a set of participating agents. A common ontology defines the vocabulary with which queries and assertions are exchanged among agents. For example, the words in the planning ontology*

[17] Esse tema é retomado na seção 4.2.

> *are technical terms that govern the form of inputs and the interpretation of outputs. The definitions tell the user of a planning system what information must be given about an "event" or "resource" in order for the planner to be able to use the information. Each program must commit to the semantics of the terms in the common ontology, including axioms about the properties of objects and how they are related. However, there need be no commitment to the form or content of knowledge* internal *to the agent.*
> *The axiomatization in an ontology need not be a complete functional specification of the behavior of an agent. Common ontologies typically specify only some of the formal constraints that hold over objects in the input and output in the domain of discourse of a set of agents. They do not say which queries an agent is guaranteed to answer. Thus, a commitment to a common ontology is a guarantee of consistency, but not completeness, with respect to queries and assertions using the vocabulary defined in the ontology. Specifying the inferential capabilities of agents is an open research problem*

1.3.4 Por que a escolha da UFO?

A UFO é escolhida devido aos seus fundamentos coerentes logicamente e alinhados com os paradigmas lógico-filosóficos do grupo de pesquisa que os pesquisadores autores deste texto estão envolvidos. Destaca-se, quanto a esse alinhamento, a abordagem modal dada à ontologia (GUIZZARDI, 2007a), o reconhecimento da distinção entre sujeito e objeto e da limitação sensorial, observado por Albuquerque e Guizzardi (2013) na distinção entre *Qualia* e *Quality Region*, a abordagem fundamental de generalização de conceitos, entre outras características. Ressalta-se que a abordagem modal da Lógica é estudada, por exemplo, desde a tese de doutoramento do orientador deste trabalho (LIMA-MARQUES, 1992); a distinção entre sujeito e objeto é base para a Fenomenologia, característica marcante dos trabalhos do grupo de pesquisa CPAI[18]; e abordagem de generalização de conceitos e de construção de teorias de visão de mundo está presente em trabalhos como a Proposta de Teoria Geral para a Arquitetura da Informação de Lima-Marques (2011), entre outros trabalhos.

Além da congruência das linhas de pesquisa, as seguintes características da UFO a torna adequada para a condução deste trabalho:

a) trata-se de uma ontologia de fundamentação construída de forma unificadora a partir de resultados de outras ontologias relevantes na literatura, a saber, GFO/GOL (DEGEN et al., 2001; DEGEN et al., 2011)[19] e OntoClean/DOLCE (WELTY; GUARINO, 2001; GUARINO; WELTY, 2002; GUARINO; WELTY, 2008), embora atualmente seja desenvolvida de forma independente dessas. A seção 4.3 descreve com mais detalhes da UFO e suas correlações com as demais ontologias;

[18] Presente desde o trabalho de Soares (2004), até o nosso (ARAUJO, 2012), entre dezenas de outros do mesmo orientador.
[19] Ver <http://www.onto-med.de/>.

b) a UFO aborda de forma abrangente temas relevantes à modelagem conceitual, como estruturas taxonômicas (GUIZZARDI et al., 2004), relações parte-todo (GUIZZARDI, 2007a; GUIZZARDI, 2009; GUIZZARDI, 2010a; GUIZZARDI, 2010b; GUIZZARDI, 2011), propriedades intrínsecas e espaços de valores (GUIZZARDI; MASOLO; BORGO, 2006; ALBUQUERQUE; GUIZZARDI, 2013; GUIZZARDI; ZAMBORLINI, 2014), papeis, (MASOLO et al., 2005; GUIZZARDI, 2006), propriedades relacionais (GUIZZARDI; WAGNER, 2008; GUARINO; GUIZZARDI, 2015), entidades perdurantes caracterizados como eventos com duração delimitada, compostos de partes temporais (GUIZZARDI; FALBO; GUIZZARDI, 2008; GUIZZARDI et al., 2013), entidades sociais e intencionais, incluindo aspectos linguísticos (GUIZZARDI, 2006; GUIZZARDI; FALBO; GUIZZARDI, 2008; GUIZZARDI; GUIZZARDI, 2010; BRINGUENTE; FALBO; GUIZZARDI, 2011), entre outras (Capítulo 4);

c) existem diversas pesquisas científicas concluídas desenvolvidas com base na UFO (MEDEIROS, 2011a; MEDEIROS, 2011b; FARINELLI; MELO; ALMEIDA, 2013; LI et al., 2014), inclusive no grupo de pesquisa deste trabalho (AZEVEDO, 2014). Essas pesquisas apresentam resultados positivos pela implementação de métodos sugeridos pela ontologia em tela;

d) há trabalhos que investigam aspectos pragmáticos de construção de ontologias com base em interação homem-máquina, como o método desenvolvido por Graças (2010) e Graças e Guizzardi (2010), que visa construir ontologias com base na UFO por meio de perguntas em interfaces com usuários. Esses trabalhos podem servir de referência para a integração futura de abordagens baseadas nos resultados desta pesquisa;

e) há livros publicados por autores não envolvidos diretamente com a pesquisa original, o que confirma a relevância da ontologia para outros grupos de interesse (HENDERSON-SELLERS, 2012; HENDERSON-SELLERS, 2011);

f) há aplicações práticas da UFO documentadas na literatura, como:

   – Guizzardi et al. (2009): descreve um caso de aplicação da ontologia no domínio de petróleo e gás;

   – Carolo e Burlamaqui (2011): apresentação realizada por representantes do Portal de jornalismo online G1[20], das Organizações Globo, que expõe o sucesso no uso de UFO para reformulação do portal de esportes da empresa;

g) há uma pluralidade de ontologias construídas sobre ontologias de fundamentação em geral, a exemplo da ontologia para o domínio artefatos multimídia de Arndt et al. (2007). Usando a UFO, de forma não exaustiva, pode-se mencionar:

[20] <http://g1.globo.com/>

- a ontologia de mensuração de software de Barcellos, Falbo e Dal Moro (2010);
- as ontologias de processos de software de Guizzardi, Falbo e Guizzardi (2008) e de Bringuente, Falbo e Guizzardi (2011), em que ambas também desenvolvem conceitos ontológicos a respeito de entidades sociais;
- a ontologia de arquitetura de continuidade serviços de software de Silva et al. (2012);
- a *core-ontology* de serviços proposta por Nardi et al. (2013)[21] e aprimorada em QUIRINO et al. (2015) com base na ideia de *"enterprise ontology pattern language"* de Falbo et al. (2014);
- a ontologia de referência para atos normativos de Barcelos, Guizzardi e Garcia (2013);
- a ontologia de referência de conceitos jurídicos de Griffo, Almeida e Guizzardi (2015);

h) ontologias de fundamentação também são usadas para realizar interoperabilidade semântica entre diferentes domínios (GUIZZARDI et al., 2011), o que é de interesse da Arquitetura da Informação, especialmente considerando a natureza ubíquoa da informação (RESMINI; ROSATI, 2009). Nessa linha de aplicação, Gonçalves, Guizzardi e Filho (2011) descreve como a UFO é usada no contexto de adequação aos padrões de dados da área de eletrocardiograma.

Em seu trabalho, Guizzardi desenvolve não apenas a UFO como evolução de ontologias anteriores[22], mas também estabelece critérios de avaliação de adequação de linguagens usadas na formalização de ontologias, além de, com base nesses critérios, propor uma série de ajustes à linguagem UML (OBJECT MANAGEMENT GROUP, 2011b), para que esta se adeque ao propósito de formalizar conceituações. Esse novo perfil da linguagem UML proposta pelo autor foi denominada OntoUML. Trata-se de uma linguagem diagramática, com a qual se pode construir especificações de ontologias de domínio com base na UFO. A OntoUML traz avanço significativo à área de Modelagem Conceitual e de Arquitetura da Informação, pois permite que vários ganhos sejam obtidos, especialmente o de ligação e da coerência teórica da metafísica de uma ontologia fundada em semântica do mundo real – no caso, a UFO – com a prática da construção de uma conceituação específica, o que é percebido devido a redução de ambiguidade dessas conceituações. Essa abordagem ontologicamente consistente de modelagem permite que outros instrumentos de formalização de ontologias sejam construídos com base, ou a partir dela. Por exemplo, o grupo *Ontology & Conceptual Modeling Research Group* da Universidade Federal do Espírito Santo (NEMO)[23] tem desenvolvido, com base em tecnologias livres e na linguagem

[21] No âmbito da Arquitetura da Informação, um estudo sobre o tema arquitetura de serviços é realizado por Rios Filho (2014).
[22] Ver seção 4.3.
[23] O grupo de pesquisa NEMO é apresentado em Guizzardi et al. (2009) e em Guizzardi et al. (2011).

Java, um editor para OntoUML que permite uma série de verificações adicionais sobre os modelos construídos com a linguagem, como a definição de restrições sobre modelos definidas com *Object Constraint Language* (OCL) (ISO/IEC, 2012b), a identificação de padrões e anti-padrões (GUIZZARDI, 2009; GUIZZARDI; GRAÇAS; GUIZZARDI, 2011; GUIZZARDI, 2014; SALES; BARCELOS; GUIZZARDI, 2012) e a integração com Alloy (GUERSON; ALMEIDA; GUIZZARDI, 2014; BRAGA et al., 2010). O editor é chamado de *OntoUML Lightweight Editor* (OLED)[24]. No caso, como um instrumento de *model checking*, Alloy é utilizado pelos autores mencionados como um tipo de validação da modelagem, baseado na busca de contra-modelos que são consistentes com as especificações da ontologia descrita (satisfatibilidade), e também aqueles que eventualmente não sejam coerentes com a intenção dos autores da ontologia.

Ao analisar a abordagem adotada nas pesquisas sobre Arquitetura da Informação realizadas grupo do CPAI, é possível observar uma espécie de objetivo compartilhado entre os pesquisadores a respeito do desenvolvimento de uma ontologia sobre AI. Esse é um objetivo, por exemplo, presente nos trabalhos de Siqueira (2008), Albuquerque (2010), Albuquerque e Lima-Marques (2011) e de Siqueira (2012a). Os próprios autores desta pesquisa investigaram conceitos sobre *arquitetura* e *configuração* com intuito de servirem de base para a construção futura de uma ontologia de referência para a área (ARAUJO, 2012). Nesse contexto, a escolha da UFO também se justifica no sentido de que pode trazer às pesquisas futuras fundamentos para o desenvolvimento de uma ontologia de referência sobre AI que seja cognitiva e filosoficamente bem-fundamentada. Dessa forma, as sistematizações teóricas referentes à UFO realizada neste trabalho, especialmente na subseção 4.5.1 e no Glossário da UFO, são contribuições diretas a esse pleito.

Por fim, cabe assentar que a escolha do fragmento endurante da UFO, chamada de UFO-A, é justificada exclusivamente para limitação do escopo de pesquisa. De toda forma, não se trata de uma escolha arbitrária. Ela reflete a estratégia adotada em Guizzardi (2005), que dedicou-se inicialmente a desenvolver essa ontologia antes de avançar sobre a ontologia de perdurantes. Por isso, registra-se na seção 11.3 que trabalhos futuros podem estender os resultados obtidos nesta pesquisa com intuito de incluir conceitos que envolvem as ontologias de perdurantes (UFO-B), de objetos sociais (UFO-C) e de serviços (UFO-S), apresentadas na seção 4.3, além de outras ontologias de fundamentação.

1.3.5 Por que a integração de Programação em Lógica e UFO?

De acordo com Guizzardi e Zamborlini (2014), a modelagem conceitual exige uma abordagem dupla no que tange às linguagens de modelagem: em primeiro lugar, é exigido que se obtenha acordos explícitos quanto aos compromissos ontológicos assumidos com

[24] Até a conclusão deste texto, o trabalho podia ser consultado neste endereço: <https://github.com/nemo-ufes/ontouml-lightweight-editor>.

determinada representação de discursos. Em segundo lugar, tornar explícitos esses acordos pode deixar a linguagem de modelagem proibitiva do ponto de vista computacional:

> *The expressivity in a modeling language needed to make explicit the ontological commitments of a complex domain tends to make this language prohibitive from a computational point of view in tasks such as automated reasoning. Conversely, computationally tractable logical languages tend to lack the expressivity to handle this essential aspect of semantic interoperability. For this reason, as defended in Guizzardi (2007b), Masolo et al. (2001), we need to account for a two-step separation of concerns in domain modeling: (i) firstly, we should develop conceptual models as rich as possible to efficiently support the tasks of meaning negotiation and semantic interoperability across* "communities, organizations, viewpoints, and authorities"*; (ii) once the proper relationship between different information models is established, we can generate (perhaps several different) implementations in different logical languages addressing different sets of non-functional design requirements.*

Ainda sobre esse mesmo tema, em Guizzardi (2007b) encontra-se:

> *On one side we need a language that commits to a rich foundational ontology. This meta-ontology, however, will require the use of* ***highly expressive formal languages*** *for its characterization, which in general,* ***are not interesting from a computational point of view****. On the other side, languages that are efficient computationally, in general, do commit to a suitable meta-conceptualization. The obvious question is then: how can we design a suitable general ontology representation language according to these conflicting requirements?*
> *The position advocated here is analogous to the one defended in Masolo et al. (2001), namely, that we actually need two classes of languages. [. . .] On one hand, in a conceptual modeling phase in Ontology Engineering, highly-expressive languages should be used to create strongly axiomatized ontologies that approximate as well as possible to the ideal ontology of the domain. The focus on these languages is on representation adequacy, since the resulting specifications are intended to be used by humans in tasks such as communication, domain analysis and problem-solving. The resulting domain ontologies, named* reference ontologies *in Guarino (1998), should be used in an* off-line *manner to assist humans in tasks such as* ***meaning negotiation*** *and* ***consensus establishment****. On the other hand, once users have already agreed on a common conceptualization, versions of a reference ontology can be created. These versions have been named in the literature* lightweight ontologies*.* (grifos nossos)

Como geralmente o ontologista analisa a realidade de forma assistida por outros indivíduos, espera-se que exista certo grau de acordo ou de consenso sobre a ontologia construída. Portanto, é natural que a escolha da linguagem, utilizada para comunicação entre os sujeitos, leve em consideração certos aspectos, como laconicidade, correção, clareza (GUIZZARDI, 2007b; GUIZZARDI, 2013), expressividade, simplicidade, estabilidade semântica, existência de mecanismos de validação e de abstração e base formal (HALPIN; MORGAN, 2008, p. 60-62). Entre outras características, geralmente adota-se uma linguagem visual, como as apresentações diagramáticos de UML (OBJECT MANAGEMENT

GROUP, 2011b) e de BPMN (OBJECT MANAGEMENT GROUP, 2011a), com vias de se facilitar acordos por múltiplas pessoas. Nesse sentido, conforme Colomb (1998, p. 92), ferramentas de modelagem conceitual devem atender a dois aspectos, as estruturas de dados e a representação visual:

> *Modelling tools generally involve two aspects: data structures, which are a bridge between the natural language descriptions of the universe of discourse and the formal representations in the ultimate implementation; and a graphical representation of the data structures, which is intended to facilitate understanding of the model both by the domain expert and by the systems analyst or knowledge engineer. Typically some of the detail of the data structure is represented in graphical form and some is represented in more or less formal textual languages.*

A estrutura de dados e a representação visual, ou mesmo as linguagens textuais, referem-se à um caráter descritivo da Lógica, que é mais importante à Inteligência Artificial e à Modelagem Conceitual do que é normativo à Lógica. Isso se dá pelo fato de a Lógica se atentar não apenas à capacidade de descrição, mas especialmente aos modos de raciocínio sobre os conceitos representados pelas linguagem. Porém, embora existam abordagens que exploram com mais afinco um ou outro aspecto, nenhum deles pode ser desprezado e devem estar em consonância, o que, idealmente, pode se dar tanto do ponto de vista matemático, ou computacional, quanto do ponto de vista antropomórfico de entendimento, de conhecimento e de raciocínio cognitivo.

> *Logical representations grew out of the efforts of philosophers and mathematicians to characterize the principles of correct reasoning. The major concern of logic is the development of formal representation languages with sound and complete inference rules. As a result, the semantics of predicate calculus emphasizes truth-preserving operations on well-formed expressions. An alternative line of research has grown out of the efforts of psychologists and linguists to characterize the nature of human understanding. This work is less concerned with establishing a science of correct reasoning than with describing the way in which humans actually acquire, associate, and use knowledge of their world. This approach has proved particularly useful to the AI application areas of natural language understanding and commonsense reasoning* (LUGER, 2009, p. 228-229).

Embora concorde-se com as posições apresentadas referentes à limitação imposta por forças tradicionalmente opostas na definição de linguagens – "expressividade" *contra* "decidibilidade" – espera-se que com uma linguagem construída especialmente para descrição de acordos sobre compromissos ontológicos seja possível balancear a clareza nas descrições com características de dedução sobre os modelos construídos, de modo que se possa extrair informações (novas ou atuais) dos discursos sobre ontologias descritas formalmente na linguagem. Mesmo que não se consiga atingir propriedades como decidibilidade e completude em nossa abordagem, Prolog é uma escolha sensata, uma vez que, conforme descrito na subseção 1.3.2, há diversos recursos para definir linguagens específicas de

contexto, sem perder a conexão direta com a Lógica Clássica, Lógica Modal ou mesmo Lógica Paraconsistente, entre outras, e ainda manter o aspecto prático da capacidade de *aplicação imediata* dos programas construídos com base na formalização dos sistemas lógicos. Embora não se possam garantir certas propriedades lógicas, é possível avançar em outros nichos, como a capacidade de construir diferentes tipos de inferências sobre as formalizações de discursos, realizar consultas definidas à base de dados que represente a ontologia descrita, incluir características de raciocínios não-monotônicos (ANTONIOU, 1997), como os que envolvam regras padrão e afirmações retratáveis (KOONS, 2014), entre outras possibilidades. Nesse sentido, o desenvolvimento de uma linguagem estruturada textual pode ser desenvolvida em Prolog para ser usada como elemento facilitador da formalização e da comunicação de acordos ontológicos de modo mais rico do que o que é possível atualmente com as ferramentas disponíveis.

Uma característica da UML é a ausência de recursos para formalizar diretamente modalidades lógicas. Por isso, o importante aspecto modal da UFO é perdido em modelos conceituais construídos com base nessa ontologia quando formalizados em OntoUML, pois este herda aquela limitação. Com uma Lógica Modal, como as disponíveis em Prolog por meio de MProlog (seção 3.2), é possível resgatar parte da semântica modal alética da UFO, ou mesmo desenvolver outros tipos interpretações modais, como modalidades referentes a múltiplos agentes, como de fato é realizado neste trabalho. Reforça-se, portanto, o uso de Prolog, especialmente por sua disponibilidade de módulos de lógicas não clássicas.

A sistematização da UFO em uma linguagem de Programação em Lógica traz ainda ganhos em diferentes dimensões:

a) **aplicação imediata**: por usar um paradigma lógico coerente, a implementação de regras e conceitos da UFO em Prolog servem como um tipo de formalização da ontologia. Além disso, a implementação torna a formalização passível de execução com objetivos finalísticos próximos às necessidades de aplicação em situações reais. Dessa forma, avanços teóricos da lógica subjacente à ontologia, ou no próprio sistema de conceitos da ontologia de fundamentação, podem ser desenvolvidos e passarem a entregar benefícios diretos às aplicações práticas;

b) **ganhos pedagógicos**: ferramentas e programas executáveis podem funcionar como instrumentos pedagógicos no ensino de Lógica, de linguagens de computador e de ontologias. Programas construídos com base em conceitos ontologicamente bem estudados possibilitam a otimização da relação ensino-aprendizagem, uma vez que exercitam diferentes aspectos do tratamento da informação, o que estimula o raciocínio por meio de diferentes paradigmas de pensamento;

c) **gestão do "conhecimento"**: cada vez mais cresce o volume de informação científica e filosófica. Um estudante precisa ler centenas ou até milhares de

artigos e livros para entender o *status quo* de determinada área. Porém, se a base de informação cresce rapidamente, o papel do cientista da informação em sistematizar esse conhecimento é cada vez mais importante. Desse modo, um ganho característico de formalizações em geral é também com a gestão do "conhecimento", no sentido de sistematização de informações, a respeito de determinada área. Portanto, ressalta-se a importância da sistematização terminológica com a construção de glossários e tesauros, como o glossário produzido neste trabalho.

Dependendo da forma como a arquitetura do sistema lógico em Prolog é preparada, não apenas a UFO, mas outras ontologicas de fundamentação, podem também ser formalizadas. Isso traz generalização e amplia a relevância da pesquisa no sentido de trazer ao universo de Programação em Lógica os benefícios do uso de ontologias de fundamentação por meio de metamodelagem. Esse tipo de qresultado é, de fato, obtido neste trabalho, conforme descrito na Parte III.

1.4 Metodologia de pesquisa

1.4.1 Classificação da pesquisa

Esta pesquisa é classificada como explicativa[25] com propostas de novas teorias e implementação inéditas de protótipos. Ela busca o aprimoramento de fundamentos científicos relacionados ao estudo da Arquitetura da Informação. O campo de investigação é transdisciplinar com base teórica na Arquitetura da Informação proposta pelo grupo de pesquisa no qual os autores deste trabalho estão inseridos[26]. A base empírica utilizada é concentrada nos temas Modelagem conceitual e Ontologia, Lógica Clássica, Lógicas Modais, Inteligência Artificial, Programação em Lógica Clássica e Programação em Lógica Modal.

O procedimento técnico adotado na Parte II e no Glossário da UFO é o levantamento bibliográfico exploratório, elaborado principalmente a partir de material já publicado, constituído de livros, artigos de periódicos, de revistas e de anais de eventos, dissertações e teses concluídas e em andamento, normativos, guias, arcabouços de boas práticas e documentos públicos governamentais relacionados aos temas pesquisados. Já o procedimento adotado nos capítulos da Parte III é o de construção de modelos e de prototipagem.

No que tange à UFO, o levantamento bibliográfico inclui cobertura de um espaço temporal de cerca de 10 anos de publicações sobre a ontologia, contados a partir de 2005. No caso, investiga-se especialmente os trabalhos publicados pelo autor da UFO,

[25] A classificação é baseada nas classes de investigação científica propostas por Melo (2010, p. 151-162).
[26] <http://www.cpai.unb.br/>

individualmente ou em colaboração. Essa estratégia de investigação reflete nas demasiadas referências às obras indicadas nas entradas de "Guizzardi", na seção de Referências.

1.4.2 Modelo de referência e Visão de Mundo

Um modelo de referência complementar que pode ser utilizado para classificar esta pesquisa é o método de *modelagem* e de *aplicação*, proposto por van Gigch e Pipino (1986) e defendido por Lacerda e Lima-Marques (2014) e Lacerda (2005) como um método adequado para pesquisas em Arquitetura da Informação. Conhecido como M^3, ele se sustenta em três níveis de análise incidentes sobre o objeto científico, conforme segue. A Tabela 1 sistematiza uma visão desses níveis.

a) **Epistemológico** – também denominado estratégico ou "de metamodelagem" – constitui o arcabouço conceitual e metodológico de uma comunidade científica específica e objetiva investigar a origem do conhecimento da disciplina ou campo de conhecimento, justificar seus métodos de raciocínio e enunciar sua metodologia;

b) **Científico** – também denominado tático ou "de modelagem" – compreende a previsão, descrição e explicação para os problemas e respectivas soluções, por meio do desenvolvimento de modelos e teorias;

c) **Prático** – também denominado operacional ou "de aplicação" – aplicação efetiva dos modelos, teorias, técnicas e tecnologias desenvolvidos nos demais níveis para a solução dos problemas reais.

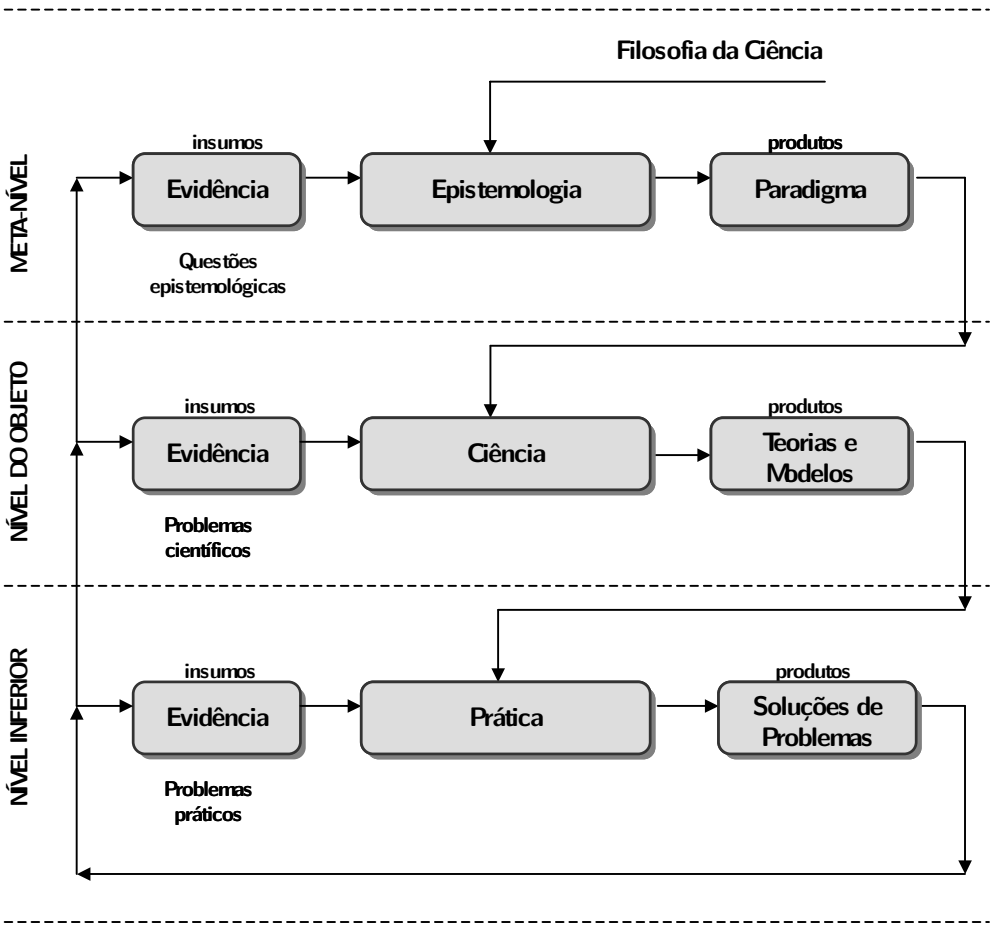
Tabela 1 – Níveis de investigação

Nível de Investigação	Insumos	Sistemas de Investigação	Produtos
Meta-nível	Filosofia da Ciência	Epistemologia	Paradigma
Nível do objeto	Paradigmas do metanível e evidências do nível inferior	Ciência	Teorias e modelos
Nível inferior	Modelos e métodos do nível do objeto e problemas do nível inferior	Prática	Solução de problemas

Fonte: van Gigch e Pipino (1986)

O diagrama apresentado na Figura 1, adaptado do texto de van Gigch e Pipino (1986, p. 74) e traduzido por Lacerda (2005, p. 17), representa a hierarquia de sistemas de investigação científica e as relações entre eles, conforme a M^3. Nessa representação, as questões epistemológicas são formuladas tanto a partir de insumos da Filosofia da Ciência quanto por evidências produzidas nos níveis inferiores, o científico e o prático. Os níveis inferiores, por sua vez, servem de evidência para os níveis superiores de investigação, de modo que a Ciência encontra evidências nas Prática.

Figura 1 – Metodologia de Metamodelagem (M^3): hierarquia de sistemas de investigação



Fonte: van Gigch e Pipino (1986, p. 74)

Pela M^3, os sistemas de investigação científica podem ser classificados com base na finalidade:

a) *conceituais*, quando tratam de questões filosóficas, epistemológicas e teóricas sobre a Ciência;

b) *de modelagem*, quando se referem ao desenvolvimento, formulação e validação de modelos, dos limitados aos genéricos; e

c) *empíricos*, quando são utilizados para observar o relacionamento entre variáveis, testar sua invariância sob determinadas condições e inferir generalizações para contextos mais abrangentes. Nesta classe encontram-se os estudos de caso, os estudos e os testes de campo, e os estudos laboratoriais.

A Parte II (Referenciais teóricos) contém elementos dos três níveis de investigação. Já na Parte III (Resultados) apresenta-se resultados referentes a teorias e métodos para formalização de ontologias, bem como exemplos de aplicação a solução de problemas concretos, portanto, focados nos dois níveis inferiores ilustrados pela Figura 1.

1.4.3 Resultados no âmbito da Arquitetura da Informação

A Ciência da Informação, especialmente a Arquitetura da Informação, é tradicionalmente vista como uma ciência transdisciplinar, ou seja, como uma ciência que surge de pesquisas que relacionam conhecimento teórico comum que interconectam diferentes epistemologias científicas. Este trabalho, como uma ocorrência de uma pesquisa em Arquitetura da Informação, contém uma interseção científica entre Lógica, Ciência da Computação, Programação em Lógica, Modelagem Conceitual e Ciência da Informação. No entanto, os resultados da pesquisa não são reclamados como frutos individuais de nenhuma das primeiras áreas, a não ser da última. Dessa forma, embora contemple tópicos tratados pela Lógica, não é escopo deste trabalho apresentar detalhes e resultados específicos referentes a novos mecanismos de dedução, ou apresentar provas de propriedades meta-lógicas dos sistemas abordados. No entanto, apresenta-se resultados no contexto de Arquitetura da Informação e de Ciência na Informação que tangem ao *tratamento lógico da informação* no contexto de *lógica aplicada*. No caso, da informação a respeito de ontologias. Ou seja, a exemplo do que se pode conceber como "tradicional" numa área transdisciplinar como a Ciência da Informação, esta pesquisa reúne resultados de diferentes áreas científicas para responder às questões no âmbito da Ciência da Informação.

1.4.4 Aspectos técnicos do texto

Normalmente aspectos técnicos da manifestação textual da obra não são discutidas em uma tese de doutorado. Entretanto, entende-se oportuno e conveniente mencionar e esclarecer determinadas decisões que tangem ao estilo e à técnica de produção deste texto. Parte desses destaques são incluídos para atender a sugestões de revisores.

1.4.4.1 Textos mantidos em língua estrangeira

Como em toda tradução existe o risco de se introduzir ideias diferentes daquelas intencionadas pelo autor original, e uma vez que a língua nativa da maioria dos textos utilizados nesta pesquisa não é o português, opta-se por não traduzir citações em inglês ou em espanhol. No entanto, com a intenção de reduzir a qualificação mínima necessária para a leitura deste texto, citações em outras línguas são mantidas no idioma original, mas traduções nossas para o português são fornecidas.

1.4.4.2 Tradução de "açúcar sintático"

No âmbito do desenvolvimento de linguagens de programação, *"syntactic sugar"*, ou *"syntactic sugaring"*, são compreendidas como construções internas à sintaxe de linguagens formais que visam tornar as construções mais fáceis de serem escritas e lidas por seres humanos. Em geral, uma *"syntactic sugar"* pode ser traduzida para expressões primitivas normalmente mais complexas da linguagem alvo. Por isso, elas exercem importante papel

pragmático nas linguagens de programação. Atribui-se o uso original da expressão em inglês, *"syntactic sugar"*, à Landin (1964, p. 315), que a usou para descrever uma estratégia de construção de uma sintaxe simplificada para ALGOL.

A expressão em português "açúcar sintático" – que também poderia ser traduzida como "sintaxe açucarada" ou "atalho sintático" – é comumente encontrada em livros da área de Computação publicados com alvo no público lusófono, como é o caso da tradução da obra de Russell e Norvig (2010), originalmente em língua inglesa: "Como ocorre com os conjuntos, é comum usar açúcar sintático em sentenças lógicas que envolvem listas." (RUSSELL; NORVIG, 2013, seção 8.3.3). Textos sobre linguagens de programação escritos originalmente na língua nacional também utilizam a expressão, como em Araújo e Acióly (2008, p. 53), que descrevem uma das possíveis notações de conjuntos matemáticos do Haskell como um caso de açúcar sintático: "A sintaxe da compreensão de listas corresponde muito de perto à notação de conjuntos matemáticos, sendo considerada um açúcar sintático."

Pelo fato de ser corrente em textos na língua portuguesa, o uso da expressão "açúcar sintático" como tradução de *"syntactic sugar"*, é preferível neste texto. A expressão é usada, por exemplo no Capítulo 7.

1.4.4.3 Código-fonte anexo e padrão PDF/A

Dentre os resultados obtidos por esta pesquisa, está um software escrito em Prolog que serve de implementação de referência do resultado sobre Ontoprolog. Os códigos-fonte dos programas de computador produzidos estão anexados à versão eletrônica deste documento. Eventualmente pode ser necessário ajustar o nível de segurança do leitor de arquivos em formato *Portable Document Format* (PDF) para que o arquivo anexo com o código-fonte possa ser aberto. Alternativamente a isso, a seção 10.1 apresenta sugestões de ferramentas gratuitas capazes de ler os arquivos anexos à versão em PDF deste documento.

O documento eletrônico que suporta este texto atende a especificação PDF/A-3b para documentos de preservação de longo prazo da ISO 19005:2012 (ISO, 2012a). Isso significa que o documento é produzido considerando um formato de arquivo idealizado para guarda permanente. Atender a essa especificação em trabalhos acadêmicos é defendido, por exemplo, por Arms et al. (2014, p. 11). Os autores apresentam como cenário de uso de documentos PDF/A-3 os anexos de arquivos de dados de pesquisas acadêmicas, pois é um método que garante a recuperação do texto e dos arquivos que auxiliaram ou sustentaram o desenvolvimento da pesquisa.

A comprovação do atendimento ao formato PDF/A-3b pode ser obtida por meio do software *Adobe Acrobat Pro Preflight*. Uma versão gratuita por 30 dias pode ser obtida do sítio <http://www.adobe.com/br/products/acrobatpro.html>, acesso de 21 jan 2015.

Para visualizar o arquivo anexo com o código-fonte e comprovar o atendimento à

especificação ISO, é necessário utilizar um leitor de documento PDF que reconheça esse padrão.

1.4.4.4 Referências cruzadas

Este texto é preparado com intuito de enriquecer a experiência de leitura a partir da versão digital, em detrimento da versão textual impressa. Embora aspectos tipográficos específicos para a versão impressa tenham sido observados, devido à numerosa quantidade de páginas da obra, a versão digital deste documento contém uma série de referências cruzadas que visam otimizar a leitura do texto digital. Dessa forma, o leitor pode utilizar recursos como *hyper link* para avançar ou retroceder nas páginas. O recurso é especialmente útil quando relacionam as partes textuais com entradas do Glossário da UFO, das Referências e também para seções, Figuras, Definições, Quadros, entre outros. As entradas do glossário geralmente são indicadas entre parênteses, mas eventualmente são usadas no curso do texto. Em todos os casos, as palavras com *hyper links* são destacadas na cor azul.

1.4.4.5 Referências reversas da bibliografia

Com intuito de favorecer análises técnicas desta obra, nas Referências extrapola-se as recomendações da ABNT NBR 6023:2002 (*Informação e documentação - Referência - Elaboração*) ao incluir referências da bibliografia para a citação no texto. Adicionalmente, contagem do número de vezes que a obra listada é mencionada no texto também é incluída.

Parte II

Referenciais teóricos

2 Arquitetura da Informação

Este capítulo apresenta o contexto de Arquitetura da Informação (AI) em que este trabalho está inserido. Ele consiste em uma visão geral da área, com intuito de apresentar a característica eminentemente transdisciplinar da AI, bem como salientar como as diferentes disciplinas científicas são abordadas para alcance dos resultados pretendidos da pesquisa. Além disso, este capítulo exerce papel complementar de justificação desta pesquisa, uma vez que apresenta argumentos que visam ratificar a relevância e a adequação da investigação.

Com esses objetivos, a seção 2.1 apresenta linhas de pesquisa e áreas de atuação comumente encontradas na literatura a respeito de Arquitetura da Informação. Na seção 2.2 tenta-se criar certa harmonização dessas diferentes disciplinas e mostrar como é possível organizá-las para obter resultados cientificamente relevantes. A seção 2.3 aborda a questão dos *discursos*, e como eles são interpretados nesta pesquisa. Na seção 2.4 introduz-se a noção de *engenharia lógica* e argumenta-se quanto à adequação da abordagem desta pesquisa com esse conceito. Por fim, a seção 2.5 contém um breve fechamento do capítulo que sumariza as principais ideias desenvolvidas.

2.1 A transdisciplinar Arquitetura da Informação

A *Science of Information* (SI), conforme proposta por Hofkirchner (2011, p. 372), é uma disciplina que lida com processos de informação de sistemas naturais, sociais e tecnológicos de modo mais amplo do que a tradicional *Information Science* (IS). Enquanto esta é uma área proveniente da Biblioteconomia e da Documentação, aquela é uma disciplina complementar e emergente que busca um entendimento mais amplo do conceito de informação. Isso inclui integração epistemológica com áreas como Lógica, Matemática, Ciência da Computação, Física, Cibernética, entre outras. Nesse contexto, Lima-Marques (2011) propõe a *Arquitetura da Informação* (AI) como uma área da *Science of Information* que trata a *informação* como *configurações do mundo* que podem ser experienciada por sujeitos sociais. Dessa forma, a AI estuda a informação enquanto entidade ontológica do mundo, os sujeitos e a experiência dos sujeitos com a informação (ARAUJO; LIMA-MARQUES, 2014). É, portanto, uma área transdisciplinar, construída como produto e intersecção de diferentes epistemologias científicas. Conforme Lacerda e Lima-Marques (2014, p. 3):

> *Information architecture is established in a context where the values of universality and certainty have given place to plurality and complexity. Thus, its nature is inherently transdisciplinary, and its methods, models*

> *and theories are strongly influenced by or even derived from a number of external sources and disciplines, including information science, architecture, design, ergonomics, usability, computer science, business administration, philosophy, cognitive science, and linguistics, to cite a few.*

Entre as diferentes abordagens da área, a que poderia ser dita como uma das mais egrégias, resolve-se em aspectos pragmáticos da organização e da apresentação da informação em mídias digitais de comunicação (como os sítios da Internet). Essa abordagem herda seus pressupostos, teorias e técnicas da tradicional Ciência da Informação, esta como organizadora da informação sobre a Ciência[1]. A abordagem da organização das mídias digitais ganhou reconhecimento mundial principalmente devido ao famoso "Livro do Urso Polar" de Morville e Rosenfeld (2006)[2], que levou holofotes à área graças a sua abordagem direta e pragmática no tratamento, organização e publicação da informação digital.

Devido ao estrondoso sucesso do livro, que se encontra em sua terceira edição, a Arquitetura da Informação tem colhido frutos dessa exposição com o aumento dos interessados em pesquisas na área, mas também tem sofrido com uma visão turva de seu objeto de pesquisa, uma vez que é comum encontrar visões sobre a Arquitetura da Informação como mera produtora de websites (RESMINI; ROSATI, 2009). Embora seja justificável e compreensível usar conhecimentos para se aprimorar a experiência em sítios da Internet, desviar o foco do objeto real de pesquisa da área parece ser injustificável: o objeto de pesquisa da AI não poderia ser outro, senão a própria informação, o que, por si só, deve ser suficiente para posicionar os ramos de pesquisa referentes aos websites como acessórios e contingentes.

Além da abordagem de websites, destacam-se, igualmente notórias, outras abordagens ditas "de arquitetura da informação". Por exemplo, menciona-se a proposta de informação pervasiva de Resmini e Rosati (2011), que propõem trabalhar com a informação de forma onipresente na realidade; de design da informação de Passos, Lima-Marques e Mealha (2011), que apresenta conceitos de desenhos gráficos, representações visuais e se sugestiona na linha filosófica de Flusser (2007) para propor que a informação é algo que pode ser configurada e manipulada; a abordagem de informação ubíqua de Morville (2005), Resmini e Rosati (2009); a abordagem organizacional, que lança mão de conceitos de administração, gestão, governança e segurança para construir arcabouços de arquitetura da informação (DUARTE; LIMA-MARQUES, 2009; DIETZ, 2009; GODINEZ et al., 2010), inclusive com foco na ideia de organização e integração de espaços (DING; LIN, 2010); as de natureza filosófica de busca de grandes categorias epistemológicas (ALBUQUERQUE, 2010;

[1] A Biblioteconomia, por exemplo, é uma dessas disciplinas historicamente evoluída com objetivo de ordenar a classificação, indexação, catalogação e pesquisa de informação registrada.

[2] Na linha de arquitetura da informação para web, destaca-se também a obra de Wodtke e Govella (2009) que apresenta abordagem mais centrada na experiência do usuário e menos técnica do ponto de vista de Ciência da Informação, quando comparada com a obra de Morville e Rosenfeld (2006).

SIQUEIRA, 2012a; SIQUEIRA, 2012b); e de natureza dura, que lançam mão de conceitos da Física com vias de construir uma " Teoria Unificada da Informação" (HOFKIRCHNER, 1998; CRNKOVIC; HOFKIRCHNER, 2011; HOFKIRCHNER, 2011; BURGIN, 2010; BURGIN, 2011) e para a ideia de *Arquitetura da Informação* (LIMA-MARQUES, 2011); entre outras.

A linha da informação ubíqua e pervasiva, também relacionada à ideia de *Internet das Coisas* tem sido pauta de estudos recentes na AI, especialmente porque têm trazido novas abordagens e instrumentos, que vão além dos tradicionais métodos e ferramentas da Biblioteconomia:

> *Either way, through the past fifteen years digital libraries have matured mostly inside the womb of library and information science, with a little contribution, and most of this eminently technical, coming from the field of computer science. Whatever the reasons behind this, digital libraries have basically perpetuated on the World Wide Web the ordered, static model of the traditional bibliographical record* (RESMINI, 2010, p. 8).

De certa forma, Resmini (2010) e Morville e Rosenfeld (2006) seguem uma linha semelhante: enquanto os últimos dedicaram-se a trazer os conceitos da biblioteconomia à AI, o primeiro traz os conceitos de design e de ubiquidade – ou como ele prefere, "pervasividade" – das interações homem/informação (que vão além da relação da tradicional homem/computador e da contemporânea homem/dispositivo). Nas palavras de Resmini (2010, p. 17):

> *Information Architecture (IA) is an emerging discipline and community of practice focused on bringing principles of, among others, livrary science, design and architecture to information space (Rosenfeld; Morville, 2002*[3]*; (RESMINI; ROSATI, 2010)).*

Com base no trabalho de White (2004), Resmini (2010, p. 24-27) apresenta que a Arquitetura da Informação possui principalmente três abordagens diferentes:

a) **abordagem do design**, cuja proposta inicial foi cunhada por Wurman em 1975[4], que apresentada a ideia de "arquitetura da informação" para uma platéia de arquitetos interessados em design. Na ocasião, Wurman expôs a ideia de que o design e a arquitetura são a base para uma ciência e uma arte de criar "instruções para espaços organizados" (WURMAN, 1997);

[3] A obra citada pelo autor nessa passagem não foi encontrada na bibliografia listada por ele, de modo que não se pode confirmar a referência apresentada. Porém, provavelmente se trata da segunda edição da obra Morville e Rosenfeld (2006), publicada em 2002.

[4] Nesse caso, embora não seja mencionado pelos autores, a obra original cujo o termo "Arquitetura da Informação" foi cunhada por Wurman provavelmente é a indicada na entrada Wurman e Katz (1975) das Referências.

b) **abordagem da ciência**, cujos principais representantes são Morville e Rosenfeld, que trouxeram metodologias da Ciência da Informação – ou, mais especificamente, da Biblioteconomia – para serem usadas na navegação, rotulação e estrutura de sites;

c) **abordagem do recurso**, apresentada como uma abordagem baseada numa "lógica de sistemas de informação" ao invés da lógica da "experiência do usuário". Esta abordagem é apresentada pelo autor como uma abordagem "minoritária". Os principais representantes desta abordagem, segundo Resmini, são Evernden e Evernden (2003).

Ocorre, porém, que a essa lista apresentada por Resmini (2010, p. 24-27) é demasiadamente limitada. Há várias outras abordagens para a Arquitetura da Informação que vão além do paradigma do Design, da Biblioteconomia – que é uma parte Ciência da Informação – e dos sistemas de informação. Defende-se que é a própria Informação que integra a Arquitetura às pessoas, de modo que diferentes epistemologias devem ser usadas para compor um arcabouço inteligível para área. Isso é defensável especialmente devido ao caráter transdisciplinar do próprio objeto da ciência de AI: a *Informação*. Nesse sentido, Díaz Nafría e Alemany (2011) destacam que é possível a construção de uma Teoria Unificada da Informação e propõem um conjunto de domínios e de arcabouços teóricos que devam ser investigados com o intuito de atingir o objetivo final de construir essa teoria unificada. Para os autores, como a natureza ampla dos fenômenos da informação perpassam diferentes aspectos naturais, sociais e técnicos do mundo, as investigações deveriam ser articuladas entre os seguintes domínios:

a) Aspectos formais da informação: tópicos lógicos e matemáticos;

b) Ciências físicas: tópicos físicos e químicos;

c) Ciências da vida: tópicos biológicos;

d) Cognição e psicologia: epistemologia, cognição, consciência e outros tópicos psicológicos;

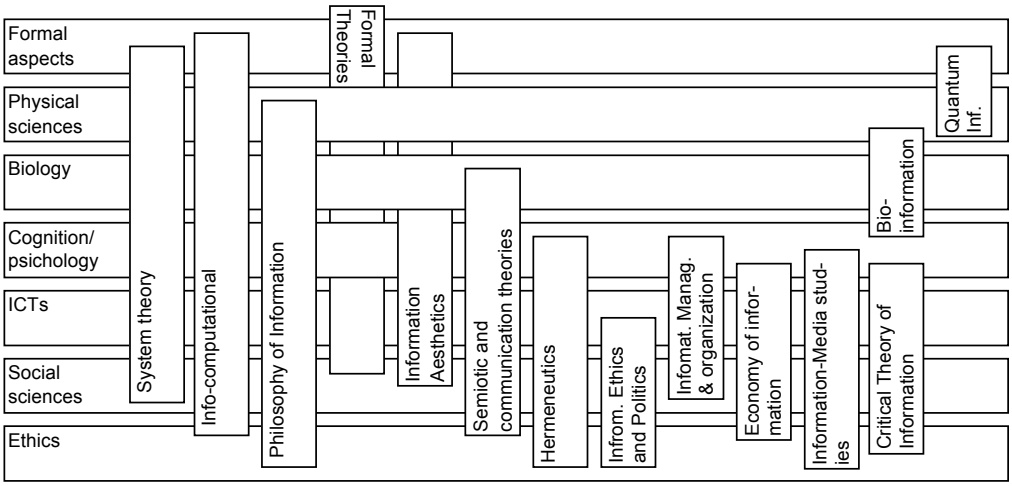
e) Informação e tecnologias de comunicação: tópicos sociais, de engenharia e antropologia;

f) Ciências sociais: tópicos em comunicação, sociedade da informação, economia, sustentabilidade.

Díaz Nafría e Alemany (2011, p. 290) apresentam alguns arcabouços básicos que fornecem uma ligação preliminar transversal entre diferentes domínios de investigação e que podem ser combinados numa pesquisa transdisciplinar. Os arcabouços são ilustrados na Figura 2:

> *– Formal framework: Studies logical and mathematical tools and concepts from main current formal theories, such as channel theory, general information theory and universal logic; pursuing the elaboration of a comprehensive theory which embraces all formal theories of information.*
> *– System Theory framework, around self-organisation from physical structures to complex societies.*
> *– Info-Computational framework, considering information as structure and computation as its dynamics in different systems.*
> *– Philosophical framework, clarifying the very root concepts used in different information domains, contrasting and providing arguments for an elucidation of the informational contents.*
> *– Communicational framework, seeking awareness on the complexity of the processes of significance, communication and interpretation.*
> *– Societal framework, centered in ethical and critical aspects of information and the information society.*
> *– Information Management framework, from the data realm to a sustainable information society (data, information, knowledge management).*
> *– Bio-Informational framework, from molecular complexity to life and cognition. Evolution, genetic information and neural processes. It aims at bridging between the organization of matter, the evolution of life and cognition.*
> *– Quantum-Information framework, from the formality of statistical physics, measurement proc- esses, de-coherence, qbits, and the relation to consciousness. It aims at establishing a strong basis for bridging over physical, life and cognition domains.*

Figura 2 – Arcabouços e áreas de pesquisas transdiciplinares



Fonte: Díaz Nafría e Alemany (2011, p. 291)

Por fim, os autores apresentam o trabalho do grupo BITrum (<http://en.bitrum.unileon.es/>), presente também em Díaz Nafría (2010), que, segundo o website do grupo, trata-se de um grupo de pesquisa constituído para desenvolvimento de uma clarificação teórica da informação:

> *research group BITrum was constituted to develop a conceptual and theoretical clarification of information, intending to gather all the relevant*

> *points of view and pursuing to preserve all the interests at stake (scientific, technical and social). Born at the First international Meeting of Experts in Information Theories-An interdisciplinary approach (León, November 2008, in colaboration with INTECO, University of León and Sierra-Pambley Foundation), BITrum group has deployed a set of activities, publications and initiatives which are here accounted for. BITrum allegorically refers to the conjuction of the information unit "BIT" and the Latin term vitrum (standing for the assembly of a multiplicity of colours).*

Dentre os trabalhos do grupo destaca-se o *Glossarium BITri*[5], um glossário de conceitos, metáforas, teorias e problemas em torno da informação. Trata-se de um glossário online bilingue produzido em colaboração por diferentes cientistas de diversos países. Nesse dicionário, a entrada "arquitetura da informação" apresenta o seguinte conteúdo resumido:

> *(. . .) En un sentido técnico, se trata de una disciplina (y, a la vez, una comunidad de práctica) centrada en los principios del diseño y la arquitectura de espacios digitales, de forma que cumplan criterios de usabilidad y recuperación. O dicho en otros términos, se trata de una disciplina que se encarga de estructurar, organizar y etiquetar los elementos que conforman los entornos informacionales para facilitar la búsqueda y recuperación de la información que contienen y mejorar, así, la utilidad y el aprovechamiento de la misma por parte de sus usuarios. (. . .)*
> *Una de las principales características de la arquitectura de la información de un entorno informacional (una página web, por ejemplo) es que no suele ser observable por parte de sus usuarios. En cierta manera, esa arquitectura es invisible para el usuario. (. . .)*
> *Los sistemas de organización son clasificaciones que permiten estructurar y organizar los contenidos de un sitio web. Los sistemas de etiquetado, en cambio, definen los términos utilizados para nombrar las categorías, opciones y enlaces utilizados en la web en un lenguaje útil para los usuarios. Los sistemas de navegación permiten navegar o movernos por una web para poder localizar la información que necesitamos; nos permitirán entender dónde estamos y dónde podemos ir dentro de la estructura de un sitio. Los sistemas de búsqueda habilitan la recuperación de la información dentro de la web utilizando recursos como el índice. Por último, en este contexto, los vocabularios controlados (o lenguajes documentales) son recursos documentales diseñados para facilitar la búsqueda y recuperación de información.*

Observa-se na definição extraída do glossário BITri, um evidente retrocesso à visão de Arquitetura da Informação como mera organizadora de websites. Por outro lado, diferente da definição apresentada no glossário BITri, outros autores, como Ding e Lin (2010) apresentam visão mais ampla da disciplina, que considera a AI como uma área que constrói e lida com sistemas de formação de modo multi-disciplinar:

> *Information Architecture is about organizing and simplifying information, designing and integrating information spaces/systems, and creating ways for people to find and interact with information content. Its goal is to help people understand and manage information and make right decisions*

[5] <https://sites.google.com/site/glosariobitrum/>

> *accordingly. In the ever-changing social, organizational and technological contexts, Information Architects not only design individual information spaces (e.g., individual websites, software applications, and mobile devices), but also tackle strategic aggregation and integration of multiple information spaces across websites, channels, modalities, and platforms. Not only they create predetermined navigation pathways, but also provide tools and rules for people to organize information on their own and get connected with others.*
>
> *Information Architects work with multi-disciplinary teams to determine the user experience strategy based on user needs and business goals, and make sure the strategy gets carried out by following the user-centered design (UCD) process via close collaboration with others.*

Embora não haja consenso sobre exatamente quais são as disciplinas que formam a Arquitetura da Informação, é prosaico entre os pesquisadores que AI está sempre no limiar entre pessoas e informação. Trata-se de uma disciplina que tem o objetivo de tornar a informação acessível e, para isso, a arquitetura, a *arquitetura da informação*, deve ser *invisível*. Nesse sentido de *não ser percebida*, a AI enquanto disciplina remete o papel de outra área do Design, tão técnica quanto artística: a Tipografia. Segundo Bringhurst (2005, p. 23), a Tipografia, enquanto *arquitetura de tipos gráficos*, tem um papel quase contraditório, de chamar e de não chamar a atenção:

> Em um mundo repleto de mensagens que ninguém pediu para receber, a tipografia precisa frequentemente chamar a atenção para si própria antes de ser lida. Para que ela seja lida, precisa contudo abdicar da mesma atenção que despertou. A tipografia que tem algo a dizer aspira, portanto, a ser uma espécie de estátua transparente.

Dessa forma, com sua vertente artística, como bem coloca o arquiteto urbanista Lúcio Costa (1902-1998), a Arquitetura é "construção concebida com o propósito primordial de organizar e ordenar o espaço para determinada finalidade e visando a determinada intenção" (COSTA, 2003, p. 19-20). A Arquitetura da Informação é construção com informação, é a arte de ocultar a arquitetura para destacar o objeto informação. Porém, como a informação é objeto demasiado heterogêneo, disciplinas e instrumentos heterogêneos devem ser utilizados. Por isso, a *Arquitetura*, que pode variar desde a arquitetura da tipos gráficos, ao design, às construções civis, é eleita como aquela que também deve reger a informação.

2.2 Contribuições harmonizadoras

Nessa panaceia de disciplinas – que é própria e característica da Ciência da Informação – reverbera-se diversas questões de cunho epistemológico. Por exemplo, se a informação é o objeto de estudo da Arquitetura da Informação, qual é a diferença entre AI e Ciência da Informação? Quais são os limites do conceito de informação adotado na

AI? O que se entende por "Arquitetura"? Algumas dessas questões são abordadas por trabalhos anteriores dos autores (ARAUJO, 2012; ARAUJO; LIMA-MARQUES, 2014; LIMA-MARQUES, 2011), mas não é objetivo discorrer sobre elas nesta tese. Porém, no seio dessas perguntas, encontra-se uma proposta de abordagem da área por meio do produto e da intersecção de três grandes fundamentos epistemológicos: a *Psicologia*, a *Ontologia* e a *Lógica*. Essencialmente, toma-se como pressuposto que a informação possui natureza ontológica: é coisa tangível, inerente à energia do cosmos. Com isso em tela, corrobora-se a posição de colegas que propõem uma abordagem filosófica para a Arquitetura da Informação, esta baseada em uma Epistemologia Fenomenológica[6] construída a partir de Husserl (2001) (LACERDA, 2005; LIMA-MARQUES, 2011; SIQUEIRA, 2012a). Dessa forma, concebem *objeto* como coisa para um *sujeito* que o percebe e captura suas propriedades. Essas propriedades se tornam conhecimento do objeto no sujeito. Assim, ambos, sujeito e objeto, só existem na correlação de um com o outro; só são o que são enquanto o são um para o outro (HESSEN, 1998, p. 26), de modo que são subjugados entre si. O objeto, então, é "objeto da consciência do sujeito" e não necessariamente precisa ser físico, caso em que é uma "intencionalidade", como a percepção, memória, fantasia (LIMA-MARQUES, 2011). Enquanto sujeito humano, é a Psicologia a disciplina que supre as teorias e práticas para se abordar o estudo desse ser complexo. Enquanto objeto do mundo, a Ontologia, enquanto estudo dos discursos sobre os entes, é incluída no rol de disciplinas bases porque vem dela os conhecimentos milenares que caracterizam o estudo dos "entes", ou seja, dos objetos para os sujeitos. Por fim, como sistematizadora de todas essas abordagens, a Lógica tem o papel de estudar relações entre os objetos e entre os objetos e os sujeitos. Quando essas relações são de consequência, a título de exemplo, diz-se que o papel da Lógica é o estudo das consequências (lógicas) dessas relações entre entidades.

Dessa abordagem transdisciplinar, deve resultar uma disciplina unificada, no caso, a própria Arquitetura da Informação, e não apenas um conjunto de retalhos de diferentes áreas. Caso contrário, trataria-se apenas de uma multidisciplinaridade, o que não é o caso proposto para a AI. Em resumo, essa disciplina, construída intuisticamente, tem o desafio de estudar as estruturas e a configuração da informação em diversos aspectos, especialmente aqueles que envolvem a relação (ou a experiência) entre os sujeitos e os objetos com a informação.

Além do próprio estudo e enriquecimento da Ciência sobre os objetos e sujeitos envolvidos, um propósito da práxis da Arquitetura da Informação é a construção de "arquiteturas da informação aplicadas" (COSTA, 2010). As arquiteturas da informação aplicadas (aia, em letras minúsculas) tratam-se da configuração da informação e dos

[6] Conforme Cerbone (2012, p. 13; 20), a "palavra 'fenomenologia' significa 'o estudo dos fenômenos', onde a noção de um fenômeno e a noção de experiência, de um modo geral, coincidem. Portanto, prestar atenção à experiência em vez de àquilo que é experimentado é prestar atenção aos fenômenos [...] A Fenomenologia está precisamente ocupada com os modos pelos quais as coisas aparecem ou se manifestam para nós, com a forma e estrutura da manifestação".

sujeitos de modo a atingir um pré-determinado objetivo. Com base no sentido de design ontológico de Willis (1999), e de configuração da informação de Araujo e Lima-Marques (2014), quando a informação está numa arquitetura, ela é configurada para atingir um determinado efeito em um ou mais sujeitos. Dessa forma, a arquitetura produzida interfere na própria existência do sujeito, de modo que este se torna, e age, com novos limites ou fronteiras não existentes antes dessa experiência. Para a construção de aia, uma das etapas necessárias é a abstração de modelos, no sentido de imagens do mundo. Considerando o *Método para Arquitetura da Informação* (Maia) proposto por Costa (2010, p. 95), é no "*momento* escutar"[7] (ECHEVERRÍA, 2005) que o arquiteto apreende o mundo que ele pretende introduzir mudanças, e construir aia. Uma das técnicas úteis nesse momento é a construção de "ontologias", no sentido de discursos concretos especificados em uma determinada linguagem, para uma determinada comunidade de outros arquitetos[8].

De modo sintético, o processo apresentado pelo Maia, aprimorado por Araujo (2012, p. 240-243), indica que no desenvolvimento de arquiteturas da informação aplicadas são produzidas especificações que representam discursos, possivelmente de diferentes arquitetos da informação, sobre a Ontologia do domínio em foco. Segundo o Método, essas especificações são produtos dos momentos *escutar* e *pensar* e insumos para os momentos *construir* e *habitar*. Dessa forma, defende-se que, com o objetivo de capturar a experiência de diferentes arquitetos com a aia, a construção de ontologias de domínio se dá a partir de uma ontologia bem-fundamentada em termos lógicos e filosóficos – tomada *a priori* como ontologia subjacente – e por meio de uma linguagem formal com expressividade tal que seja possível capturar e fazer inferências sobre o discurso de diferentes arquitetos no transcorrer do processo de construção de aia.

Com esse objetivo em tela, e considerando a estratégia de abordagem dos problemas da informação de forma transdisciplinar, um arcabouço ideal de conhecimento para um Arquiteto da Informação na tarefa de construir arquiteturas da informação aplicadas poderia ser sistematizado do seguinte modo:

a) um conjunto de pressupostos epistemológicos como base para tratar o comportamento dos sujeitos envolvidos na construção da arquitetura da informação;

b) um conjunto de pressupostos ontológicos que idealmente proveriam fundamentação teórica e visão de mundo compartilhada entre diferentes sujeitos (também entendidos como agentes) na arquitetura;

c) um conjunto de ferramentas lógicas que possam prover uma ou mais linguagens formais para que se possa expressar as estruturas de aia que mantenham

[7] O Maia é um método que propõe que um sujeito constrói arquiteturas da informação por meio de um processo iterativo, dividido em quatro momentos – Escutar, Pensar, Construir e Habitar — apresentados como espécies de fases cíclicas constituídas de atos.

[8] Geralmente aquele que construir "ontologias" é chamado de "ontologista", mas defende-se que o nome "arquiteto da informação" é mais adequado no contexto apresentado.

> compromisso ontológico com o conhecimento dos sujeitos e com a estrutura do mundo, além de arcabouço dedutivo sobre essas estruturas, de modo a ser possível inferir novos conhecimentos a partir dos existentes.

Um arcabouço epistemológico adequado à construção de AI está disponível, por exemplo, como é apresentado por Siqueira (2012a), Siqueira (2012b), mas que, propositalmente, não é detalhado neste texto por estar além dos objetivos desta pesquisa. Embora indique-se as obras mencionadas, não se defende que elas tenham encerrado as discussões quanto à epistemologia da Arquitetura da Informação. Porém, apresentam contribuições relevantes à área, que são utilizadas como ponto de partida para o alcance do Item a).

No que tange aos pressupostos ontológicos – Item b) – defende-se que UFO (Capítulo 4) provê uma ontologia base adequada para os objetivos de construção de AI, especialmente referentes à atividade de modelagem conceitual de domínios específicos. Isso é devido ao fato de UFO se tratar de uma ontologia genérica, filosoficamente bem-fundamentada, que pode ser usada como metateoria (ou teoria de base) para teorias específicas de domínio. No que tange à Lógica – Item c) – a UFO lança mão de ideias modais aléticas para incorporar conceitos já conhecidos como rigidez e antirigidez, mas não contém, em si, estruturas lógicas epistemológicas, que são potencialmente necessárias para construção colaborativa de especificações, ou de "ontologias" compartilhadas por diferentes arquitetos.

Nesse sentido, parece ser justificável a integração teórica de arcabouços disciplinares da Ontologia e da Lógica para produzir novas teorias e técnicas que sejam adequadas para os objetivos da Arquitetura da Informação.

2.2.1 O projeto "Configuração da informação na Arquitetura da Informação"

As investigações e os resultados obtidos nesta pesquisa vão ao encontro de trabalho futuro sugerido em pesquisa anterior dos autores (ARAUJO, 2012, p. 255, item 9), o que torna este trabalho um avanço no projeto *configuração da informação na Arquitetura da Informação*.

No processo de construção de arquiteturas da informação, advoga-se que a configuração da informação é uma composição de objetos rígidos inseridos em um espaço juntamente com sujeitos que sofrem e exercem ações de *transformação* dessas configurações. O processo de construção de configurações pode ser dividido em funções, fases ou etapas. Uma das mais importantes delas refere-se ao processo de Identificação da Configuração[9] (ARAUJO, 2012). A função de Identificação da Configuração é aquela em que se identificam os itens, ou os objetos, que compõe o espaço de informação que será configurado para atingir os objetivos determinados. A identificação da configuração, ou dos itens de configuração, é

[9] As outras são: "Planejamento e Gestão", "Controle das mudanças da configuração", "Relato da situação da configuração" e "Auditoria da Configuração".

crucial para o sucesso da empreitada de proposição de uma arquitetura, porque é nessa fase que se estabelece o vocabulário do domínio que os arquitetos trabalharão, e que se define a linguagem e os insumos para a arquitetura a ser construída. Nessa atividade de identificação, é necessário que se façam abstrações sucessivas e que se construam artefatos que sirvam de objetos de comunicação entre os arquitetos, e a comunidade alvo do domínio de atuação.

No âmbito da área de *Modelagem conceitual*, esses artefatos poderiam ser chamados de "modelos", ou "modelos conceituais", geralmente representados por uma linguagem artificialmente construída com o propósito específico de servir de língua franca entre as pessoas envolvidas na modelagem. Dessa forma, no contexto de construção de arquiteturas da informação por meio da configuração da informação, na etapa de identificação da configuração, sugere-se que a produção de ontologias seja eficaz no que tange o descobrimento da realidade e a construção de acordos entre múltiplos arquitetos, no sentido de artefatos que representem o estudo ou os discursos sobre os entes e seres de determinado domínio de atuação.

Essa visão pragmática e reducionista de "ontologia" remonta à ideia de que esta, enquanto modelo conceitual, não produz nem é a configuração em si, mas provê suporte para o entendimento de alguns aspectos da configuração. Dessa forma, a Arquitetura da Informação não se limita ao estudo da ontologia, mas sem esta aquela seria muita mais complexa. Por isso, a ontologia, por meio da modelagem conceitual, é vista como um meio de entender o *estar* da informação e por isso é relevante à disciplina AI.

2.2.1.1 O operador *zoom*

Um dos resultados do trabalho que motivou esta pesquisa refere-se à discussão a respeito de operadores que podem agir sobre configurações de informação. As configurações podem ser concebidas como objetos relacionamentos por algum tipo de relação que denote composição. Determinados níveis de composições entre os objetos em uma mesma configuração são reconhecidos como objetos distintos. Por exemplo, um "determinado automóvel é objeto em uma configuração de carros em um pátio. Esse automóvel também é uma configuração de chassi, bancos, volante, escapamento, motor, etc. O motor, por sua vez, é uma configuração de pistão, cilindro, biela... e assim sucessivamente" (ARAUJO, 2012, p. 204).

Por isso, "navegar" pelas composições das configurações e explicitar as distinções nessas camadas é um tipo de operação concebida naquela obra como "operador *zoom*"[10]:

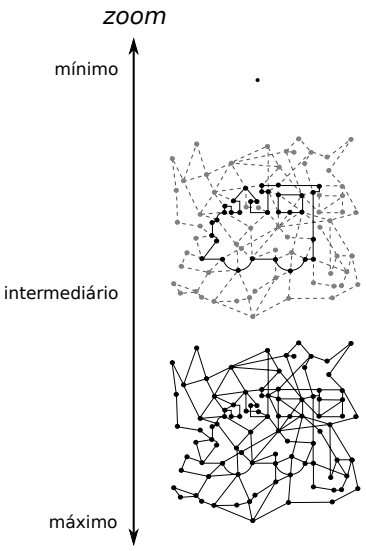
> Definição 9.3 (Operador *zoom*) Zoom é o operador de identificação de distinções entre configurações. A ação *zoom out* (z-) sobe e a ação zoom in (z+) desce na hierarquia de composições de configurações.

[10] ARAUJO, 2012, loc. cit.

> O *zoom in* (z+) desce na hierarquia de composições de configurações, de tal modo que identifica mais configurações diferentes. Ele aumenta a resolução das configurações. É o mesmo que reduzir o número de composições. De modo oposto, o *zoom out* (z-) reduz a resolução de distinções e identifica menos configurações ao subir pela hierarquia de composições. É o mesmo que aumentar o número de composições.

A Figura 3 ilustra a ideia do operador. Em um emaranhado de objetos compostos, deve haver um meio de distinguir diferentes níveis ou de se identificar objetos diferentes. Em um máximo de distinções, não é possível perceber os níveis mais abstratos de objetos, mas apenas os componentes indivisíveis. Da mesma forma, em um mínimo de distinções, a configuração é entendia como uma única coisa sem partes. Em ambos os casos, a configuração não apresenta "utilidade" ao sujeito. O *zoom* é justamente o operador que adentra na composição e permite identificar distinções e, por isso, tornar a configuração adequada a determinado contexto de informação, a partir de determinado critério.

Figura 3 – Operador *zoom*



Fonte: Araujo (2012, p. 205)

Este tópico é utilizado na subseção 8.4.3 como analogia a determinadas características de resultados obtidos nesta pesquisa.

2.3 Sobre o discurso do sujeito

Um discurso sobre a realidade pode ser tratado como uma representação sobre a configuração dos objetos do mundo, esta entendida como uma *ontologia de domínio*. Neste contexto, o discurso não é isento de ideologias ou é ermo de contextos, ele pode ser interpretado como uma forma de atuar, de agir sobre o outro: um instrumento de ação sobre o mundo.

Neste pesquisa aplica-se um conceito específico de *discurso*. A ideia de *discurso* está relacionada ao ato de fala (AUSTIN, 1962) de um sujeito, ou de uma sociedade de sujeitos, a respeito de determinada realidade. Especialmente, interessa-se pela descrição, ou formalização do discurso sobre entidades ontológicas presentes em determinado universo ou domínio. A noção de discurso é obtida com fundamento teórico na Linguística, especialmente influenciada pela escola francesa da Análise do Discurso, cujos principais expoentes são Michel Foucault (1926-1984) e Michel Pêcheux (1938-1983). A escola francesa é muito presente nas pesquisas brasileiras da área, cuja principal representante é a Dra. Eni Puccinelli Orlandi. Conforme a pesquisadora, etimologicamente, discurso possui a ideia de curso, de percurso, de correr por, de movimento. Portanto, discurso é palavra em movimento, é prática da linguagem:

> [...] a linguagem como mediação necessária entre o homem e a realidade natural e social [...] é o discurso [que] torna possível tanto a permanência e a continuidade quanto o deslocamento e a transformação do homem e da realidade em que ele vive. O trabalho simbólico do discurso está na base da produção da existência humana (ORLANDI, 2009, p. 15-16).

As alíneas a seguir apresentam características fundamentais do que é chamado *discurso* para Brandão (2009). Essas alíneas foram baseadas e extraídas quase que literalmente da obra citada:

a) o discurso deve ser compreendido como algo que ultrapassa o nível puramente gramatical, linguístico. O nível discursivo apoia-se sobre a gramática da língua (o fonema, a palavra, a frase), mas nele é importante considerar também (e sobretudo) os interlocutores (com suas crenças, valores) e a situação (lugar e tempo geográfico, histórico) em que o discurso é produzido;

b) no nível do discurso, os falantes/ouvintes, escritor/leitor devem ter conhecimentos não só do ponto de vista linguístico (dominar a língua, as regras de organização de uma narrativa, de uma argumentação etc.), mas também conhecimentos extralinguísticos: conhecimento para produzir discursos adequados às diferentes situações em que atua na vida; conhecimentos de assuntos, temas que circulam na sociedade; conhecimento das finalidades da troca verbal, e para isso são importantes a imagem que o sujeito faz de si, da própria posição, a imagem que tem das pessoas com quem fala, imagens que vão determinar a maneira como falar com essas pessoas;

c) o discurso é contextualizado, isto é, do ponto de vista discursivo, toda frase (ou melhor, enunciado) só tem sentido no contexto em que é produzido. Assim, um mesmo enunciado, produzido em momentos diferentes (quer seja pelo mesmo sujeito ou por sujeitos diferentes) terá sentidos diferentes e, portanto, pode corresponder a discursos diferentes;

d) o discurso é produzido por um sujeito – um EU que se coloca como o responsável pelo que se diz (de forma explícita, como num diário de adolescente[11] ou implícita, como no discurso da ciência) e é em torno desse sujeito que se organizam as referências de tempo e de espaço. Por exemplo, no enunciado: "Hoje, meu depoimento será sobre a infância vivida na casa de minha avó", os termos "hoje", "meu", "minha" devem ser entendidos em relação ao sujeito que fala e que se coloca como EU do discurso. E esse sujeito que fala assume uma atitude, um determinado comportamento (de firmeza, dúvida, opinião) em relação àquilo que diz (usa para isso recursos da língua como: infelizmente, talvez, certamente, na verdade, eu acho) e em relação àquele com quem fala (explicitamente por expressões do tipo você, caro leitor, ou escolhendo os termos adequados ao seu nível sócio-cultural, usando uma linguagem mais informal, gírias ou linguagem mais formal de acordo com a situação);

e) o discurso é interativo, pois é uma atividade que se desenvolve, no mínimo, entre dois parceiros (marcados linguisticamente pelo binômio Eu-Você). A conversação é o exemplo mais evidente dessa interatividade: os parceiros monitoram a sua fala de acordo com a reação do outro. Mas, no discurso escrito, o locutor está também preocupado com seu leitor, a ele dirigindo-se explicitamente (como em "meu caro leitor") ou procurando uma linguagem adequada a ele (um livro de literatura infantil, um guia médico para pais leigos em assuntos médicos têm toda uma linguagem voltada para o público que se quer atingir) ou utilizando-se de estratégias de discurso para se defender, antecipar a contra-argumentação do leitor;

f) o discurso é uma forma de atuar, de agir sobre o outro. Quando se realiza promessas, ordens, perguntas, etc., pratica-se uma ação pela linguagem (um ato de fala) que tem por objetivo modificar uma situação. Por exemplo, o "eu te batizo X" pronunciado pelo padre numa cerimônia de batismo muda a situação da pessoa no quadro da religião católica; numa passeata, um cartaz com o enunciado "Não à corrupção" visa modificar comportamentos de pessoas envolvidas nesse ato e mostra a atitude de indignação daqueles que levam esse cartaz;

g) o discurso trabalha com enunciados concretos, falas/escritas realmente produzidas (e não idealizadas, abstratas, como as frases da gramática) e os estudos que se fazem deles visam descrever suas normas, isto é, como funciona a língua no seu uso efetivo. Por exemplo, se alguém faz uma pergunta, pressupõe-se que ele não ignore a resposta e tenha interesse nessa resposta; e, ainda, que aquele a quem é feita a pergunta tem condições de responder. Se essas regras não são

[11] Ou, de modo mais contemporâneo que exemplifique prática verdadeira: como com amigos reais, conectados por redes sociais privadas, baseadas em comunicação por dispositivos móveis...

obedecidas, por exemplo, se ele sabe a resposta, mas pergunta assim mesmo, é porque o locutor tem intenções implícitas. O interlocutor se pergunta então "por que razão, sabendo a resposta, ele me fez a pergunta assim mesmo?", e por uma série de raciocínios (inferências) vai procurar o sentido que está por trás;

h) um princípio geral rege o discurso: o princípio do dialogismo. A palavra dialogismo vem de diálogo – "conversa", "interação verbal" que supõe pelo menos dois falantes. Quando se fala dirige-se sempre a um interlocutor; mesmo num monólogo (quando se fala consigo mesmo), num diário, criam-se um personagem (um outro eu) com quem imaginariamente se dialoga.

A formalização de ontologias, embora deva ser atividade realizada de forma precisa e livre de ambiguidades, não pode desconsiderar o fato de ser essencialmente um discurso de sujeitos humanos, necessariamente imersos na história, carregados por ideologias[12] e preconceitos. Conforme Brandão (2004, p. 11), a linguagem enquanto discurso:

> não constitui um universo de signos que serve apenas como instrumento de comunicação ou suporte de pensamento; a linguagem enquanto discurso é interação, e um modo de produção social; ela não é neutra, inocente e nem natural, por isso o lugar privilegiado de manifestação de *ideologia*. (grifo nossos)

Portanto, quando um ou mais sujeitos criam uma ontologia, eles essencialmente formalizam seus próprios *discursos* a respeito das entidades ontológicas que estão experimentando. Por proferir um discurso, no sentido apresentado pelos parágrafos anteriores, sua fala é influenciada por tudo aquilo que influencia seus autores. Por isso, é importante salientar que devem existir mecanismos que visem suprimir, ou ao menos reduzir, as características subjetivas e ambíguas dos discursos. É para tentar atingir esse objetivo que ontologias de fundamentação, lógicas e linguagens formais são combinadas nesta pesquisa. No entanto, reconhece-se que a meta de tornar absolutamente objetiva determinada formalização de um discurso não pode ser alcançada, uma vez que há no mínimo uma ideologia subjacente que não pode ser explicitada, restando sempre algum grau de imprecisão em relação às entidades da realidade, ao sujeito que profere o discurso, e ao resultado da descrição da ontologia. Por isso, considerando objetivos pragmáticos de construção de arquiteturas da informação aplicadas, e obtenção de configurações específicas como meta de um determinar design, a imprecisão deve ser reduzida ao ponto que não se torne relevante, ao menos no que tange a propósitos específicos.

[12] Em Japiassu e Marcondes (2008) encontra-se uma definição de ideologia: "conjunto de idéias, princípios e valores que refletem uma determinada visão de mundo, orientando uma forma de ação, sobretudo uma prática política." Conforme o filósofo em tela, são exemplos de ideologia a ideologia fascista, ideologia de esquerda, a ideologia dos românticos, etc.

2.4 Engenharia Lógica

Em sua tese de doutoramento, Lima-Marques (1992, p. 7-10) propõe a " Engenharia Lógica" como uma disciplina que visa a aplicação da lógica para solução de problemas de informação. Nas palavras do autor:

> *Le but de cette nouvelle discipline est principalement d'aider la communication entre les logiciens et les experts de développement de systèmes à base de connaissances intéressés par l'application des logiques pour la solution de problèmes*[13] (LIMA-MARQUES, 1992, p. 10).

Segundo o autor, o engenheiro lógico é aquele que exerce um papel misto de logicista e de engenheiro cognitivo. O engenheiro cognitivo atua com o especialista no domínio na análise e no tratamento do problema a ser resolvido, ou ao menos descrito, em um formalismo adequado ao objetivo a ser atingido. Com foco em soluções de problemas, o engenheiro lógico deve conhecer diferentes tipos de lógicas e de ferramentas meta-lógicas, com visão teórica e aplicada. O engenheiro lógico deve trabalhar para atingir seus objetivos, muitas vezes, fora do "projeto lógico", uma vez que seu compromisso é com o entendimento do problema e com a proposição de soluções àquela demanda. Dessa forma, a atuação do engenheiro lógico é quase sempre qualificada como *lógica aplicada*. Isso difere do "projeto lógico" no sentido que este geralmente estuda teoricamente a Lógica pela e na própria Lógica, não necessariamente com objetivo de resolver algum problema concreto.

Este trabalho é desenvolvido nessa linha disciplinar de Engenharia Lógica, em que o papel de engenheiro lógico é exercido para desenvolvimento de uma formalização em Programação em Lógica de métodos para tratamento de formalizações de discursos sobre ontologias (seção 1.2).

A abordagem de uso de ontologias para solução de problemas cria uma camada intermediária adicional entre o problema e a solução, uma vez que insere uma ontologia como mediatária entre a realidade do problema e a Lógica da solução. Por isso, a própria descrição da ontologia exige uma lógica subjacente, que possivelmente é diferente da lógica a ser aplicada na solução do problema final. No caso, esta pesquisa foca nas investigações dessa lógica intermediária, referente à formalização de ontologias sobre a realidade. Isso, de certa forma, pode ser entendido como um aprimoramento da ideia original de Lima-Marques (1992), uma vez que este abordou diretamente o problema concreto com base em uma combinação de famílias de lógicas, sem usar uma camada intermediária de formalização do problema com base em categorias ontológicas.

[13] "O objetivo desta nova disciplina é principalmente ajudar a comunicação entre lógicos e os especialistas em desenvolvimento de sistemas baseados em conhecimento interessados na aplicação da lógica para solução de problemas." (tradução nossa)

2.5 Fechamento

A defesa de que a Informação é a integradora da Arquitetura às pessoas, discutida na seção 2.1, reflete um tipo específico de visão sobre a Arquitetura da Informação: a AI é uma disciplina científica que visa configurar informação como se essa fosse um material que pudesse ser moldado, combinado e usado como instrumento para construir espaços a serem habitados por pessoas. Por tratar a AI como área da Ciência da Informação, esse tipo de visão exige uma posição necessariamente abrangente, que não seja restrita epistemologicamente. Portanto, essa abordagem integradora é característica notória da proposição do arcabouço ideal de tratamento de ontologias pela AI descrito na seção 2.2.

Os resultados de trabalhos anteriores dos autores são usados como base nas investigações em curso, conforme apresentado na subseção 2.2.1. O projeto de *configuração da informação* tem a intenção de construir arcabouço no âmbito da Arquitetura da Informação que possibilite a construção no espaço de informação. Por isso, esta pesquisa é importante para o alcance dessa agenda maior, que também é compartilhada por outros colegas do Grupo de Pesquisa que este trabalho se desenvolve.

A ideia de *discurso* como um *ato de fala*, que é capaz de atuar, de agir sobre o outro (seção 2.3), está alinhado com a aceitação da epistemologia fenomenológica da Arquitetura da Informação. Juntas, ambas resultam em uma posição ideológica que considera a formalização do discurso não apenas uma forma de descrição da realidade, mas um meio de atuar, modificar e configurar a informação presente no espaço. Portanto, assume-se que os enunciados concretos proferidos por um sujeito – falados oralmente ou escritos em uma linguagem – têm o poder de configurar o design ontológico do ambiente (WILLIS, 1999) e de determinar esses mesmos sujeitos pelo fenômeno da experiência.

O projeto de produzir um glossário que sistematiza os conceitos da Arquitetura da Informação (*Glossarium BITri*), conforme apresentado a partir da Figura 2.1, é corajoso e louvável, especialmente porque a área, por ainda estar em plena infância, possui abordagens díspares e, por isso mesmo, a produção de uma sistematização terminológica é relevante ao estabelecimento e aprazamento de conceitos. Ao menos no que tange à ideia de construção de um glossário, essa iniciativa serve de motivação para que o Glossário da UFO contido neste trabalho seja produzido. Evidentemente, o glossário desta pesquisa concentra-se num escopo muito mais restrito, mas que os resultados são igualmente relevantes tanto para pesquisadores da área de Ontologia, como de Arquitetura da Informação.

Por fim, a ideia de Engenharia Lógica é adequada à pesquisa em curso, especialmente devido ao seu caráter aplicado, uma vez que nesta pesquisa não se busca atingir resultados a respeito das meta-lógicas das teorias investigadas. Por isso, as soluções lógicas investigadas e desenvolvidas com foco nos problemas enfrentados pela Arquitetura da Informação são realizadas tendo a Engenharia Lógica como norteadora dos trabalhos.

3 Programação em Lógica

O objetivo desde capítulo é expor um fragmento dos achados bibliográficos de pesquisa exploratória realizada sobre o tema Programação em Lógica. Além disso, o capítulo visa complementar argumentos apresentados na subseção 1.3.2 em defesa da abordagem de Programação em Lógica Clássica e Lógicas Modais para tratamento de ontologias.

Conforme destacado na Por que a aplicação de Programação em Lógica?, Prolog é o nome atribuído ao *paradigma* de Programação em Lógica, e não se refere a uma ferramenta ou implementação específica. Por isso, a expressão e seu acrônimo são utilizados indistintamentes em todo o texto. Quando " Prolog" denotar uma implementação de um provador, ou mesmo um programa específico, o novo significado é explicitamente salientado.

Este capítulo propositalmente não contempla a descrição detalhada de vários dos achados bibliográficos conhecidos durante as investigações dos referenciais teóricos, mas indica precisamente suas referências bibliográficas e o conteúdo sintético que colaborou com o alcance dos resultados da pesquisa. O capítulo também não abrange conceitos básicos ou de fundamentação teórica de Prolog, que podem ser obtidos em Araribóia (1988), Saint-Dizier (1990), Nilsson e Maluszynski (2000). Para iniciação ao tema, indica-se também a *Wiki Book de Prolog*, disponível no sítio <https://en.wikibooks.org/wiki/Prolog> (em inglês), ou ainda, as primeiras sete páginas do clássico texto de Warren (1980). Por não ser o foco deste trabalho discutir teoricamente a Programação em Lógica, não estão incluídas formalizações a respeito da semântica de modelos, nem os demais detalhes técnicos do sistema de dedução de programas lógicos, por entender que esses resultados já estão devidamente registrados na literatura da área. De toda forma, o fundamento teórico utilizado no restante do trabalho é o presente em Lloyd (1993). Os detalhes do processo de Resolução SLD e $SLDNF$ (SDL com negação) não são tampoucos apresentados. Para isso, a bibliografia indicada e também Kowalski (1974) e Rodrigues (2010, p. 19-46) podem ser consultados.

O restante deste capítulo está estruturado da seguinte maneira. A seção 3.1 apresenta um breve histórico da Programação em Lógica, mostra uma relação entre linguagem clássica da Lógica de Primeira Ordem (FOL) e a linguagem usada para Programação em Lógica, descreve uma técnica para otimização de regras com negações, utilizadas amplamente nos resultados desta pesquisa, e aborda a *semântica de base de dados*, que consiste na junção das hipóteses de nome único e de mundo fechado. A seção 3.2 aborda a Programação em Lógica Modal e especialmente apresenta os trabalhos do Molog e do

MProlog. A seção 3.3 consiste em um conjunto de tópicos referentes à especificação e implementação de linguagens de programação com Prolog. Essas técnicas são investigadas com intuito de justificar a formalização de Ontoprolog em Prolog. Na seção 3.4 apresenta-se brevemente três trabalhos com abordagem similar à desenvolvida neste trabalho, sendo duas realizadas especificamente com Programação em Lógica (Telos e MoMat), e uma focada na arquitetura da OMG (OntoDSL). A seção 3.5 consiste em uma breve menção ao estilo de programação de códigos-fonte Prolog adotado neste trabalho. Já na seção 3.6 alguns comentários sobre o capítulo são traçados a título de enceramento.

3.1 Programação em Lógica Clássica: Prolog

3.1.1 Breve histórico

Conforme Sebesta (2012, p. 79) e Kowalski (1988), *Prolog* foi originalmente definido como uma linguagem de programação desenvolvida em 1972 por Alain Colmerauer e Philippe Roussel no *Grupo de Inteligência Artificial* da *Universidade de Aix-Marseille*. Em conjunto com Robert Kowalski do *Departamento de Inteligência Artificial* da *Universidade de Edimburgo*, eles desenvolveram o design fundamental da linguagem, baseado no algoritmo de Resolução aperfeiçoado por Robinson (1965). O nome "Prolog" foi definido pela esposa de Roussel, Jacqueline, como abreviação para a expressão em francês *"programmation en logique"*.

Kowalski (1988, p. 38) resume a abordagem de Programação em Lógica da seguinte forma:

> *Logic programming shares with mechanical theorem proving the use of logic to represent knowledge and the use of deduction to solve problems by deriving logical consequences. However, it differs from mechanical theorem proving in two distinct but complementary ways: (1) It exploits the fact that logic can be used to express definitions of computable functions and procedures; and (2) it exploits the use of proof procedures that perform deductions in a goal-directed manner, to run such definitions as programs.*

Lloyd (1993, p. 2) destaca que a invenção de Prolog revolucionou o modo de usar a Lógica. Ela poderia ser usada não apenas para descrever conceitos, mas também para instruir computadores e, com isso, obter resultados práticos que são logicamente coerentes. Essa posição ressalta a relevância da invenção, especialmente como instrumento que permite simultaneamente formalizar uma lógica e obter imediatamente a resposta dos comportamentos que ela implica:

> *The idea that first order logic, or at least substantial subsets of it, could be used as a programming language was revolutionary, because, until 1972, logic had only ever been used as a specification or declarative language in computer science. However, what Kowalski (1974) shows is that logic has a*

> procedural interpretation, *which makes it very effective as a programming language. Briefly, a program clause* $A \leftarrow B_1, \ldots B_n$ *is regarded as a* procedure definition. *If* $\leftarrow C_1, \ldots, C_k$ *is a goal, then each* C_j *is regarded as a* procedure call. *A program is run by giving it an initial goal. If the current goal is* $\leftarrow C_1, \ldots, C_k$*, a step in the computation involves unifying some* C_j *with the head* A *of a program clause* $A \leftarrow B_1, \ldots, B_n$ *and thus reducing the current goal to the goal* $\leftarrow (C_1 \ldots, C_{j-1}, B_1, \ldots, B_n, C_{j+1}, \ldots, C_k)\Theta$*, where* Θ *is the unifying substitution. Unification thus becomes a uniform mechanism for parameter passing, data selection and data construction. The computation terminates when the empty goal is produced.*

Desde a criação de Prolog, diversas abordagens foram proposta com base em seus algoritmos de Resolução, como programação por restrições (FRÜHWIRTH, 1998; WALLACE et al., 2004; SNEYERS et al., 2010), assim como outras técnicas de inferências foram desenvolvidas, como as aplicadas em banco de dados dedutivos (COLOMB, 1998).

3.1.2 Linguagem da Programação em Lógica

Com intuito de construir um vocabulário comum, esta seção inclui, com algumas adaptações, definições apresentadas por Lloyd (1993, p. 4-19) a respeito da Lógica de Primeira Ordem referentes a conceitos básicos da Programação em Lógica utilizados no decorrer deste trabalho. No âmbito da pesquisa do referencial teórico, também são considerados os trabalhos de Kowalski (1974), de Kowalski (1979) e de van Emden e Kowalski (1976). No restante desta seção, assume-se as definições básicas nos referidos textos, especialmente do primeiro, e evita-se realizar novas citações dos mesmos autores.

Para outros detalhes técnicos que extrapolam os apresentados nesta seção, remete-se às obras indicadas, uma vez que não é objetivo exaurir, nem sequer ser demasiadamente rígido em aspectos formais de FOL. Uma apresentação mais moderna de Prolog é realizada por Sebesta (2012, p. 727-757). Para uma introdução à Programação em Lógica mais direcionada à programação de computadores do que à formalização de conceitos, consultar Saint-Dizier (1990).

As notações de primeira ordem definidas nesta são usadas para denotar o significado formal das especificações realizadas na linguagem de Ontoprolog, conforme a seção 6.1. A notação refere-se a uma adaptação das fórmulas-bem-formadas de teorias de Primeira Ordem conforme utilizadas no âmbito da Programação em Lógica.

Os fundamentos sobre Programação em Lógica adotados neste trabalho são aqueles apresentados por Lloyd (1993), e não são transcritos neste trabalho.

Definição 3.1 (Alfabeto)**.** Um *alfabeto* consiste em sete classes de símbolos, conforme seguem escritos já com convenções a serem seguidas no restante deste trabalho:

a) *variáveis*: normalmente denotadas por letras maiúsculas ou palavras formadas por essas letras, como U, V, W, X, Y e Z, possivelmente tipografadas em subscrito;

b) *constantes*: normalmente apresentadas com letras minúsculas ou palavras formadas por essas letras, como a, b e c, possivelmente em subscrito;

c) *símbolos funcionais*: de aridade > 0 são denotadas por letras como f e g;

d) *símbolos predicativos*: de aridade ≥ 0 são normalmente escritos denotados por letras minúsculas como p, q, r e s, possivelmente subscrito;

e) *conectivos*: $\neg$, $\wedge$, $\vee$, $\rightarrow$ e $\leftrightarrow$;

f) *quantificadores*: $\forall$ e $\exists$;

g) *símbolos de pontuação*: "(", ")" e ",", sem as aspas.

Para qualquer alfabeto, considera-se que apenas as classes descritas no Item b) e no Item c) podem ser vazias.

Com intuito de evitar excessos de parênteses nas fórmulas, adota-se as seguinte ordem de precedência, em que a maior precedência é escrita primeiro:

$$\begin{gathered} \neg, \forall, \exists \\ \vee \\ \wedge \\ \leftarrow, \leftrightarrow \end{gathered}$$

—

Definição 3.2 (Variável anônima)**.** A *variável anônima* é definida pelo símbolo de sublinhado (_). Trata-se de uma variável especial existencialmente quantificada no processo de unificação. —

Definição 3.3 (Definição de conectivos)**.** De forma usual, os conectivos $\rightarrow$, $\wedge$ e $\leftrightarrow$ são definidos classicamente, com base em $\neg$ e $\vee$:

$$\begin{aligned} \varphi \rightarrow \psi &:= \neg\varphi \vee \psi \\ \varphi \wedge \psi &:= \neg(\neg\varphi \vee \neg\psi) \\ \varphi \leftrightarrow \psi &:= \varphi \rightarrow \psi \wedge \psi \rightarrow \varphi \end{aligned}$$

para todo φ e todo ψ. —

Definição 3.4 (Quantificador existencial)**.** Da mesma forma, o quantificador existencial $\exists$ é definido classicamente com base no quantificador universal $\forall$:

$$\exists\varphi := \neg\forall\neg\varphi$$

para todo φ.

Consequentemente, a negação do quantificador existencial é:

$$\neg\exists\varphi := \forall\varphi$$

—

Definição 3.5 (Termo)**.** Termo é definido indutivamente da seguinte maneira:

a) Uma variável é um termo;

b) Uma constante é um termo;

c) Se f é uma função n-ária, e $t_1, \ldots t_n$ são termos, então $f(t_1, \ldots t_n)$ é um termo.

—

Definição 3.6 (Átomo, fórmula ou fórmula bem-formada)**.** Fórmula é definida indutivamente da seguinte maneira:

a) Se p é um símbolo predicativo n-ário, e $t_1, \ldots t_n$ são termos, então $p(t_1, \ldots t_n)$ é uma *fórmula* (neste caso, chamada de *fórmula atômica* ou de *átomo*)[1];

b) Se φ e ψ são fórmulas, então $\neg(\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ e $(\varphi \leftrightarrow \psi)$ também são fórmulas;

c) Se φ é uma fórmula e X uma variável, então $(\forall X \varphi)$ e $(\exists X \varphi)$ são fórmulas.

Pode ser conveniente para o restante deste documento escrever $(\varphi \rightarrow \psi)$ como $\psi \leftarrow \varphi)$.

Fórmulas, fórmulas bem-formadas ou átomos são convenientemente escritos como letras gregas minúsculas neste documento.

Um átomo pode ser identificado por seu símbolo predicativo seguido de / e um número natural que denota sua aridade.

Exemplo 3.1. O átomo $dio(a, b)$ pode ser identificado como $dio/2$.

—

Dadas essas definições, apresenta-se a definição de Linguagem de Primeira Ordem.

Definição 3.7 (Linguagem de Primeira Ordem)**.** Consiste em um conjunto de todas as fórmulas construídas a partir de símbolos de um Alfabeto.

Exemplo 3.2. Considere as fórmulas: $(\forall X(\exists Y(p(X, Y) \rightarrow q(X))))$, $(\neg(\exists X(p(X, a) \wedge q(f(X))))$ e $(\forall X(p(X, g(X)) \leftarrow (q(X) \wedge (\neg r(X)))))$. Essas fórmulas podem ser escritas sem os parênteses usando a convenção estipulada anteriormente, de modo que podem ser escritas, respectivamente, como: $\forall X \exists Y(p(X, Y) \rightarrow q(X))$, $\neg\exists X(p(X, a) \wedge q(f(X)))$ e, por fim, como $\forall X(p(X, g(X)) \leftarrow q(X) \wedge \neg r(X))$. As fórmulas são simplificadas neste texto sempre que possível.

Informalmente, a semântica dos quantificadores e dos conectivos é definida conforme a Lista de símbolos, de modo que a semântica informal de $\forall X(p(X, g(X)) \leftarrow q(X) \wedge \neg r(X)))$ é "para todo X, se $q(X)$ é verdadeiro e $r(X)$ é falso, então $p(X, g(X))$ é verdadeiro". —

[1] Observar que átomos são sempre positivos.

Definição 3.8 (Escopo)**.** O *escopo* de $\forall X$ ($\exists X$) em $\forall X\varphi$ ($\exists X\varphi$) é φ. Uma *ocorrência ligada* de uma variável em uma fórmula é uma ocorrência que segue imediatamente um quantificador ou uma ocorrência no escopo de um quantificador, que possui a mesma variável imediatamente após o quantificador. Qualquer outra ocorrência de uma variável é dita *livre*.

Exemplo 3.3. Na fórmula $\exists X p(X,Y) \wedge q(X)$, as duas primeiras ocorrências de X são ligadas, enquanto a terceira é livre, já que na fórmula o escopo de $\exists X$ é $p(X,Y)$. Já no caso de $\exists X(p(X,Y) \wedge q(X))$, todas as ocorrências de X são ligadas.

—

Definição 3.9 (Fórmula fechada)**.** Uma fórmula é fechada se não ocorrem variáveis livres. —

Definição 3.10 (Fecho universal e Fecho existencial)**.** Se φ é uma fórmula, então $\forall(\varphi)$ denota o *fecho universal* de φ, que é obtido por meio de uma fórmula fechada pelo quantificador universal para cada variável livre ocorrendo em φ. De forma similar, o *fecho existencial*, denotado por $\exists(\varphi)$, é obtido por meio de uma fórmula fechada pelo quantificador existencial para cada variável livre ocorrendo em φ.

Exemplo 3.4. Se φ é $p(X,Y) \wedge q(X)$, então $\forall(\varphi)$ é $\forall X \forall Y(p(X,Y) \wedge q(X))$, enquanto $\exists(\varphi)$ é $\exists X \exists Y(p(X,Y) \wedge q(X))$.

—

Definição 3.11 (Ocorrência positiva)**.** Um átomo φ ocorre *positivamente* em φ. Se um átomo ocorre positivamente (negativamente) em uma fórmula ψ, então φ *ocorre positivamente* (ocorre negativamente) em $\exists X\psi$, $\forall X\psi$, $\varphi \wedge \alpha$, $\varphi \vee \alpha$ e $\varphi \leftarrow \alpha$.

Se um átomo φ ocorre positivamente (negativamente) em uma fórmula ψ, então *phi ocorre negativamente* (positivamente) em $\neg\psi$ e em $\alpha \leftarrow \psi$. —

Definição 3.12 (Literal)**.** Um literal é um átomo ou a negação de um átomo. Um *literal positivo* é um átomo. Um *literal negativo* é a negação de um átomo. —

Definição 3.13 (Cláusula)**.** Uma cláusula é uma fórmula na forma:

$$\forall X_1 \ldots \forall X_s(\varphi_1 \vee \ldots \vee \varphi_m)$$

sendo que cada φ_i é um literal e $X_1 \ldots X_s$ são todas variáveis ocorrendo em $\varphi_1 \vee \ldots \vee \varphi_m$.

Exemplo 3.5. As seguintes fórmulas são cláusulas:

$$\forall X \forall Y \forall Z(p(X,Z) \vee \neg p(X,Y) \vee \neg r(Y,Z))$$
$$\forall X \forall Y(\neg p(X,Y) \vee r(f(X,Y),a))$$

Pelo fato de cláusulas serem comuns em Programação em Lógica, é conveniente adotar uma notação especial. Dessa forma, as cláusulas na forma:

$$\forall X_1 \ldots \forall X_s(\varphi_1 \vee \ldots \vee \varphi_k \vee \neg\psi_1 \vee \ldots \vee \neg\psi_n)$$

são escritas como:

$$\varphi_1, \ldots, \varphi_k \leftarrow \psi_1, \ldots, \psi_n$$

sendo que $\varphi_1, \ldots, \varphi_k$ e $\psi_1, \ldots, \psi_n$ são átomos e $X_1 \ldots X_s$ são todas variáveis ocorrendo nesses átomos.

Dessa forma, na *notação clausal*, todas as variáveis são assumidas como universalmente quantificadas, as vírgulas no antecedente $(\psi_1, \ldots, \psi_n)$ denotam conjunção, enquanto as vírgulas no consequente $(\varphi_1, \ldots \varphi_k)$ denotam disjunção. Essas convenções são justificadas porque

$$\forall X_1 \ldots \forall X_s(\varphi_1 \vee \ldots \vee \varphi_k \vee \neg\psi_1 \vee \ldots \vee \neg\psi_n)$$

é classicamente equivalente a

$$\forall X_1 \ldots \forall X_s(\varphi_1 \vee \ldots \vee \varphi_k \leftarrow \psi_1 \wedge \ldots \wedge \psi_n)$$

—

Definição 3.14 (Cláusula definida)**.** Uma cláusula definida de programa é uma cláusula na forma

$$\varphi \leftarrow \psi_1, \ldots \psi_n$$

que contem precisamente um átomo (φ) em seu consequente. No caso, φ é chamado de *cabeça* e $\psi_1, \ldots \psi_n$ de *corpo* da cláusula do programa. —

Definição 3.15 (Fato ou cláusula unitária)**.** Uma cláusula unitária, também chamada de fato, é uma cláusula na forma:

$$\varphi \leftarrow$$

ou seja, um fato é uma cláusula definida de programa com o corpo vazio.

A semântica informal de $\varphi \leftarrow, \psi_1, \ldots \psi_n$ é "para cada valoração de cada variável, se $\psi_1, \ldots, \psi_n$ são todos verdadeiros, então φ é verdadeiro". Dessa forma, se $n > 0$, a cláusula é *condicional*. Por outro lado, uma cláusula unitária $\varphi \leftarrow$ é incondicional. Sua semântica informal é "para cada valoração de cada variável, φ é verdadeiro". —

Definição 3.16 (Programa definido)**.** Um programa definido é um conjunto finito de cláusulas definidas. —

Definição 3.17 (Definição de Símbolo predicativo)**.** Em um programa definido, o conjunto de todas as cláusulas com o mesmo símbolo predicativo p na cabeça é chamado de *definição*

de p (LLOYD, 1993, p. 109 e 145). Nesse caso,

$$h \leftarrow \psi_1, \ldots, \psi_n$$
$$\ldots$$
$$h \leftarrow \psi_{n+1}, \ldots, \psi_m$$

denota

$$h \leftarrow (\psi_1 \wedge \ldots \wedge \psi_n) \vee \ldots \vee (\psi_{n+1} \wedge \ldots \wedge \psi_m)$$

—

Definição 3.18 (Consulta definida)**.** Uma consulta (*goal*) definida é uma cláusula na forma:

$$\leftarrow \psi_1, \ldots \psi_n$$

ou seja, é uma cláusula cuja cabeça (o consequente) é vazia. Cada $\psi_i (i \in \{1, \ldots, n\})$ é chamado de *subconsulta* (*subgoal*) da consulta.

Se $Y_1, \ldots Y_r$ são variáveis da consulta

$$\leftarrow B_1, \ldots B_n$$

então essa notação clausal é uma abreviação de

$$\forall Y_1 \ldots \forall Y_r (\neg\psi_1 \vee \ldots \vee \neg\psi_n)$$

ou, de forma equivalente,

$$\neg\exists Y_1 \ldots \exists Y_r (\psi_1 \wedge \ldots \wedge \psi_n)$$

—

Definição 3.19 (Cláusula vazia)**.** Uma cláusula vazia, denotada por $\emptyset$, é uma cláusula cujo o antecedente e o consequente são vazios. Esta cláusula pode ser entendida como uma "contradição". —

Definição 3.20 (Cláusula de Horn)**.** Uma cláusula de Horn é uma cláusula definida de programa ou uma consulta definida. —

Definição 3.21 (Termo *ground* e Átomo *ground*)**.** Um termo *ground* é um termo que não contém variáveis. Da mesma forma, um átomo *ground* é um átomo que não contém variáveis. —

A semântica da Programação em Lógica é dada de forma usual com Teoria de Modelos, especialmente referentes às interpretações dos Modelos de Herbrand. A Lógica de Primeira Ordem Clássica provê métodos para dedução de teoremas a partir da teoria composta por axiomas, ou seja, deduzir fórmulas que são consequência lógica dos axiomas

da teoria e, dessa maneira, que são "verdadeiras" em todas as interpretações em que há um modelo para cada axioma da teoria. No caso da Programação em Lógica, utiliza-se um tipo específico de regra de inferência, chamada *Resolução*, introduzido por Robinson (1965) e aperfeiçoado por Kowalski (1974) como SLD *(Selection-rule driven Linear resolution for Definite clauses)*, que se restringe ao tipo específico de fórmulas (chamadas cláusulas) utilizados no paradigma de Programação em Lógica. Informalmente, suponha que se deseja provar que a fórmula:

$$\exists Y_1 \ldots \exists Y_r(\varphi_1 \wedge \ldots \wedge \varphi_n)$$

é uma consequência lógica de um programa P. Nesse caso, o método de *Resolução* é um sistema de refutação. A negação da fórmula que se deseja demonstrar é adicionada à lista de axiomas e busca-se obter uma contradição. No caso, quando se nega a fórmula acima obtém-se uma consulta:

$$\leftarrow \varphi_1, \ldots \varphi_n$$

Utilizando-se um processo chamado *backtraking*[2], que é um processamento em árvore *top-down*, o sistema de prova deriva sucessivas consultas. Se uma cláusula vazia $\emptyset$ é derivada, então uma contradição foi obtida, de modo que se pode assumir que

$$\exists Y_1 \ldots \exists Y_r(\varphi_1 \wedge \ldots \wedge \varphi_n)$$

é de fato uma consequência lógica do programa P.

Do ponto de vista de tratamento lógico da informação, o que se está interessado geralmente não é exatamente se uma fórmula é ou não consequência da base de axiomas, mas sim quais são os valores atribuídos a variáveis usados como contra exemplos para derrogar a contradição incluída pela consulta. Em outras palavras, um analista da informação está interessado em obter valores para suas consultas a partir da base fatos do programa. Tecnicamente, os valores para as variáveis são obtidas pelo processo de *unificação*.

3.1.3 Otimização de negações

A respeito do uso da negação em programas lógicos, conforme Naish (1986, p. 6):

> *We want* not *to fail as quickly as possible, without retrying the call more than necessary. Making* not *succeed quickly is sometimes useful, but can also lead to greater inefficiency. If the rest of the computation is nondeterministic, then* not *is called fewer times if it succeeds quickly. If the negated call is nondeterministie, however, delaying further can speed*

[2] Conforme Sebesta (2012, p. 742): *"When a goal with multiple subgoals is being processed and the system fails to show the truth of one of the subgoals, the system abandons the subgoal it cannot prove. It then reconsiders the previous subgoal, if there is one, and attempts to find an alternative solution to it. This backing up in the goal to the reconsideration of a previously proven subgoal is called back- tracking. A new solution is found by beginning the search where the previous search for that subgoal stopped. Multiple solutions to a subgoal result from different instantiations of its variables."*

> *up the execution of* not. *Also, the more* not *is retried, the more time will be wasted when it can neither succeed or fail. We therefore suggest only retrying* not *when one of the previous proofs may no longer bind any of the calls variables.*

Por isso, considerando a semântica operacional de Prolog, para que o algoritmo de *backtraking* do método de *Resolução* seja mais eficiente, negações nos antecedentes de regras não determinísticas devem ser posicionados mais ao final possível das regras. Por exemplo, ao invés de se escrever uma regra conforme a Equação 3.1:

$$\begin{aligned} ngrule(deo\_non\_pt\_extends\_pt, C, W) \leftarrow & \neg pt(C) \\ & deo(C, W), \\ & pt(W) \end{aligned} \tag{3.1}$$

Deve-se optar forma da Equação 3.2:

$$\begin{aligned} ngrule(deo\_non\_pt\_extends\_pt, C, W) \leftarrow & deo(C, W), \\ & pt(W), \\ & \neg pt(C) \end{aligned} \tag{3.2}$$

Outros detalhes técnicos a respeito dessa característica dos programas lógicos com negação, consultar Naish (1986).

3.1.4 Hipótese do nome único e Hipótese do mundo fechado

A hipótese do nome único, em inglês *Unique-Name Assumption*, é uma abordagem utilizada especialmente em sistemas de banco de dados que afirma que cada entidade representada em uma base de dados é identificada por exatamente um por um nome, e cada nome na base de dados refere-se a exatamente uma entidade. Dessa forma, não é o caso que existam duas entidades com o mesmo nome, nem um nome que refira-se a duas entidades diferentes. Já a hipótese do mundo fechado, em inglês *Closed World Assumption*[3] (CWA), é a abordagem que estabelece que tudo o que não é dito no mundo não é o caso. Ou, posto de outra forma, só é o caso o que é conhecido pela base de dados. As duas hipóteses juntas são conhecidas como *semântica de base de dados*. Conforme a obra-prima em Inteligência Artificial de (RUSSELL; NORVIG, 2010, p. 299-300, 344):

[3] Conforme Lloyd (1993, p. 72), a CWA foi introduzida em Reiter (1978) e denota aspecto não-monotônico nas regras de inferências que é adequado tanto à inteligência artificial quanto para banco de dados, uma vez que permite inferir conhecimento negativo sobre a base conhecimento. A regra é não monotônica porque o acréscimo de fatos a uma base pode revogar conclusões anteriores a respeito daquela base.

> *The unique names assumption says that every Prolog constant and every ground term refers to a distinct object, and the closed world assumption says that the only sentences that are true are those that are entailed by the knowledge base. There is no way to assert that a sentence is false in Prolog. This makes Prolog less expressive than first-order logic, but it is part of what makes Prolog more efficient and more concise. [...] One proposal that is very popular in database systems works as follows. First, we insist that every constant symbol refer to a distinct object—the so-called* ***unique-names assumption****. Second, we assume that atomic sentences not known to be true are in fact false – the* ***closed-world assumption****. Finally, we invoke* ***domain closure****, meaning that each model contains no more domain elements than those named by the constant symbols. Under the resulting semantics, which we call* ***database semantics*** *to distinguish it from the standard semantics of first-order logic [...]. Database semantics is also used in logic programming systems [...]*

A semântica de base de dados não é assumida em *description logics* (DL) em geral. Nesse tipo de abordagem, a hipótese de mundo adotada é a de mundo aberto, assim como ocorrem em FOL. Dessa forma, sempre é necessário existir um axioma que afirme quando algo não é o caso. O mesmo refere-se à hipótese de nome único. Em DL, se nada é dito sobre a identidade de dois nomes, não se pode assumir que se tratam de entidades diferentes, iguais.

Como as investigações desta pesquisa estão relacionadas à abordagem de Prolog, é natural que a semântica de base de dados vincule os resultados obtidos, o que de fato é o que ocorre.

3.2 Programação em Lógica Modal: MProlog

Esta seção descreve o paradigma de Programação em Lógica Modal adotado neste trabalho. O objetivo é apresentar o arcabouço lógico modal provido pelo MProlog, utilizado na obtenção de resultados específicos desta pesquisa, conforme o Capítulo 9.

MProlog é um dos produtos da tese de doutoramento do vietnamita Anh Linh Nguyen (NGUYEN, 1999), que desde 1999 tem publicado uma série de artigos e relatórios (NGUYEN, 2001; NGUYEN, 2003; NGUYEN, 2004a; NGUYEN, 2004b; NGUYEN, 2005; NGUYEN, 2007; NGUYEN, 2009; NGUYEN, 2010) a respeito de resultados em raciocínio modal automatizado com aplicações para a Inteligência Artificial, especialmente os banco de dados dedutivos modais[4]. Nessa área o autor apresentou resultados interessantes, como a demonstração dos algoritmos de transformação de *magic sets multimodais* que essencialmente são algoritmos de transformação do tipo *bottom-up* que simulam, de forma mais eficiente, os resultados padrões da abordagem *top-down* do processo de *Resolução*. Os Banco de Dados Dedutivos (conhecidos como Datalog) são amplamente utilizados no contexto de *big-data* (BU et al., 2012).

[4] A teoria do banco de dados dedutivo multimodal do autor é chamada de MDatalog.

Nesse contexto modal, Nguyen apresenta não apenas as bases e as provas teóricas de sua abordagem de Programação em Lógica Modal, como também provê uma implementação concreta em Prolog de algoritmos de Resolução Modal *top-down*. MProlog é concebido como um módulo para Prolog tão expressivo quanto o fragmento de cláusulas de Horn usadas para a programação clássica. Porém, adiciona expressividade para figurar modalidades e multimodalidades. MProlog é um arcabouço extensível que permite que diferentes cálculos de consequência lógica sejam implementados. Em Nguyen (2003) encontra-se um arcabouço para desenvolvimento da *least model semantics*, *fixpoint semantics* e *SLD-resolution calculi* para programas modais positivos em lógicas modais cujas restrições consistem em condições seriais (por exemplo, $\forall x \exists y R_i(x, y)$ para todo inteiro i) e também para cláusulas clássicas de Horn, de modo que é possível combinar cláusulas clássicas e modais em um mesmo programa. O arcabouço tem sido aplicado para lógicas monomodais e multimodais seriais, ou seja, que incorporam o axioma $(D) : \Box_i \psi \rightarrow \Diamond_i \psi$ (NGUYEN, 2007).

Segundo Nguyen, é possível classificar as abordagens de Programação em Lógica Modal em duas categorias: a abordagem de tradução e a abordagem direta[5]. Dos trabalhos de abordagem direta, há conhecimento que as únicas implementações existentes são MProlog e Molog[6] (FARIÑAS DEL CERRO, 1986). De fato, o Molog foi estudado por Nguyen, que identificou suas principais características e deficiências e propôs o MProlog como uma evolução daquela solução no que tange aos cálculos seriais.

Com Molog é possível construir uma lógica modal e definir as regras que resolvem os operadores modais. Diferentemente de MProlog, Molog possibilita que diferentes tipos de lógicas modais sejam construídas, e não apenas restritas aos cálculos seriais. A implementação de Molog, no entanto, não permite que sejam incorporados programas Prolog diretamente em seu corpo. Para que isso seja possível, introduz-se um predicado especial que tem essa atribuição. No entanto, essa estratégia pode ser inconveniente em programas longos. Sem contar que pode inviabilizar a integração de programas clássicos existentes com o paradigma modal. Isso se deve, por exemplo, ao fato de que em Molog os operadores `&` e `<--` são usados para a conjunção e implicação, respectivamente, e não os símbolos `,` e `:-` tradicionais dos programas Prolog. Por esse motivo, o uso da ferramenta se torna menos adequado em contextos em que é necessário intercalar os dois tipos de programas: clássicos e modais. Além disso, em Molog as fórmulas são tratadas internamente como fórmulas de Skolem, o que pode tornar a depuração (*debug*) confusa para programadores que não conheçam detalhadamente a implementação do Molog. Nessa questão, MPrologusa uma estratégia de computação diferente, porque usa a técnica de *labeling* para operadores existenciais modais, ao invés de *skolemização*, além de oferecer um processo próprio de

[5] Ver Nguyen (2010) para um *survey* sobre as abordagens direta e de tradução da Programação em Lógica Modal.

[6] O Molog, e ferramentas derivadas, como o *Toulouse Inference Machine* (TIM) (BALBIANI; HERZIG; LIMA-MARQUES, 1991) e TARSKI (BALBIANI; HERZIG; LIMA-MARQUES, 1992), foram usadas na tese de doutoramento do orientador desta tese (LIMA-MARQUES, 1992).

depuração que facilita a interpretação das fórmulas modais. Há também outras características que distinguem os dois arcabouços. Em MProlog as modalidades possuem formas normais e relações de pré-ordem entre operadores modais, e essa ferramenta é capaz de produzir respostas em Prolog padrão, o que engloba instanciação de variáveis em consultas definidas, ao invés de prover meras respostas computadas (como as do tipo 'sim' ou 'não') do Molog.

No arcabouço do MProlog, uma série de cálculos de resolução SLD (*Selective Linear Definite*) para diferentes tipos de lógicas são providas, com foco em lógicas epistêmicas seriais. Dentre o sistemas incorporados, destaca-se KD, T, KB, KDB, B, $K5$, $KD5$, $K45$, $KD45$, $KB5$, $S5$, $KDI4_s$, $KDI4_s5$, $KDI45$, $KD4_s5_s$ e $KD4I_g5_a$, que podem ser usadas para raciocinar sobre estados epistêmicos de agentes. As provas de corretude e completude para esses cálculos são apresentadas pelo autor nas obras indicadas no primeiro parágrafo. Comparado ao MProlog, o MProlog é mais específico (ou restrito), uma vez que é limitado aos cálculos seriais, mas é também mais profundo, pois incorpora e permite que se defina uma vasta gama desses sistemas.

No relatório técnico de Nguyen (2010, p. 21-25) encontra-se uma apresentação concisa da linguagem de Programação em Lógica Modal MProlog, baseada em um excerto do formalismo da Lógica Modal positiva do arcabouço. Conforme o texto, MProlog é restrito a construções modais seriais normais (baseadas no axioma K e D, ver próxima seção). Além da apresentação técnica, no relatório citado consta também as duas primeiras páginas de exemplos de programas modais construídos com o arcabouço. O documento é um relatório de 147 páginas que sumariza o trabalho de MProlog e do MDatalog, e concentra o conteúdo apresentado em diferentes publicações sobre o tema, realizadas no período de 1999 a 2010. As principais publicações de Nguyen mencionadas no relatório estão listadas nas Referências. No documento completo é possível encontrar a descrição detalhada do arcabouço proposto pelo autor, as provas de corretude e completude da resolução SLD, exemplos de instanciações do arcabouço, em que são detalhadas as regras para os principais sistemas lógicos epistêmicos de multiagentes. Otimizações e detalhes de implementação também são abordados pelo documento. Em Nguyen (2006), encontra-se uma apresentação formal ainda mais detalhada do arcabouço de programação em lógica modal. Já em Nguyen (2009) é possível encontrar uma apresentação mais concisa do formalismo de MProlog.

3.2.1 Sistemas epistêmicos de MProlog

Uma lógica pode ser definida por um conjunto de fórmulas bem-formadas, uma classe de interpretações possíveis e uma relação de satisfação. A classe de interpretações admissíveis para uma lógica modal L é geralmente especificada por restrições aos enquadramentos (*frames*) de Kripe. Em Nguyen (2010, p. 13-15) e em Nguyen (2006,

p. 250-252) encontram-se uma formulação tradicional, no estilo Hilbert, de uma Lógica modal quantificada. Nessa lógica, termos são definidos indutivamente como variáveis, símbolos constantes e funções. Sintaticamente, as fórmulas são definidas como predicados n-ários e por operadores sintáticos de disjunção ($\vee$) e negação clássica ($\neg$), o quantificador universal ($\forall$) e o (operador modal universal) $\Box$, além da interdefinição de conectivos, nos moldes da Definição 3.3 (Definição de conectivos). A semântica é dada em termos de enquadramentos e modelos de Kripe e por meio de relações de acessibilidade. A axiomatização desse sistema é dada como uma lógica modal *normal* ou seja, que possui no axioma K, conforme os seguintes axiomas:

a) axiomas da Lógica Clássica de Predicados sem identidade;

b) o axioma $K = \Box_i(\varphi \rightarrow \psi) \rightarrow (\Box_i\varphi \rightarrow \Box_i\psi)$, também chamado de *distribuição epistêmica* (van BENTHEM, 2010, p. 139);

c) o axioma da fórmula de Barcan de alternação entre quantificadores e modalidades: $\forall x.\Box_i\varphi \rightarrow \Box_i\forall x.\varphi$;

d) a definição da modalidade existencial: $\Diamond_i\varphi \equiv \neg\Box_i\neg\varphi$;

e) a regra de modus ponens: $\varphi, \varphi \rightarrow \psi \vdash \psi$;

f) a regra de generalização: se $\vdash \varphi$, então $\vdash \forall x.\varphi x$;

g) a regra de necessitação: se $\vdash \varphi$, então $\vdash \Box_i\varphi$.

Por questão de brevidade, por não apresentar contribuição relevante e por se tratar de construção padrão, obstém-se de transcrever nesta seção as definições presentes na obra citada para essa lógica. Porém, esse sistema é utilizado pelo autor para discutir como a combinação de diferentes axiomas pode ser incluída nessa lógica para criar sistemas cujas aplicações são de interesse desta pesquisa. Em vista disso, com o objetivo de explorar sistemas para formalização de raciocínios sobre epistemologias, a Tabela 2 sistematiza um conjunto desses axiomas, as relações de acessibilidade que eles denotam entre os mundos e um significado possível para cada um deles.

Com base nos axiomas generalizados da Tabela 2, em Nguyen (2010, p. 14-18) explora-se a combinação arbitrária deles com a lógica apresentada para obtenção de sistemas com propriedades específicas referentes ao raciocínio sobre a crença de vários agentes. Os axiomas seguintes, por exemplo, podem ser combinados para criação de lógicas que se prestam a descrever estados epistêmicos de agentes sobre graus de conhecimentos:

Tabela 2 – Axiomas referentes a modalidades epistêmicas

Axioma	Esquema	Restrição	Exemplo de significado
(D)	$\Box_i\varphi \rightarrow \neg\Box_i\neg\varphi$, que é o mesmo que $\Box_i\varphi \rightarrow \Diamond_i\varphi$	$\forall u\ \exists v\ R_i(u,v)$	a crença é consistente: afirmações aleticamente necessárias são também afirmações possíveis
(I)	$\Box_i\varphi \rightarrow \Box_j\varphi$ se $i > j$	$R_j \subseteq R_i$ se $i > j$	o índice indica grau de crença
(4)	$\Box_i\varphi \rightarrow \Box_i\Box_i\varphi$	$\forall u,v,w\ (R_i(u,v) \wedge R_i(v,w) \rightarrow R_i(u,w))$	a crença satisfaz introspecção positiva (transitividade)
(4_s)	$\Box_i\varphi \rightarrow \Box_j\Box_i\varphi$	$\forall u,v,w\ (R_j(u,v) \wedge R_i(v,w) \rightarrow R_i(u,w))$	a crença satisfaz introspecção positiva forte (transitividade)
(5)	$\neg\Box_i\varphi \rightarrow \Box_i\neg\Box_i\varphi$	$\forall u,v,w\ (R_i(u,v) \wedge R_i(u,w) \rightarrow R_i(v,w))$	a crença satisfaz introspecção negativa (euclidiana)
(5_s)	$\neg\Box_i\varphi \rightarrow \Box_j\neg\Box_i\varphi$	$\forall u,v,w\ (R_j(u,v) \wedge R_i(u,w) \rightarrow R_i(v,w))$	a crença satisfaz introspecção negativa forte (euclidiana)

Fonte: Os autores. Adaptado de Nguyen (2010, p. 17, 18) e de Nguyen (2004b, p. 268).

$$KDI4_s = K_(m) + (D) + (I) + (4_s)$$
$$KDI4 = K_(m) + (D) + (I) + (4)$$
$$KDI4_s5 = K_(m) + (D) + (I) + (4_s) + (5)$$
$$KDI45 = K_(m) + (D) + (I) + (4) + (5)$$

Nesses sistemas, o axioma (I) faz com que $\Box_i\varphi$ signifique "acredita-se em φ até o grau i", e que $\Diamond_i\varphi$ possa ser lido como "é possível fracamente até o grau i que φ". Os axiomas (5) são controversos, por serem são muito fortes, uma vez que poderia ser interpretados como "aquilo se crê como não sendo o caso, acredita-se que necessariamente não é o caso". Por isso, os sistemas KDI4 e KDI4_s[7] foram desenvolvidos, como uma alternativa mais fraca do que os que contém (I).

Para os sistemas multiagentes, os índices dos operadores modais $\Box$ e $\Diamond$ denotam agentes e assume-se que $\Box_i\varphi$ representa "o agente i acredita que φ é verdadeiro" e $\Diamond_i\varphi$ representa que "φ é considerado possível pelo agente i".

Em sistemas de crença distribuída, os agentes possuem acesso completo às bases de crenças uns dos outros. Nesse caso, os agentes são *amigos* e colaboram uns com os outros em um sistema unificado. Para sistemas de crença distribuída com colaboração, a seguinte composição de axiomas pode ser utilizada:

$$KD4_s5_s = K_(m) + (D) + (4_s) + (5_s)$$

[7] Observar que o axioma ($5s$) pode ser obtido de KDI$4s5$.

Outra forma de combinação de axiomas permite criar um sistema de multiagentes em que os integrantes são *oponentes*, e trabalham uns contra os outros. Uma interpretação desse tipo de sistema permite que sejam desenvolvidas situações em que um agente tenta simular os estados epistêmicos dos outros. Uma composição adequada para esse tipo de sistema é esta, em que o índice m é utilizado apenas para indicar que se trata de um sistema com múltiplos agentes:

$$KD45_m = K_(m) + (D) + (4_s) + (5_s)$$

O tipo de interpretação apresentada por esses sistemas é baseada na ideia *crença*, e não exatamente de *conhecimento*. Por isso, também poderiam ser chamados de lógicas doxásticas. O axioma (D) é especialmente importante para as interpretações doxásticas/epistêmicas porque denota que o agente não acredita simultaneamente em uma afirmação e sua negação, ou seja, trata-se de um tipo de agente racional (SMULLYAN, 1986).

O objetivo desses axiomas e sistemas é capturar propriedades importantes de crença e de crença compartilhada que são úteis aos objetivos desta pesquisa, e não investigar exaustivamente essas formulações. Esses sistemas epistêmicos são explorados no Capítulo 9.

3.2.2 Sintaxe base de teorias MProlog

Sintaticamente, em MProlog modalidades são representadas como listas, como nos exemplos seguintes. À esquerda estão fórmulas lógicas tradicionais (por exemplo, conforme são apresentadas por Carnielli e Pizzi (2008)), e à direita estão transliterações dessas fórmulas em sentenças de MProlog, sendo que `b` é um símbolo para "*box*", `d` para "*diamond*", `bel` para "*believe*", e `pos` para "*possible*":

$\Box\Diamond q(x,y)$	`[b, d] : q (X, Y)`
$\Box_i\langle X_3\rangle\Diamond_j q(a)$	`[bel(I), pos(3,X), pos(J)] : q (a)`
$christian(x) \rightarrow \Box_x god\_exists$	`[bel(X)] : god_exists :- christian(X)`

Ainda no ponto de vista sintático, programas Prolog são programas MProlog, de modo que fragmentos modais em programas MProlog podem conter diretivas e cláusulas clássicas. Assim, cada cláusula que faz parte do conjunto de cláusulas de um programa MProlog possui uma das seguintes formas:

```
Context : (Head :- Body).
```

```
Head :- Body.
```

em que `Context` é uma lista que representa modalidades, `Head` ou possui a forma E ou a forma $M : E$, sendo que E é um átomo clássico, e M é uma lista contendo uma ou mais modalidades. Com essa forma, uma sentença em MProlog pode tanto conter expressões clássicas de Prolog, como modais de MProlog. Isso é útil aos objetivos estipulados na

seção 1.2, por exemplo, porque permite combinar regras lógicas que validam restrições em modelos ontológicos classicamente, e regras epistêmicas modalmente.

3.2.3 Principais predicados de MProlog e bibliotecas de cálculos

De acordo com Nguyen (2010, p. 94) e Nguyen (2004b, p. 275), há três grupos de predicados pré-definidos úteis aos usuários da implementação de MProlog: predicados principais (utilizados para consulta, execução e depuração), predicados para obtenção do estado do sistema e predicados para modificação do programa em execução. Os predicados principais são estes:

a) `consult_calculi(Files)`: utilizado para carregar um cálculo de resolução SLD para lógicas modais. O argumento é o nome de um arquivo ou de uma lista de arquivos. O autor do arcabouço em tela provê os seguintes conjuntos de cálculos:

   - `belief.cal`: cálculos de resolução SLD para lógicas modais de múltiplos agentes. Suporta os seguintes sistemas lógicos: $KD4_s5_s$, $KD45_m$, $KDI4_s5$, $KDI4_s$, $KDI4$ e $KDI45$;
   - `belief_box.cal`: cálculos de resolução SLD para lógicas modais para uma variação de MProlog, chamado MProlog-$\Box$. Suporta os mesmos tipos de cálculos de `belief.cal`, com os devidos ajustes nas regras de inferências;
   - `groups.cal`: cálculo de resolução SLD para o sistema $KD4I_g5_a$. A implementação contempla apenas programas modais cujas cláusulas possuam contexto modal com tamanho 0 ou 1;
   - `smnm.cal`: cálculos de resolução SLD para lógicas monomodais seriais. Os seguintes sistemas são contemplados: KD, T, KDB, B, $KD4$, $S4$, $KD5$, $KD45$, $S5$.

b) `mconsult(ProgramFile, Calculus)` e `mconsult(ProgramFile)`: esse predicado é utilizado para carregar um programa no âmbito de um cálculo. Embora os programas MProlog possam incorporar cláusulas clássicas e modais, e portanto conter um programa Prolog clássico, os programas modais devem ser escritos em um arquivo próprio, de modo que não seja nem consultado, nem compilado diretamente em Prolog. Por isso, sugere-se que os programas MProlog sejam escritos em arquivos com a extensão `.mpl`, alterno à extensão padrão, `.pl`, de Prolog, para que possam ser consultados pelo programa Prolog que o controla. Portanto, para um programa modal com a extensão `.mpl`, sempre haverá ao menos um programa Prolog, com a extensão `.pl`, que o consulta e controla;

c) `mcall(Goal, Calculus)` e `mcall(Goal)`: esses predicados são utilizados para executar uma consulta definida à base de fatos modais. Variáveis e unificações

ocorrem normalmente entre programas clássicos e modais;

d) `mtrace` e `nomtrace`: tratam-se de predicados que ligam e desligam o modo depuração, respectivamente.

Para ilustrar o uso dos predicados, o Código 1 contém um programa Prolog que carrega o arcabouço do MProlog, consulta a biblioteca de cálculos seriais e executa o programa modal contido no Código 2.

Código 1 – Exemplo simples de MProlog: código Prolog

```
% load MProlog
:- include('../../../ontoprolog/modal/mprolog2-custom/mprolog').

% Consult calculi
:- consult_calculi('../../../ontoprolog/modal/mprolog2-custom/smnm.cal').

% Consult modal program
:- mconsult('simple_example.mpl').
```

O programa do Código 2, por sua vez, define uma teoria modal simples, em que se exercita cláusulas com expressões modais. No caso, `[b]` denota a modalidade universal $\Box$ e `[d]` a modalidade existencial $\Diamond$.

Código 2 – Exemplo simples de MProlog: especificação modal

```
:- calculus cKD.

[b] : (likes_football(husband(X)) :-
      woman(X),
      likes_football(X)).

[d] : likes_football(X) :- man(X).
man(husband(X)) :- woman(X).
[b] : woman(X) :- woman(X).
[b] : man(X) :- man(X).
[b] : likes_football(X) :- likes_football(X).
man(tom).
woman(jane).
woman(mary).
likes_football(jane).

/*
  % goals to answer:
  mcall([b]:likes_football(X), cKD).
  mcall([d]:likes_football(X), cKD).
*/
```

Executar a primeira consulta mencionada no código retorna os seguintes resultados:

```
| ?- mcall([b]:likes_football(X), cKD).
X = husband(jane) ? ;
X = jane ? ;
no
```

Informalmente, é fácil verificar que apenas atenderia à definição de $\Box likes\_football(X)$ as instâncias da definição na linha 3 do Código 2, uma vez que é a única definição para o predicado contido na teoria. Para atender esse predicado da linha 3, é necessário provar que X sucede para $woman(X)$ e $likes\_football(X)$, o que de fato é o caso nas linhas 13 e 15, respectivamente.

O resultado da execução da segunda consulta é este:

```
| ?- mcall([d]:likes_football(X), cKD).
X = husband(jane) ? ;
X = husband(mary) ? ;
X = tom ? ;
X = jane ? ;
no
```

É possível verificar que, pela definição do axioma (D) (Tabela 2), o resultado da primeira execução deve estar incluído no resultado da segunda, como de fato é o caso. O resultado $husband(mary)$ e tom são devidos à definição da linha 7, que afirma $\Diamond likes\_football(X)$ para todo $man(X)$: $man(husband(mary))$ sucede devido à definição da linha 8 e tom pela definição $man(tom)$ da linha 12.

3.3 Prolog e especificação de linguagens formais

Nesta seção apresenta-se resultados de pesquisa sobre publicações referentes ao desenvolvimento de DSL em Prolog. Mais de uma abordagem é considerada, investigada e prototipada. Porém, a escolha pela definição de linguagem interna com operadores, conforme estabelecido no Capítulo 7, é devida à possibilidade de combinar sentenças da DSL com programas comuns. As seções seguintes descrevem parte dos resultados dessas investigações.

3.3.1 Semântica formal de linguagens

Semântica formal de linguagens diz respeito à especificação de significado, ou comportamento, de programas de computador, partes de hardware, sistemas lógicos, ..., enfim, de linguagens artificiais em geral. Conforme Nielson e Nielson (2007, p. 1), a necessidade pela especificação formal surge porque:

a) ela pode revelar ambiguidades e complexidades sutis e definições aparentemente claras, por exemplo em manuais e outros documentos; e

b) ela serve de base para a implementação, análise e verificação, especialmente, quanto a provas de corretude.

Em Reimer e Hahn (1983), encontra-se a seguinte contribuição a essa discussão. Na obra, o autor apresenta uma semântica operacional para *frame data model*:

> *The lack of semantic specifications in knowledge representation languages may cause severe problems (e.g. unpredictable side effects and inconsistent changes in the data base), which obviously become more and more serious when large numbers of knowledge structures are managed. We consider knowledge base systems to be realizations of formally defined data models.*

Na história da computação, diversas estratégias para se atribuir semântica de linguagens de programação foram propostas. Como exemplo de três grandes gêneros, pode-se mencionar a:

a) semântica estática *Attribute Grammar*, proposta por Knuth (1968);

b) *semântica denotacional* que, segundo Schmidt (1986), foi desenvolvida pelo Grupo de Pesquisa em Programação da Universidade de Oxford, comandado por Christopher Strachey (1916–1975), nos anos 1960. Uma das obras centrais dessa abordagem é o trabalho de Scott e Strachey (1971);

c) *semântica axiomática*, de Hoare (1969), conhecida também como *Hoare logic*.

Curiosamente, Moss (1982) inicia da seguinte maneira seu artigo com o objetivo de apresentar uma abordagem de como definir uma linguagem usando Prolog:

> *There have been many papers, conferences and theses concerned with the task of defining programming languages. Yet, twenty years after McCarthy's (1962) seminal paper there is still no widely accepted method which will deal with all parts of a language clearly and concisely.*

O artigo de Moss é de 1982, ou seja, ele já possui mais de 30 anos, o que, somando aos 20 anteriores desde o artigo de McCarthy (1962), temos mais de meio século de desenvolvimento de linguagens e, como se observa em textos mais recentes, como Nielson e Nielson (1999), Nielson e Nielson (2007), Watt (2004) e Turbak e Gifford (2008), ainda resta a ausência de um consenso sobre os métodos a serem adotados em cada situação. No entanto, percebe-se que nas (inúmeras) estratégias que envolvem semântica denotacional e operacional tenham convergido para estratégias mais ou menos baseadas em λ-calculus (Cálculo Lambda) e *Hoare logic*, respectivamente.

No entanto, outros tipos de estratégias têm sido adotadas, especialmente mais recentemente. Tratam-se, geralmente, de definições ainda denotacionais e operacionais, mas auxiliadas por computador. Isso tem permitido uma rápida proliferação de linguagens, especialmente as DSL. Em Sadilek (2008) e Sadilek e Wachsmuth (2009), por exemplo, os autores apresentam a implementação de uma ferramenta baseada no arcabouço da IDE Eclipse[8] para definir formalmente a semântica de linguagens, de modo que linguagens específicas de domínio (DSL) possam ser rapidamente modeladas, prototipadas e implementadas.

3.3.2 Funções de mapeamento: Ligação entre sintaxe e semântica

Do ponto de vista denotacional, atribuir semântica à sintaxe de uma linguagem é definir uma função de mapeamento cujo domínio são as estruturas sintáticas e a imagem

[8] <http://www.eclipse.org/>

são objetos matemáticos, representados concretamente em algum tipo de formalismo, com Cálculo Lambda. De certa forma, trata-se de um tipo de tradução de uma estrutura em outra:

> *The syntax comprises the* abstract syntax *representing the essence of the concepts of the language. The abstract syntax can be mapped to several* concrete syntaxes *(textual representation, graphical visualization, etc). These two components specify the different concepts of the language and how they are represented. The semantics of a formalism is specified by a unique* semantic mapping function *which maps each model element in the language onto an element in a* semantic domain*. For example, the meaning of a Causal Block Diagram is given by mapping it onto an Ordinary Differential Equation. For practical reasons, semantic mapping is usually applied to the abstract rather than to the concrete syntax of a model. Note that the semantic domain is a modelling language in its own right which needs to be properly modelled (and so on, recursively).* ***In practice, the semantic mapping function maps abstract syntax onto abstract syntax*** (SYRIANI, 2011, p. 14-15). (grifo final nosso)

Em Harel e Rumpe (2000, p. 10), há uma adequada visão geral sobre sintaxe e semântica de linguagens. Inclusive, dessa obra extrai-se uma apresentação sobre as funções de mapeamento, em que $\mathcal{L}$ refere-se aos conceitos sintáticos da linguagem e $\mathcal{S}$ ao domínio semântico:

> Given a syntax $\mathcal{L}$ and a semantic domain $\mathcal{S}$, the final step in defining a semantics is to relate the syntactic concepts to those of the semantic domain. Each syntactic creature is mapped to some semantic element. [...] Often the mapping is explained informally, by examples and in plain English. But regardless of the degree of formality of its representation, the semantic mapping itself should be a function from $\mathcal{L}$ to $\mathcal{S}$, i.e.,
>
> $$\mathcal{M} : \mathcal{L} \rightarrow \mathcal{S}$$

Em Sebesta (2012, p. 142-147), um dos livros mais recentes sobre conceitos de linguagens de programação, encontra-se uma introdução à semântica denotacional, na qual o autor apresenta uma forma de traduzir sintaxe em objetos matemáticos, que representam o domínio semântico:

> *The mapping functions of a denotational semantics programming language specification, like all functions in mathematics, have a domain and a range. The domain is the collection of values that are legitimate parameters to the function; the range is the collection of objects to which the parameters are mapped. In denotational semantics, the domain is called the* ***syntactic domain****, because it is syntactic structures that are mapped. The range is called the* ***semantic domain*** (SEBESTA, 2012, p. 142).

A respeito da abordagem, Decker (1998) é um trabalho com um objetivo semelhante ao desta pesquisa, especialmente ao que se refere à construção de uma linguagem para

descrição de conceitos. Porém, o foco daquele autor é a criação de uma DSL cuja semântica é dada em termos de *Frame-Logic* e *Linear Temporal Logic*. O objetivo desta pesquisa é diferente, uma vez que deseja-se sustentar a abordagem de descrição no paradigma de Programação em Lógica. No entanto, a justificativa de seu trabalho é também válida para o caso desta pesquisa, bem como está alinhado com o escopo desta seção, ao menos no que tange à metáfora entre FOL e Máquina de Turing:

> *Logic based knowledge representation and specification has a long tradition in AI [...], so for knowledge acquisition and engineering tasks many representation formalisms are available. However, most of them do not support the elicitation and representation of domain specific knowledge. Using the syntax of First Order Predicate Logic (FOL) for knowledge representation tasks is comparable to using a Turing Machine for programming tasks: it is a sufficient formalism for studying theoretical properties, but does not support real life tasks. Of course a turing machine is powerful enough from a principal point of view, but not from a practical one: the encoding is often to expensive. The reason for this is, that the language (FOL) does not support thinking in domain terms. Instead, the domain terms must be translated to the terms of the representation language (e.g. relations). So to establish a communication between a knowledge engineer and a domain expert, the representation must always translated back to the domain terms. This, however is fortunately a matter of the syntax of the representation language. Therefore, similar to the development of high level programing languages, we propose to use the syntax of First Order Logic as a kind of low level language for building domain-specific declarative languages (DSDLs) (see [DSL97][9]), such that higher level representation languages can be compiled into this language.*

A abordagem de Decker é interessante porque, de forma simples e direta, propõe uma integração do poder expressivo de *Frame-Logic* (KIFER; LAUSEN, 1989), de *Chronolog*[10] (ORGUN, 1996) e uma tradução para FOL. A Tabela 3 contém um exemplo da tradução apresentada pelo autor no que toca expressões de Frame-Logic para FOL por meio da instaxe concreta de cláusulas de Horn. Trata-se de uma apresentação semi-formal e incompleta, mas que serve para exemplificar a abordagem do autor.

No que se refere à Prolog, sendo notável na definição de semânticas em linguagens de programação por meio de cláusulas de Horn, Gupta (1998)[11,12], apresenta diversos resultados nessa área. Ele defende o uso de lógica de Horn, ou mesmo *Constraint Logic*, ao invés de λ-cálculo para expressar semântica denotacional. A justificativa se resume ao fato de, segundo o autor, a abordagem lógica ser mais clara, no sentido de mais simples de

[9] *idem.*

[10] Trata-se de uma linguagem baseada numa lógica temporal de tempo linear construída como um banco de dados dedutivo.

[11] Gopal Gupta mantém o sítio <http://www.cs.nmsu.edu/~gupta/sem.html> com uma lista de artigos sobre cláusulas de Horn usadas para denotação de semântica em várias áreas de pesquisa.

[12] Outros artigos do autor sobre uso de Prolog para semântica de linguagens de Gupta: Gupta e Pontelli (1999), Gupta e Pontelli (2002). Em Pontelli et al. (2002) encontra-se um estudo de caso da abordagem de Gupta.

Tabela 3 – Esquema de tradução de Frame-Logic para cláusulas de Horn.

Object Oriented		First Order
`C::D`	⇒	`sub(C,D)`
`O:C`	⇒	`instance(O,C)`
`O[M->>V]`	⇒	`method(O,M,V)`
`C[M=>>D]`	⇒	`methodtype(C,M,D)`
`O:C[M->>V:D]`	⇒	`instance(O,C)` ∧ `method(O,M,V)` ∧ `instance(V,D)`

Fonte: Decker (1998)

ler, do que a abordagem comumente usada baseada em λ-cálculo além do fato de que, na abordagem lógica, a semântica é imediatamente executável:

> *Use of Horn Clause Logic to specify denotational semantics is based on the observation that a notation based on Horn Clauses (pure Prolog) and Constraints facilitates inference of program properties, program transformations, etc., much more easily than the Lambda Calculus. Constraints allow one to model time, as well as automatically derive parallelizing compilers from language specifications. Language Syntax can also be very easily specified in Horn Logic.*

Gupta (1998, p. 152) nomeia de *filtro* o sistema de tradução, ou a função de tradução, de uma especificação escrita em uma linguagem e traduzida para outra:

> *Logical denotations also provide a formal theory of* porting *or* language filtering. *Proting can involve migrating a software from one machine/-operation system to another (e. g. moving an application from a Sun Sparc running Solaris to a PC running Linux), or migrating a system written in one notation to another on the same machine (e. g. porting a database written in Oracle SQL to IBM DB2). It is the latter type of porting, that we are interested in. Essentially, this type of porting involves development of a semantic translation system to translate programs written in a language £$_s$ to another language £$_t$ (such a system is called a* filter*). Specification of a filter can be seen as an exercise in semantics. Essentially, the meaning or semantics of the language £$_s$ can be given in terms of the constructs of the language £$_t$.*

No caso, o o autor apresenta um exemplo de tradução da linguagem *Braille Nemeth Math* (AMERICAN ASSOCIATION OF WORKERS FOR THE BLIND, 1987), uma notação em Braille, para LaTeX. A tradução consiste na especificação de um filtro usando a especificação em DCG da linguagem Nemeth Brasille e a semântica especificada como árvores sintáticas produzida em termos de LaTeX. O resultado da especificação é totalmente executável, uma vez que ambas, sintaxe e semântica, são apresentadas em Prolog padrão.

No contexto de funções de tradução, Armstrong, Virding e Williams (1992) define o conceito de *semantic embedding*. Segundo o autor, trata-se dos casos em que a semântica

de uma determinada expressão é traduzida para Prolog e passa a significar o que quer que Prolog lhe dê significado. Desse modo, a semântica da linguagem subjacente, ou hospedeira (no caso, definidas para o próprio Prolog), se torna *"accidentally embedded"* na linguagem que está sendo definida. Esse tipo de semântica é especialmente útil, principalmente no que se refere à definição de linguagens internas, uma vez que permite compor constructos – ou padrões – semanticamente densos e sintaticamente simples que são traduzidos para a própria linguagem que servem de extensão.

3.3.3 Semântica formal de linguagens com Prolog

O uso de Prolog para definição de semântica e para implementação de linguagens, como alternativas às abordagens tradicionais ou não auxiliadas por computador, encontra adeptos há vários anos. Uma das obras mais antigas encontrada a esse respeito, Warren (1980), descreve a implementação de um compilador em Prolog de modo que esse compilador denote a semântica operacional da linguagem compilada por ele. Esse tipo de demonstração também é realizado por Cohen e Hickey (1987). Dez anos depois de Warren, os autores Bowen, Jifeng e Pandya (1990) também mostram que a especificação de um compilador (ou um interpretador) é similar a um programa em Lógica. Segundo os autores, isso justifica a naturalidade em realizar protótipos de compiladores em Prolog.

De fato, Prolog é usado não só na implementação de compiladores, mas especialmente na atribuição de semântica de linguagens. A estratégia de uso de Prolog para especificação de semântica de programas é defendida principalmente por Bryant e Pan (1989), que diversos tipos de semântica podem ser expressas com Prolog com as seguintes vantagens:

> *1) Because the notations used in formal semantics are based upon logic, it is straightforward to translate these into Prolog, thus making the conformity of the Prolog specifications with the original specifications more reliable;*
> *2) Specifications expressed in Prolog are executable and may therefore be considered as prototype implementations of the semantics; and*
> *3) Prolog exhibits the maximum amount of parallelism in the specifications, indicating where production quality systems might be implemented in parallel for improved performance.*
> *Finally, Prolog allows for a complete programming language specification which integrates all aspects of programming language semantics into a single unified framework.* (quebras de linhas nossas)

Na mesma linha de Bryant e Pan (1989), os pesquisadores Slonneger e Kurtz (1995) descrevem detalhadamente o tópico semântica denotacional e mostram o uso de Prolog como ferramenta de implementação não só desse tipo de semântica, mas também para gramática de dois níveis, *Attribute grammars*, semântica operacional, e outros tipos de semântica. A obra dos autores apresenta uma abordagem que eles chamam de *"a laboratory based*

approach", porque para cada tipo de abordagem, é apresentada a especificação equivalente em Prolog. Já Moss (1982) mostra uma abordagem completa de definição sintática e semântica de linguagens em Prolog com base em uma técnica chama *Metamorphosis Grammar* (M-grammar), que envolve semânticas relacionais e axiomáticas:

> *van Emden e Kowalski (1976) argue that the fixpoint semantics of the procedural interpretation of loqic is a special case of the model theoretic semantics. Also negation seen as finite failure may be regarded as the maximal fixpoint definition. However, these definitions are primarily for the experts. All that the practical language designer needs to remember is that a semantic definition is more than a program. It is a specification. The definitions presented here emphasize simplicity while leaving scope for others to develop the analytic techniques for transformation and proof. Two methods of semantics will be presented briefly: relational and axiomatic.*

Mais recentemente, em Wang, Gupta e Leuschel (2005) mostra-se que a notação de gramática de cláusulas definida (DCG) pode ser utilizada para especificar tanto a sintaxe quanto a semântica de linguagens imperativas, e também mostram que a semântica de continuação (*Continuation Semânticas*) também pode ser expressa pelas cláusulas de Horn. Já em Wang e Gupta (2005), a abordagem é exercitada na construção de uma DSL, cuja semântica é atribuída via técnica que utiliza DCG.

3.3.4 Linguagem interna e pré-processada

Em geral, toda *Application Programming Interface* (API) é uma linguagem específica interna, ou uma linguagem de propósito geral embutida na linguagem subjacente. Isso pode ser afirmado porque conhecer as interfaces de determinado programa é similar a conhecer um novo idioma: aspectos semânticos e sintáticos estão envolvidos tanto em uma simples uma API quanto em linguagens completas. Para mencionar um exemplo, no contexto de redes sociais digitais, um programador pode utilizar a API do Google+ (<https://plus.google.com>) para publicar uma mensagem, ou uma foto, na página de um usuário que ele tenha permissão. Publicar a mensagem é a própria semântica, resultado de um conjunto de operações sintaticamente realizadas por métodos, funções ou mensagens das interfaces da API em tela.

A respeito de *linguagens internas*, encontra-se em Fowler e Parsons (2010, p. 67):

> *Unlike external DSLs, you don't need to learn about grammars and language parsing, and unlike language workbenches, you don't need any special tools. With internal DSLs you work in your regular language environment. As a result, it's no surprise that there's been a lot of interest in internal DSLs in the last couple of years.*
> *When you use internal DSLs, you are very much constrained by your host language. Since any expression you use must be a legal expression in your host language, a lot of thought in internal DSL usage is bound up in language features.*

Mernik, Heering e Sloane (2005) apresenta um detalhado estudo sobre DSL, o que inclui critérios de avaliação de linguagens de programação em geral. Interessante observar que o autor apresenta mais formas de classificação de linguagens. A classe de DSL interna é também chamada de embutida (*embedding*), além de que ele destaca a classe *preprocessor*. Trata-se de um tipo de linguagem interna em que os construtores são traduzidos para construtores mais elementares na própria linguagem. Considerando essa classificação, a linguagem desenvolvida na Parte III é uma linguagem interna em Prolog. Porém, devido à estratégia de tradução da árvore sintática dos operadores Prolog, conforme detalhado no Capítulo 7, a linguagem poderia ser melhor classificada como *preprocessor*.

3.3.4.1 Características de uma DSL

Walter, Parreiras e Staab (2009) desenvolveram uma pesquisa sobre *Language-oriented programming* via DSL. Nesse trabalho eles apresentam uma lista de requisitos desejáveis nesse tipo de solução:

> ***Sufficient semantic expressiveness:*** *The DSL must allow to express logic necessary for the respective application domain directly in the language, e.g., rules in a knowledge-based system. This is necessary for implementing the application at all.*
>
> ***Conciseness and comprehensibility:*** *Statements should be expressed adequately, i.e., as concisely and comprehensibly as possible. Boilerplate code should be avoided. Inadequate representation has several drawbacks. Firstly, developing unnecessarily voluminous code increases the costs for developing the application. Even higher are the follow-up costs for maintenance. Secondly, the quality of the solution decreases since errors cannot be detected as easily. The more adequate the representation – i.e., the closer to the problem domain – the more problem domain experts can be involved in the development.*
>
> ***Integration:*** *It must be possible to integrate the DSL application with other applications, possibly implemented in other DSLs. This includes invocation at run-time and, ideally, integrated development including integrated debugging. Integration is mandatory if the DSL application is a component of a larger application, e.g., a business information system. Integrated development decreases development costs, particularly for testing and debugging.*
>
> ***Performance:*** *Applications implemented in the DSL must meet the respective performance requirements.*
>
> *Those aspects help the* application developer *(designer and implementer). Language-Oriented Programing adds another role: the* language developer *(designer and implementer). The following aspect helps the language developer.*
>
> ***Efficient DSL development:*** *The design, implementation including validation logic, testing, and maintenance of DSLs should be as efficient as possible. Efficient DSL development directly affects the total development cost.*

Essas características são importantes para orientar o desenvolvimento de linguagens, uma vez que refletem parâmetros de qualidade salutares.

3.3.5 Modelagem de linguagem de programação via DSL

A modelagem de linguagem de programação via DSL é um caminho natural na medida que instrumentos nas próprias linguagens hospedeiras estão disponíveis para extensão da linguagem base. Isso é cada vez mais comum em linguagens modernas, como o caso de Haskell (JONES, 2003; HUDAK; WADLER, 1990). Erwig e Walkingshaw (2012), para mencionar um exemplo, apresenta a definição de uma DSL interna ao Haskell, e realiza diversas discussões a respeito de semântica de linguagens construídas dessa forma, ao invés de se dar tratamento operacional de forma tradicional.

No âmbito de Prolog, Kosar e Mernik (2006) apresenta um detalhado texto sobre como desenvolver uma linguagem interna em Prolog usando operadores, e também sobre como definir sua semântica por meio cláusulas de Horn cláusula de Horn. Da mesma forma, Decker (1998) é um trabalho semelhante ao de Kosar e Mernik, ao menos em termos de atribuição de semântica a sentenças de primeira ordem via DSL, uma vez que usa uma abordagem similar às cláusulas Horn usadas em Prolog.

O uso de operadores provê representação e acesso direto às árvores sintáticas de sentença, de modo que a análise léxica de uma linguagem construída internamente em Prolog é realiza de forma automática. Por exemplo, a definição dos operadores `::`, `extends`, `extend`, `cover`:

```
:- op(500, yfx, [::, extends, extend, cover]).
```

permitem construir as seguintes sentenças sintaticamente válidas em Prolog:

```
carro :: kind.
carro extends veiculo.
fusca :: subkind extends carro.
[bicicleta, onibus] :: subkind extend veiculo.
[fusca_usado, fusca_zero] :: phase cover fusca.
```

Os operadores das sentenças são automaticamente convertidos para predicados, que são construídos conforme as prioridades da árvore sintática, de modo que podem ser representados por termos, conforme segue:

```
::(carro,kind).
extends(carro,veiculo).
extends(::(fusca, subkind), carro).
extend(::([bicicleta, onibus], subkind), veiculo).
cover(::([fusca_usado, fusca_zero], phase), fusca).
```

Uma vez convertidas as sentenças em termos, pode-se atribuir predicados semânticos, como nos casos abaixo, em que os conceitos de `instance/2` e `subsumption/2` são definidos à partir da árvore sintática produzida pela conversão dos operadores de Prolog:

```
instance(Instance, Meta) :- ::(Instance, Meta).
instance(Instance, Meta) :- extends(::(Instance, Meta), _).

subsumption(Particular, General) :- extends(Particular, General) .
subsumption(Particular, General) :- extends(::(Instance, _), General) .
```

Ressalta-se que mesmo no caso de uso de operadores Prolog, a definição semântica é necessária. No entanto, ela pode ser realizada no próprio Prolog, conforme apresentado no último código.

3.4 Abordagens similares

Esta seção discute brevemente abordagens que compartilham semelhanças com as características presentes neste trabalho referente à definição de linguagem formal subjacente a programas lógicos, ao tratamento de ontologias e ao uso de metamodelagem para descrição de conceitos.

3.4.1 Telos

Telos é um exemplo de linguagem declarativa para representação do conhecimento construída no final dos anos 1980 e início dos 1990 como uma proposta de formalização de redes semânticas (MYLOPOULOS et al., 1990; MYLOPOULOS, 1992):

> *In a nutshell, the contribution of Telos lies in its adaptation of ideas from knowledge representation, deductive databases and requirements modeling languages in order to offer a language that can be used to tackle a broader class of modeling tasks – arising from information system development tasks – than those attempted by other proposals* (MYLOPOULOS et al., 1990, p. 359)[13].

Telos é uma linguagem com alta expressividade, orientada a *frames*, que permite a declaração de classes, propriedades de classes como atributos de primeira ordem, *tokens* e relações. Como exemplos de relações, destacam-se relações de subsunção e instanciação (de uma classe por um *token*). A linguagem permite também a declaração de dados temporais, caso que inclui uma característica modal às instâncias de classes. Além disso, é possível a especificação de restrições (*constraints*) tanto referentes à cardinalidade e multiplicidade de relações quanto referentes às instâncias e outras relações formais das estruturas descritas na linguagem. Nesse caso, uma sintaxe com operadores que lembram uma linguagem de primeira ordem é usada. Uma característica interessante do Telos é a possibilidade de declarar instâncias em diferentes níveis. Ou seja, uma instância de uma classe pode ser uma classe para outra instância. Essa é uma característica semelhante às instâncias entre camadas da UML. A sintaxe proposta pela *Human-Usable Textual Notation (HUTN) Specification* (OBJECT MANAGEMENT GROUP, 2004), também baseada em *frames*, é semelhante à sintaxe do Telos.

Publicações referentes ao *Telos* foram analisadas nesta pesquisa de modo a se herdar algumas características daquela linguagem, como as propriedades básicas das relações de

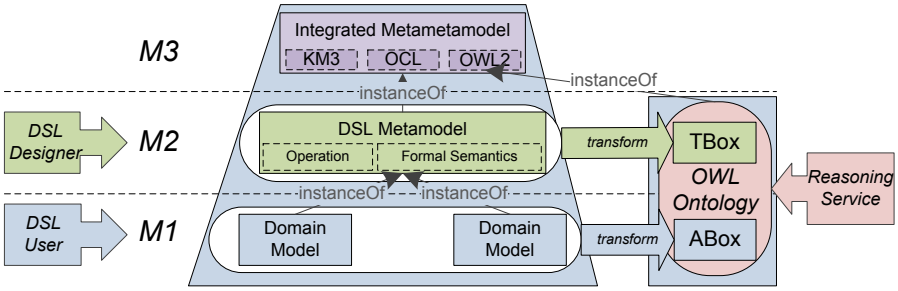
[13] Conforme a obra citada, um protótipo do *Telos* foi implementado em Prolog no final dos anos 1980.

subsunção e instanciação, e o fato de relacionamentos não serem elementos de primeira classe na linguagem. Embora tenha servido de referência, Telos não é aplicado diretamente nessa pesquisa porque os objetivos deste trabalho não seria completamente atendidos com essa ferramenta. Especialmente, as características modais necessárias à formalização de ontologias de forma colaborativa não poderiam ser descritas por meio das regras temporais modais embutidas em Telos, uma vez que deseja-se explorar semânticas modais com interpretações epistêmicas/doxásticas de vários agentes.

3.4.2 OntoDSL

No contexto de propostas de definição de semântica de linguagens, Walter, Parreiras e Staab (2009) e Walter (2009) desenvolvem uma proposta chamada *OntoDSL*. Segundo os autores, *OntoDSL* consiste em uma abordagem que permite o uso de ontologias para descrever DSL. Essencialmente, usam uma abordagem em camadas (M3), na qual na M1 estão as instâncias de modelos de domínio desenvolvidas por usuários da DSL. Na M2 estão as operações possíveis e a semântica formal da DSL. É nesta camada que o designer da DSL define efetivamente a sintaxe linguagem. Ambas as camadas, M1 e M2, são transformadas em descrições em Lógica Descritiva por meio de ABox e TBox, respectivamente. Por fim, na M3 concentra-se os *metametamodelos* usados para construir as demais camadas. Os autores propõem o uso de KM3 (JOUAULT; BÉZIVIN, 2006) – um subconjunto simplificado de MOF (OBJECT MANAGEMENT GROUP, 2014a) – para definir a estrutura sintática da linguagem, OWL2 (W3C, 2012) para definir sua semântica e OCL (ISO/IEC, 2012b) para as operações. A Figura 4 ilustra a abordagem dos autores.

Figura 4 – Camadas do OntoDSL



Fonte: Walter, Parreiras e Staab (2009)

Entre os pontos positivos que se pode destacar na abordagem, está a integração com as tecnologias correntes propostas pela W3C (OWL2, OCL), que possuem um vasto número de ferramentas e de fornecedores, o que é reflexo de grande aceitação no mercado. O principal ponto negativo talvez se refira a críticas à referentes aos requisitos impostos à OWL: apenas um fragmento decidível das lógicas são incorporadas as suas tecnologias, o que limita os resultados à apenas uma parte das variadas possibilidades oferecidas

pelos avanços teóricos da Lógica, muitos deles já disponíveis para uso, como banco de dados dedutivos paraconsistentes (PAIS, 2004; AMO; PAIS, 2007), Programação em Lógica Modal (NGUYEN, 1999; NGUYEN, 2001; NGUYEN, 2009), entre outras. Crítica semelhante é realizada por Störrle (2007b, p. 73) no que tange ao uso de OCL:

> *The OCL is somewhat more general in that, theoretically, it should fit to any UML model/tool combination. In practical settings, this is not true, however. Also, OCL is extremely difficult to read and write and there is very scarce tool support* (STÖRRLE, 2007b, p. 73).

A proposta do OntoDSL é justificada pelos criadores como uma abordagem que visa resolver desafios na construção de linguagens específicas de domínio. Baseado em Gray et al. (2008), os autores argumentam que são cinco os desafios principais na construção de DSL:

a) **Desafio 1**: Ferramentas (*"debuggers"*, ferramentas de teste);

b) **Desafio 2**: Interoperabilidade com outras linguagens;

c) **Desafio 3**: Semântica formal;

d) **Desafio 4**: Curva de aprendizagem;

e) **Desafio 5**: Análise de domínio.

Para atender o terceiro desafio, Walter, Parreiras e Staab mencionam que Guizzardi, Pires e Sinderen (2006) propõem o uso de *"upper ontologies"* para desenhar e avaliar conceitos de domínio. De fato, o artigo citado, que é baseado nos primeiros capítulos da tese do autor (GUIZZARDI, 2005), discorre sobre um método baseado em ontologia para avaliar *domain-specific visual modeling languages* (DSVL), que é um tipo de DSL focado em linguagens visuais. Para Guizzardi (2006), há pelo menos dois critérios importantes a serem considerados na avaliação de linguagens desse tipo:

> *(i) it should easy for a user of the language to communicate, understand and reason with the produced models (comprehensibility appropriateness); (ii) The language should be truthful to the domain in reality that it is supposed to represent (domain appropriateness).*

No contexto de DSVL, do ponto de vista pragmático a separação entre sintaxe e semântica é muito mais tênue do que no caso de linguagens baseadas em frases, uma vez que aquelas, idealmente, devem embutir na representação concreta visual aspectos semânticos em relação ao domínio que se planeja modelar. Ou seja, no caso das DSVL, a sintaxe visual pode explorar a capacidade de informar o usuário sobre aspectos semânticos do domínio que aquela linguagem se presta a descrever. Essa característica, quando explorada cuidadosamente de modo a considerar símbolos significativos e não ambíguos, possui a vantagem de melhorar a compreensão das construções realizadas. Por outro lado, as

linguagens visuais, quando projetadas sem considerar o domínio específico e o significado que os símbolos possam induzir, possuem o risco de produzir significados indesejados e ambíguos.

3.4.3 MoMaT

Há diversos trabalhos que investigam a verificação de modelos UML e de OCL em Prolog (PAP; PATARICZA; SZEGI, 2001; MOTA et al., 2003; CHIMIAK-OPOKA et al., 2008; KHAI; NADEEM; LEE, 2011; ZARETSKA et al., 2012). Para apresentar um exemplo, em Störrle (2007b) e em Störrle (2007a) desenvolve-se um sistema batizado de *Model Manipulation Toolkit* (MoMaT):

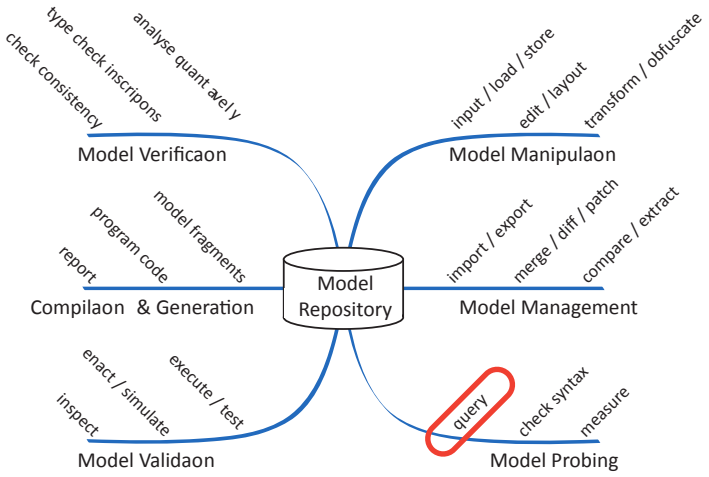
> *Over the last years, we have created a system called the Model Manipulation Toolkit (MoMaT) which allows us to experiment with models in general. In MoMaT, models are imported into a Prolog-based model repository by a set of transformers for several modeling languages like UML, ARIS/EPC, and Use Case Maps (in this paper, we will only focus on UML models, though)* (STÖRRLE, 2007b, p. 72).

O autor propõe um modelo de representação em Prologe uma interface de consulta para análise de modelos na abordagem MDD (*Model Driven Development*). No MoMat, modelos e consultas são representados em cláusulas padrões Prolog. Pode-se realizar consultas sobre essas bases de fatos de modo a identificar elementos, propriedades e submodelos. As representações do autor são em forma de listas de mapas de classes, propriedades e operações de classes. A proposta é interessante, porque permite a identificação de diferentes versões dos objetos, numa semântica temporal. Uma característica do MoMat é a possibilidade de se trabalhar com diferentes tipos de modelos, e não apenas modelos baseados nos padrões da OMG ou W3C. Para isso, uma etapa de conversão de modelos padroniza a forma dos dados para fatos Prolog, de modo que esse padrão funcione como semântica comum para os diferentes modelos. A Figura 5 ilustra esse processo.

Já a Figura 6 apresenta uma visão esquemática do MoMat, em que se destacam as funcionalidades da ferramenta, salientada a capacidade de realização de consultas.

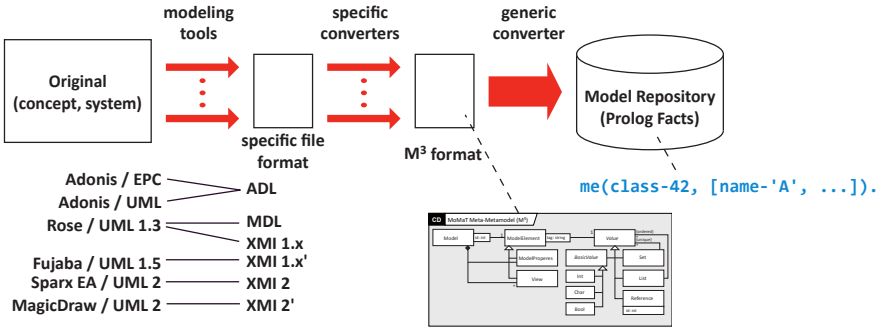
A abordagem geral realizada no MoMat serviu de referência para o desenvolvimento do Ontoprolog, especialmente no que se refere à representação de dados de modelos baseados em Programação em Lógica, e as características de realização de consultas à base de fatos que representa os discursos sobre ontologias.

Figura 5 – Classes de operações sobre diagramas potencialmente relevantes para modelagem industrial



Fonte: Störrle (2007b, p. 72)

Figura 6 – Visão esquemática do MoMaT



Fonte: Störrle (2007b, p. 72)

3.5 Estilos de Programação em Lógica

Embora seja uma linguagem quase quadragenária, apenas em 2012 foi publicado um artigo inteiramente dedicado a sugerir um estilo consistente de codificação para Prolog. O artigo referido trata-se do trabalho de Covington et al. (2012), que é a evolução de um rascunho não publicado, mas disponível online (COVINGTON, 2002), escrito uma década antes. O texto é um guia de estilos de orientações gerais de codificação em Prolog e consiste em sugestões de boas práticas baseadas na experiência dos próprios autores e também baseadas no clássico livro estilos de programação de Kernighan e Plauger (1978) – que apresenta exemplos em PL/I e em Fortran – e em uma obra mais recente do autor (KERNIGHAN; PIKE, 2009) – em que já são apresentados exemplos em Java, embora não seja um livro dedicado exclusivamente aos estilos de programação. O guia de Covington et al. é composto por 17 sugestões referentes a layout do código, 14 referentes a convenção de

nomes, 5 sobre documentação, 13 dedicadas a detalhes da linguagem e 18 gerais, referentes ao desenvolvimento, teste e depuração.

Os estilos de programação são importantes porque simplificam a leitura do código por tentar impor um determinado modo de codificação que não desvie o foco de um leitor da lógica e da linha de raciocínio denotadas pelo programa. Porém, a definição de regras de estilos de programação trata-se também de arte, e não somente de ciência, uma vez que não é possível estabelecer um padrão único, pré-determinado, adequado para todas as situações ou claro o suficiente para que seja possível apresentar argumentos puramente lógicos que justifiquem sua escolha. Nesse sentido, por envolverem arte e ciência, poderia ser argumentado que se tratam de um aspecto de "arquitetura"[14], no caso, algo como a "arquitetura do estilo de programação".

Todos os códigos Prolog presentes neste texto, e componentes dos softwares produzidos nesta pesquisa (Capítulo 10), são produzidos considerando as sugestões de Covington et al. (2012), embora não necessariamente se atenda a todas as recomendações dos autores. Quando isso não acontece, fica evidenciado o fragmento não científico inerente ao arbítrio da proposta.

Opta-se por não incluir neste texto as sugestões em si dos autores, uma vez que tratam-se de demasiados detalhes sobre os quais não se pretende dissertar, apenas seguir e cumprir conformidade. De toda forma, as sugestões se presam ao objetivo de ter um padrão de qualidade a ser seguido. Além disso, o artigo referido está disponível online, conforme entrada das Referências.

3.6 Fechamento

Destaca-se que no contexto da seção 3.1, a subseção 3.1.2 apresenta definições, baseadas em Lloyd (1993), que visam introduzir a anotação das formulações lógicas utilizada no restante deste trabalho. Essas definições são realizadas com intuito de construir uma ligação da notação padrão de Lógica Clássica de Primeira Ordem com a notação comumente utilizada na Programação em Lógica, baseada em cláusulas de Horn.

A *semantic embedding*, proposta para Prolog por Armstrong, Virding e Williams (1992), abordada na seção 3.3, é o tipo de semântica utilizada neste trabalho, conforme mencionado na seção 6.1. Isso ocorre porque utiliza-se as semânticas atribuídas a programas Prolog, como a semântica baseada em modelos de Herbrand, como significado das relações do universo semântico das sentenças sintáticas de Ontoprolog.

O nome *função filtro*, usado no título da seção 7.3, é obtido de Gupta (1998),

[14] Argumentos que objetivam justificar a ideia de arquitetura como (a intersecção entre) arte e ciência é apresentado, por exemplo, por Costa (2003, p. 19-21), Winters (2007, p. 162), Castelnou (2011) e outros arquitetos citados em Araujo (2012, p. 65-74).

também abordado na seção 3.3. Porém, não apenas o título, mas a abordagem do autor para atribuição de semântica de sintaxe de linguagem é de fato a utilizada nas definições do significado da sintaxe de Ontoprolog em termos relações semânticas de Prolog puro, conforme descrito naquela seção.

A estratégia de Kosar e Mernik e de Decker é a adotada neste trabalho no que se refere à definição de Ontoprolog como uma DSL interna em Prolog construída por meio de operadores. A semântica dessas construções via operadores é dada como tradução (filtro) para o domínio semântico da linguagem Prolog (*semantic embedding*), o que inclui a *semântica de base de dados* (subseção 3.1.4), conforme detalhado na Parte III.

Quando a MProlog, de modo geral seu é justificado no caso de prototipagem de lógicas que estabeleçam relações de dedução modal. O arcabouço lógico apresentado brevemente na seção 3.2 é utilizado como sustentação de lógicas subjacentes à extensão modal da linguagem de especificação de ontologias produzida nesta pesquisa. Isso é realizado no Capítulo 9.

4 Ontologia de fundamentação UFO

Este capítulo descreve o fragmento da *Unified Foundational Ontology* (UFO) utilizada como ontologia subjacente à definição da linguagem descrita na Parte III. O capítulo contém os resultados de levantamento bibliográfico exploratório realizada em algumas das dezenas de obras[1] do autor da UFO e de autores que colaboram com ele a respeito dessa ontologia de fundamentação. A revisão em tela tem o objetivo estrito de obter as regras lógicas que ontologistas devem atender ao formalizar discursos sobre ontologias de domínio em que a base meta-teórica seja a ontologia de fundamentação proposta por Guizzardi (2005). Dessa forma, este capítulo provê um marco teórico referente à ontologia de fundamentação em tela, tendo como foco a sustentação teórica para construção de linguagens que tenham a UFO como ontologia subjacente. Por esse objetivo, estuda-se a linguagem OntoUML[2], que é um exemplo de linguagem construída tendo a UFO como ontologia subjacente, com intuito de identificar as regras que a modelagem deve observar.

Não é foco deste capítulo, ou deste trabalho em si, descrever os argumentos utilizados por Guizzardi na construção de sua ontologia de fundamentação, pois a obra original já o faz de forma apropriada para seus objetivos. Tampouco é escopo deste texto destacar as alterações que ocorreram na UFO no decorrer do tempo – embora estejam presentes alguns comentários nesse sentido – ou prover qualquer análise filosófica ou teórica do tema que não esteja alinhada com o fim específico de identificar as regras a serem usadas para construção de linguagens de especificação de ontologias. Por isso, não é objetivo realizar uma avaliação crítica da UFO. Além disso, com intuito de tornar o texto o mais direto possível, furta-se inclusive de se reescrever apresentações das categorias ontológicas propostas pela UFO. Para isso, além da obra já mencionada de referência principal de Guizzardi, um texto mais conciso está presente em Guizzardi (2007b). Além desse, o trabalho de Zamborlini (2011, p. 24-47) apresenta uma resumida introdução ao fragmento da UFO que é escopo desta tese, a UFO-A. Nas páginas citadas, o autor se dedica a introduzir a ontologia de fundamentação por meio de descrições conceituais sobre as principais categorias da UFO e abusa de exemplos didáticos. A obra da autora está disponível online e, diante de sua clareza, assume-se que a introdução à UFO, especialmente a UFO-A, pode ser realizada a partir do trabalho citado e, por isso, obstem-se de repetir introdução semelhante neste trabalho. Dessa forma, ao invés de apresentações e exposições em prosa, os conceitos da UFO trabalhados no marco teórico deste capítulo são sistematizados e sumarizados em forma de regras presentes na subseção 4.5.1 e de verbetes definidos no Glossário da UFO, que deve ser lido em conjunto com este capítulo. Referências aos demais fragmentos da

[1] A lista completa de obras consideradas estão indicadas nas Referências.
[2] GUIZZARDI, op. cit.

UFO são listados na seção 4.3.

Conforme já estabelecido na subseção 1.4.4.1, e a exemplo de Zamborlini (2011, p. 29), a maioria dos termos referentes a conceitos específicos das ontologias tratados nesta seção são mantidas em inglês, embora o Glossário da UFO apresente, entre parênteses, sugestões em nomes em português para os referidos termos.

O restante deste capítulo é organizado do seguinte modo: A seção 4.1 contém uma brevíssima apresentação sobre o conceito de "ontologia", *modelagem conceitual* e *ontologia de domínio*. A seção 4.2 inicia o capítulo com uma breve apresentação sobre classificações de ontologias, e tem o objetivo de construir um conceito sobre "ontologia de fundamentação". Apesar das ponderações realizadas no parágrafo anterior, a seção 4.3 consiste em uma apresentação da UFO e do Glossário da UFO presente na página 437, que contém a descrição terminológica e conceitual da ontologia. Como preparação para a próxima seção, a seção 4.4 aborda tópicos técnicos utilizados na modelagem de ontologias com base na UFO e na OntoUML. Tratam-se de informações preliminares que devem ser lidas em conjunto com as definições do Glossário da UFO. A seção 4.5 contém a parte central do capítulo, que refere-se a uma sistematização do marco teórico da ontologia em tela. Tratam-se de quadros que sumarizam as regras a serem observadas na construção de especificações de discursos ontológicos por meio da UFO, ou seja, de restrições que linguagens baseadas na UFO devem observar. Os quadros são referenciados no Apêndice C, que contém a formalização das regras informais apresentadas nessa seção já na linguagem desenvolvida como resultado desta pesquisa. Por fim, a seção 4.6 apresenta um breve fechamento deste capítulo.

4.1 Sobre "ontologia"

Esta brevíssima seção apresenta alguns achados bibliográficos referentes a definições e explicações sobre o termo "ontologia". Inicia-se com acepções encontrados em dois dicionários de filosofia. Para uma longa e mais aprofundada descrição, a entrada "ontologia" pode ser consultada em Mora (1994, p. 2142-2149).

a) Durozoi e Roussel (1993, p.349):

> **ôntico, ontologia, ontológico**. Etimologicamente, o adjetivo **ôntico** qualifica o que se relaciona com o ser. Porém, a filosofia atual, depois de Heidegger, o emprega sobretudo para designar o que se refere aos entes.
> Em filosofia clássica, a **ontologia** é a ciência do ser em geral, ou seja, do "ser por onde ele é ser" (Aristóteles): equivalente da metafísica ou "filosofia primeira". Na Filosofia contemporânea, a palavra designa o estudo ou as concepções da existência em geral, como as encontramos principalmente nas diferentes versões do existencialismo. [...]

b) Abbagnano (2007): a entrada "ontologia" remete ao verbete "metafísica"[3], que apresenta uma ampla visão histórico-filosófica que aborda a natureza amálgama de "ontologia" com metafísica. Ao invés de incluir excertos da obra neste texto, indica-se consulta à obra original. De toda forma, uma breve citação referente ao verbete *ôntico* é apresentada na sequência:

> **ôntico**. Existente: distinto de ontológico, que se refere ao ser categorial, isto é, à essência ou à natureza do existente. P. ex., a propriedade empírica de um objeto é uma propriedade O.; a possibilidade ou a necessidade é uma propriedade ontológica. Essa distinção foi ressaltada por Heidegger: " 'Ontológico', no sentido dado à palavra pela vulgarização filosófica (e aqui se mostra a confusão radical) significa aquilo que, ao contrário, deveria ser chamado de O., ou seja, uma atitude tal em relação ao ente que o deixe ser em si mesmo, no que é e como é. Mas nem por isso se propôs ainda o *problema do ser*, e muito menos se atingiu aquilo que deve constituir o fundamento para a possibilidade de uma 'ontologia' " Ibid., p.727.

Num contexto contemporâneo de "ciências de informação", como aquelas vinculadas à organização e distribuição de informação, o termo "ontologia" têm sido usado como quase sinônimo de modelagem "semântica" representada em determinada linguagem. Nessa linha, ressalta-se a definição de ontologia de Gruber (2009):

> *In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application. In the context of database systems, ontology can be viewed as a level of abstraction of data models, analogous to hierarchical and relational models, but intended for modeling knowledge about individuals, their attributes, and their relationships to other individuals. Ontologies are typically specified in languages that allow abstraction away from data structures and implementation strategies; in practice, the languages of ontologies are closer in expressive power to first-order logic than languages used to model databases. For this reason, ontologies are said to be at the "semantic" level, whereas database schema are models of data at the "logical" or "physical" level. Due to their independence from lower level data models, ontologies are used for integrating heterogeneous databases, enabling interoperability among disparate systems, and specifying interfaces to independent, knowledge-based services. In the technology stack of the Semantic Web standards, ontologies are called out as an explicit layer. There are now standard languages and a variety of commercial and open source tools for creating and working with ontologies.*

Apresenta-se, na sequência, excerto de Guizzardi et al. (2011) que aborda a posição dos autores e de alguns de seus colaboradores a respeito de três conceitos, a saber *Modelagem*

[3] ABBAGNANO, op. cit., p. 660-667.

conceitual, *ontologias* e *ontologia de domínio*. A citação é importante, porque realça com louvável clareza a posição do discurso científico e filosófico em que a pesquisa dos autores está posicionada e que, por conseguinte, é a adotada neste trabalho. Grifos nossos:

> Em *modelagem conceitual* e áreas afins (ex. Modelagem organizacional), o termo é usado de acordo com sua definição original em filosofia, a saber, como uma referência a um sistema formal e filosoficamente bem-fundamentado de categorias que pode ser usado para articular conceituações e modelos em domínios específicos do conhecimento. Em contraste, nas outras áreas supracitadas, o termo ontologia tem sido usado como: (i) artefato concreto de engenharia, projetado com um propósito específico e sem prestar nenhuma ou quase nenhuma atenção a aspectos teóricos de fundamentação; (ii) modelo de um domínio específico do conhecimento (ex., biologia molecular, finanças, logística, doenças infecciosas) expresso em uma linguagem de representação do conhecimento (ex., RDF, OWL, F-Logic) ou modelagem conceitual (ex., UML, EER, ORM).
> *Ontologias*, no sentido filosófico, têm sido desenvolvidas em filosofia desde Aristóteles com sua teoria de Substância e Acidentes e, mais recentemente, várias dessas teorias têm sido propostas sob o nome de Ontologias de Fundamentação (Foundational Ontologies). Desde o fim da década de oitenta, observa-se um crescente interesse no uso dessas ontologias de fundamentação no processo de avaliação e (re)engenharia de linguagens de modelagem conceitual. A hipótese inicial, e que foi posteriormente confirmada por várias evidências empíricas, pode ser explicada através da seguinte argumentação: (i) modelos conceituais são artefatos produzidos com o objetivo de representar uma determinada porção da realidade segundo uma determinada conceituação; (ii) Ontologias de Fundamentação descrevem as categorias que são usadas para a construção dessas conceituações. Podemos, portanto, concluir que **uma linguagem adequada de modelagem conceitual deveria possuir primitivas de modelagem que refletissem as categorias conceituais definidas em uma Ontologia de Fundamentação.**
> Uma *ontologia de domínio*, no sentido usado pelas demais comunidades em computação, é um tipo particular de modelo conceitual. Em particular, é um modelo conceitual que deve satisfazer o requisito adicional de servir como uma representação de consenso (ou modelo de referência) de uma conceituação compartilhada por uma determinada comunidade. Portanto, se uma ontologia de domínio é, antes de qualquer coisa, um modelo conceitual, uma linguagem adequada para representação de ontologias de domínio deve satisfazer os requisitos gerais de uma linguagem adequada para modelagem conceitual, ou seja, deve ter como teoria subjacente uma ontologia de fundamentação. Em outras palavras, ontologias (no sentido adotado em filosofia e em modelagem conceitual) representam ferramentas conceituais de importância fundamental para a criação de ontologias de domínio de qualidade (no sentido adotado nas demais áreas).

Em Almeida (2014) encontra-se interessante, embora breve, estudo sobre conceitos de "Ontologia" e "ontologias" no âmbito da Ciência da Informação, Ciência da Computação e Filosofia. A Tabela 4 sintetiza o estudo de diferentes usos dos termos nas áreas mencionadas realizadas pelo autor. Essas posições conceituais sobre *Modelagem conceitual*, *ontologias* e *ontologia de domínio* servem de base para análise da ideia de *ontologia de fundamentação*, conforme apresentado pela seção 4.2.

Tabela 4 – Quadro de visões sobre Ontologia e ontologias

Distinção	Campo	O que é?	Propósito	Exemplo
Ontologia como uma disciplina	Filosofia	Ontologia como um sistema de categorias	Entender a realidade, as coisas que existem e suas características	Sistemas de Aristóteles, Kant, Husserl
Ontologia como um artefato	Ciência da Computação	ontologia como uma teoria (baseada em lógica)	Entender um domínio e reduzi-lo à modelos	BFO, DOLCE (genéricas)
		ontologia como um artefato de software	Criar um vocabulário para representação em sistemas e para gerar,inferências	OWL (linguagem de RC)
	Ciência da Informação	ontologia como uma teoria (informal)	Entender um domínio e classificar termos	Sistema de classificação de Ranganathan
		ontologia como um sistema conceitual informal	Criar vocabulários controlados para recuperação da informação a partir de,documentos	um catálogo, um glossário, um tesauros

Fonte: Almeida (2014, p. 252)

4.2 Ontologia de fundamentação

Antes de se adentrar no marco teórico da UFO em si, esta seção busca responder esta pergunta: o que é uma ontologia de fundamentação? Para respondê-la, apresenta-se uma proposta abrangente de classificação de ontologias realizada por Oberle (2006, p. 43-49). Segundo o autor, as ontologias podem ser classificadas com base em três parâmetros, conforme seguem.

Ontologias classificadas quanto:

a) ao **propósito**: que podem ser classificadas como

   – *aplicação*: usada durante a execução de uma modelagem específica de um universo de discurso;

   – à *referência*: usada durante o desenvolvimento de uma ontologia de aplicação. Serve para entendimento mútuo entre homens e máquinas, entre agentes de diferentes comunidades, para estabelecer consenso em uma comunidade ou simplesmente para explicar o significado de um termo para alguém novo à comunidade.

b) à **expressividade**:

   – *pesada (heavyweight)*: ontologias que tentam explicar o significado de um vocabulário o mais preciso possível. A motivação inicial desse tipo de ontologia é possibilitar o entendimento mútuo em ambientes heterogêneos. São extensivamente axiomatizadas, de modo a obter um compromisso ontológico

explicitamente. O propósito primeiro é eliminar ambiguidades;

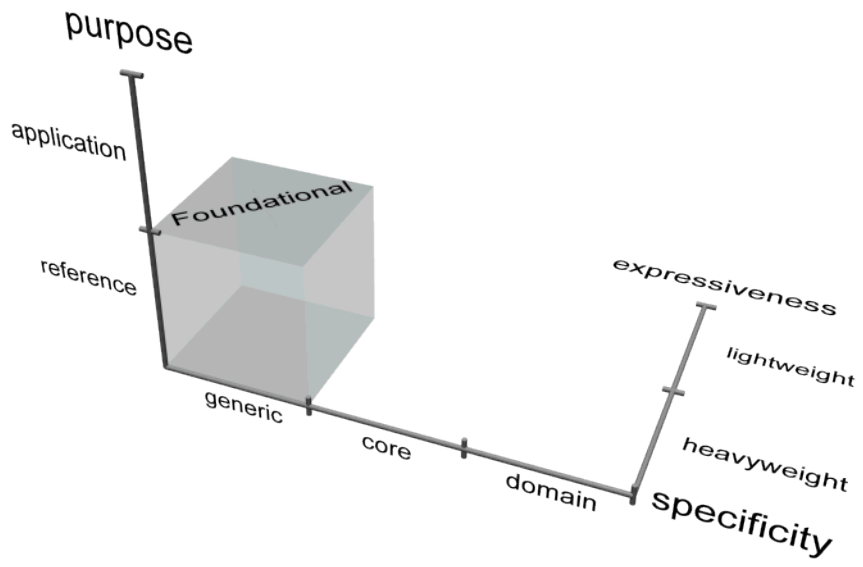
- *leve (lightweight)*: ontologias leves podem consistir de um conjunto mínimo de axiomas escritos em uma linguagem de expressividade limitada. Ontologias desse tipo geralmente suportam apenas um conjunto limitado de significados, e visam o compartilhamento de conceitos entre usuários que já concordam com uma conceituação base;

c) **especificidade**:

- *genérica (generic)*: ontologias que definem conceitos genéricos em diversos campos de conhecimento. Ontologias genéricas, "*upper level*" ou "*top-level*" geralmente definem conceitos como estado, evento, processo, ação, componente, etc.;
- *central, principal ou nuclear (core)*: ontologias nucleares definem conceitos que são genéricos em um determinado conjunto de domínios. Desse modo, elas estão situadas entre os extremos das ontologias genéricas e das de domínio. As fronteiras dessas ontologias não são claras, uma vez que geralmente não enumeram exatamente os limites ou áreas de conhecimento a que se referem.
- *domínio (domain)*: ontologias de domínio são específicas de um universo de discurso. Geralmente os conceitos das ontologias de domínio são especializações das ontologias nucleares e genéricas.

A Figura 7 sumariza a classificação de ontologias proposta por Oberle (2006).

Figura 7 – Classificação de ontologias e o papel das ontologias de fundamentação



Fonte: Oberle (2006, p. 49)

Com base na classificação apresentada, sobre ontologias de domínio Oberle (2006, p. 48) afirma que: *"They are* heavyweight*, i.e., extensively axiomatized and* generic*, i.e., domain-independent. Their main purpose is to have a concise* reference *at development time."* O autor continua e escreve que *"However, lightweight versions of a foundational ontology can also be used for reasoning at run time."* Isso significa que, embora geralmente as ontologias de fundamentação são descritas em linguagens lógicas muito expressivas – como lógicas de primeira ordem e utilizando conceitos modais – durante o processo de produção de uma ontologia específica de um domínio geralmente são usadas ontologias mais leves, e descritas em lógicas decidíveis. Isso faz com que as ontologias de fundamentação se posicionem como ontologias *genéricas*, no sentido que servem a diferentes domínios, porém, se posicionam entre ontologias de *referência* e de *aplicação*, uma vez que possuem o compromisso de servirem de referência mas ao mesmo tempo possuem um propósito de serem auxiliares à produção de ontologias de aplicação. Da mesma forma, o número de axiomas dessas ontologias de fundamentação devem ser suficientes e expressivos, mas não ao ponto de deixarem de ser úteis, compreensíveis e, idealmente, logicamente decidíveis, características que as levam a se posicionarem entre as ontologias *leves* e as *pesadas*, conforme descrito na Figura 7.

Por fim, Scherp et al. (2011, p. 20-21) sumariza a noção de ontologias de fundamentação e de ontologias nucleares do seguinte modo (mantivemos as referências originais):

> *Foundational ontologies are generic across many fields (OBERLE, 2006). They have a large scope and are highly reusable in different modeling scenarios (BORGO; MASOLO, 2008). Thus, foundational ontologies serve reference purposes (OBERLE, 2006) and aim at modeling the very basic and general concepts and relations (OBERLE, 2006, 2006) that make up our world, e.g., objects, events, participation, and parthood. Foundational ontologies are heavyweight as they are rich in axiomatization (BORGO; MASOLO, 2008), precisely defining the concepts in the ontology and their relations (OBERLE, 2006). Synonyms of the term foundational ontology are generic ontology, upper level ontology, and top-level ontology (EUZENAT; SHVAIKO, 2007; OBERLE, 2006)*[4].
> [...] *In contrast to foundational ontologies that span across many fields and model the very basic and general concepts and relations (OBERLE, 2006, 2006) that make up our world, core ontologies provide a detailed abstract definition of structured knowledge in one of these fields, e.g., medicine, law, software services, personal information management, multimedia annotations, and others. By their nature, foundational ontologies are much broader than core ontologies.*

A respeito da comparação entre ontologias de referência e ontologias leves, Guizzardi (2007b, p. 16) apresenta o seguinte:

> *Contrary to reference ontologies, lightweight ontologies are not focused on representation adequacy but are designed with the focus on guaranteeing*

[4] (EUZENAT; SHVAIKO, 2007, p. 68)

> *desirable computational properties. Examples of languages suitable for lightweight ontologies include OWL and LINGO. An example of an ontology representation language that is suitable for reference ontologies is the UML profile [...] (as demonstrated in Guizzardi (2005)). [...] Finally, a phase is necessary to bridge the gap between the conceptual modeling of references ontologies and the coding of these ontologies in terms of specific lightweight ontology languages. Issues that should be addressed in such a phase are, for instance, determining how to deal with the difference in expressivity of the languages that should be used in each of these phases, or how to produced lightweight specifications that maximize specific non-functional requirements (e.g., evolvability vs. reasoning performance).*

Ainda sobre a obra Guizzardi (2007b), o artigo é introduzido com a definição de "ontologia" obtida do dicionário idiomático *The Webster* o qual, segundo o autor, apresenta três definições para o termo[5]:

> *(D1). a branch of metaphysics concerned with the nature and relations of being;*
> *(D2). a particular theory about the nature of being or the kinds of existents;*
> *(D3). a theory concerning the kinds of entities and specifically the kinds of abstract entities that are to be admitted to a language system.*

Essas definições de "ontologia" são mencionadas nas conclusões do artigo, que resumem o posicionamento conceitual do autor a respeito de temas como *ontologia formal* e *ontologia de fundamentação*. Apresenta-se a tradução de um excerto dessas conclusões:

a) *Ontologia formal*, no modo como concebido por Husserl, é parte da disciplina de Ontologia na filosofia (no sentido D1), a qual, por sua vez, é o mais importante ramo da metafísica;

b) Ontologia formal almeja o desenvolvimento de teorias gerais que tratem aspectos da realidade que não são específicos de nenhum campo da ciência, seja ela física ou de modelagem conceitual (sentido D2);

c) Essas teorias descrevem o conhecimento sobre a realidade de modo independente de linguagem, de determinados *states of affairs* (estados do mundo), ou de estados epistêmicos de agentes. Essas linguagens independentes de teorias são chamadas (meta)*conceituações*. A representação dessas teorias em um artefato de concreto é uma *ontologia de fundamentação*;

d) Uma Ontologia de Fundamentação é uma Ontologia de Referência independente de domínio. Ontologias de Referência tentam caracterizar, o mais precisamente possível, a conceituação que elas são comprometidas;

[5] Nas referências da obra citada, o autor esclarece que a consulta foi realizada online no sítio <www.m-w.com> em 2004. Refizemos a consulta no mesmo endereço em 26.10.2014, e o resultado não apresentou a definição D3.

e) Ontologias de fundamentação, em sentido filosófico, podem ser usadas para prover semântica de mundo real para *linguagens de representação de ontologias* em geral e para restringir as interpretações possíveis das primitivas dessas linguagens. Do mesmo modo, uma adequada linguagem de representação de ontologias devem se comprometer com um sistema de distinções ontológicas (no sentido filosófico), ou seja, devem realmente pertencer ao *Nível Ontológico*;

f) Uma ontologia pode ser entendida como uma especificação de um metamodelo para uma linguagem ideal com intuito de representar fenômenos de um determinado domínio da realidade, ou seja, uma linguagem que apenas admita representações possíveis de *state of affairs* reais (relacionado com o sentido D3);

g) Linguagens gerais de podem ser usadas no desenvolvimento de ontologias de referência de domínio, o que, entre outros propósitos, pode ser usado para caracterizar o vocabulário de uma *linguagem de domínio específico*.

Com base nisso, pode-se entender que ontologias de fundamentação são ontologias não específicas a um único domínio, no sentido que as distinções categóricas que provêm são genéricas mas também servem de referência para diferentes áreas de aplicação e, embora o poder expressivo da ontologia de fundamentação seja limitado, ainda é expressiva o suficiente para que se possa descrever diferentes domínios com base nas categorias fundamentais de referência providas por ela.

Com essa visão de ontologias de fundamentação, a seção 4.3 descreve a *Unified Foundational Ontology* (UFO), foco deste capítulo.

4.3 *Unified Foundational Ontology* (UFO)

Em seu seminal trabalho de doutorado, Guizzardi (2005, p. 382) propõe uma ontologia de fundamentação "centrada nas categorias ontológicas de endurantes[6] e de endurantes universais", criada com o propósito específico de servir como uma teoria base para modelagem conceitual. Trata-se de um desenvolvimento com abordagem interdisciplinar de Ontologia Formal, Lógica Filosófica, Linguística e Psicologia Cognitiva[7] que é parte de um programa maior de ontologias de fundamentação, chamadas de *Unified Foundational Ontology* (UFO). Conforme Guizzardi e Wagner (2005b), Guizzardi e Wagner (2005a) e

[6] Do inglês *endurants* (Endurant).

[7] Quanto à abordagem cognitiva, estratégia adotada na UFO, Gangemi et al. (2001) explica a ideia com um exemplo de constelação: uma constelação é uma entidade ontológica genuína ou é apenas um arranjo de estrelas? Quando se vê o arranjo de estrelas e o chama de constelação de Touro, por exemplo, não se está assumindo que existe um touro no céu. Porém, a ideia de poder nomear um conjunto de estrelas consiste em reconhecer a existência de "entidades cognitivas", dependente do modo de como determinados arranjos de estrelas são vistos. Então, segundo Gangemi et al. (2001, p. 23-24), *"Including cognitive entities in our ontology seems therefore a good idea if natural language plays a relevant role in our applications.*

Guizzardi e Wagner (2010, p. 175), UFO é derivada de uma síntese e do aperfeiçoamento de duas outras ontologias de fundamentação: GFO/GOL (DEGEN et al., 2001; DEGEN et al., 2011) (ver <http://www.onto-med.de/>) e OntoClean/DOLCE (WELTY; GUARINO, 2001; GUARINO; WELTY, 2002; GUARINO; WELTY, 2008), mas é atualmente desenvolvida de forma independente daquelas.

A UFO consiste em um sistema de categorias que reflete características próprias da realidade, como relações parte-todo, princípio de identidade, diferentes aspectos de dependência e relações entre objetos, entre outras. A UFO é organizada em três camadas:

a) **UFO-A** (*Ontology of Endurants*): uma ontologia de endurantes (Endurant), ou de objetos sem partes temporais, que persistem no tempo e mantêm sua identidade. A principal referência sobre UFO-A é a referida tese de Guizzardi, mas também alguns trabalhos posteriores como Guizzardi (2006), Guizzardi (2009), entre outros;

b) **UFO-B** (*Ontology of Perdurants*): uma ontologia que incrementa a UFO-A com a introdução da noção de perdurantes (Perdurant), caracterizados como eventos com duração delimitada, compostos de partes temporais (GUIZZARDI; FALBO; GUIZZARDI, 2008; GUIZZARDI et al., 2013);

c) **UFO-C** (*Ontology of Social and Intentional Entities*): com base na UFO-A e na UFO-B, a UFO-C é uma ontologia de *entidades sociais* e intencionais, incluindo aspectos linguísticos. A UFO-C é desenvolvida essencialmente por Guizzardi (2006), mas também está presente em Guizzardi, Falbo e Guizzardi (2008), Guizzardi e Guizzardi (2010), Bringuente, Falbo e Guizzardi (2011) e em outras obras.

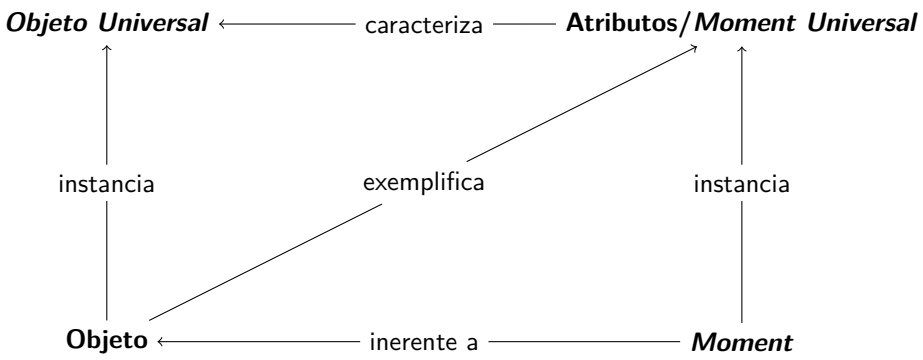
Conforme Guizzardi et al. (2011), a UFO-A é o núcleo da ontologia UFO, porque sistematiza conceitos como tipos e estruturas taxonômicas (GUIZZARDI et al., 2004), relações parte-todo (GUIZZARDI, 2007a; GUIZZARDI, 2009; GUIZZARDI, 2010a; GUIZZARDI, 2010b; GUIZZARDI, 2011), propriedade intrínsecas e espaços de valores (GUIZZARDI; MASOLO; BORGO, 2006), papeis, (MASOLO et al., 2005; GUIZZARDI, 2006), propriedades relacionais (GUIZZARDI; WAGNER, 2008), entre outros. Ainda conforme os autores, "esse fragmento constitui uma teoria estável, [...] possuindo forte suporte empírico promovido por experimentos em psicologia cognitiva".

Como desdobramento das UFO-A, UFO-B e UFO-C, Nardi et al. (2013) propuseram uma UFO-S, como uma ontologia bem-fundamentada para tratar de conceitos de *serviços* de forma independente a um determinado domínio ou ontologia de aplicação. A UFO-S é classificada pelos autores como uma ontologia nuclear (*core ontology*). Porém, no nome adotado pelos autores ressalta um problema terminológico (e, talvez, ontológico): UFO refere-se à *Unified Foundational Ontology*, mas UFO-S não é uma ontologia de

fundamentação, mas sim uma ontologia nuclear. Desse modo, entende-se que nomear a ontologia de serviço dos autores como UCO-S: *Unified Core Ontology for Services* seria mais adequado. De toda forma, à parte dessa questão terminológica, a UFO-S é um bom exemplo de como a ontologia de fundamentação proposta por Guizzardi se presta como fundamentos para a produção de novas ontologias mais específicas.

A parte mais fundamental da UFO exemplifica o quadrado ontológico aristotélico, representado na Figura 8 (GUIZZARDI; WAGNER, 2010, p. 176). A primeira distinção da UFO é referente às categorias dos Particulares (Particular) – ou Individuais[8] – e Universais (Universal). Os particulares são entidades que existem na realidade e possuem uma identidade única. Os universais, por outro lado, são padrões de características independentes de espaço-tempo que são realizados em diferentes indivíduos. Os Momentos (Moment), são a categoria de elementos que dependem de ao menos outro, do qual são inerentes, para existir. Diz-se que são existencialmente dependentes (Existential Dependence), ou seja, apenas existem em outros indivíduos. Já os objetos (Substantials) são existencialmente independentes (Independence), ou seja, não há nenhum outro indivíduo que precise existir para que um *substantial* exista.

Figura 8 – As quatro categorias ontológicas aristotélicas



Fonte: Os autores

Nota: Adaptado de Lowe (2006, p. 18), Guizzardi e Wagner (2010, p. 178) e Guizzardi e Zamborlini (2014).

Essas categorias estão presentes como primeiras distinções da UFO, em que os universais podem se subdividir em Monadic Universal ou em Relation. *Universais Monádicos* são conceitos que se referem a uma única entidade, como Pessoa, Cão, Flor, Banco de Dados, Bruxa. *Relações* são universais poliádicos, ou seja, que se referem a mais de uma entidade, como Casamento (no sentido de que são necessários ao menos duas entidades para que uma relação como essa possa existir). Os *Universais Monádicos* se subdividem em Substantial Universal e Moment Universal. Entidades substanciais são existencialmente

[8] O nome *Individual* aparece em trabalhos mais recentes, como em Guizzardi (2006).

independentes, ou seja, não há nenhum outro indivíduo que precise existir para que um Substantial exista. Conforme já abordado no parágrafo anterior, isso não é o caso para a categoria dos Moment Universal.

Diferentemente das tradicionais teorias de universais, que estabelece que duas classes são idênticas se possuem os mesmos indivíduos, Guizzardi (2005, p. 219) não propõe a UFO com uma teoria extensional, em que os indivíduos se comportem sempre como conjuntos da Teoria de Conjuntos e atendem um Princípio de Extensionalidade (Extensionality Principle), mas com uma teoria de universais que, amparado em evidências cognitivas e psicológicas, possuem elementos que atendem ao princípio da extensionalidade (por exemplo, Quantities) e outros que não atendem a esse princípio necessariamente, como o caso dos Category. Dessa forma, conforme o autor:

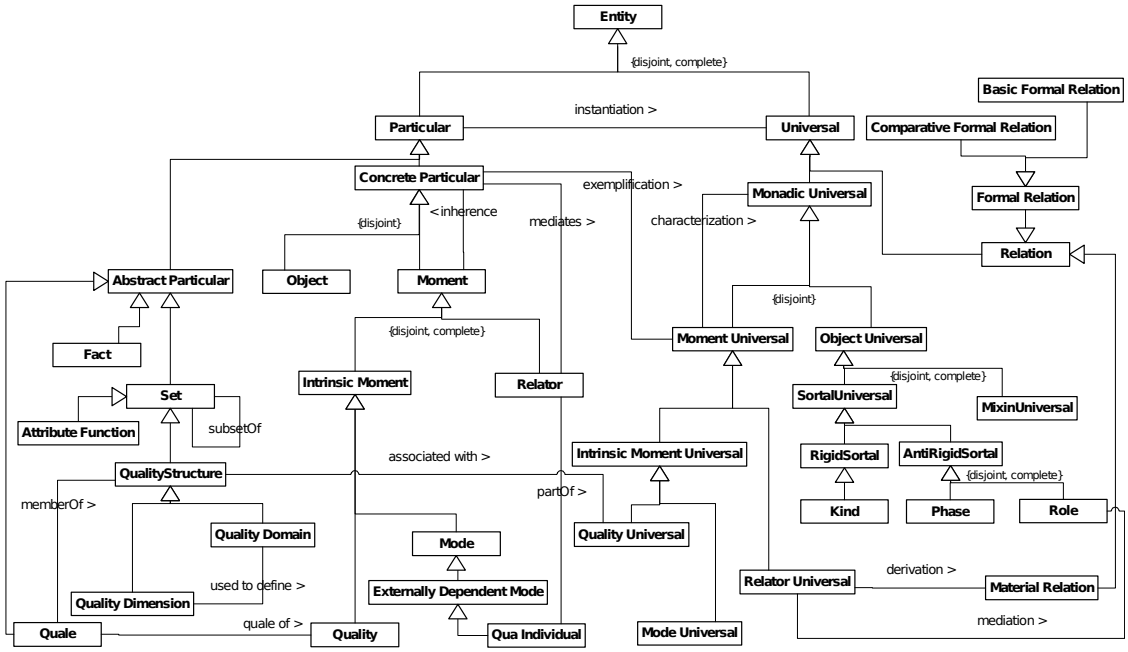
> *For every universal U there is a set $Ext(U)$, called its* extension*, containing all instances of U as elements. However, even if two universals U_1 and U_2 have identical extensions ($Ext(U_1) = Ext(U_2)$), they are not necessarily considered to be identical.*

A Figura 9 contém uma figura extraída de Guizzardi e Wagner (2010). Trata-se de uma das figuras produzida pelo autor da UFO que melhor sumariza os conceitos da Ontologia de fundamentação, embora seja ainda uma figura incompleta. Para tentar completar a Figura 9, a Figura 60 do Apêndice C contém um diagrama com os principais conceitos encontrados na bibliografia da UFO que visa ser o mais preciso possível em relação aos textos originais, embora fatalmente ainda seja incompleta.

Conforme apresentado na introdução deste capítulo, para uma introdução à UFO, que parte do prefácio dos parágrafos anteriores, indica o trabalho de Zamborlini (2011, p. 24-47) é indicado como antecipação à obra principal (GUIZZARDI, 2005). Porém, a lista exaustiva do fragmento dos conceitos da UFO abordados por este trabalho está contida na Figura 59 do Apêndice B, em que os conceitos são apresentados na forma de uma ilustração em UML. As entidades destacadas em cinza escuro, exercem o papel de conceitos utilizados como estereótipos para a OntoUML, conforme descritos na seção 4.5. Na referida seção 4.5, as regras da UFO a serem observadas para cada um dos conceitos são sistematizados em quadros, um para cada conceito fundamental e outros para construtores adicionais. A descrição desses conceitos fundamentais utilizados nesta pesquisa está presente no Glossário da UFO, que se trata de uma sintetização teórica produzida com base nas obras listadas nas Referências[9]. O glossário está organizado em ordem alfabética de termos conforme aparecem nas obras referenciadas, por isso, geralmente estão escritos em inglês. Citações precisas das obras das quais foram extraídos são apresentadas no corpo de cada entrada. Dessa forma, tem-se o seguinte:

[9] O referido glossário possui outras entradas referentes a conceitos que não necessariamente são trabalhos nesta pesquisa, mas se referem à UFO.

Figura 9 – Fragmento da *Unified Foundational Ontology (UFO)*



Fonte: Guizzardi e Wagner (2010)

a) para uma introdução didática à UFO-A, consultar a obra Zamborlini (2011, p. 24-47), Guizzardi (2007b), ou diretamente a obra original Guizzardi (2005);

b) a Figura 59 do Apêndice B contém a lista exaustiva dos conceitos da UFO abordados por esta pesquisa. Trata-se do escopo de Ontoprolog, conforme desenvolvido na Parte III;

c) a seção 4.5 sistematiza em quadros as regras que linguagens baseadas em *frames* que tenham a UFO como ontologia subjacente devem observar;

d) o Glossário da UFO detalha cada um dos conceitos da UFO, mencionados nos quadros, e apresenta referências e citações precisas de obras da literatura das quais cada conceito foi obtido;

e) a Figura 60 do Apêndice D contém uma tentativa de construir um diagrama completo com todos os conceitos fundamentais da UFO-A encontradas na bibliografia da área.

4.4 Tópicos abordados pela modelagem conceitual

As seções seguintes abordam observações ou comentários referentes a tópicos técnicos utilizados na modelagem de ontologias com base na UFO e OntoUML. Tratam-se de informações analíticas preliminares que devem ser lidas em conjunto com as definições do Glossário da UFO e que complementam os quadros da subseção 4.5.1.

4.4.1 Instâncias

O conceito de *instância* denota uma relação formal entre dois elementos do modelo conceitual em que um elemento particular (a instância) é um exemplo de um conceito geral (o tipo). Em UML (OBJECT MANAGEMENT GROUP, 2011b, p. 82-83) essa relação é definida pela `InstanceSpecification`, definida como:

> *An instance specification is a model element that represents an instance in a modeled system.*
> *An instance specification may specify the existence of an entity in a modeled system. An instance specification may provide an illustration or example of a possible entity in a modeled system. An instance specification describes the entity. These details can be incomplete. The purpose of an instance specification is to show what is of interest about an entity in the modeled system. The entity conforms to the specification of each classifier of the instance specification, and has features with values indicated by each slot of the instance specification. Having no slot in an instance specification for some feature does not mean that the represented entity does not have the feature, but merely that the feature is not of interest in the model.*

Apesar de haver críticas que envolvem a distinção ontológica dos conceitos de instanciação e subsunção, conforme mencionado na subseção 4.4.10, neste trabalho assume-se ao menos diferenças lógicas entre as relações de instanciação e de subsunção, assim como ocorre com OntoUML, que herda essa característica de UML. Essas diferenças são baseadas na citação apresentada, mas são formalizadas na Parte III.

4.4.2 Generalização ou Subtipos

O conceito de *generalização* (extensão, subtipo ou especialização), representado por uma seta com o lado aberto apontando para o lado mais geral, na forma $A \triangleleft\!\!- B$, em que A é estendido por B. O conceito é usado amplamente em modelagem UML e é descrita na versão 2.0 Object Management Group (2003, p. 67) usada por Guizzardi (2005) e na versão corrente Object Management Group (2011b, p. 70-71) da UML da seguinte forma:

> *A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier inherits the features of the more general classifier. [...] Where a generalization relates a specific classifier to a general classifier, each instance of the specific classifier is also an instance of the general classifier. Therefore, features specified for instances of the general classifier are implicitly specified for instances of the specific classifier. Any constraint applying to instances of the general classifier also applies to instances of the specific classifier.*

Baseado nessa definição[10] e na artigo de Guizzardi, Wagner e van Sinderen (2004), em sua dissertação de mestrado Benevides (2010, p. 31) estende a noção de generalização da UML para considerar aspectos modais, ausentes na UML. Nesse sentido, considerando uma linguagem lógica modal normal[11,12] alética[13] com semântica de Kripke sem restrições de acessibilidade, no contexto da relação de generalização da UML, o autor propõe a seguinte definição de *subtipo*:

> ***(subtype)**: Where a generalization relates a specific non relational Classifier $C1$ to a general non relational Classifier $C2$, each instance x of $C1$ is also an instance of $C2$ in every world w in which x is an instance of $C1$: subtype$(C1, C2) \coloneqq \Box(\forall x(C1(x) \rightarrow C2(x)))$*[14].

A definição de Benevides é importante porque retrata a noção de rigidez da relação de generalização, ou de subtipo. Porém, ela só é usada explicitamente numa linguagem que contemple expressividade modal, o que não é o caso da OntoUML.

4.4.3 *Generalization Sets*

Conforme Object Management Group (2011b, p. 74-75), *Generalization Sets* são conjuntos de associações de generalização de classes específicas para um mesmo *Classifier*:

> *Each Generalization is a binary relationship that relates a specific Classifier to a more general Classifier (i.e., from a class to its superclasses). Each GeneralizationSet defines a particular set of Generalization relationships that describe the way in which a general Classifier (or superclass) may be divided using specific subtypes. For example, a GeneralizationSet could define a partitioning of the class Person into two subclasses: Male Person and Female Person. Here, the GeneralizationSet would associate two instances of Generalization. Both instances would have Person as the general classifier; however, one Generalization would involve Male Person as the specific Classifier and the other would involve Female Person as the specific classifier. In other words, the class Person can here be said to be* partitioned *into two subclasses: Male Person and Female Person. Person could also be divided into North American Person, Asian Person, European Person, or something else. This collection of subsets would define a different GeneralizationSet that would associate with three other Generalization relationships. All three would have Person as the general Classifier; only the specific classifiers would differ (i.e., North American Person, Asian Person, and European Person).*

Generalization Sets possuem dois atributos específicos:

[10] O referido autor usou como referência a versão 2.2 da UML (OBJECT MANAGEMENT GROUP, 2009).

[11] Segundo Benevides, trata-se de uma lógica GS5.

[12] Uma lógica modal é considerada normal se possui o axioma $\Box(\alpha \rightarrow \beta) \rightarrow (\Box\alpha \rightarrow \Box\beta)$, ou algumas de suas formas multimodais, e a regra de necessitação $(\alpha \vdash \beta \implies \alpha \vdash \Box\beta)$ (CARNIELLI; PIZZI, 2008, p. 32).

[13] Lógica alética é aquela na qual se interpreta os operadores modais como "necessidade" e "possibilidade" (CARNIELLI; PIZZI, 2008, p. 30).

[14] Notação simbólida adaptada pelos autores.

a) `isCovering : Boolean`

> *Indicates (via the associated Generalizations) whether or not the set of specific Classifiers are covering for a particular general classifier. When isCovering is true, every instance of a particular general Classifier is also an instance of at least one of its specific Classifiers for the GeneralizationSet. When isCovering is false, there are one or more instances of the particular general Classifier that are not instances of at least one of its specific Classifiers defined for the GeneralizationSet.*

b) `isDisjoint : Boolean`

> *Indicates whether or not the set of specific Classifiers in a Generalization relationship have instance in common. If isDisjoint is true, the specific Classifiers for a particular GeneralizationSet have no members in common; that is, their intersection is empty. If isDisjoint is false, the specific Classifiers in a particular GeneralizationSet have one or more members in common; that is, their intersection is* not *empty.*

A combinação das propriedades `isCovering` e `isDisjoint` é interpretado na OntoUML conforme a Tabela 5. Cada relação de generalização que participa do *Generalization Set* deve possuir um rótulo com a notação indicada na tabela, exceto no caso de `isCovering = False` e `isDisjoint = True`, que é o estado padrão.

Tabela 5 – Notação de atributos de *Generalization Sets*.

isCovering	isDisjoint	Notação preferível	Notação padrão
True	True	{complete,disjoint}	{complete,disjoint}
True	False	{complete,overlapping}	{complete}
False	True	{incomplete,disjoint}	{incomplete,disjoint}
False	False	{incomplete,overlapping}	{incomplete}

Fonte: Produzido pelos autores.

Nota: Baseado em Object Management Group (2011b, p. 76).

Em Ontoprolog os *Generalization Sets* não são representados diretamente, mas podem ser inferidos nas traduções de Ontoprolog para OntoUML, uma vez que há constructos lógicos suficientes para resgatar esse conceito da UML (Capítulo 6).

4.4.4 Representação de qualidades

No que tange à construção de Ontoprolog com base na UFO, um dos aspectos de maior desafio e também de maiores controvérsias teóricas aos autores deste trabalho é a definição das Qualidades (Quality). De forma intuitiva, qualidades são propriedades que um tipo universal e seus indivíduos possuem. “Trata-se de uma tentativa de modelar a relação

entre Intrinsic Moments e sua representação em estruturas cognitivas" (GUIZZARDI; WAGNER, 2010). Conforme Probst (2007, p. 74):

> *Qualities are understood as individually existing entities. Qualities are the only entities that can be directly observed. All other entities, such as endurants, perdurants or abstracts are characterized via their qualities. This implies that any perceptions of physical reality are intrinsically tied to qualities. In this sense, any observation or measurement process has a quality as target.*

Embora Guizzardi (2005, p. 201-278) tenha dedicado um capítulo inteiro de sua tese de doutoramento à questão das *Qualities*, o tema não se encerrou naquele texto, tanto é que, em um trabalho recente (ALBUQUERQUE; GUIZZARDI, 2013), o autor aperfeiçoa a abordagem original apresentando novas construções para a linguagem OntoUML. Originalmente (GUIZZARDI, 2005, p. 222-235; 243-253) as propriedades, ou qualidades, foram propostas baseadas na "Teoria de Espaços Conceituais" de Gärdenfors (2000), Gärdenfors (2004). Já em Albuquerque e Guizzardi (2013), os autores lançam mão das teorias de "Espaços de Referência Semântica" de Probst (2007) e Probst (2008). As entradas Quale, Quality, Quality Universal, Association Relation, Quality Structure e Mode Universal do Glossário da UFO possuem comentários referentes aos dois trabalhos de Guizzardi.

O que segue desta seção é uma interpretação das teorias de qualidades propostas para a UFO, ajustadas aos objetivos desta tese, com intuito de permitir que as teorias referentes aos Intrinsic Moments sirvam como metamodelos a ontologias de domínio. O foco desta seção é discorrer sobre aspectos referentes a modelagem, e não discutir a natureza ontológica dos conceitos. Ainda assim, trata-se de uma visão simplificada sobre o tema dos *intrinsic moments*, ajustada aos objetivos do trabalho. Uma abordagem mais completa pode considerar a taxonomia de qualidades e de espaços de qualidade propostas por Probst (2007) e os demais conceitos de qualidade propostos em Albuquerque e Guizzardi (2013), conforme sugestões a trabalhos futuros anotada na seção 11.3. Como de praxe, os detalhes sobre o significado de cada conceito e diferentes definições para os conceitos encontradas na literatura, encontram-se no Glossário da UFO.

Quality Structures representam álgebras de um ou mais Conceptual Space. Estes podem ser definidos matematicamente fora da linguagem de especificação de ontologias com base nas dimensões que compõem os Quality Structures. As Quality Structures, por sua vez, são representadas em OntoUML por meio de DataTypes. Como Quality Structures possuem duas especializações, Quality Dimension e Quality Domain, o `Datatype` recebe igualmente duas especializações, cada uma para representar uma das estruturas. As diferenças estão no número de dimensões básicas: Quality Dimensions são representados pelo Classifier `Simple Datatype`, e Quality Domains são representados por `Structured Datatypes`. Como `Simple Datatype` e `Structured Datatypes` não estavam presentes na figura original de Guizzardi (2005, p. 334), mas fazem parte do metamodelo da OntoUML

(BENEVIDES, 2010, p. 41), apresenta-se a referida ilustração com customizações que incluem essas especializações, conforme a Figura 17.

Attribute Functions são usadas para relacionar Quality Universals e Quality Structures:

> *attribute functions are therefore the ontological interpretation of UML attributes, i.e., UML Properties which are owned by a given classifier. That is, an attribute of a class C representing a universal U is interpreted as an attribute function derived from the elementary specification of the universal U* (GUIZZARDI, 2005, p. 325).

Conforme Guizzardi (2005, p. 326), uma Attribute Function é representada como uma propriedade do universal a que está relacionada quando o co-domínio da função é uma Quality Dimension. No caso em que o co-domínio é um Structured DataType, ou seja, um Quality Domain, a Attribute Function é representada por um DataType Relationship (uma relação direcionada) partindo do Classifier do universal possuidor, e apontando para o DataType relacionado, de modo que o nome da relação é o nome da função de atribuição. Por isso, as funções de atribuição são navegáveis e necessariamente devem ser nomeadas.

Em Guizzardi (2005, p. 224), um Quality Universal U é relacionado a uma única Quality Structure x por meio de uma Association Relation[15] $assoc$, conforme segue, em que $\exists!$ deve ser lido como "existe apenas um" (notação nossa):

$$qualityUniversal(U) \coloneqq intrinsicMomentUniversal(U) \wedge \exists! x QS(x) \wedge assoc(x, U) \quad (4.1)$$

Da mesma forma, uma Quality Structure x é sempre relacionada a uma única Quality Universal U:

$$QS(x) \coloneqq \exists! U \; qualityUniversal(U) \wedge assoc(x, U) \quad (4.2)$$

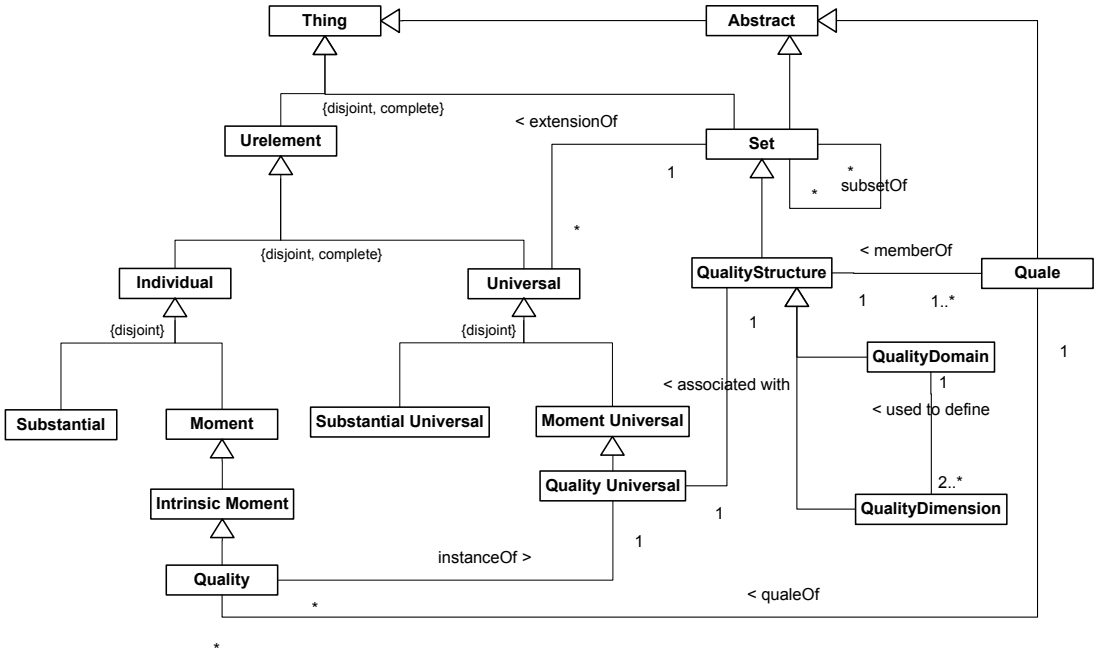
Essa relação um para um entre Quality Structures e Quality Universals é ilustrada pela Figura 10, em que "$<$ *associated with*" relaciona as duas entidades.

Porém, para o escopo deste trabalho, assume-se a alteração nas regras de relações entre Quality Structures e Quality Universals na Equação 4.2 e na Equação 4.1 apresentada em Albuquerque e Guizzardi (2013), de modo que passem a ter a forma da Equação 4.3:

$$\begin{aligned} QS(x) &\coloneqq \exists U \; qualityUniversal(U) \wedge assoc(x, U) \\ qualityUniversal(U) &\coloneqq intrinsicMomentUniversal(U) \wedge \exists x QS(x) \wedge assoc(x, U) \end{aligned} \quad (4.3)$$

[15] Em Albuquerque e Guizzardi (2013), a relação de associação é chamada de *"structuration"*. Porém, no decorrer deste texto, o nome original é mantido, embora concorde-se que o nome "association", usado originalmente, sofra de imprecisão terminológica, algo que espera-se ser superado com as definições apresentadas neste texto.

Figura 10 – Qualitites, Quality Universals e respectivas associações



Fonte: Guizzardi (2005, p. 235)

Desse modo, pela Equação 4.3, cada Quality Structures está relacionado a *ao menos um*, e não somente a exatamente um, Quality Universal. Com isso, os Quality Universals podem representar álgebras de Conceptual Spaces de mais de um Quality Universal.

Conforme Guizzardi (2005, p. 246-248, 325), geralmente os Quality Universals não são representados explicitamente em modelos OntoUML, mas sim via Attribute Functions. Isso significa que em modelos OntoUML as Qualidades, que estão associadas a uma ou mais Estruturas de Qualidade, são representadas no compartimento de propriedades das entidades. A Qualidades, por sua vez, está ligada à entidade que a detém pela relação de Caracterização (Characterization), no nível de tipos, e, no nível de particulares, pela relação de Inerência (Inherence), instância de Caracterização.

Com base nas discussões apresentas nesta seção, o Quadro 13 (Quality), Quadro 14 (QualityDimension como SimpleDataType), Quadro 15 (QualityDomain como StructuredDataType) e Quadro 16 (Quale) sumarizam as regras de construção de modelos conceituais referentes à qualidades. Esses quadros são baseados nas observações presentes em Guizzardi (2005, p. 323-327), e nos seguintes princípios, extraídos da mesma obra:

> *Principle 6.2: Every attribute function derived from the elementary specification of the universal U may be represented as an attribute of the class C_U (representation of the universal U) in a* CM-ontology[16]*; every* quality dimension *which is the co-domain of one these functions may*

[16] Abreviação que indica que se trata de uma ontologia de domínio na forma de uma especificação de um modelo conceitual.

> *be represented as data types of the corresponding attributes in this* CM-ontology*. Finally, relations constraining and informing the geometry of a quality dimension may be represented as constraints in the corresponding data type* (GUIZZARDI, 2005, p. 246).

Esse princípio é generalizado por este, que trata também Quality Domains:

> *Principle 6.3: Every quality dimension D associated to a quality universal Q may be represented as a datatype DT in a* CM-ontology*; Relations constraining and informing the geometry of a quality dimension D may be represented as operators in the corresponding datatype DT. A collection of integral dimension* $D_1 \ldots D_n$ *(represented by data types* $DT_1 \ldots DT_n$*) constituting a quality domain QD can be grouped in structured datatype W representing quality domain QD. In this case, every quality dimension* D_i *of QD may be represented by a field of W of type* DT_i*. Moreover, the relations between the dimensions* D_i *of QD may be represented by constraints relating the fields of data type W* (GUIZZARDI, 2005, p. 251).

As Regiões de Qualidade (Quality Region) são regiões de valores de qualidade em determinado Espaço Conceitual contido em um Quality Structure específico. Devido às limitações sensoriais inerentes a qualquer sujeito cognoscente, assume-se a impossibilidade de conhecer direta e absolutamente o valor de uma qualidade, mesmo em um espaço de valores discreto. Por isso, as Regiões de Qualidade são aproximações de valores pressupostos como absolutos, mas inacessíveis completamente. Em essência, os diversos espaços possíveis definidos logicamente pelas Quality Structure devem poder ser representados pelas Regiões de Qualidade, de modo que as Regiões de Qualidade representem regiões de valores contidos em uma Quality Structure. Dessa forma, Regiões de Qualidade possuem representação (léxica) próxima do valor absoluto de determinada observação da qualidade, o quale. Guizzardi (2005, p. 251) aborda a questão dos Quales:

> *every quality universal Q that is associated to a quality domain in an elementary specification of universal U can be represented in a conceptual model via attribute functions mapping instances of U to quale vectors in the n-dimensional domain associated with Q. The n-dimensional domains should be represented in a conceptual model as an n-valued structured data type.*

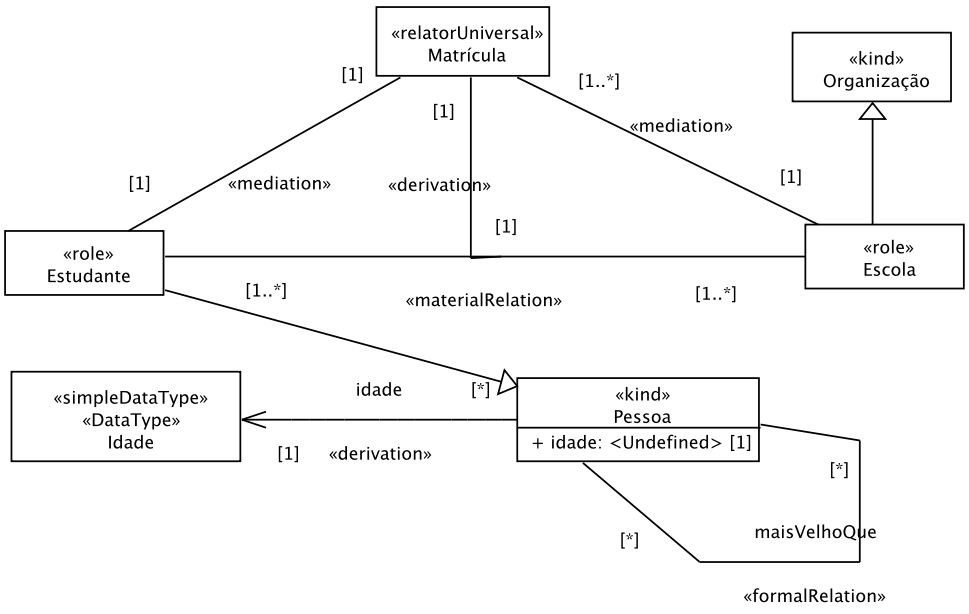
A semântica para os Quales não é alterada neste texto, e permanece como apresentada no Glossário.

4.4.5 Relações formais, materiais e de derivação

Conceitos centrais na OntoUML referem-se às Relações (Relation), que se subdividem em Relações Formais (Formal Relation) e Relações Materiais (Material Relation), todas definidas em Guizzardi (2005), cujas definições entre parênteses remetem ao Glossário da UFO.

Apenas com o intuito de exemplificar o uso dessas relações, destaque-se exemplo da dissertação de mestrado de Benevides (2010, p. 44-45), sintetizada na Figura 11, e que muito bem sistematiza o uso dessas relações. A figura contém um Kind `Pessoa` com um atributo `Idade` que é um número natural representado pelo Quality Dimension `Idade`, o que leva a existência de um Formal Relation chamado `maisVelhoQue`. Uma instância de `Pessoa` pode exercer o Role `Estudante` quando matriculado (`Matrícula`) em uma `Escola`. Isso é representado pela relação Mediation Relation (m) entre uma instância de Relator Universal `Matrícula` e `Pessoa`. Se forma similar, um Kind `Organização` exerce um Role `Escola` quando provê serviços educacionais por meio da relação Mediation Relation (m) entre uma instância do Relator Universal `Matrícula` e `Escola`. Quando uma instância x de `Matrícula` media uma instância y de `Pessoa` ($m(x, y)$) e uma instância z de `Organização` ($m(x, z)$), então há uma relação "estuda" do tipo Material Relation entre y and z. Esse fato é simbolizado pela relação Derivation Relation entre a Material Relation e o Relator Universal `Matrícula` (BENEVIDES, 2010, p. 44-45).

Figura 11 – Exemplo de Relator Universal, Formal Relation, Material Relation, Mediation e Derivation Relation



Fonte: Os autores

Adaptado de: Benevides (2010, p. 45)

4.4.6 Relações sobre relações: *subsetting*, *specialization* e *redefinition*

Nesta seção aborda-se a proposta de Costal, Gómez e Guizzardi (2011a) referente à significação ontológica das relações de *subsetting* (Subsetting relation), *specialization*

(Specialization relation (for relations)) e *redefinition* (Redefinition relation), que são tipos de relações da UML. Isso é realizado porque a distinção entre essas relações não estava clara na especificação da UML 2. A proposta discutida por Costal, Gómez e Guizzardi, que não estava presente na obra original de Guizzardi (2005), é incorporada à linguagem Ontoprolog (Capítulo 7) porque consiste em regras precisas sobre o uso dos três subtipos de relações, e porque estão alinhadas aos objetivos deste capítulo.

O tema das relações também é abordado por Nieto, Costal e Gómez (2011), por Olivé (2007), sendo este um livro que trata de diversos outros detalhes sobre modelagem conceitual no contexto de sistemas de informação, e também por Guizzardi e Wagner (2008), que apresenta uma análise ontológica sobre relações. Nesse contexto, o relatório técnico de Costal, Gómez e Guizzardi (2011b, p. 202) apresenta uma coletânea de referências sobre o estudo de diferentes formas de se lidar com as relações de *subsetting*, *specialization* e *redefinition* na UML:

> *This catalog has been obtained from the papers reviewed in our related work section, thus, comprising a set of ten examples produced by different authors (which can be considered experts in the field).*

O restante desta seção é baseada em Costal, Gómez e Guizzardi (2011a) e por isso omite-se novas referências a essa obra. Essencialmente, os autores abordam tipos de refinamentos, no sentido de aprimoramento[17], que relações podem formal e explicitamente receber durante a modelagem de domínio. São elas:

a) Subsetting relation, detalhada no Quadro 29;

b) Specialization relation (for relations), detalhada no Quadro 30; e

c) Redefinition relation, detalhada no Quadro 31.

O primeiro significado dessas relações refere-se a restrições sobre a estrutura dos modelos referente à inclusão das instâncias da relação refinadora na extensão das instâncias da relação que foi refinada.

Ainda de forma comum, os limites inferior e superior das restrições de cardinalidade das relações que redefidem outra por *Subsetting*, *Specialization* ou por *Extension* podem ser iguais ou mais restritos do que as restrições das entidades refinadas, mas não o contrário.

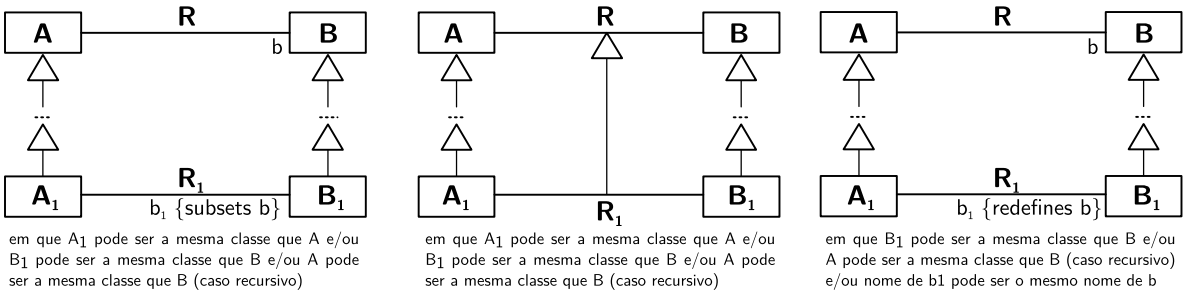
Ambos os lados das relações refinadoras devem participar as entidades ou especializações das entidades refinadas.

No âmbito de tipos, as relações refinadas precisam instanciar o mesmo *meta type* que as relações refinadas, embora *subsetting*, *redefinition* e *extention* sejam definidas apenas para relações do tipo Material Relation.

A Figura 12 ilustra a representação em UML das relações tratadas nesta seção.

[17] A palavra "refinamento" é usada como gênero para *subsetting*, *redefinition* e *extention*.

Figura 12 – Associações de *subsetting*, *specialization* e *redefinition* e regras sintáticas e semânticas em UML



Fonte: Adaptado de Costal, Gómez e Guizzardi (2011a, p. 192) e de Nieto, Costal e Gómez (2011, p. 184-185).

4.4.7 Regras de cardinalidade de «relator», «mediation», «derivation» e «material»

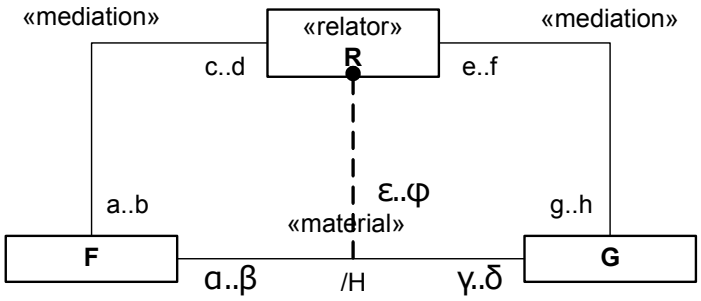
Em Guizzardi (2005, p. 253-262) desenvolve-se a análise das relações ontológicas e, baseado em sistemática avaliação de Material Relation, deriva-se os conceitos de Relator, Mediation Relation, Derivation Relation. A conclusão dessa análise, cujos detalhes abstemo-nos de transcrever, é apresenta na forma do seguinte princípio:

> *Principle 6.4: In a* CM-ontology*, any formal relation universal R_F of the domain may be directly represented as a standard association whose links represent the tuples in the extension of R_F. Conversely, a material relation R_M of the domain may be represented in a* CM-ontology *by a complex construct composed of: (i)* CM-class *stereotyped as «relator» representing the* relator universal*. The relator universal is associated to CM-Classes representing mediated entities via associations stereotyped as «mediation»; (ii) a standard association stereotyped as «material» representing a material relation whose links represent the tuples in the extension of R_M; (iii) a dashed line with a black circle in one of the ends representing the formal relation of derivation between (i) and (ii), in which the black circle lies in the association end of the relator universal* (GUIZZARDI, op. cit., p. 262).

O principal objetivo dos Relators é resolver problemas de ambiguidade em modelos OntoUML referentes à questão das restrições de cardinalidade entre as relações materiais, um problema conhecido como *"counting problem"* (GUIZZARDI, 2006). Alinhado com os objetivos deste capítulo, apresenta-se exclusivamente as regras a serem observadas na construção de especificações de relacionamentos materiais dependentes (Relational dependence) e entidades mediadas (Mediation) por um Relators. Com base nisso, a Figura 13 apresenta um esquema geral que exemplifica uma forma que diferentes configurações de entidades podem assumir no que tange ao tema em tela.

Considerando $a, b, c, d, e f, g, h$ variáveis para restrições dadas de cardinalidade entre entidades, calcula-se o valor de $\alpha, \beta, \delta, \gamma, \phi, \epsilon$, conforme segue (GUIZZARDI, 2005, p. 331):

Figura 13 – Regras de cardinalidade de relações materais



Fonte: Guizzardi (2005, p. 331)

Limites inferiores:

$$\gamma := c \times g$$
$$\alpha := e \times a$$
$$\epsilon := a \times g$$

Limites superiores:

$$\delta := d \times h$$
$$\beta := f \times b$$
$$\phi := b \times h$$

Essas fórmulas se justificam porque a Relação Material e a Relação de Derivação, identificadas na figura por /H e pela linha tracejada, respectivamente, tratam-se de relações baseadas e dependentes existencialmente (Existential Dependence) do «relator» identificado por R.

As regras descritas nesta seção são usadas nas restrições do Quadro 21 (Derivation Relation) e do Quadro 23 (Material).

4.4.8 Meronímias

As relações de meronímias representam teorias de "partes", e não teorias de "todos". Por isso se coloca que um x é $subComponentOf$ de um y, e não que um y é um todo composto por partes x. Há diversas teorias mereológicas (Mereologies) que abordam as relações parte-todo (Meronymic relations). A UFO utiliza uma axiomatização baseada em axiomas clássicos como Ground Mereology, Weak Supplementation e Strong Supplementation, Minimal Mereology e Extensional Mereology, entre outros, presentes no Glossário da UFO. Com base nisso, Guizzardi (2005, p. 141-200, 294-309, 339-352)[18] define quadro tipos específicos de relações parte-todo, todas caso específicos do gênero Part of, a saber:

a) Component of/Component-Functional Complex Relation, abordada no Quadro 25 (ComponentOf);

[18] E trabalhos mais recentes, como os citados na seção 4.3.

b) Subquantity of/Mass-Quantity Relation, abordada no Quadro 26 (SubQuantityOf);

c) Subcollection of/Subcollection-Collection Relation, abordada no Quadro 27 (SubCollectionOf);

d) Member of/Member-Collection Relation, abordada no Quadro 28 (MemberOf).

As regras a serem observadas referentes a cada tipo de relação parte-todo estão detalhadas nos respectivos quadros. O Quadro 24 (Meronymic) apresenta regras gerais, observáveis em todos tipos de relação.

No Glossário da UFO, a entrada Mereology pode ser usada como ponto de referência principal, uma vez que todos os conceitos mereológicos utilizados pelo fragmento da UFO abordado neste trabalho estão listados naquela entrada.

Foram encontradas inconsistências nas propriedades das relações de meronímia entre as obras Guizzardi (2005), Guizzardi (2010a), Guizzardi (2010b) e Guizzardi (2011). Sempre que isso aconteceu, anotou-se no Glossário da UFO o fato e adotou-se nos quadros da subseção 4.5.1 a versão mais recente encontrada.

A subseção 4.4.8.1 contém um breve resumo axiomático dos principais conceitos usados nos quadros mencionados nesta seção. Todas as fórmulas foram extraídas no Glossário da UFO, exceto às referentes às propriedades básicas de relações, como simetria, reflexibilidade, transitividade e ordem. O intuito dessa seção é exclusivamente auxiliar na leitura dos respectivos quadros.

4.4.8.1 Axiomatização das meronímias

Reflexividade

Definição 4.1 (Relação reflexiva)**.** Seja R uma relação em A. Então, R é chamada *reflexiva* se $(x, x) \in R$, ou seja, quando todas as coisas estão relacionadas consigo mesmas:

$$R \text{ é reflexiva: } \forall x \in A \ (xRx)$$

Por exemplo, a relação *ter a mesma altura que* é reflexiva. —

Definição 4.2 (Relação irreflexiva)**.** Seja R uma relação em A. Então, R é chamada *irreflexiva* se $(x, x) \notin R$, ou seja, que nenhuma coisa está relacionada consigo mesma:

$$R \text{ é irreflexiva: } \forall x \in A \ \neg(xRx)$$

Por exemplo, a relação de *paternidade* é irreflexiva: ninguém é pai de si mesmo. —

Definição 4.3 (Relação não reflexiva)**.** Seja R uma relação em A. Então, R é chamada *não reflexiva* se não é uma Relação reflexiva, nem uma Relação irreflexiva, ou seja, quando algumas coisas estão relacionadas consigo mesmas e outras não:

$$R \text{ é não reflexiva: } \neg(\forall x \in A\ (xRx)) \wedge \neg(\forall x \in A\ \neg(xRx))$$

—

Simetria

Definição 4.4 (Relação simétrica)**.** Seja R uma relação em A. Então, R é chamada *simétrica* se R é inversa a si mesmo, ou seja, se $(x, y) \in R$ sempre implica em $(y, x) \in R$:

$$R \text{ é simétrico: } \forall x \forall y \in A\ (xRy \rightarrow yRx)$$

Por exemplo, a relação *é irmão de* é simétrica. Se Aldo é irmão de Beto, então Beto é também irmão de Aldo. —

Definição 4.5 (Relação assimétrica)**.** Seja R uma relação em A. Então, R é chamada *assimétrica* se $(x, y) \in R$ sempre implica em $(y, x) \notin R$:

$$R \text{ é assimétrico: } \forall x \forall y \in A\ (xRy \rightarrow \neg yRx)$$

Por exemplo, a relação de *paternidade* é assimétrica: se Aldo é pai de Beto, então Beto não é pai de Aldo. —

Teorema 4.1 (Se R é uma Relação assimétrica, então R é uma Relação irreflexiva.)**.** ■

Demonstração. Usando a notação de dedução natural de Jaśkowski (1934), temos que:

1.	$\forall x \forall y\ (xRy \rightarrow \neg yRx) \vdash \forall x\ \neg(xRx)$	
2.	$\forall x \forall y\ (xRy \rightarrow \neg yRx$	Premissa
3.	$\forall y\ (aRy \rightarrow \neg yRa)$	2, $\forall$ Eliminação
4.	$aRa \rightarrow \neg aRa)$	3, $\forall$ Eliminação
5.	aRa	Hipótese para *Reductio ad Absurdum*
6.	$\neg aRa$	4,5 Modus Ponens
7.	$aRa \wedge \neg aRa$	5,6 $\wedge$ Introdução
8.	$\neg aRa$	5-7 *Reductio ad Absurdum*
9.	$\forall x\ \neg(xRx)$	8 $\forall$ Introdução

□

Definição 4.6 (Relação anti-simétrica)**.** Seja R uma relação em A. Então, R é chamada *anti-simétrica* se $(x, y) \in R$ e $(y, x) \in R$ sempre implica em $x = y$, ou seja, a relação anti-simétrica implica que apenas a mesma coisa pode estar relacionada consigo mesma:

$$R \text{ é anti-simétrica: } \forall x \forall y \in A \ (xRy \wedge yRx \rightarrow x = y)$$

Por exemplo, a relação *não é menor que* é anti-simétrica: se x não é menor que y, e y não é menor que x, então x e y referem-se ao mesmo número. —

Definição 4.7 (Relação não simétrica)**.** Seja R uma relação em A. Então, R é chamada *não simétrica* se não é uma Relação simétrica nem uma Relação assimétrica:

$$R \text{ é não simétrica: } \neg(\forall x \forall y \in A \ (xRy \rightarrow yRx)) \wedge \neg(\forall x \forall y \in A \ (xRy \rightarrow \neg yRx))$$

—

Transitividade

Definição 4.8 (Relação transitiva)**.** Seja R uma relação em A. Então, R é chamada *transitiva* se $(x, y) \in R$ e $(y, z) \in R$ sempre implica em $(x, z) \in R$:

$$R \text{ é transitiva: } \forall x \forall y \forall z \in A \ (xRy \wedge yRz \rightarrow xRz)$$

Por exemplo, a relação *é mais velho que* é transitiva: Se Aldo é mais velho que Beto e Beto é mais velho que Carlos, então Aldo é mais velho que Carlos. —

Definição 4.9 (Relação intransitiva)**.** Seja R uma relação em A. Então, R é chamada *intransitiva* se $(x, y) \in R$ e $(y, z) \in R$ sempre implica em $(x, z) \notin R$:

$$R \text{ é intransitiva: } \forall x \forall y \forall z \in A \ (xRy \wedge yRz \rightarrow \neg xRz)$$

Por exemplo, a relação de *paternidade* é intransitiva: Se Aldo é pai de Beto e Beto é pai de Carlos, então Aldo não é pai de Carlos. No caso, Aldo é avô de Carlos. —

Definição 4.10 (Relação não transitiva)**.** Seja R uma relação em A. Então, R é chamada *não intransitiva* se não é uma Relação transitiva nem uma Relação intransitiva, uma vez que podem ser definidas relações existenciais que hora se comportam como transitivas e hora como não transitivas:

R é não transitiva: $\neg(\forall x \forall y \forall z \in A\,(xRy \wedge yRz \rightarrow xRz)) \wedge \neg(\forall x \forall y \forall z \in A\,(xRy \wedge yRz \rightarrow \neg xRz))$

—

Ordem

Definição 4.11 (Relação de ordem parcial)**.** R é uma *relação de ordem parcial* se R é uma Relação reflexiva, uma Relação anti-simétrica e uma Relação transitiva:

1. $\forall x \in A\,(xRx)$ (Relação reflexiva);
2. $\forall x \forall y \in A\,(xRy \wedge yRx \rightarrow x = y)$ (Relação anti-simétrica);
3. $\forall x \forall y \forall z \in A\,(xRy \wedge yRz \rightarrow xRz)$ (Relação transitiva);

—

Definição 4.12 (Relação de ordem parcial estrita)**.** É a relação Relação irreflexiva, Relação assimétrica e Relação transitiva:

1. $\forall x \in A\,\neg(xRx)$ (Relação irreflexiva);
2. $\forall x \forall y \in A\,(xRy \rightarrow \neg yRx)$ (Relação assimétrica);
3. $\forall x \forall y \forall z \in A\,(xRy \wedge yRz \rightarrow xRz)$ (Relação transitiva);

—

Definições mereológicas gerais

Definição 4.13 (Ground mereology)**.** Considerando a entrada Ground Mereology do Glossário da UFO, a relação *parte de* ($\leq$) é uma Relação de ordem parcial. Logo, $\leq$ pode ser definido como:

1. $\forall x\,(x \leq x)$ (Relação reflexiva);
2. $\forall x, y\,(x \leq y) \wedge (y \leq x) \rightarrow (x = y)$ (Relação anti-simétrica);
3. $\forall x, y, z\,(x \leq y) \wedge (y \leq z) \rightarrow (x \leq z)$ (Relação transitiva).

—

Definição 4.14 (Proper part)**.** Considerando a entrada Proper part do Glossário da UFO e a Definição 4.13 (Ground mereology), a relação *parte própria de* ($<$) é uma Relação de ordem parcial estrita definida como:

1. $(x < y) \coloneqq (x \leq y) \wedge \neg(y \leq x)$

Ou seja, *parte própria de* é uma Relação assimétrica e uma Relação transitiva, da qual uma Relação irreflexiva decorre (Teorema 4.1). Desse modo, a relação em tela pode ser representada por:

1. $\forall x, y\ (x < y) \rightarrow \neg(y < x)$ (Relação irreflexiva);
2. $\forall x\ \neg(x < x)$ (Relação assimétrica);
3. $\forall x, y, z\ (x < y) \wedge (y < z) \rightarrow (x < z)$ (Relação transitiva).

Definição 4.15 (Overlapping)**.** Considerando a entrada Overlapping do Glossário da UFO e a Definição 4.13 (Ground mereology), trata-se de Relação reflexiva, Relação simétrica e Relação não transitiva, que é equivalente à relação de intersecção da teoria de conjuntos. A relação de *overlap* – ou sobreposição – ($\bullet$) é definida como:

1. $(x \bullet y) \coloneqq \exists z\ (z \leq x) \wedge (z \leq y)$ (Relação não transitiva)
2. $\forall x\ (x \bullet x)$ (Relação reflexiva)
3. $\forall x, y\ (x \bullet y) \rightarrow (y \bullet x)$ (Relação simétrica)

Definição 4.16 (Disjointness)**.** Considerando a entrada Disjointness do Glossário da UFO e a Definição 4.15 (Overlapping), trata-se de uma relação Relação simétrica que afirma que se dois elementos não se sobrepõem, então eles são disjuntos:

1. $(x \int y) \coloneqq \neg(x \bullet y)$

A simetria é herdada da Overlapping (Definição 4.15)

1. $\forall x, y\ (x \int y) \rightarrow (y \int x)$ (Relação simétrica)

Definição 4.17 (Weak Supplementation)**.** Considerando a entrada Weak Supplementation do Glossário da UFO a Disjointness (Definição 4.16) e a Proper part (Definição 4.14), trata-se de:

1. $\forall x, y\ (y < x) \rightarrow \exists z\ ((z < x) \wedge (z \int y))$

   Que, expandindo $\int$ (Disjointness), é equivalente a:

1. $\forall x, y\ (y < x) \rightarrow \exists z\ ((z < x) \wedge \neg(z \bullet y))$

   Expandindo $\bullet$ (Overlapping), temos a forma final:

1. $\forall x, y\ (y < x) \rightarrow \exists z\ ((z < x) \wedge \neg(\exists h\ (h \leq z) \wedge (h \leq y)))$

—

Definição 4.18 (Minimal Mereology)**.** Considerando a entrada Minimal Mereology do Glossário da UFO, a chamada *mereologia mínima* é definida como a Ground mereology (Definição 4.13) estendida do axioma Weak Supplementation (Definição 4.17):

1. $\forall x\ (x \leq x)$ (Relação reflexiva);
2. $\forall x, y\ (x \leq y) \wedge (y \leq x) \rightarrow (x = y)$ (Relação anti-simétrica);
3. $\forall x, y, z\ (x \leq y) \wedge (y \leq z) \rightarrow (x \leq z)$ (Relação transitiva);
4. $\forall x, y\ (y < x) \rightarrow \exists z\ ((z < x) \wedge (z \int y))$ (Weak Supplementation)

—

Definição 4.19 (Existência na teoria de indivíduos)**.** Conforme comentários a respeito do predicado $\mathcal{E}$ apresentado na entrada Existential Dependence do Glossário da UFO, o predicado $\mathcal{E}$ denota "existência na teoria de indivíduos"[19], e é interpretado do seguinte modo:

$$\mathcal{E}(x) \coloneqq \exists T\ x :: T$$

Sendo que :: é apresentado na entrada Instantiation Relation do Glossário da UFO. —

[19] Em efeito, o conceito de "teoria de indivíduos" não está presente nas obras mencionadas naquela entrada do Glossário da UFO.

Definição 4.20 (Essential part)**.** Considerando a entrada Essential Part do Glossário da UFO, a Definição 4.19 (Existência na teoria de indivíduos), e a Definição 4.13 (Ground mereology), o conceito de *parte essencial* EP é definido como, em que x é a parte e y o todo:

$$EP(x, y) \coloneqq \Box(\mathcal{E}(y) \rightarrow (x \leq y))$$

—

Definição 4.21 (Inseparable part)**.** Considerando a entrada Inseparable Part do Glossário da UFO, a Definição 4.19 (Existência na teoria de indivíduos), e a Definição 4.13 (Ground mereology), o conceito de *parte inseparável* IP é definido como, em que x é a parte e y o todo:

$$IP(x, y) \coloneqq \Box(\mathcal{E}(x) \rightarrow (x \leq y))$$

—

Nota 4.1 (Diferença entre Essential part e Inseparable part). Observar que no caso de Inseparable part, sempre que uma parte x existir, ela existirá como parte de um todo y. Já no caso de Essential part, sempre que o todo y existir, ele existirá tendo um x como parte. A esse respeito, em Guizzardi (2011) encontra-se:

> *Essential parthood can be defined as a case of* existential dependence *between individuals, i.e., x is an* essential part *of y iff y cannot possibly exist without having that specific individual x as part (GUIZZARDI, 2007a). This specific mode of existential dependence can also be defined from the part x to the whole y. We say that x is an* inseparable part *of y iff x cannot possibly exist without being a part of that specific individual y (GUIZZARDI, 2007a). A stereotypical example of an essential part of a car is its chassis, since that specific car cannot exist without that specific chassis (changing the chassis legally changes the identity of the car); A stereotypical example of an inseparable part of a living cell is its membrane, since the membrane cannot exist without being part of that particular cell.*

—

Definição 4.22 (Mandatory part)**.** Considerando a entrada Mandatory part do Glossário da UFO, a Definição 4.19 (Existência na teoria de indivíduos), e a Definição 4.14 (Proper part), o conceito de *parte mandatória* MP é definido como:

$$MP(T, y) \coloneqq \Box(\mathcal{E}(y) \rightarrow \exists x\ (x :: T \wedge x < y))$$

—

Definição 4.23 (Extensional individual)**.** Considerando a entrada Extensional Individual do Glossário da UFO, um indivíduo y é chamado extencional se, e somente se, para todo x tal que x é parte de y, x é uma Essential part (Definição 4.20) EP de y :

$$E(y) \coloneqq \Box(\forall x\ (x < y) \rightarrow EP(x, y))$$

Logo, se um indivíduo y é classificado como extencional, então todas as suas partes são essenciais. —

4.4.9 *Power types*

Provavelmente o conceito de *power types* foi proposto originalmente em Odell (1994, p. 10, 12), que usou a intuição da ideia de *power set*, conjunto potência ou conjunto de partes, da Teoria de Conjuntos[20], para apresentar a seguinte definição:

> *A **power type** is an object type having instances that are subtypes of another object type.*

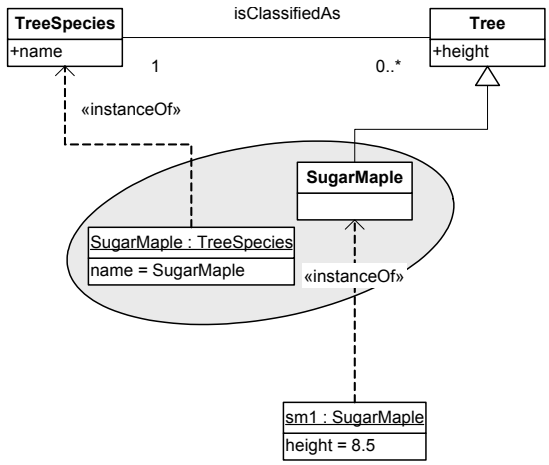
Nessa definição, *power type* refere-se a um padrão de design de modelagem visual que denota que uma classe W, o *power type*, está relacionado a uma classe S, e que instâncias w_i de W são extensões de S, de modo cada w_i é simultaneamente tanto uma instância de W como um subtipo de S.

Odell (1994) explica seu conceito com base em um exemplo famoso, utilizado quase sempre na explicação dessa ideia, mesmo em textos de autores diferentes. Trata-se do exemplo das árvores e espécies de árvores (*TreeSpecies*). Utiliza-se, no entanto, a ilustração contida em Henderson-Sellers e Gonzalez-Perez (2005) e em Gonzalez-Perez e Henderson-Sellers (2008, p. 126), que adaptam o exemplo de Odell, também presente de forma semelhante em Fowler (1996, p. 25), à notação UML, conforme Figura 14. Na figura, o *power type* é `TreeSpecies`, cujas instâncias estendem `Tree`. `SugarMaple`, nesse caso, é tanto uma instância, quanto uma classe. Por isso a notação acinzentada, proposta pelo autor para denotar essa entidade *sui generis*.

A noção de *power types* é abordada pela especificação da UML Object Management Group (2011b, p. 75-82), que apresenta uma detalhada explicação desse conceito lançando mão de diversos exemplos, entre eles o exemplo das árvores e espécies de árvores de Odell (1994, p. 11). Também apresenta a notação a ser utilizada em diagramas que representam o conceito, conforme ilustrado pela Figura 15. Em Object Management Group (2011b, p. 54) temos que:

[20] No âmbito da Teoria de Conjuntos, o conjunto potência de um conjunto A é o conjunto de todos os subconjuntos formados pelos elementos de A, denotado por $\wp(A)$.

Figura 14 – Árvores e espécies de árvores como *power type*



Fonte: Henderson-Sellers e Gonzalez-Perez (2005, p. 84)

> *The notion of power type was inspired by the notion of power set. A power set is defined as a set whose instances are subsets. In essence, then, a power type is a class whose instances are subclasses. The powertypeExtent association relates a Classifier with a set of generalizations that a) have a common specific Classifier, and b) represent a collection of subsets for that class.*

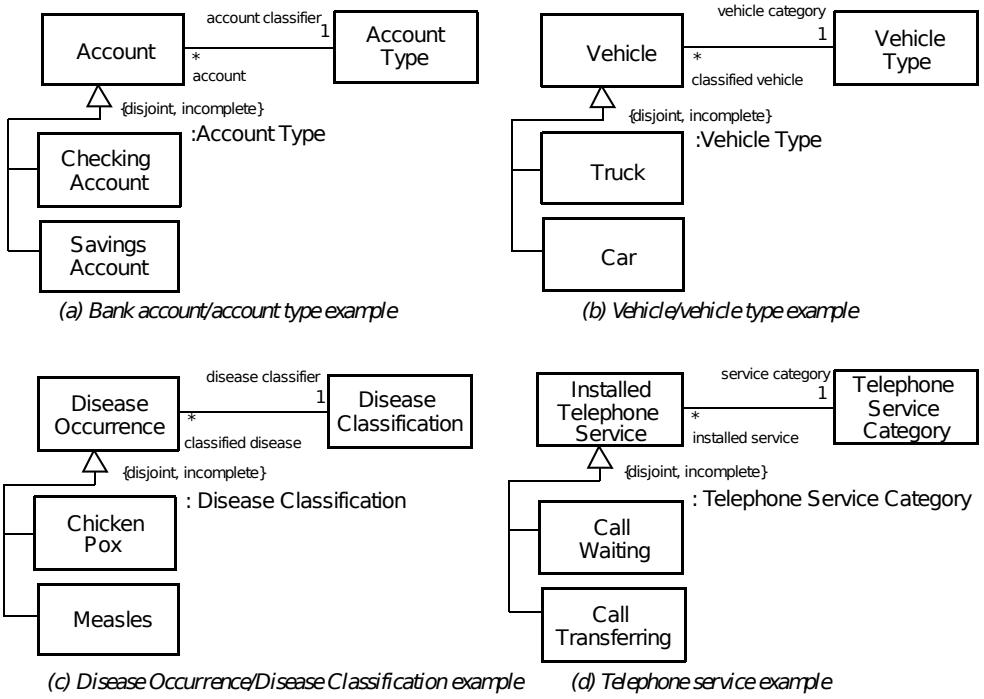
De forma mais detalhada, em Object Management Group (2011b, p. 80) encontra-se a seguinte explicação sobre a ideia de *power types*:

> *As established above, the instances of Classifiers can also be Classifiers. (This is the stuff that metamodels are made of.) These same instances, however, can also be specific classifiers (i.e., subclasses) of another classifier. When this occurs, we have what is called a power type. Formally, a* power *type is a classifier whose instances are also subclasses of another classifier.*
> *In the examples above, Tree Species is a power type on the Tree type. Therefore, the instances of Tree Species are subtypes of Tree. This concept applies to many situations within many lines of business. Figura 15* [21] *depicts other examples of power types.* ***The name on the generalization set beginning with a colon indicates the power type****. In other words, this name is the name of the type of which the subtypes are instances* (grifo forte nosso).
> *Diagram (a) in the figure below, then, can be interpreted as: each instance of Account is classified with exactly one instance of Account Type. It can also be interpreted as: the subtypes of Account are instances of Account Type. This means that each instance of Checking Account can have its own attributes (based on those defined for Checking Account and those inherited from Account), such as account number and balance. Additionally, it means that Checking Account as* an object in its own right *can have attributes, such as interest rate and maximum delay for withdrawal. (Such attributes are sometimes referred to as class variables, rather than instance variables.) The example (b) depicts a vehicle-modeling example.*

[21] Na obra citada a figura é Figure 7.50

> *Here, each Vehicle can be subclassed as either a Truck or a Car or something else. Furthermore, Truck and Car are instances of Vehicle Type. In (c), Disease Occurrence classifies each occurrence of disease (e.g., my chicken pox and your measles). Disease Classification is the power type whose instances are classes such as Chicken Pox and Measles.*

Figura 15 – Exemplos de *power type*



Fonte: Object Management Group (2011b, p. 81)

A respeito de questões referentes à implementação de sistemas considerando *power types*, a especificação da UML orienta o seguinte:

> *Power types are a conceptual, or analysis, notion. They express a real-world situation; however, implementing them may not be easy and efficient. To implement power types with a relational database would mean that the instances of a relation could also be relations in their own right. In object-oriented implementations, the instances of a class could also be classes. However, if the software implementation cannot directly support classes being objects and vice versa, redundant structures must be defined. In other words, unless you're programming in Smalltalk or CLOS, the designer must be aware of the integrity problem of keeping the list of power type instances in sync with the existing subclasses. Without the power type designation, implementors would not be aware that they need to consider keeping the subclasses in sync with the instances of the power type; with the power type indication, the implementor knows that a) a data integrity situation exists, and b) how to manage the integrity situation. For example, if the Life Policy instance of Insurance Line were deleted, the subclass called Life Policy can no longer exist. Or, if a new subclass of Policy were added, a new instance must also be added to the appropriate power type* (OBJECT MANAGEMENT GROUP, 2011b, p. 87).

Como a intenção deste trabalho é a construção de uma linguagem para especificação de ontologias baseadas no mundo real, essa observação da especificação da UML não é uma restrição ao uso de *power types*. De toda forma, o reconhecimento de que *power types* expressam situações do mundo real está alinhado com os objetivos traçados na seção 1.2.

Os defensores *power types* argumentam que esse padrão de modelagem reflete situações reais em que não há uma clara separação entre as relações de subsunção entre classes e de instâncias entre classes e particulares, de modo que a própria de ideia de universal e de particular seja de difícil separação, assim como aponta MacBride (2005). Com base nas posições desse autor, os particulares seriam apenas os elementos nas folhas das classificações realizadas de forma indistina entre subsunção e instanciação. No entanto, embora não se tome, por hora, partido nessa discussão, reitera-se a necessidade e a conveniência da existência desse tipo de padrão de modelagem baseada em *power types*, pois permite que situações do mundo real, em que a diferenciação entre instância e subsunção não seja conveniente ou ontologicamente justificável, sejam modeladas.

4.4.10 Clabjects

Se um *power type* é uma classe cujas instâncias são subsunções de outra classe, então como poderia ser chamado essa instância que tanto é classe como é instância? Segundo Atkinson e Kühne (2000), o nome sugerido é *clabject*: uma mistura de classe com objeto. Já Lowe (2004), no contexto de defender distinções entre as relações de instanciação e subsunção, sugere o nome de *'lower-order' universals*:

> *Within this ontological system, objects and modes are accorded the status of* particulars*, while kinds, properties and relations are accorded the status of* universals*. This is because the best way, in my view, to capture the traditional distinction between particulars and universals is by appeal to the* instantiation *relation. Universals are entities that are instantiated – that is, they have in- stances – while particulars are the entities that instantiate them. (I leave aside here the possibility of so-called 'higher-order' universals – universals that sup- posedly have other, 'lower-order' universals as their instances – because I am sceptical about the need to include them in our ontology.)*

Para os propósitos desta pesquisa, sugere-se o nome *clabject*, devido a sua clareza e simplicidade.

4.4.11 *Hyperspaces*

A noção de *Hyperspace*, uma abordagem de desenvolvimento e de gerenciamento de configuração de software, foi desenvolvida por pesquisadores da IBM e é descrita em Ossher e Tarr (2002). Trata-se de uma abordagem em que o relacionamento entre artefatos de um sistema computacional são organizados de maneira multi-dimensional.

De forma simplificada, essencialmente um *hyperspace* é um conjunto de *units* e um conjunto de *concerns* distribuídos numa matriz multidimensional que esgota as combinações possíveis entre *units* e *concerns*. As *units* (e os *compound units*, como composições dos primeiros) são um ou um conjunto de elementos que se deve agrupar e encapsular (no sentido de se destacar ou separar) das demais *units*. Exemplos, segundo os autores, são uma sentença sintática em alguma linguagem, uma classe (de uma linguagem de programação ou de uma linguagem de representação), um requisito de software, etc. Enquanto os *concerns* são aspectos ou características que agrupam e integram as *units*. Por exemplo, no contexto de desenvolvimento de software, a característica de "desenhar na tela" pode ser distribuída entre diferentes *units*. Dessa forma, cada *concern* se torna uma dimensão na matriz, de modo que se pode atribuir características e outros aspectos às *units*. Nas palavras dos autores:

> *Hyperspaces permit the explicit* identification *of any concerns of importance,* encapsulation *of those concerns, identification and management of* relationships *among those concerns, and integration of concerns [...] A* hyperspace *is a concern space specially structured to support our approach to multi-dimensional separation of concerns. Its first distinguishing characteristic is that its units are organized in a multi-dimensional matrix. Each axis represents a dimension of concern, and each point on an axis a concern in that dimension. This makes explicit all the dimensions of interest, the concerns that belong to each dimension, and which concerns are affected by which units. The coordinates of a unit indicate all the concerns it affects; the structure clarifies that each unit affects exactly one concern in each dimension. Each dimension can thus be viewed as a partition of the set of units: a particular software decomposition. Any single concern within some dimension defines a hyperplane that contains all the units affecting that concern. The matrix structure permits uniform treatment of all kinds of concerns, and it allows developers to navigate or slice through the matrix according to any desired concerns* (OSSHER; TARR, 2002, p. 302-304).

Essa intuição básica de *hyperspace* foi usada pelos autores para desenvolver diversas ferramentas, entre elas o *HyperJ* (TARR; OSSHER, 2000), que é uma API para a linguagem de programação Java construída com base nessas ideias.

Nesse contexto, Lohmann e Ebert (2003) apresenta proposta de generalização dos *hyperspaces* usando meta-modelos, e Philippow, Riebisch e Boellert (2003) estendem a ideia para modelagem UML.

A ideia *hyperspace*, no sentido de uma abordagem multidimensional, é utilizado na definição do conceito de Metateoria de *Hypertypes* (Definição 5.10) (seção 5.6).

4.5 Sistematização das regras da UFO para formalização de ontologias

Dentre as contribuições principais da tese de Guizzardi (2005), além das categorias iniciais da UFO, destaca-se o *profile* da *Unified Modeling Language* 2.0 (UML) (OBJECT MANAGEMENT GROUP, 2003) construído com base nos conceitos da ontologia de fundamentação proposta por ele. Dessa forma, o autor atinge um objetivo de produzir uma linguagem para modelagem conceitual. O referido *profile*, criado como uma extensão à UML e nomeado *OntoUML*, permite que ontologistas construam modelos conceituais e, especialmente, ontologias de domínio em UML baseados na ontologia de fundamentação UFO. Uma das vantagens dessa abordagem é permitir que uma linguagem efetivamente popularizada entre profissionais e pesquisadores das áreas de tecnologia da informação e analistas de informação pudessem aproveitar o conhecimento prévio em UML para produzir modelos conceituais ontologicamente bem fundamentados. Conforme Guizzardi et al. (2011), a OntoUML foi proposta como:

> *[...] uma linguagem de modelagem conceitual que contempla como primitivas de modelagem as distinções ontológicas proposta pela ontologia UFO-A. Essa linguagem (atualmente chamada de OntoUML) foi construída seguindo um processo no qual: (i) o metamodelo da linguagem original (no caso, a UML 2.0) é reparado para garantir um isomorfismo em seu mapeamento para a estrutura definida pela ontologia de referência (no caso, UFO-A); (ii) em segundo lugar, a axiomatização da ontologia de fundamentação é transferida para o metamodelo da linguagem, por meio de restrições formais incorporadas a esse metamodelo. O objetivo dessa etapa é garantir que a linguagem só admitirá como modelos gramaticamente válidos aqueles modelos que satisfazem (do ponto de vista lógico) a axiomatização de UFO, ou seja, aqueles modelos que são considerados válidos segundo essa teoria. Essa linguagem também incorpora um conjunto de padrões de modelagem de ontologias (ontological design patterns) para solução de alguns problemas clássicos de modelagem no que diz respeito a, por exemplo, modelagem de papéis e modelagem de relações de constituição (GUIZZARDI et al., 2004), separação de qualidades dos espaços de valores de attributos (GUIZZARDI; MASOLO; BORGO, 2006), modelagem de quantidades (GUIZZARDI, 2010a), resolução do problema de transitividade da relação todo-parte (GUIZZARDI, 2009) e resolução do problema de colapso de restrições de cardinalidade* (GUIZZARDI; WAGNER, 2008).

A extensão à UML foi necessária porque essa linguagem não é suficientemente expressiva para que os conceitos da UFO pudessem ser representados de forma precisa, e também porque a UML é demasiadamente ambígua, e não possui fundamentos ou semântica ontológica bem formada. Isso é o caso, por exemplo, na noção de *association classes* e de *Interfaces* que, segundo Guizzardi (2005, p. 329, 333, 339), as primeiras são casos de ambiguidades da linguagem (uma vez que há mais de uma forma de se representar associações) e a segunda não possui sustentação ontológica.

Porém, nem mesmo com a referida extensão todos os conceitos da UFO podem ser expressos. Por exemplo, em UML não há claro suporte para que noções modais sejam representadas, o que força que alternativas sejam adotadas, como o uso de subtipos para representar tipos alternativos possíveis (GUIZZARDI et al., 2004, p. 20), a criação de "todos" (Whole) que tenham apenas uma única parte, entre outros (BENEVIDES, 2010, p. 27). De toda forma, o estudo da construção da OntoUML, quando se observa as limitações a respeito de modalidades, é de extrema valia aos objetivos desta pesquisa, pois se trata da proposição de uma linguagem de especificação de ontologias que se alinha com as propostas deste trabalho, uma vez que tanto Ontoprolog como OntoUML são linguagens baseadas em UFO. Além disso, por esse motivo, é desejável que existam traduções para as especificações (ou modelos) produzidas em uma linguagem na outra. Isso é justificável, por exemplo, porque OntoUML é uma linguagem visual, diagramática, que possui características intrínsecas desejáveis, como a facilidade de comunicação e de compartilhamento entre seres humanos, não tão ricas quanto linguagens textuais declarativas, como é o caso de Ontoprolog. Por outro lado, a facilidade de inferências lógicas e de raciocínio automatizado de Ontoprolog, inclusive modais, completa lacunas inerentes à linguagem visual, tornando uma linguagem um complemento da outra. De toda forma, é importante destacar que OntoUML não se limita a uma linguagem de diagramas. De fato, UML, a linguagem base de OntoUML não se trata apenas de uma linguagem de diagramas. A representação diagramática é apenas uma das formas de apresentar conceitos modelados em UML. Além disso, é possível utilizar regras OCL (*Object Constraint Language*) (ISO/IEC, 2012b) para descrever restrições à UML. Esse é o escopo do trabalho de Benevides et al. (2009), Benevides (2010), Guerson, Almeida e Guizzardi (2014), por exemplo, que sistematizam as regras da UFO em restrições OCL, além de apresentarem interessantes resultados na verificação de modelos modais com Alloy (JACKSON, 2012). O que se argumenta, no entanto, é que Ontoprolog é uma alternativa que complementa essas abordagens "tradicionais" de modelagem com UML.

Nesse contexto, a construção da OntoUML pode ser analisada a partir das interligações do metamodelo da UML com conceitos da UFO da Figura 9. Dessa forma, essencialmente o que se tem são a transformação de folhas da taxonomia da UFO em estereótipos e restrições para Classifiers da UML. A interligação dos conceitos da UFO com o metamodelo da UML é ilustrado pelas seguintes figuras, em que as entidades da OntoUML são representadas em cinza escuro e as da UML em branco:

a) Figura 16: descreve a interligação do metamodelo da UML com os conceitos da UFO referentes à categoria dos *Substance Universals*;

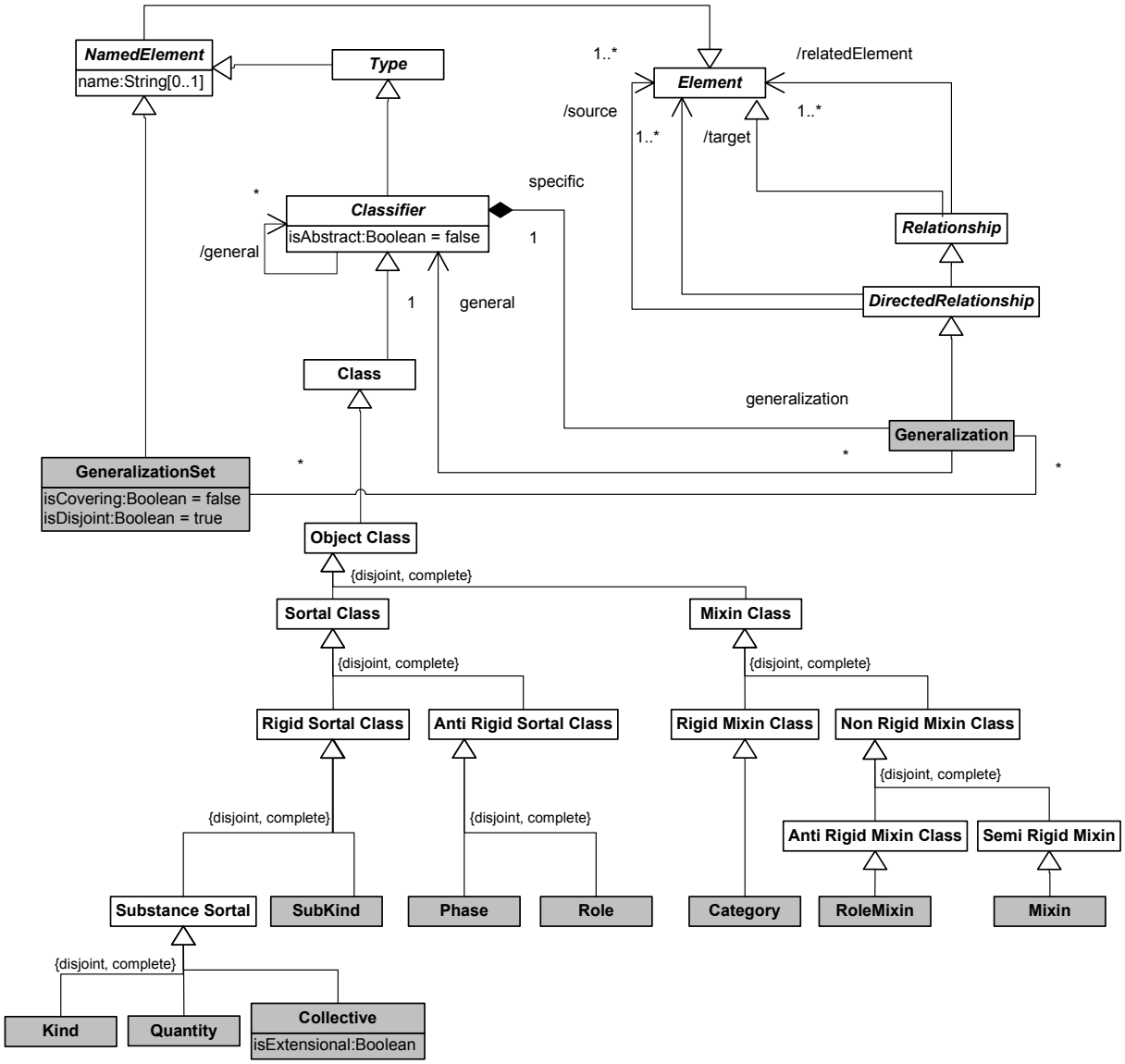
b) Figura 17[22]: responsável pela descrição da interligação de elementos da UML com as categorias *Relation*, *Moment* e categorias relacionadas;

c) Figura 18: apresenta a interligação das Meronymic relations.

[22] A seção Figura 17 sofreu adaptações que incluíram as entidades `SimpleDataType` e `StructuredDataType`.

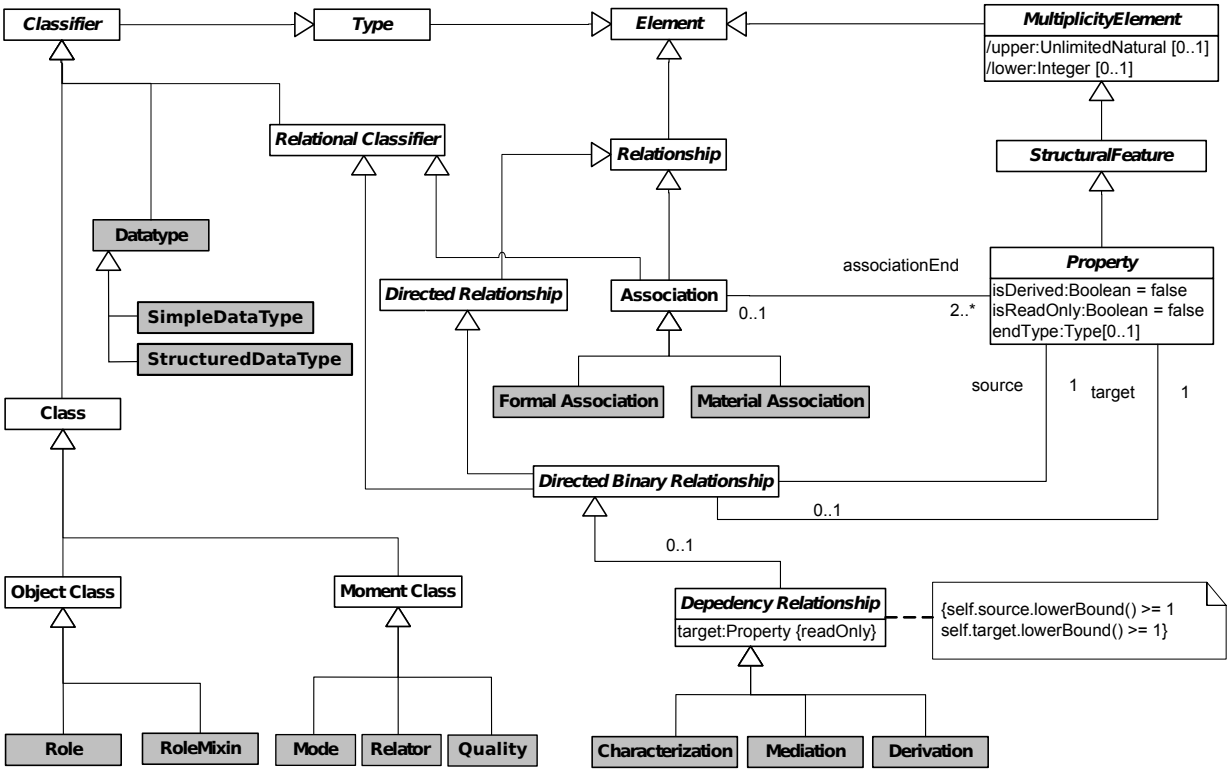
É importante destacar que, embora a Figura 59 contenha uma representação em UML, assim como a Figura 16, a Figura 17 e a Figura 18, a primeira faz uso da "linguagem unificada" para descrever os conceitos da UFO, enquanto as três últimas usa a linguagem UML para descrever o fragmento da própria linguagem UML que é estendida com o *profile* da UFO.

Figura 16 – Interligação do metamodelo da UML com os conceitos da UFO referentes à categoria dos *Substance Universals*



Fonte: Guizzardi (2005, p. 316)

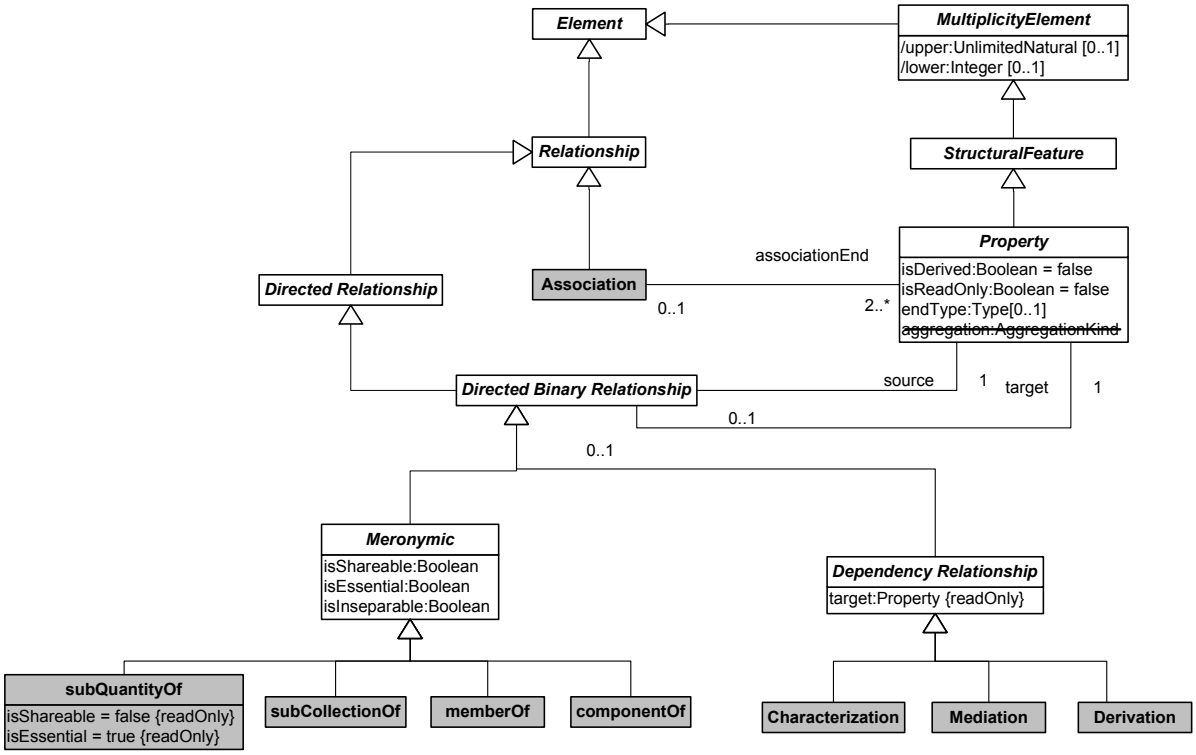
Figura 17 – Interligação do metamodelo da UML com os conceitos da UFO referentes às categorias *Relation*, *Moment* e categorias relacionadas



Fonte: Os autores

Notas: Adaptado de Guizzardi (2005, p. 334) com acréscimo das categorias `SimpleDataType`, `StructuredDataType` e `Quality`. `Quality` é adicionado baseado em Guizzardi e Zamborlini (2014)

Figura 18 – Interligação do metamodelo da UML com os conceitos da UFO referentes aos diferentes tipos de relações de meronímia



Fonte: Guizzardi (2005, p. 348)

4.5.1 Quadros de regras

Esta seção contém a sistematização das restrições semânticas da UFO. Em cada quadro, um conjunto de restrições são apresentadas sequencialmente. Os quadros são traduções, adaptações e extensões das tabelas presentes em Guizzardi (2005, p. 317-320, 334-338, 348-352), Benevides (2010, p. 36-39, 45-50, 55-58) e trabalhos posteriores. Nessa forma, tratam-se não apenas de uma transcrição, mas de resultado referente a sistematização dessas regras. Sempre que conveniente e adequado, referências a entradas do Glossário da UFO são providas, geralmente, entre parêntese.

As regras destes quadros encontram-se definidas em Ontoprolog, conforme a especificação contida no Apêndice C, no Apêndice F e no Apêndice G. O código da referida especificação indica como restrições dos quadros desta seção são implementadas em Ontoprolog.

Quadro 1 – Substance Sortal

Substance Sortal

Trata-se de uma meta classe abstrata que representa propriedades gerais – como rigidez (Rigidity) – de todos os Substance Sortals relacionalmente independentes de objetos universais que forneçam um princípio de identidade (Principle of identity) para suas instâncias. O Substance Sortal é um Rigid Sortal que não possui sintaxe concreta. Dessa forma, representações simbólicas são definidas apenas por suas subclasses concretas.

Tipo superior de

«kind» (Quadro 2);

«quantity» (Quadro 3);

«collective» (Quadro 4);

Restrições

1. Todo Substantial Object deve ser uma instância de um Substance Sortal, direta ou indiretamente. Isso significa que todo elemento concreto em um diagrama de classe deve estar numa hierarquia que inclui uma classe com um dos estereótipos: «kind», «quantity» ou «collective»;
2. Um Substantial Object não pode ser instância de mais do que um Substance Sortal;
3. Um Substance Sortal não pode incluir outro Substance Sortal nem um «subkind» (Quadro 5) em sua hierarquia de classes, ou seja, um Substance Sortal não pode ter uma superclasse membro de {«kind», «subkind», «quantity», «collective»};
4. Uma classe que representa um *rigid substantial universal* (Rigid Sortal) não pode ser uma subclasse de uma classe que representa um Anti Rigid Universal. Dessa forma, um Substance Sortal não pode ter como superclasse um membro de {«phase», «role», «roleMixin»}.

Quadro 2 – Kind

Kind

«kind»
A

Um «kind» (Kind) representa um Substance Sortal (Quadro 1) que provê princípio de identidade (Principle of identity) a suas instâncias (GUIZZARDI, 2005, p. 108).

Exemplos

Tipos Naturais (como Pessoa, Gato, Flor) e artefatos (Cadeira, Carro, Celular).

Subtipo de

Substance Sortal (Quadro 1)

Restrições

(todas as restrições do Quadro 1 se aplicam)

Quadro 3 – Quantity

Quantity

«quantity»
A

Um «quantity» (Quantity) representa um Substance Sortal (Quadro 1) no qual as instâncias são "quantidades". De acordo com Guizzardi (2005, p. 179), *"a piece of a quantity K is a maximally self-connected object constituted by portions"*.

Exemplos

Exemplos em linguagem natural incluem termos gerais de massa, como Ouro, Água, Areia, Argila.

Subtipo de

Substance Sortal (Quadro 1)

Restrições

1. Quantities sempre possuem a propriedade {extensional} (Extensional Individual), de modo que todas as suas partes são Essential Part (Quadro 24);
2. *(todas as restrições do Quadro 1 se aplicam)*.

Quadro 4 – Collective

Collective

«collective»
A

Um «collective» (Collective) representa Substance Sortals (Quadro 1) nas quais as instâncias são coleções de entidades conectadas por uma relação de unificação, que possuem uma estrutura uniforme (GUIZZARDI, 2005, p. 181).

Exemplos

Exemplos incluem baralhos de cartas, grupo de pessoas, pilha de tijolos.

Collectives geralmente se relacionam com Complexes por meio de uma *relação de constituição*.

Exemplos

Uma pilha de tijolos que *constitui* uma parede, um grupo de pessoas que *constituem* um time de futebol.

Nos casos de *relação de constituição*, os Collectives geralmente possuem um princípio de identidade extensional, em contraste aos Complexes que eles constituem.

Exemplos

A banda de rock dos anos 1960 *The Beatles* foi constituída por um pelo coletivo {`John`, `Paul`, `George`, `Pete`} e posteriormente alguns de seus membros foram alterados e o grupo passou a ser formado por {`John`, `Paul`, `George`, `Ringo`}. A substituição de Pete Best por Ringo Star não alterou a identidade da *banda*, mas criou um *grupo de pessoas* numericamente diferente.

Subtipo de

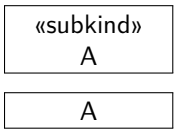
Substance Sortal (Quadro 1)

Restrições

1. Um Collective pode ser extensional. Nesse caso, o meta-atributo {extensional} (apresentado na Figura 16 como {extensional}) deve ser atribuído como `True`. Isso significa que todas as suas partes (Quadro 24) são Essential Part (propriedade {essential}) e que a mudança (ou a destruição) de qualquer de seus elementos encerra a existência do Collective.

Quadro 5 – Subkind

Subkind



Um «subkind» é uma restrição relacionalmente independente (Independence) de um Substance Sortals (Quadro 1) que carrega um princípio de identidade (Principle of identity) fornecido por este.

Exemplos

Um exemplo de *subkind* pode ser `PessoaHomem` e `PessoaMulher` como tipos de um *kind* `Pessoa`.

Subtipo de

Rigid Sortal

Restrições

1. Se um elemento em um modelo não possui um estereótipo, então ele deve ser considerado um «subkind»[a];
2. Como um tipo que representa um universal rígido não pode especializar (restringir) um tipo anti-rígido (GUIZZARDI, 2005, p. 103), um «subkind» não pode ter como tipo superior um membro de {«phase», «role», «roleMixin»};
3. Todo «subkind» deve estender um «kind»[b].

[a] Isso está de acordo com Guizzardi (2005, p. 314), que estabelece que classes sem estereótipos devem ser interpretadas como Substantial Universal, que é um tipo superior de Subkind.

[b] Esta regra era implícita pela própria definição de «subkind».

Quadro 6 – Phase

Phase

«phase»
A

Um «phase» (Phase) representa um Phased Sortal. Trata-se de um Anti Rigid Universal *relacionamente independente* (Independence) definido como parte de um partição de um Substance Sortal

Exemplos

As fases ⟨`Criança`, `Adulto`⟩ particionam um Kind `Pessoa`.

Subtipo de

Anti Rigid Universal

Restrições

1. Como um tipo que representa um universal rígido não pode especializar (restringir) um tipo anti-rígido (GUIZZARDI, 2005, p. 103), um «phase» não pode ser um tipo superior de um Rigid Universal, ou seja, não pode ser um tipo superior de {«kind», «subkind», «quantity», «collective», «category»}, nem de um «role» (Quadro 7);
2. As diferentes «phases» *ontologicamente dependentes*[a] devem ser agrupadas e representadas num diagrama de classes como um *generalization set* (subseção 4.4.3) do tipo disjunto e completo. Ou seja, Phases são representadas por meio do elemento `Generalization Set` da UML com as propriedades `isCovering = true` e `isDisjoint = true`.

[a] *Ontologicamente dependente* significa que Phases mantêm relação baseada no compromisso ontológico que representam. Isso significa que pode haver mais de um grupo de Phases que são dependentes de diferentes compromissos ontológicos.

Quadro 7 – Role

Role

«role»
A

Um «role» (Role) representa um Phased Sortal. Trata-se de um Anti Rigid Universal *relacionamente dependente* (Relational dependence). Um Role é uma restrição em um «kind» (Quadro 2) por meio de uma condição de restrição relacional.

Exemplos

O papel Estudante é exercido por uma instância de um Kind Pessoa. Desse modo, o papel de Estudante é parte da definição final do conceito de Pessoa (nesta conceituação).

Subtipo de

Anti Rigid Universal

Restrições

1. Como um tipo que representa um universal rígido não pode especializar (restringir) um tipo anti-rígido (GUIZZARDI, 2005, p. 103), um «role» não pode ser um tipo superior de um Rigid Universal, ou seja, não pode ser um tipo superior de {«kind», «subkind», «quantity», «collective», «category»}, nem de um «phase» (Quadro 6);
2. Toda classe com o estereótipo «role» deve estar conectada a um associação de uma relação do tipo «mediation» (Quadro 19).

Quadro 8 – Mixin Class

Mixin Class

A *Mixin class* (Mixin Universal) é uma metaclasse abstrata que representa propriedades gerais de todos os Mixins. *Mixin classes* não possuem sintaxe concreta. Dessa forma, representações simbólicas são definidas apenas por suas subclasses concretas.

Tipo superior de

«category» (Quadro 9);

«roleMixin» (Quadro 10);

«mixin» (Quadro 11);

Restrições

1. Classes que representam Non Sortals não podem ser subclasses de uma classe que representa um Sortal. Como consequência, um *mixin class* não pode ter como tipo superior um membro de {«kind», «quantity», «collective», «subkind», «phase», «role»};
2. Qualquer classe que instanciar algum dos Non Sortals não podem ter instâncias diretas. Dessa forma, qualquer instância de um dos *mixin class* deve sempre ser uma classe abstrata ({abstract}).

Quadro 9 – Category

Category

«category»
A

Um «category» representa um Rigid Mixin relacionamente dependente (Relational dependence), ou seja, trata-se de um Dispersive Sortal que agrega propriedades essenciais que são comuns a diferentes Substance Sortals (Quadro 1).

Exemplos

A categoria `EntidadeRacional` como uma generalização de `Pessoa` e `AgenteInteligente`.

Subtipo de

Mixin class (Quadro 8)

Restrições

1. Pelo fato de representar um universal rígido (que não pode especializar ou restringir um tipo anti-rígido[a]), e também pela combinação das regras do Quadro 8, um «category» não pode ter como tipo superior um membro de {«kind», «quantity», «collective», «subkind», «phase», «role», «roleMixin»};
2. Um «category» sempre é definido como uma classe abstrata ({abstract});
3. Um «category» só pode ser subdividido em outro «category» ou em um «mixin».

[a] (GUIZZARDI, 2005, p. 103)

Quadro 10 – RoleMixin

RoleMixin

«roleMixin»
A

Um «roleMixin» representa um Anti Rigid Mixin externamente dependente (External Dependence), ou seja, um Dispersive Sortal que agrega propriedades essenciais a diferentes Roles (Quadro 7).

Exemplos

Papeis formais como *todo*, *parte* e *iniciador*.

Subtipo de

Mixin class (Quadro 8)

Restrições

1. O tipos superiores de um «roleMixin» não pode ser um membro de {«kind», «quantity», «collective», «subkind», «phase», «role»};
2. Um «roleMixin» sempre é definido como uma classe abstrata;
3. Toda classe com o estereótipo «roleMixin» deve estar conectada a um associação de uma relação do tipo «mediation» (Quadro 19);
4. As restrições de cardinalidade descritas na subseção 4.4.7.

Quadro 11 – Mixin

Mixin

«mixin» A	Um «mixin» (Mixin) representa um conjunto de propriedades que são essenciais para algumas de suas instâncias e acidentais para outras.

Exemplos

Um exemplo é o Mixin `Sentável`, que representa uma propriedade que pode ser considerada essencial para *kinds* (Quadro 2) como `Cadeira` e `Banco`, mas são acidentais para `Engradado`, `CaixaDePapel` ou `Pedra`.

Subtipo de

Mixin class (Quadro 8)

Restrições

1. Um «mixin» não pode ter um «roleMixin» como um tipo superior;
2. Um «mixin» sempre é definido como uma classe abstrata;

Quadro 12 – Moment

Moment

Trata-se de uma meta classe abstrata que representa propriedades gerais de todos os Moment Universals. Moment não possui sintaxe concreta. Dessa forma, representações simbólicas são definidas apenas por suas subclasses concretas.

Tipo superior de

Intrinsic Moment Universal

«quality» (Quadro 13);

«mode» (Quadro 17);

«relator» (Quadro 18);

Restrições

1. Todo Moment deve (direta ou indiretamente) estar conectado a uma associação com pelo menos uma relação do tipo «characterization» (Quadro 20);

Quadro 13 – Quality

Quality

«quality»
A

Um «quality» (Quality Universal) representa um Intrinsic Moment relacionado via Association Relation a um Quality Structure e é existencialmente dependente (Existential Dependence) de exatamente uma entidade.

A base teórica deste quadro é discutida na subseção 4.4.4. Ver também Quadro 17.

Subtipo de

Moment Universal (Quadro 12)

Intrinsic Moment Universal

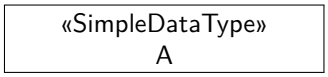
Restrições

1. Todo tipo «quality» deve estar associado a ao menos um Quality Structure via Association Relation;
2. Toda instância de «quality» deve estar associado (Inherence) a um Endurant, no sentido que o Quality Particular é inerente ao Endurant, o que simétrico ao Endurant possuir o Quality Particular (Bearer of a Moment);
3. Os Qualities podem ser representados via Attribute Functions[a];
4. Uma Attribute Function deve ser representada como propriedade do Classifier que a possui, no caso de o co-domínio da função ser um Quality Dimension (Quadro 14);
5. Uma Attribute Function deve ser representada como relação direcionada partindo do Classifier ao co-domínio, no caso de o co-domínio da função ser um Quality Domain (Quadro 15);
6. Qualities instanciam um único Quality Universal;

[a] (GUIZZARDI, 2005, p. 327).

Quadro 14 – QualityDimension como SimpleDataType

QualityDimension como SimpleDataType



Os Quality Dimensions tratam-se de Quality Structures na forma de grandezas que fazem parte de Quality Domains (Quadro 15). Em conjunto com Quality Domains, representam o Conceptual Space de Qualities (Quadro 13).

Exemplo

Na modelagem do espaço de cores RGB, os componentes R, G e B são três dimensões que compõe o domínio em tela.

Subtipo de

Quality Structure

Restrições

1. Em diagramas OntoUML, todo Quality Dimension pode ser representado como uma propriedade de um Quality Domain (Quadro 15), ou como uma «SimpleDataType» caso não esteja associado a um domínio;
2. Todo Quality Dimension deve estar vinculado a um Quality Domain ou a associado (Association Relation) a um «quality» ou Attribute Function (Quadro 13).

Quadro 15 – QualityDomain como StructuredDataType

QualityDomain como StructuredDataType

«StrutucturedDataType»
A

Os Quality Domains tratam-se de Quality Structures na forma de composições de Quality Dimensions (Quadro 14). Em conjunto com Quality Dimensions, representam o Conceptual Space de Qualities (Quadro 13).

Exemplo

Na modelagem do espaço de cores formado pelos componentes R, G e B, o domínio RGB são as três dimensões em tela e as respectivas restrições do espaço de valores conceituais que regem a configuração das dimensões.

Subtipo de

Quality Structure

Restrições

1. Em diagramas OntoUML, as dimensões do Quality Domain devem ser representadas como propriedades do «StructuredDataType»;
2. Todo Quality Domain deve estar vinculado a *pelo menos um*[a] «quality», ou ser o co-domínio de ao menos uma Attribute Function (Quadro 13);
3. Todo Quality Domain pode conter restrições lógicas sobre as relações das Quality Dimensions que o compõe, escritas fora do escopo da linguagem de representação[b].

[a] Ver observações a respeito desta regra na subseção 4.4.4.
[b] Essas restrições representam matematicamente o Conceptual Space.

Quadro 16 – Quale

Quale

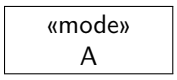
O Quale é representação de um ponto no Conceptual Space representado pelo Quality Domain ou Quality Dimension. Trata-se de um conceito que diz respeito apenas a Particulars.

Restrições

1. Um Quale é relacionado via Attribute Function (Quadro 13) a uma instância de um Endurant e a um *valor abstrato* contido no Conceptual Space do Quality Structure co-domínio da Attribute Function;
2. O valor do Quale deve atender às restrições impostas pelo Quality Structure ao qual se refere.

Quadro 17 – Mode

Mode



Um «mode» é o estereótipo de um Mode Universal, um tipo de Intrinsic Moment Universal. Toda instância de um Mode Universal é existencialmente dependente (Existential Dependence) de exatamente uma entidade.

Ver também Quadro 13.

Exemplos

Habilidades, pensamentos, crenças, intenções, sintomas.

Subtipo de

Moment Universal (Quadro 12)

Intrinsic Moment Universal

Restrições

1. Todo Moment deve (direta ou indiretamente) estar conectado a uma associação com pelo menos uma relação do tipo «characterization» (Quadro 20);

Quadro 18 – Relator

Relator

«relator»
A

Um «relator» (Relator Universal) é um Moment Universal, cujo toda instância é existencialmente dependente (Existential Dependence) de ao menos duas entidades. Relators são as instâncias de propriedades relacionais.

Exemplos

Casamento, beijo, aperto de mão, acordos, compras.

Subtipo de

Moment Universal (Quadro 12)

Restrições

1. Todo «relator» deve (direta ou indiretamente) estar conectado a uma associação com pelo menos uma relação «mediation» (Quadro 19);
2. Todo relator deve mediar ao menos duas entidades diferentes, dessa forma, seja R uma classe com o estereótipo «relator»; $\{C_1, \ldots, C_n\}$ um conjunto de Universals relacionados a R via uma relação «mediation» (Universals mediados por R); $lower_{C_i}$ sendo $(1 \leq i \leq n)$ o valor mínimo da restrição de cardinalidade da associação conectada à C_i na relação «mediation». Dessa forma, tem-se a restrição:

$$(\sum_{i=1}^{n} lower_{c_i}) \geq 2$$

3. Além da restrição anterior, adicionalmente, ao menos dois Universals mediados por R relacionados ao «relator» devem ser diferentes;
4. Todo Moment deve (direta ou indiretamente) estar conectado a uma associação com pelo menos uma relação do tipo «characterization» (Quadro 20);

Quadro 19 – Mediation

Mediation

«mediation»	Um «mediation» é uma relação formal (Formal Relation) entre um Relator Universal, portanto entre um relação do tipo «relator» (Quadro 18) e um Endurant Universal.

Exemplos

Um Universal `Casamento` media os Roles `Marido` e `Esposa`; o Universal `Matriculado` media `Estudante` e `Universidade`;

Subtipo de

Formal Relation

Restrições

1. As associações do tipo «mediation» devem ser binárias;
2. Uma associação com o estereótipo «mediation» deve ter como origem da associação uma classe com o estereótipo «relator» (Quadro 18) que represente o Relator Universal correspondente;
3. A restrição de cardinalidade dos dois lados da associação do «mediation» deve ser igual ou maior a 1[a];
4. A associação entre o «mediation» e o Endurant Universal deve ter a propriedade `isReadOnly = true`;

[a] Esta regra equivale à combinação das regras 2 e 4 das *constraints* do estereótipo «mediation» de Guizzardi (2005, p. 335-336).

Quadro 20 – Characterization

Characterization

«characterization» Um «characterization» é uma relação formal Formal Relation entre um Mode Universal e um Endurant Universal.

Exemplos

Os Universals `NivelAlegria` e `ComprimentoPerna` caracterizam um Universal `Pessoa`.

Subtipo de

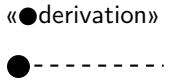
Formal Relation

Restrições

1. As associações do tipo «characterization» devem ser binárias;
2. Uma associação com o estereótipo «characterization» deve ter como origem da associação uma classe com o estereótipo «mode» (Quadro 17) que represente o Mode Universals correspondente;
3. A restrição de cardinalidade do lado da associação do Endurant Universal deve ser exatamente 1;
4. A restrição de cardinalidade do lado da associação do «mode» deve ser igual ou maior a 1;
5. A associação entre o «characterization» e o Endurant Universal deve ter a propriedade `isReadOnly = true`;

Quadro 21 – Derivation Relation

Derivation Relation



Um «●derivation», ou «●derivationRelation»[a], representa a relação formal (Formal Relation) de derivação (Derivation Relation) entre alguma relação material (Material Relation) e um Relator Universal – ou seja, uma relação com o estereótipo «relator» (Quadro 18) – na qual essa relação material é derivada.

[a] Originalmente, Guizzardi (2005, p. 337) não usou um nome entre as marcas de estereótipo «e »(em francês, *"guillemets"*), mas tão somente *"Derivation Relation"*. Isso corre porque a entidade em tela já possui um sinal gráfico representativo. Dessa forma, o nome entre a notação de estereótipo é desnecessária. Porém, a título de padronização, mantemos as duas notações, de modo que elas se tornem intersubstituíveis. Nesse caso, o *bullet* (●) no nome do estereótipo indica o lado da relação com o círculo fechado na representação gráfica.

Exemplos

A relação material `casado-com`, que é derivada do Relator Universal `Casamento`; a relação material `beijado-por`, derivada de um Relator Universal `Beijo`; a relação material `comprado-de`, derivada de um Relator Universal `Compra`.

Subtipo de

Formal Relation

Restrições

1. As associações do tipo «derivation» devem ser binárias;
2. O lado do círculo negro deve estar conectado a um Relator Universal – ou seja, conectado a uma relação com o estereótipo «relator» (Quadro 18) – e o outro a uma relação material «material» (Quadro 23);
3. A restrição de cardinalidade do lado do círculo negro deve ser exatamente 1;
4. O lado do círculo negro deve ter a propriedade `isReadOnly = true`;
5. A restrição de cardinalidade do lado da associação conectado à relação material deve ser igual ou maior a 1. Porém, observar as regras adicionais na subseção 4.4.7;

Quadro 22 – Formal

Formal

«formal»	Uma associação com o estereótipo «formal» é uma Formal Relation que representa uma relação comparativa (Comparative Formal Relation) derivada de propriedades intrínsecas das entidades ou relação interna (Basic Formal Relation, Internal Relation).

Exemplos

`Pessoa é mais vela do que Pessoa`; um `Átomo é mais pesado do que outro Átomo`;

Subtipo de

Formal Relation

Observações

1. Usa-se o símbolo / da UML de relações *derivadas* para representar relações comparativas.

Quadro 23 – Material

Material

«material»	Uma associação «material» representa um Material Relation, ou seja, uma relação que é induzida por um Relator Universal (Quadro 18).

Exemplos

Um `Estudante estuda em Universidade`; `Paciente é tratado em Unidade Médica`; `Pessoa é casada com Pessoa`.

Restrições

1. As associações do tipo «material» devem estar associadas a exatamente um Derivation Relation, ou seja, à uma classe com o estereótipo «derivation» (Quadro 21);
2. (seção 7.5) As associações do tipo «material» devem ter a propriedade {derived};
3. A restrição de cardinalidade dos dois lados da associação «material» deve ser igual ou maior a 1. Porém, observar as regras adicionais na subseção 4.4.7;
4. As instâncias da associações do tipo «material» são existencialmente dependentes do «relator» ao qual estão relacionadas via «derivation».

Quadro 24 – Meronymic

Meronymic

Trata-se de uma meta classe abstrata que representa propriedades gerais de todas as relações de meronímias (Meronymic relation), ou seja, de relações do tipo Part of. Meronymic relation não possui sintaxe concreta. Dessa forma, representações simbólicas são definidas apenas por suas subclasses concretas.

Tipo superior de	**Meta-propriedades**
«componentOf» (Quadro 25)	Irreflexividade
«subQuantityOf» (Quadro 26)	Anti-simetria
«subCollectionOf» (Quadro 27)	Não-transitividade
«memberOf» (Quadro 28)	Weak Supplementation

Restrições válidas para todas as relações derivadas de Part of

1. ***Weak Supplementation***: Seja U um Universal cujas instâncias são Wholes e seja $C_1, \ldots, C_2$ um conjunto dos universais relacionados a U via relações de agregação que funcionam como partes de U. Seja $lower_{C_i}$ o valor mínimo da restrição de cardinalidade da associação conectada à C_i na relação de agregação. Dessa forma, tem-se a seguinte restrição:

$$(\sum_{i=1}^{n} lower_{c_i}) \geq 2$$

2. ***Essential Parthood*** (Essential Part) e ***Immutable Part***: O atributo {essential} indica que a parte é essencial ao todo. O atributo {immutable}[a] diz respeito à parte ser essencial a um todo Anti Rigid.
   a) No caso de um Classifier conectado no lado associação que representa o Whole ser um Anti Rigid, então o valor atributo {essential} da parte precisa ser `False`, ao passo que o atributo {immutable} (Immutable Part) pode, contingencialmente, ser `True`;
   b) Porém, se {essential} recebe `True` (no caso de um Classifier rígido com partes essenciais), então {immutable} deve também receber `True`;
   c) Sempre que um Whole receber o atributo {extensional}, o que sempre é o caso para «quantity» (Quadro 3) e contingencialmente para «collective» (Quadro 4), então o terminal da parte deve receber o atributo {essential} e, consequentemente, {immutable};
3. ***Inseparable Parthood*** (Inseparable Part): O atributo {inseparable} indica que o todo é essencial à parte, portanto, a cardinalidade mínima no lado do Whole deve ser 1;
4. ***Shareable Parthood*** (Shareable): O atributo {shared} indica se a parte pode ser relacionada a mais do que um Whole do mesmo tipo, ou seja, se não é uma Exclusive part;
5. ***Mandatory part*** **e** ***Mandatory Whole***: O atributo {mandatory} é automaticamente inferido sempre que a restrição mínima de cardinalidade for 1. No caso, trata-se de Mandatory part se a restrição mínima for no lado da parte, ou de Mandatory Whole no caso de a restrição mínima estar definida no lado do Whole;

[a] Em UML padrão, o atributo {immutable} refere-se à propriedade `isReadOnly` no no terminal da relação que se refere à parte.

Quadro 25 – ComponentOf

ComponentOf

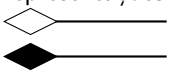
A relação «componentOf» (Component of) trata-se de Meronymic relation estabelecida entre dois Complexes. Elas podem ser compartilháveis ou não compartilháveis Quadro 24.

Exemplos

A mão que é parte do braço; o motor que é parte do carro; o coração que é parte do sistema circulatório.

Representações visuais

Como este é o tipo de relação mereológica mais usada em modelagem conceitual, usa-se as representações tradicionais de agregação e composição da UML:



Para relações do tipo «componentOf» compartilháveis

Para relações do tipo «componentOf» não compartilháveis

Subtipo de

Meronymic (Quadro 24);

Meta-propriedades

Irreflexividade

Anti-simetria

Não-transitividade (transitivo nos casos apresentados na Component-Functional Complex Relation do Glossário da UFO)

Weak Supplementation

Restrições

1. (Definição formal 8.5 e Definição formal 8.6) As classes ligadas em ambos os lados da associação «componentOf» devem representar Universals cujas instâncias são Functional Complexes. Para isso, um Universal X deve atender às seguintes condições:
   a) Se X é um Sortal Universal, então ele deve ter o estereótipo «kind» (Quadro 2) ou ser um subtipo de uma classe que tenha esse estereótipo;
   b) Caso contrário, se X é um Mixin Universal, para todas as classes Y, tal que Y seja subtipo de X, Y não pode ter o estereótipo «quantity» (Quadro 3) nem «collective» (Quadro 4), nem ser um subtipo de uma classe com um desses estereótipos.
2. Pode ser modificado com as meta propriedades {essential}, {immutable}, {inseparable}, {shared}, conforme Quadro 24.

Quadro 26 – SubQuantityOf

SubQuantityOf

A relação «subQuantityOf» (Subquantity of) trata-se de relação mereológica estabelecida entre duas Quantities (Quadro 3).

Exemplos

Álcool é parte de Vinho; Plasma é parte do Sangue; Açúcar é parte do Sorvete

Representação visual

Subtipo de

Meronymic (Quadro 24);

Meta-propriedades

Irreflexividade

Assimetria

Transitividade

Strong Supplementation (Extensional Mereology)

Restrições

1. As classes conectadas aos dois lados da associação desta relação devem representar Universals cujas instâncias são Quantities ou estendem uma classe com o estereótipo «quantity».
2. Pelo fato de quantidades serem maximais, o valor máximo da restrição de cardinalidade da associação conectada à parte deve ser 1;
3. Pelo mesmo motivo, um Quantity não pode ter como parte outro Quantity do mesmo tipo[a];
4. Esta relação sempre é não-compartilhável ({shared} sempre recebe a valoração `False`);
5. Todas as entidades com o estereótipo «quantity» (Quadro 3) são Extensional Individuals ({extensional}) e, desse modo, todas as relações envolvendo Quantities são marcadas com a meta propriedade {essential}, como consequência, o terminal da associação ligado à parte deve ser imutável {immutable};

[a] Conforme Guizzardi (2010a): *"a type stereotyped as «quantity» in this work stands for a maximally-connected-amount-of-matter. Since a quantity is maximal, it cannot have as a part a quantity of the same kind."*

Quadro 27 – SubCollectionOf

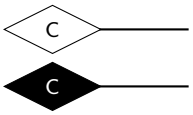
SubCollectionOf

A relação «subCollectionOf» (Subcollection of) trata-se de relação mereológica estabelecida entre dois Collectives.

Exemplos

A parte norte do Parque da Cidade é parte do Parque da Cidade; A coleção de "coringas" de uma carta de baralho é uma parte da coleção de cartas de baralho; uma coleção de homens em uma multidão é um coletivo parte da multidão

Representações visuais



relação «subCollectionOf» compartilhável

relação «subCollectionOf» não-compartilhável

Subtipo de

Meronymic (Quadro 24);

Meta-propriedades

Irreflexividade

Assimetria

Transitividade

Weak Supplementation (Minimal Mereology)

Restrições

1. As classes conectadas aos dois lados da associação desta relação devem representar Universals cujas instâncias são Collectives ou são um subtipo de uma classe com esse estereótipo;
2. O valor máximo da restrição de cardinalidade da associação conectada à parte deve ser 1[a];
3. Um «collective» não pode ter como parte outro «collective» do mesmo tipo. Como consequência, tratam-se de relações irreflexivas no nível de tipo;
4. Todos os subcoletivos são relações do tipo Inseparable Part, de modo que essas relações são marcadas como estereótipo {inseparable} e o terminal da associação colectado com o "todo" precisa ser {immutable};
5. Se um indivíduo x é uma Essential Part ({essential}) de um todo W, então todo W' que for subcoleção de W terá esse x como uma parte essencial.

[a] Conforme Guizzardi (2011): *"because collectives are maximal entities, a collective can have at maximum one subcollective of a given type. For this reason, the maximum cardinality constraint in the association end connected to the part in this relation must be one"*.

Quadro 28 – MemberOf

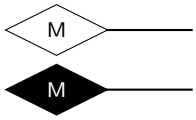
MemberOf

A relação «memberOf» (Member of) trata-se de relação mereológica estabelecida entre um Complex ou um Collective (que funciona como uma parte) e um Collective (que funciona como um Whole).

Exemplos

Uma árvore é parte de uma floresta; uma carta é parte de um baralho; um membro de um clube é parte do membro.

Representações visuais



relação «memberOf» compartilhável

relação «memberOf» não-compartilhável

Subtipo de

Meronymic (Quadro 24);

Meta-propriedades

Irreflexividade

Assimetria

Intransitividade

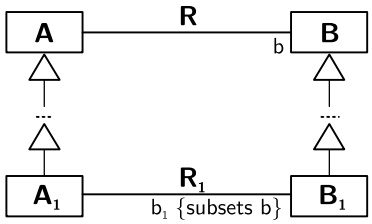
Weak Supplementation (Minimal Mereology)
Embora transitividade não seja o caso entre dois membros de relações «memberOf», uma relação «memberOf» seguida de outra «subCollectionOf» (Quadro 27) é transitiva. Ou seja, para quaisquer a, b, c, se $memberOf(a, b)$ e $memberOf(b, c)$, então $\neg memberOf(a, c)$, mas se $memberOf(a, b)$ e $subCollectionOf(b, c)$, então $memberOf(a, c)$.

Restrições

1. O Classifier conectado no lado da relação que representa o Whole deve representar um Universal cujas instâncias são Collectives, ou estendem uma classe com o estereótipo «collective»;
2. (Definição formal 8.7 e Definição formal 8.8) O Classifier conectado ao lado da relação que representa a parte deve representar um universal cujas instâncias são Collectives ou Functional Complexes;
3. Esta relação só pode representar relações parte-todo essenciais (Essential Part), marcadas com o atributo {essential}, se o objeto que representa o Whole nessa relação é um Extensional Individual (ou seja, se possui a propriedade {extensional});
4. No caso de o Whole possuir a propriedade {extensional}, então todas as relações parte-todo nas quais esse Extensional Individual participa são relações essenciais (Essential Part).

Quadro 29 – Relação *Subsetting*

Relação *Subsetting*



Subsetting relation é uma relação de subconjunto, que define que as instâncias relacionadas à relação do subconjunto são todas instâncias relacionadas à relação refinada. Trata-se de uma relação definida para os terminais das relações.

Uma relação de subconjunto deve ser definida entre duas relações materiais (Material Relation) R_2 e R_1, para (R_2 $subsets$ R_1) se, e somente se, essas relações são derivadas de Relators (Quadro 18) de tipos disjuntos e existe uma restrição de inclusão que inclui a extensão de R_2 na extensão de R_1 (COSTAL; GÓMEZ; GUIZZARDI, 2011a, postulate 1).

Exemplos

A relação material `matrícula`, que relaciona `Aluno` e `Disciplina ofertada`, é um subconjunto da relação material `Tem preferência por` entre `Aluno` e `Disciplina`, se `Disciplina ofertada` é uma extensão de `Disciplina`[a].

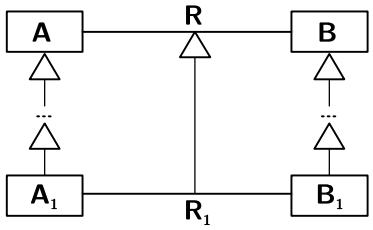
[a] Adaptado de Costal, Gómez e Guizzardi (2011a).

Restrições

1. Toda entidade ligada à relação refinadora deve também ser instância, ou instância de um tipo mais específico, da relação refinada (restrição de inclusão), de modo que a extensão da relação refinada inclua a extensão da relação refinadora;
2. A relação refinadora deve instanciar o mesmo *power type* da relação refinada (subseção 4.4.9);
3. Os limites inferior e superior das restrições de cardinalidade da relação refinadora podem ser iguais ou mais restritos do que as restrições da relação refinada;
4. Considerando a figura neste quadro, A_1 pode ser a mesma classe que A e/ou B_1 pode ser a mesma classe que B e/ou A pode ser a mesma classe que B (caso recursivo), de modo que em ambos os lados das relações refinadoras devem participar as entidades ou especializações das entidades refinadas.

Quadro 30 – Relação *Association Specialization*

Relação *Association Specialization*



A Specialization relation (for relations) é uma relação taxonômica entre uma associação mais específica de outra mais geral. A relação específica herda as características da geral. Em contrate com Subsetting relation (Quadro 29), a relação de especialização entre relações é um constructo definido para as relações em si, e não para os terminais.

Uma relação de especialização deve ser definida entre duas relações materiais (Material Relation) R_2 e R_1 ($R_1 \Leftarrow R_2$) se essas duas relações são derivadas a partir de dois Relators (Quadro 18) RR_1 e RR_2, tal que RR_2 especializa RR_1 ($RR_1 \Leftarrow RR_2$) (COSTAL; GÓMEZ; GUIZZARDI, 2011a, postulate 2).

Exemplos

A relação material `SentençaPronunciada` entre um `Tribunal` e um Réu pode ser especializada na relação `Absolve` entre um `Tribunal` e uma classe `Absolvido` que especializa Réu[a].

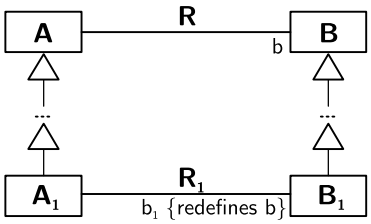
[a] Adaptado de Costal, Gómez e Guizzardi (2011a).

Restrições

(todas as restrições do Quadro 29 se aplicam)

Quadro 31 – Relação *Association Redefinition*

Relação *Association Redefinition*



A função da Redefinition relation é estabelecer uma nova definição para um ou mais terminais de outra associação de modo específico a determinado contexto. No caso, a nova definição trata-se de algo significativo no domínio da ontologia descrita. Um ou ambos os terminais de uma associação podem ser redefinidos e redefinições sempre envolvem duas associações.

Uma relação de redefinição deve ser estabelecida entre duas relações materiais R_2 e R_1 ($R_2 redefines R_1$) se: (i) ambas as relações são derivadas do mesmo Relator (Quadro 18) RR; (ii) existe um tipo A_i conectado a um dos terminais de R_2 tal que A_1 é uma especialização de A ($A \leftarrow A_1$) e A está conectado ao terminal de R_1 equivalente ao de A_1 (COSTAL; GÓMEZ; GUIZZARDI, 2011a, postulate 3).

Restrições

1. A relação refinadora deve instanciar o mesmo *power type* da relação refinada (subseção 4.4.9);
2. Os limites inferior e superior das restrições de cardinalidade da relação refinadora podem ser iguais ou mais restritos do que as restrições da relação refinada;
3. Considerando a figura neste quadro, B_1 pode ser a mesma classe que B e/ou A pode ser a mesma classe que B e/ou nome de b_1 pode ser o mesmo nome de b. Porém, a entidade conectada no lado da relação redefinadora deve ser um subtipo da relação refinada. Caso contrário, não seria possível explicar qual o novo papel dos indivíduos participantes da entidade refinada;

4.6 Fechamento

Os quadros presentes na subseção 4.5.1 são extensões dos encontrados na literatura UFO. Eles representam sistematizações de regras produzidas com base nas definições descritas no Glossário da UFO, além de outras referências bibliográficas. Portanto, embora representem um marco teórico importante para os capítulos posteriores, eles tratam-se de resultados desta pesquisa. As regras contidas nos quadros, e os tópicos de modelagem conceitual discutidos na subseção 4.5.1, são utilizados como referência para a produção das regras e das definições semânticas contidas no Capítulo 6 e no Capítulo 8.

Devido a limitações de escopo, temas importantes referentes à UFO não são abordados neste capítulo. Como exemplo, menciona-se os tópicos a respeito ade padrões e anti-padrões (GUIZZARDI, 2009; GUIZZARDI; GRAÇAS; GUIZZARDI, 2011; GUIZZARDI, 2014; SALES; BARCELOS; GUIZZARDI, 2012), e de ontologia em geral, como a questão do nível ontológico de Guarino (1995) e de Guarino (2009). No entanto, com base no marco teórico produzido nesta pesquisa, trabalhos futuros podem incorporar essas e outras características de modelagem conceitual no arcabouço lógico desenvolvido na Parte III.

No que se refere à Arquitetura da Informação, parece razoável argumentar a favor da utilização da UFO no projeto de sistematização de conceitos da AI (Capítulo 2). Ocorre que um dos temas ainda em aberto da AI é justamente o da ausência de uma linha de base teórica e conceitual que delimite as fronteiras da área. A ontologia de fundamentação provida pela UFO poderia ser utilizada como referência, e até estendida, para que conceitos próprios da AI sejam explicados, como as ideias de *espaço de informação*, *experiência do sujeito com a informação*, *configuração da informação* e até o próprio conceito de *informação*, entre tantos outros. Diante dessa possibilidade, o marco teórico produzido por este capítulo pode ser utilizado como referência para realização desse projeto.

Parte III

Resultados

5 Ontoprolog: Visão geral

Este capítulo é um prólogo dos resultados desta pesquisa. Seu objetivo é triplo: apresentar uma visão geral de Ontoprolog; introduzir conceitos lógicos e ontológicos discutidos nos capítulos posteriores; e apresentar determinadas adaptações realizadas na UFO. Detalhes técnicos, como a semântica formal e a sintaxe de Ontoprolog, são descritos nos capítulos posteriores.

A palavra "ontologia"[1] é utilizada nesta parte do trabalho para denotar um sistema metafísico de categorias sobre o mundo, ou uma instância específica de algum desses sistemas de categorias. Neste último caso, uma "ontologia" é equivalente a uma teoria, uma especificação ou configuração de entidades representada em uma linguagem. No primeiro, a palavra é usada para denotar genericamente esses sistemas. Caso se requeira o sentido da palavra "Ontologia" com outro significado, como por exemplo para denotar área ou disciplina de estudo, isso é explicitamente positivado no texto.

O restante deste capítulo é organizado do seguinte modo. A seção 5.1 descreve a abordagem adotada ao tratamento de ontologias com base em Arquitetura da Informação, UFO e Programação em Lógica. Nessa seção, central nesta obra, desenvolve-se a visão de mundo adotada pelos autores e introduz-se o nome *Ontoprolog* com designador dos diferentes arcabouços desenvolvidos nesta pesquisa. A seção 5.2 introduz aspectos referentes à sintaxe e à semântica adotada em Ontoprolog. Esses temas são retomados no Capítulo 6 e no Capítulo 7. A seção 5.3 apresenta os conceitos de *Entidade ontológica* e de *Entidade lógica*, e os fundamentos utilizados para defini-los. Esses conceitos são utilizados para distinguir objetos cuja semântica é ontológica, portanto representam algo do universo do discurso, de objetos lógicos, cuja semântica é formal e definida em termos de construções lógicas. A seção 5.4 elenca e descreve as relações predicativas primitivas utilizadas na semântica da sintaxe de Ontoprolog. A seção 5.5 discute as noções de *Nível teórico* e de *Nível teórico superior*. Essas ideias estão relacionadas com aspectos de metamodelagem introduzidos na linguagem. A seção 5.6 descreve a *Metateoria de* Hypertypes. Ela representa o arcabouço conceitual base inerente a Ontoprolog, e serve de teoria subjacente a toda especificação criada com a linguagem. A seção 5.7 continua a discussão sobre ontologias subjacentes e descreve como ontologias de domínio são modeladas com base em ontologias mais fundamentais, bem como provê definições sobre os conceitos de "Ontologia de domínio", "Ontologia de referência ou de fundamentação", entre outras. A seção 5.8 lista as principais adaptações na ontologia UFO realizadas neste trabalho. Trata-se de justificativas e de comentários referentes aos trabalhos originais da UFO, e a abordagem adotada nesta pesquisa. A seção 5.9 apresenta breves discussões referentes a interpretações dadas às

[1] A questão polissêmica do termo "ontologia" é abordado na seção 4.1.

modalidades lógicas que diferem daquelas presentes nas obras da UFO. Ao final, na seção 5.10, apresenta-se um breve fechamento do capítulo.

5.1 Ontoprolog: uma abordagem de Arquitetura da Informação ao tratamento de ontologias

Este trabalho trata o problema semiótico referente ao registro, em determinada linguagem, da expressão da experiência fenomenológica do sujeito com os objetos. O problema é abordado por meio de ontologias de fundamentação, metamodelagem, Programação em Lógica e Lógicas Modais. As ontologias de fundamentação são usadas como teorias gerais, cognitivamente coerentes e abstratas a respeito de uma realidade compartilhada entre sujeitos. A metamodelagem é usada como técnica em que teorias gerais servem de base para que outras teorias mais específicas sejam construídas – no caso, tratam-se de teorias sobre a realidade experimentada por sujeitos. A Programação em Lógica e as Lógicas Modais são usadas como instrumento de formalização e de raciocínio sobre ontologias. Adota-se interpretações epistêmicas aos diferentes tipos de modalidades lógicas aplicadas no trabalho. A partir desses elementos, os resultados da pesquisa consistem na proposição de uma linguagem textual adequada à descrição de entidades ontológicas de determinado universo de discurso e também de um arcabouço dedutivo capaz de realizar inferências e de verificar consistência das teorias sobre ontologias descritas nessa linguagem.

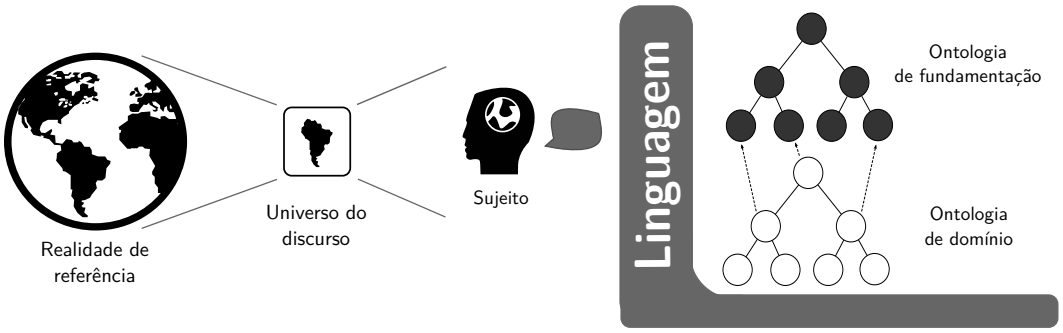
No contexto de metamodelagem, a linguagem desenvolvida nesta pesquisa depende de definições bases, organizadas na forma de uma Metateoria de *Hypertypes* (Definição 5.10), sobre as quais ontologias de fundamentação, como UFO, são descritas. A seção 5.6 aborda com detalhes essa construção base. De todo modo, adianta-se que essa metateoria é um conjunto restrito de entidades que funcionam como sustentação para a construção de teorias mais específicas. Dessa forma, um sujeito combina constructos de uma linguagem mais geral de modo a modelar um vocabulário que serve de referência para a construção de linguagens mais específicas.

Considerando um ponto de vista fenomenológico, a Figura 19 e a Figura 20 ilustram a visão geral da abordagem adotada neste trabalho. A abordagem é dita *de Arquitetura da Informação* porque se baseia em referenciais epistemológicos próprios da área, conforme mencionados no Capítulo 2. Nas figuras, uma determinada parcela da *realidade* é usada como *universo de discurso*. Esse universo é experimentado por um sujeito que, por meio de uma linguagem, descreve sua experiência com os objetos contidos naquela realidade. Essa "descrição da experiência" é entendida como um "discurso sobre o *ontos*", ou seja, sobre os objetos ou entidades ontológicas contidos naquele universo. Esse discurso é materializado como uma Ontologia de domínio (Definição 5.21). Devido à abordagem por ontologias de fundamentação, o discurso do sujeito mantém compromisso não apenas com a realidade

experimentada, mas também com a própria ontologia de fundamentação que lhe serve de vocabulário base e de indicação ideológica. Dessa forma, a ontologia de fundamentação consiste em conceitos utilizados para falar a respeito do mundo.

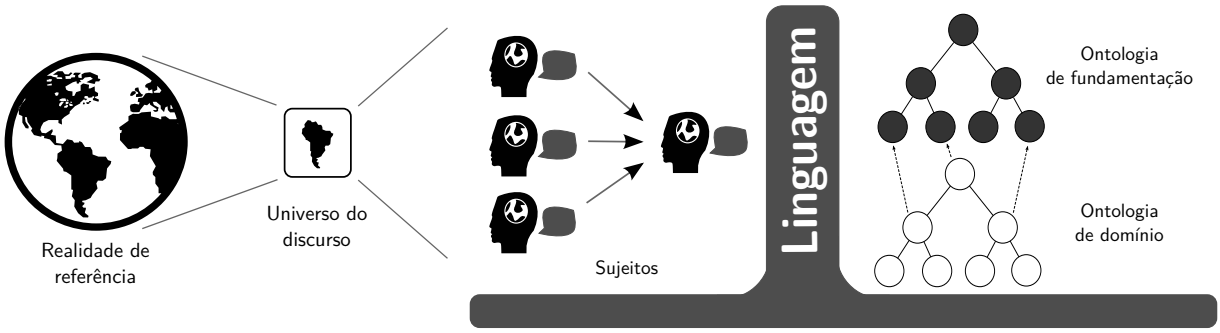
Por experimentar a realidade do universo do discurso, o sujeito é diretamente influenciado por esta e também pelo próprio discurso que ele realiza, que também passa a fazer parte do universo. Isso denota um processo contínuo e recíproco de retroalimentação de objetos na realidade. Por isso, o sujeito revê sua posição a respeito do *ontos* de forma constante. O ato de discursar sobre a realidade trata-se, então, da descrição da correlação (e não apenas de mera relação) entre o sujeito e os objetos do discurso. As duas figuras distinguem-se pelo modo de tratar logicamente o sujeito que profere o discurso. Essa distinção reflete a estratégica metodológica utilizada para obter os resultados da pesquisa, conforme descrita a seguir:

Figura 19 – Visão fenomenológica com um único sujeito



Fonte: Os autores

Figura 20 – Visão fenomenológica com múltiplos sujeitos



Fonte: Os autores

a) inicialmente propõe-se uma linguagem e uma metateoria base (Metateoria de *Hypertypes*), ambas muito fundamentais e que servem de sustentação para as demais fases. Uma semântica formal por tradução dessa linguagem é provida em termos de uma *linguagem alvo* (seção 6.1) baseada no paradigma de Programação em Lógica Clássica, conforme apresentado pelo Capítulo 7. Ainda

nessa fase, desenvolve-se também um conjunto de regras de consistência dos modelos produzidos pela teoria descrita nessa sintaxe. A todo esse arcabouço, dá-se o nome de Ontoprolog, ou Ontoprolog-Clássico, quando necessário para distinguir esse dos demais resultados. Nesse estágio, Ontoprolog contém uma linguagem permissiva e está munido com um pequeno número de restrições básicas de validade de ontologias, conforme descritas no Capítulo 6. A camada em cinza-escuro da Figura 19 com o rótulo "Linguagem" ilustra que as ontologias de fundamentação e de domínio são sustentadas pela linguagem provida por Ontoprolog. Entretanto, nem o estado epistêmico, ou a identificação de quem profere o discurso podem ser expressos nessa fase da linguagem. O sujeito utiliza a linguagem para expressar sua experiência com o universo de discurso, mas a linguagem é chamada "clássica" porque não há nenhuma característica epistemológica explícita, ou seja, o sujeito em si não está presente na linguagem. O rótulo "clássico" é mantido mesmo sendo a lógica subjacente utilizada na definição da semântica da linguagem uma lógica não-monotônica, no caso, o fragmento de Primeira Ordem utilizada por Prolog, conforme é abordado nas seções seguintes;

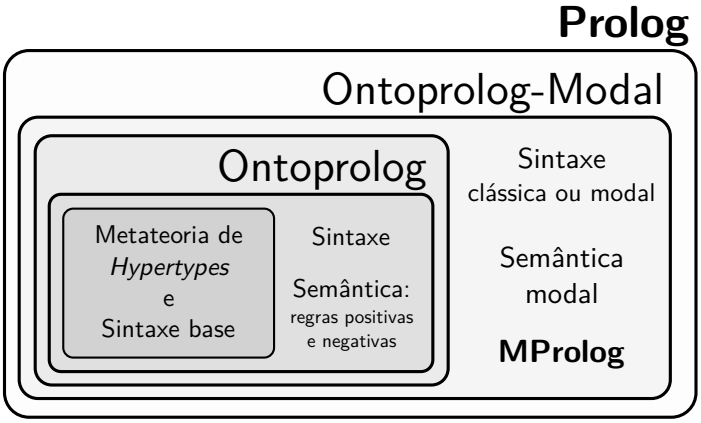
b) uma vez a linguagem e a Metateoria de *Hypertypes* construídas, define-se sobre essa metateoria a ontologia UFO-A, e todas as regras descritas nos quadros da subseção 4.5.1. Trata-se de uma instanciação do arcabouço produzido por meio de metamodelagem, conforme descrito no Capítulo 8. Nesse estágio a linguagem base é ampliada com açúcares sintáticos e com novas regras de consistência específicas para formalização de ontologias com base na UFO-A. Os açúcares sintáticos são incorporados para atender critérios como concisão e simplicidade da linguagem, de modo que ela possa ser lida e oralizada de forma inteligível. Esse complemento ainda é realizado sobre o paradigma "clássico". A ilustração da Figura 19 continua válida e a ontologia de fundamentação mencionada na figura passa a se referir especificamente à UFO-A;

c) por fim, a linguagem é ampliada para incorporar expressões modais. Com interpretação sobre o conhecimento de múltiplos agentes, o objetivo das modalidades é tornar explícito os discursos individuais de cada sujeito. Nesse caso, a ilustração da Figura 20 é mais adequada, porque ressalta os sujeitos sustentados pela linguagem, e não apenas como usuários dela. Inclusive, é possível incorporar características de diferentes tipos de lógicas epistêmicas, como conhecimento introspectivo, múltiplos agentes com conhecimento compartilhado, entre outros mencionados na seção 3.2. Para atingir esse objetivo, desenvolve-se integrações com o motor de inferências modais do MProlog (seção 3.2). Nesse caso, as sentenças de Ontoprolog podem ser tratadas como afirmações no contexto modal provido por sentenças expressas no arcabouço dos sistemas modais provido por

MProlog. Essa abordagem é discutida no Capítulo 9, que apresenta tanto a aplicação descrita neste item como expõe como outros tipos de semântica modal podem ser usadas com o arcabouço de Ontoprolog.

A Figura 21 ilustra a descrição apresentada nas alíneas anteriores. A Metateoria de *Hypertypes* e a sintaxe base da linguagem de Ontoprolog são componentes independentes de qualquer motor de consequências lógicas. Elas por si só representam resultados independentes, uma vez que podem ser utilizadas com outros tipos de semânticas operacionais, desde que mantidas a mesma semântica denotacional. Isso se justifica, por exemplo, no caso em que a sintaxe da linguagem é utilizada apenas para codificar determinada descrição, sem suporte de ferramentas de computação de inferências. Porém, esses dois componentes, aliados a um motor clássico de consequências lógicas construído sobre o paradigma de Programação em Lógica, representam o que se chama de Ontoprolog ou Ontoprolog-Clássico. Ontoprolog-Clássico, somado a uma extensão modal da linguagem base e a um motor de consequências lógicas com semânticas de múltiplos agentes provido pelo MProlog, consistem em Ontoprolog-M. A distinção das diferentes abordagens utilizadas na obtenção dos resultados referentes ao Ontoprolog-M não estão presentes no diagrama. A extensão sintática com açúcares sintáticos construída para a instância de UFO-A está contida no quadro "Ontoprolog", mas não é representada na imagem.

Figura 21 – Diagrama de inclusão dos componentes de Ontoprolog



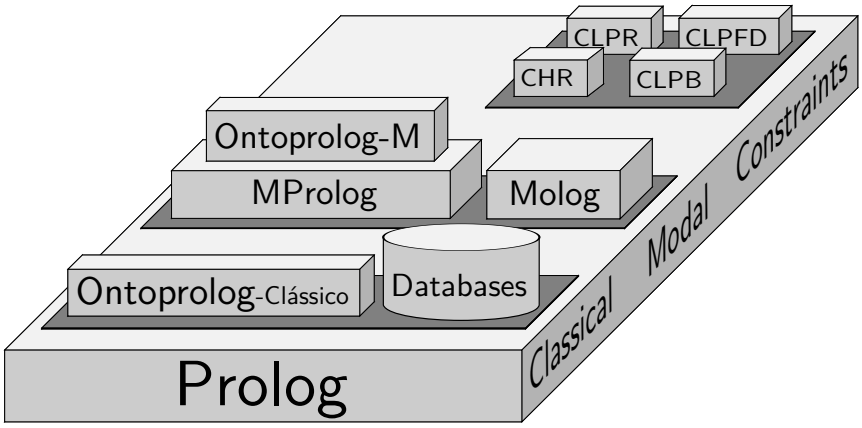
Fonte: Os autores

Ainda lançando mão de ilustrações para explicar a relação dos componentes de Ontoprolog, a Figura 22 apresenta a relação que Ontoprolog possui com Prolog, MProlog, Molog, bancos de dados e abordagens de programação por restrições. Prolog funciona como uma plataforma que sustenta todas os diferentes paradigmas baseados em Programação em Lógica. Não se trata de uma ferramenta ou implementação específica, mas de um grande arcabouço técnico e teórico sobre o qual se desenvolvem as demais lógicas e instrumentos. Sustentado sobre Prolog, encontra-se o Ontoprolog-Clássico e, para mencionar um exemplo similar, os banco de dados dedutivos. Na figura, estes estão sob o rótulo *"Classical"*.

Considerando a abordagem de banco de dados dedutivos de Colomb (1998), Ontoprolog pode ser interpretado como um banco de dados[2] de fatos sobre determinada ontologia. Nessa interpretação, as regras de verificação de consistência e as consultas definidas são interpretadas como *views* em um banco de dados. Porém, mais do que apenas um banco de fatos, Ontoprolog pode ser incorporado em programas Prolog de modo a estender a sintaxe padrão de programas existentes. Nesse caso, as bases de dados existentes em programas comuns podem servir de fonte de dados para fatos sobre ontologias. Essa abordagem é exemplificada na subseção 8.4.6. De toda forma, ressalta-se que Ontoprolog e banco de dados são abordagens alternativas, mas que são exemplos da categoria de aplicações clássicas em Programação em Lógica. Seguindo a figura, o próximo rótulo à direita da base Prolog é *"Modal"*. Nesse caso, ilustra-se os dois principais arcabouços lógicos considerados nesta pesquisa: MProlog e Molog (seção 3.2). Ambos são extensões modais de Prolog que funcionam sobre o sistema dedutivo clássico da Programação em Lógica. Ontoprolog-Modal, por sua vez, é construído sobre MProlog. Portanto, herda todas as características clássicas inerentes a qualquer programa Prolog, além de lançar mão dos raciocínios sobre modalidades providos por MProlog. A exemplo dos banco de dados, Molog é inserido na figura para exemplificar abordagem similar, mas alternativa ao MProlog. Por fim, o paradigma de programação por restrições (*"Constraints"*, em que CHR[3], CLPB[4], CLPR[5] e CLPFD[6] são exemplos) está presente na figura de modo a ilustrar que Ontoprolog, MProlog, Molog e banco de dados são todos arcabouços alternativos, que podem ser integrados, mas que são essencialmente abordagens diferentes. Dessa forma, o arcabouço completo de Ontoprolog é apresentado como um exemplo da mesma categoria em que a *Constraint Programming* pode ser enquadrada: como uma linguagem suplementar de Prolog comum e um metaprovador, assim como os programas por restrição. É importante destacar que o foco da abordagem de Ontoprolog adotada nesta pesquisa não está no *reasoning* da lógica subjacente, mas sim a linguagem de descrição, esta sustentada por regras escritas nos diferentes sistemas de raciocínio (clássico e modais).

[2] Embora possa ser interpretado como um banco de dados, Ontoprolog não é proposto como um banco de dados de dedutivo em sentido estrito, porque é construído sob o motor de inferências do tipo *backward chaining*, e não *forward chaining*, como seria adequado aos banco de dados dedutivos.
[3] *Constraint Handling Rules.*
[4] *Constraint Logic Programming over Booleans.*
[5] *Constraint Logic Programming over Rationals or Reals.*
[6] *Constraint Logic Programming over Finite Domains.*

Figura 22 – Componentes Prolog de Ontoprolog



Fonte: Os autores

5.2 Concepção da sintaxe e da semântica da linguagem

As sentenças de Ontoprolog são escritas conforme a sintaxe da linguagem e são concebidas para que seja possível descrever, de forma mais natural possível, frases que denotem discursos (seção 2.3) sobre uma Ontologia. Ou seja, a sintaxe da linguagem é uma tentativa idealizada para descrever conceituações sobre uma realidade, na forma de modelos conceituais, de modo a se parecer com um discurso oral. Essa "forma natural" é compreendida como a capacidade de as sentenças serem oralizadas por humanos, como se pudessem serem lidas naturalmente. Esse requisito da oralidade é importante ao propósito pragmático que Ontoprolog se presta: servir de instrumento para arquitetos da informação que constroem colaborativamente ontologias de domínio. Nesse sentido, a corrente definição da camada gramatical proposta para Ontoprolog (seção 7.2) visa tornar o uso da linguagem o mais natural possível para um arquiteto conhecedor de UFO e de Prolog. Considerando os critérios desejáveis de linguagens de modelagem conceitual propostos por Halpin e Morgan (2008, p. 60-62), a sintaxe de Ontoprolog é definida de modo a manter clareza e simplicidade em relação a uma semântica relevante do ponto de vista ontológico. A semântica da linguagem é definida para que mecanismos de validação de regras lógicas possam ser definidos de modo prosaico, intuitivo, e que não sacrifiquem aspectos de comunicação e de representação dos conceitos em detrimento de características como decidibilidade e eficiência computacional. De fato, a decidibilidade não é garantida em Ontoprolog, mas obtenível apenas em determinadas circunstâncias.

Uma característica das sentenças da linguagem é a natureza direta das frases. Há um esforço intencional para que essas sentenças sejam escritas (e possam ser lidas) na *voz ativa*[7], de modo a representarem afirmações diretas, em oposição a frases na ordem passiva ou indireta. Desenvolvimentos futuros poderiam acrescentar novos açúcares sintáticos

[7] Com exceção, por exemplo, do construtor `on`, usado para atribuir meta-propriedades (subseção 7.4.7), que é lido na voz passiva, conforme orientação contida na Tabela 6.

em voz passiva. Dessa forma, seria natural a substituição de um operador `relates` por `related by`, por exemplo. Essa possibilidade está registrada na seção 11.3 (Possibilidades de pesquisas futuras).

Embora o requisito desejável de oralidade seja um alvo a ser atingido, ele não é necessariamente obtido em todos os casos, nem é avaliado objetivamente. Por isso, a avaliação da eficácia da estrutura gramatical atual de Ontoprolog, nesse aspecto da oralidade, é proposta como sugestão a trabalho futuro registrada na seção 11.3. De toda forma, este trabalho apresenta como a gramática pode ser aprimorada no futuro, sem necessariamente alterar a estrutura geral de Ontoprolog, ou de seus constructos semânticos.

Outra característica pragmática incorporada à linguagem de Ontoprolog é o requisito de que toda sentença Ontoprolog válida seja também uma sentença Prolog igualmente válida. Posto dessa forma, a linguagem é concebida como um conjunto de açúcares sintáticos para a sintaxe padrão de Prolog. Portanto, toda sentença Ontoprolog pode ser incorporada em programas Prolog, inclusive os já existentes.

No lado semântico da linguagem, assume-se um contexto de um Programa em Lógica. A semântica de Ontoprolog-clássico é detalhada no Capítulo 6 e estendida na seção 8.3 tendo como base uma Lógica Clássica de Primeira Ordem cujo mecanismo de inferência é um tipo de Resolução. Nesse contexto de programação em Lógica, os axiomas e definições que regem esse sistema lógico são chamados de *regras de validade* ou *regras semânticas*, e são descritos sintaticamente por cláusulas de Horn, cujos esquemas são a Linguagem alvo (seção 6.1). Portanto, as regras de validade são predicados que regem a validade semântica dos conceitos e relações ontológicas descritos pelas sentenças. Posto dessa forma, a teoria construída na Linguagem alvo é a semântica das sentenças.

As regras semânticas são escritas na forma positiva, mas detonam condições negativas. Ou seja, cada regra de validade semântica representa consultas definidas que tentam provar que determinada propriedade indesejada ocorra. Nessa forma, para verificar que determinada relação é irreflexiva, deve haver um regra de validade semântica que tente mostrar que existe ao menos uma instância reflexiva daquela determinada relação. Se essa regra (ou regra negativa escrita de forma positiva) não puder ser provada, então, devido à hipótese do mundo fechado inerente a todo programa Prolog (subseção 3.1.4), assume-se de modo não monotônico o oposto, ou seja, que a tal relação é irreflexiva.

Para que a validade semântica possa ser verificada, um processo traduz as sentenças sintáticas (ricas em açúcares sintáticos) em termos simples de Prolog, na forma $predicado(param1, \ldots, param_n)$, para $n > 0$, em que $param$ é qualquer termo válido em Prolog (subseção 3.1.2). Essas relações clausais são chamadas de *relações semânticas*, e denotam o significado das sentenças.

Dessa forma, considerando[8] que `a :: b.` é uma sentença Ontoprolog, em:

$$\boxed{\texttt{a :: b} \Longrightarrow dio(a, b)}$$

a) `a :: b` é uma sentença (sintática) válida em Prolog quando `::` é um Operador (Definição 7.3) infixo, de modo que `a :: b.` seja um Fato ou cláusula unitária (Definição 3.15) (observar o ponto final acrescido à sentença);

b) $dio(a, b)$ é uma relação semântica que denota (parcialmente)[9] o significado da sentença `a :: b`, em que $dio/2$ é um mnemônico para *direct instance of* (seção 6.2);

c) $\Longrightarrow$ representa uma função que traduz sentenças em relações semânticas (seção 7.3).

Nesse sentido, há uma separação clara entre semântica e sintaxe em Ontoprolog, em que a sintaxe é descrita por regras EBNF (Capítulo 7) e a semântica é dada de dois modos:

a) *Semântica de tradução* (Capítulo 7): a função $\Longrightarrow$ representa um tipo de semântica da sintaxe, no caso, um tipo de semântica denotacional dada por tradução das sentenças, representadas por árvores sintáticas abstratas pela combinação de operadores Prolog, em um conjunto de relações (termos/cláusulas de Horn) simples, que representam a semântica das sentenças traduzidas;

b) *Regras de validade* (Capítulo 6) e *noção de consistência*: as regras de validade ocorrem apenas no lado das relações semânticas, e regem a validade dos modelos produzidos por essas relações. Pelo fato de essas regras serem escritas de forma positiva, mas denotarem condições negativas, uma teoria é considerada consistente quando nenhuma das regras sucede, ou seja, quando nenhuma delas é verdadeira, ou ainda, quando nenhuma delas "é o caso"[10]. Dito de outra forma, uma teoria ψ é dita *consistente* em relação à *disjunção das regras de validade* Δ quando $\psi, \Delta \models \emptyset$, para $\emptyset$ como um conjunto vazio de contra-modelos (contra-exemplos) que desautorizem a conclusão de que a teoria ψ seja inconsistente, conforme abordado no Capítulo 6. A noção de consistência pode ser ampliada nas especificações concretas por meio de entidades lógicas, conforme abordado na subseção 5.3.2.

Embora apresente-se uma *noção de consistência* de teorias sobre ontologias descritas em Ontoprolog, por ser definido como um programa Prolog, a *noção de prova* da linguagem

[8] Cada um dos temas listados são detalhados nos capítulos posteriores.

[9] Conforme detalhado no Capítulo 7, acrescente-se $entity(a)$ e $entity(b)$ às relações que denotam o significado de `a :: b.`

[10] A distinção dessas expressões não é relevante para o contexto deste capítulo, de modo que todas podem ser entendidas como sinônimos.

é a mesma utilizada do sistema lógico subjacente em que Ontoprolog é definido, ou seja, a noção de prova da linguagem é a mesma utilizada na Programação em Lógica Clássica, conforme abordado na subseção 3.1.2, ou na utilizada no âmbito de MProlog (seção 3.2), quando se trata da extensão modal de Ontoprolog (seção 5.9, Capítulo 9).

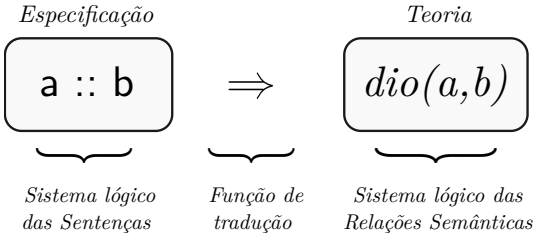
A conjunção dos elementos de um conjunto de sentenças é chamado de *especificação* (Definição 5.1, Definição 7.7); a conjunção dos elementos de um conjunto de relações semânticas é chamado de *teoria*, ou *teoria Ontoprolog* (Definição 5.2). Toda ontologia é uma teoria, embora o contrário não seja necessariamente o caso.

Definição 5.1 (Especificação)**.** Uma especificação Ontoprolog é uma conjunção de sentenças. —

Definição 5.2 (Teoria Ontoprolog)**.** Uma teoria Ontoprolog é a conjunção dos elementos de um conjunto de relações semânticas resultado da tradução de uma Especificação. —

Pelo fato de haver um sistema de tradução (de sentenças em relações semânticas), pode ser o caso de os sistemas lógicos subjacente às especificações e às teorias não serem os mesmos. Essa possibilidade é abordada no Capítulo 9, em que sentenças podem ser definidas usando um sistema clássico, mas traduzidas para relações semânticas regidas por um sistema modal. Essa possibilidade é ilustrada pela Figura 23 que também ornamenta conceitos desenvolvidos nesta seção, especialmente a relação entre sintaxe e semântica, realizada por meio da *função de tradução* de *especificações* representadas por *sentenças* que produz *teorias* representadas por meio de *relações semânticas*.

Figura 23 – Visão geral da sintaxe e da semântica das teorias de Ontoprolog



Fonte: Os autores

As teorias podem ser traduzidas para outras linguagens, como UML ou OntoUML, conforme é abordado na seção 10.3.

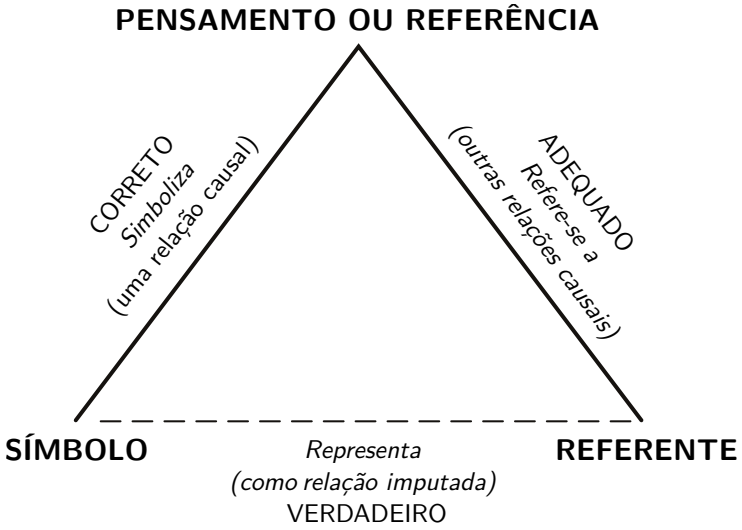
5.3 Entidades da linguagem

Na âmbito da linguagem de Ontoprolog, há dois tipos elementares de unidades de discurso: *entidades ontológicas*, ou apenas *entidades*, e *entidades lógicas*, conforme descritas nas seções a seguir.

5.3.1 Entidades ontológicas

Considerando o triângulo semiótico de Pierce presente na Figura 24, ilustrado por Ogden e Richards (1923, p. 11) com tradução nossa, um *símbolo* na linguagem representa um *referente* no universo de discurso. Esse referente é entendido fenomenologicamente como o objeto da realidade que, quando experimentado por um sujeito, denota um *pensamento*, uma *referência*, um *conceito* ou uma *imagem* do objeto no sujeito (seção 2.2). Essa imagem do objeto pode ser *simbolizada* pelo símbolo, que por sua vez, *representa* o objeto, e assim por diante. Desse modo, no âmbito da linguagem, assume-se que a representação do referente pelo símbolo é sempre *verdadeira*, no sentido em que ela denota, no pensamento, aquilo que os sentidos apresentam ao sujeito a respeito daquele referente. É patente que a experiência fenomenológica revê essa correlação entre referente e referência (entre sujeito e objeto) incessantemente, de modo que a linguagem deve sempre se adaptar para manter os símbolos corretamente representantes dos objetos.

Figura 24 – Triângulo semiótico de Pierce



Fonte: (OGDEN; RICHARDS, 1923, p. 11)

Nota: Tradução nossa

Com base nessa visão, define-se a ideia de *entidade* (ou *entidade ontológica*) e de *conceito*, como sendo a entidade (ontológica) o símbolo na linguagem e o *conceito* a imagem desse símbolo no sujeito, de modo que o conceito é sempre adequado ao objeto na realidade.

Definição 5.3 (Entidade ontológica)**.** *Entidade ontológica*, ou apenas *Entidade*, é um símbolo que representa um objeto referente ao universo do discurso adequado à imagem desse objeto no sujeito. —

Definição 5.4 (Conceito)**.** *Conceito* é a imagem, pensamento ou referência do objeto da realidade no sujeito. É o produto da correlação entre sujeito e objeto representável por símbolos. —

Em Ontoprolog assume-se a *hipótese do nome único*, abordada no subseção 3.1.4, em que cada objeto diferente recebe um identificador diferente nas especificações (um representante na linguagem), e cada identificador diferente denota um, e exatamente um, objeto da realidade, de modo que há uma correlação entre ambos. De fato, a definição de entidade ontológica apresentada pela Definição 5.3 visa salientar o caráter de *verdade* entre a representação e o referente assumido na linguagem: o símbolo *simboliza* corretamente o pensamento ou a referência a um referente no universo do discurso que ele representa. Por isso, a *entidade* na linguagem representa necessária e verdadeiramente um (único) objeto da realidade.

Como um tipo de exorbitância terminológica, é possível também que o objeto da realidade seja chamado de *conceito ontológico*. Isso se deve ao fato de que o triângulo traçado na Figura 24 mantém, idealmente, uma valoração equivalente de significado nas três entidades ilustradas. Ou seja, quando o referente (objeto) é experimentado pelo sujeito, a captura das propriedades do objeto representadas pela imagem denotam *adequadamente* aquela experiência. Desse modo, um *conceito* é ontológico quando ele se adequa à experiência do objeto (ontológico). Da mesma forma, por haver a correlação entre *entidade* (ontológica) e *referente* na realidade, no âmbito linguístico o termo *conceito* pode ser utilizado como sinônimo de *entidade*, em que ambos apontam para um mesmo referente.

Em termos pragmáticos, os símbolos que denotam entidades ontológicas são definidos como uma sequência de caracteres válidos em Prolog, necessariamente representados por átomos *ground* da linguagem subjacente, conforme descrito na subseção 7.3.4.

5.3.2 Entidades lógicas

As entidades lógicas são definidas conforme a Definição 5.5:

Definição 5.5 (Entidade lógica)**.** *Entidade lógica* representa um conceito cujo referente não se encontra na realidade do universo do discurso, mas em uma conceituação lógica. As entidades lógicas são tomadas como *a priorísticas*, e servem como instrumentos acessórios para descrever conceitos dos quais o compromisso ontológico não está na realidade do discurso.

As entidades lógicas são definidas de forma intencional: podem ser representadas por qualquer constructo válido na linguagem subjacente (no caso, Prolog), como um número, uma lista, uma cláusula, uma ou mais variáveis, etc. —

Conforme exposto na definição, nenhum compromisso ontológico é assumido a respeito das *entidades lógicas*, sendo o significado de cada entidade desse tipo dependente de explicação formal. Ontoprolog fornece duas formas de definir explicações formais. Uma, a mais geral, é diretamente por meio de regras negativas escritas de modo positivas (subseção 6.3.1) e a outra por um tipo específico dessas regras, construídos com base em meta-propriedades, conforme descrito na subseção 6.4.4.

5.4 Relações primitivas entre entidades

Como em uma rede semântica, em Ontoprolog um arquiteto da informação pode expressar conceitos e relações entre conceitos. Os conceitos, entretanto, são expressos apenas no âmbito de relações, de modo que as sentenças exprimem apenas relações formais entre entidades. Ou seja, não é possível em Ontoprolog declarar a existência de uma entidade (ontológica ou lógica) sem que ela participe de ao menos uma relação. Além disso, Ontoprolog consiste em uma lista taxativa de relações formais primitivas admissíveis na linguagem. Essa característica o distingue, por exemplo, da abordagem mais geral de lógicas descritivas, que não assumem um universo específico de relações.

Intuitivamente[11], as relações formais elementares entre entidades expressáveis em Ontoprolog, e suas respectivas intenções de compromisso ontológico, são as seguintes:

a) *Relação de subsunção*: a relação de subsunção é uma relação entre classes, que define como uma determinada classe pode ser subdividida. A relação classifica intencionalmente os indivíduos que participam da classe. Por isso, a relação é entendida como subordinação, no sentido de que uma determinada entidade c_1 é subordinada de uma entidade c_2. Considerando uma explicação com base na Teoria de Conjuntos, um conjunto C_1 é subordinado a um conjunto C_2 quando $C_1 \subseteq C_2$, ou seja, quando C_1 está contido em C_2[12];

b) *Relação de instanciação*: a relação de instanciação é uma relação de pertencimento de um indivíduo em uma classe. Ela denota que determinada entidade c é um membro de outra entidade C. Com base na Teoria de Conjuntos, c é instância de C quando $c \in C$. Dessa forma, a relação de instanciação sempre ocorre entre uma entidade c que é dita *instância de* uma entidade C, sendo esta chamada de *classe*, ou *tipo* (Instantiation Relation);

[11] A seção 6.2 introduz essas relações de modo mais formal.

[12] A ideia de subconjunto refere-se à Relação de ordem parcial (Definição 4.11), ou seja, uma relação em que valem as propriedades reflexividade, anti-simetria e transitividade. Essa ideia é utilizada para explicar o compromisso ontológico assumido pela *relação de subsunção*. A definição dessas propriedades é realizada em Ontoprolog por meio "relações básicas", regras afirmativas e negativas, conforme estrutura semântica apresentada na seção 6.2. Esse comentário também vale para as demais analogias apresentadas nesta seção.

c) *Disjunção exclusiva de conceitos*: trata-se de um tipo de restrição que se aplica às relações de instanciação e subsunção, de modo que duas entidades disjuntas não podem ser estendidas nem instanciadas simultaneamente;

d) *Subsunção completa*: trata-se de restrição que se aplica especificamente à relação de subsunção, pois indica que determinada entidade geral foi completamente detalhada por outras, mais específicas;

e) *Power types*: conforme a subseção 4.4.9, um tipo é um *power type* de outro tipo quando as instâncias do *power type* são subsunções daquele outro tipo, dito classificado pelo *power type*. Dessa forma, a declaração de *power type* permite que relações de subsunção sejam inferidas automaticamente a partir de relações de instanciação;

f) *Meta-propriedades*: permite que meta-propriedades sejam atribuídas a entidades ontológicas. As meta-propriedades são entidades lógicas, portanto, dependentes de definição formal adicional. Elas podem denotar tanto entidades específicas, que sustentam a teoria em tela, como o conceito *abstract* (Definição 5.6), como restrições, no sentido de *logic constraints*, sobre as relações em que determinados conceitos participam, podem ou venham a participar;

g) *Atribuição de propriedade* e *Valores de propriedades*: trata-se da especificação de que instâncias de determinados tipos devem possuir valores para propriedades. Os valores de propriedade podem tanto ser entidades ontológicas quanto entidades lógicas;

h) *Relação de redefinição* e *Relação de subconjunto*: tratam-se de relações especiais entre entidades que denotam relações entre entidades ontológicas. As relações dessa categoria são definidas para suprir as interpretações ontológicas descritas na subseção 4.4.6.

Outras relações entre entidades, que não sejam essas apresentadas, são construídas em Ontoprolog por meio da reificação de conceitos monádicos em conceitos diádicos, conforme abordado nas seções seguintes.

Além das restrições atribuíveis por meio das meta-propriedades, Ontoprolog pode ser expandido de modo que outras restrições entre conceitos sejam definidas. Isso é demonstrado no Capítulo 8, em que Ontoprolog é expandido para abrigar determinadas restrições específicas da UFO.

A definição do conceito *abstract* é apresentado informalmente pela Definição 5.6. A definição formal do conceito é escrita na linguagem de Ontoprolog no Apêndice A.

Definição 5.6 (Entidade abstrata)**.** Um tipo intencionalmente rotulado como abstrato não pode possuir instâncias que também não instanciem simultaneamente ao menos outro tipo não abstrato. —

Os fundamentos teóricos utilizados para a definição de *power type* encontram-se na subseção 4.4.9.

Definição 5.7 (Power Type)**.** Um *power type*[13] é uma tipo cujas instâncias são subtipos de outra entidade. —

5.5 Níveis teóricos

Assim como ocorre no Telos (subseção 3.4.1) e na UML, em Ontoprolog não há restrições sobre a quantidade de níveis em que uma determinada entidade pode ser instanciada. Os diferentes níveis de instâncias (ou graus da relação de instância de uma entidade c, que pode ser instância de uma entidade b, e instância de uma entidade a, etc.) entre entidades são chamados de *níveis teóricos*. O primeiro nível teórico, aquele que não é instância de nenhum outro, é chamado de *nível teórico superior*. Não há uma restrição sobre quantos níveis teóricos superiores podem haver, de modo que várias hierarquias paralelas de conceitos são permitidas.

Definição 5.8 (Nível teórico)**.** Nível teórico é a atribuição dada para os diferentes graus da relação de instância entre conceitos. —

Definição 5.9 (Nível teórico superior)**.** *Nível teórico superior* ou *Nível teórico raiz* é formado pelas entidades que não participam como instância de nenhuma outra entidade. —

Dessa forma, cada *modelo* (ou teoria), pode ser identificado pelo grau das relações de instanciação a partir do nível teórico superior. Sendo assim, se o nível 0 é o nível teórico superior, o nível 1 contempla as entidades instâncias do nível 0. Para as entidades do nível 1, as entidades do nível 0 são o metamodelo das primeiras.

É importante destacar que em Ontoprolog é possível descrever que determinada entidade é subsunção de mais do que uma outra entidade, ou seja, é possível representar "heranças múltiplas", assim como é possível que instâncias sejam membros de mais de uma classe, ou seja, é possível representar "instanciação múltiplas". Isso significa que uma determinada entidade pode estar simultaneamente em diferentes níveis teóricos. Seria o caso, por exemplo, de uma entidade que é instância de uma entidade de nível teórico 0 e, simultamente, de uma entidade de nível 2. Isso pode acontecer, por exemplo, a partir da relação de Power Type, uma vez que instâncias de um tipo marcado como *power type* serão subsunção de outra entidade, que eventualmente pode estar em outro nível teórico.

De fato, o conceito de *nível teórico* apresentado nesta seção não representa a ideia de modelagem multi-nível, na qual podem haver instâncias de entidades sem que isso

[13] Escrito também como *powertype*.

implique a mudança de um nível teórico. Dessa forma, em uma modelagem multi-nível, instâncias particulares de entidades universais poderiam ser classificadas no mesmo nível teórico. Uma possibilidade de representação dessa ideia, seria a distinção de dois tipos de relação de instância: uma que representa a transposição do nível teórico, ou seja, que representa a transição de um modelo para outro, e outra relação que represente a instância de um objeto de uma classe. Porém, isso não está presente na versão base de Ontoprolog. Opta-se por não incluir esse tipo distinção na relação de instanciação por questões de simplificação da proposta.

A característica de metamodelagem implica que determinadas características de ordem superior são inerentes às teorias expressáveis em Ontoprolog, como a possibilidade de atribuição de propriedades a classes, sendo estas predicados de instâncias. Porém, devido ao modo como Ontoprolog é construído, isso não significa que ele se trate de um sistema de ordem superior, conforme é apresentado nos capítulos posteriores.

A subseção 6.4.1.1 apresenta definições a respeitos dos níveis teóricos introduzidos nesta seção.

5.6 A metateoria base embutida em Ontoprolog

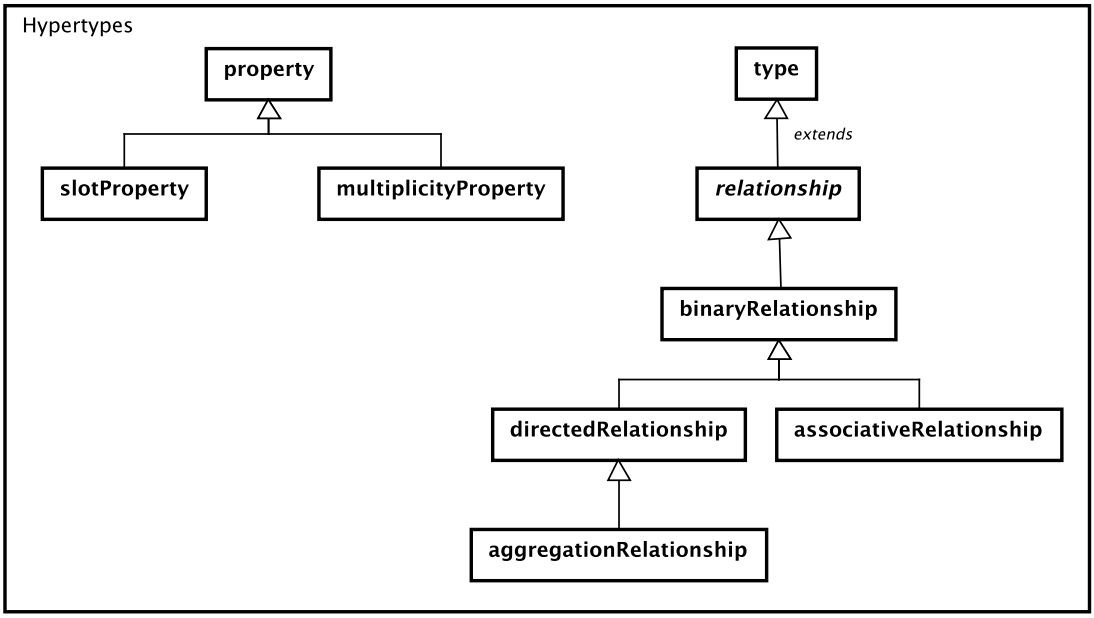
Embora Ontoprolog seja proposto nesta pesquisa motivado para a descrição de ontologias bem-fundamentadas com base na UFO, ele é construído de modo independente dessa ontologia de fundamentação. Uma camada base do arcabouço teórico sustenta logicamente uma metateoria (ou "meta-ontologia") que é usada como base para a definição da UFO, e posteriormente para a descrição das ontologias de domínio. Com isso, Ontoprolog lança mão do paradigma de metamodelagem, em que modelos mais simples são usados para definição de modelos mais complexos. Esta seção descreve essa meta-ontologia base.

Em toda teoria Ontoprolog é embutida (ou toda teoria Ontoprolog é construída sobre) uma metateoria elementar, chamada de Metateoria de *Hypertypes* (Definição 5.10)[14]. Essa metateoria fundamental consiste no Nível teórico superior (Definição 5.9), do qual todos os outros níveis são instâncias. A metateoria em tela representa o paradigma metafísico mais elementar da linguagem. Esse paradigma é baseado no quadrado ontológico aristotélico, representado na Figura 8, que por sua vez é também utilizado pela UFO como ontologia de referência. A Figura 25 descreve a Metateoria de *Hypertypes* por meio de um diagrama UML, em que a seta aponta para o conceito mais geral; os retângulos cujo rótulo estão em itálico são chamadas *entidades abstratas* (Definição 5.6) e denotam entidades que não podem possuir instâncias diretas sem que essas instâncias sejam também instâncias de ao menos outra entidade não-abstrata. Conforme se observa na figura, a Metateoria de

[14] O nome *hypertype* está alinhado com a proposta de abordagem multidimensional de Ossher e Tarr (2002) (subseção 4.4.11), uma vez que entende-se que os conceitos qualificados como *hypertype* são usados como base para que se defina conceitos em dimensões lógicas possivelmente distintas e paralelas.

Hypertypes contempla duas hierarquias distintas de entidades, os Type (Definição 5.11) e os Property (Definição 5.12). Essas entidades remetem à distinção aristotélica elementar entre *objetos* e *tropes*.

Figura 25 – Metateoria de *Hypertypes*



Fonte: Os autores

Definição 5.10 (Metateoria de *Hypertypes*)**.** A Metateoria de *Hypertypes* é a teoria formada pelas entidades qualificadas como *hypertypes* que constituem o Nível teórico superior (Definição 5.9). —

Definição 5.11 (Type)**.** Instâncias de *type* (tipo) denotam conceitos sobre os quais se mantém compromisso ontológico. —

Definição 5.12 (Property)**.** Instâncias de *property* (propriedade) denotam qualificações atribuíveis a um ou mais *types*. —

As propriedades são relações funcionais ($\mapsto$) parciais que mapeiam *entidades ontológicas* $\mapsto$ *entidades ontológicas*, ou *entidades ontológicas* $\mapsto$ *entidades lógicas*.

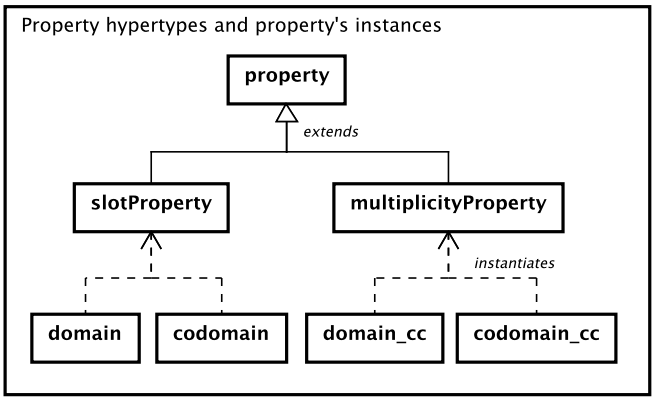
Seguindo a Figura 25, apresenta-se a definição de cada elemento na hierarquia de *properties* e de *types*:

Definição 5.13 (Slot Property)**.** Tipo específico de propriedade cujo contra-domínio é uma entidade ontológica. —

Definição 5.14 (Multiplicity Property)**.** Tipo específico de propriedade cujo contra-domínio é uma entidade lógica que denota uma restrição na cardinalidade das instâncias da relação. —

Além desses elementos, a Figura 26 ilustra quatro propriedades específicas, instâncias de Slot Property e de Multiplicity Property, presentes em todas as teorias Ontoprolog. Essas propriedades são usadas na caracterização de entidades que denotam relações de domínio, significativas no âmbito dos universos de discursos ontológicos, conforme descrito na sequência.

Figura 26 – Instâncias de propriedades embutidas em Ontoprolog



Fonte: Os autores

Paralelo aos *properties*, apresenta-se os conceitos que são subsunção de Type:

Definição 5.15 (Relationship)**.** Representa *types* não monádicos, ou seja, as instâncias de *relationship* representam conceitos que denotam relações entre conceitos, inclusive entre outras relações. As quatro instâncias de propriedades contidas na Figura 26 são atribuídas às instâncias de *relationship*. —

É por meio das instâncias desse tipo que se reifica relações ontológicas arbitrárias e as inclui em teorias Ontoprolog. Porém, *relationship* é um conceito abstrato (Definição 5.6), ou seja, uma entidade não pode instanciar diretamente *relationship* sem que seja instância de ao menos outra entidade não abstrata. Dessa forma, as instâncias de *relationship* se dão por instâncias de uma de suas subsunções, definidas conforme segue.

Definição 5.16 (Binary Relationship)**.** Representa *relationship* binários, ou seja, conceitos que relacionam exatamente duas outras entidades. —

Para o propósito deste trabalho, apenas relações binárias são abordadas.

Definição 5.17 (Associative Relationship)**.** Representa relações binárias não direcionais simétricas (Definição 4.5), em que o domínio está relacionado com o contra-domínio e vice-versa. —

Definição 5.18 (Directed Relationship)**.** Representa relações binárias direcionadas não simétricas (Definição 4.7), em que o domínio está relacionado com o contra-domínio, mas não necessariamente o contrário. —

Definição 5.19 (Aggregation Relationship)**.** Representa relações binárias direcionadas não simétricas que denotam conceitos de relações de meronímia. —

As especificação completa da Metateoria de *Hypertypes*, escrita na sintaxe de Ontoprolog proposta no Capítulo 7, está contida no Apêndice A.

Toda teoria Ontoprolog deve ser especificada a partir de instâncias dessas entidades fundamentais. Pelo fato de a ontologia base de Ontoprolog compartilhar com a UFO o mesmo paradigma ontológico, no caso, a ontologia de quatro categorias de Aristóteles, é natural que seja possível descrever as categorias conceituais da UFO com base na Metateoria de *Hypertypes*, como de fato é realizado no Capítulo 8.

5.7 As ontologias de domínio e a metateoria UFO

Uma vez apresentado o nível teórico superior com a Metateoria de *Hypertypes*, todas as instâncias dessa metateoria são teorias específicas descritas em Ontoprolog. Embora seja possível descrever conceitos de um domínio específico diretamente como instância dessa metateoria elementar, esse não é o propósito deste trabalho. De fato, o que se deseja demonstrar é que com base nela, e do arcabouço teórico elementar apresentado na seção 5.6, pode-se definir a UFO como um caso que enriquece ontologicamente a metateoria base. Dessa forma, amplia-se o vocabulário da linguagem ao passo que aumenta-se as restrições lógicas sobre a validade de determinado discurso ontológico. Com isso como meta, é necessário ajustar certo vocabulário:

Definição 5.20 (Ontologia objeto)**.** A *Ontologia objeto* é a realidade (o referente) sobre a qual se faz um discurso e se mantém compromisso ontológico. Trata-se do universo do discurso ilustrado pela Figura 19 e pela Figura 20. A ontologia objeto é entendida como acessível de forma mediada pelos sujeitos fenomenológicos, de modo que ela é externa às teorias e aos sujeitos, mas participam do fenômeno da experiência de forma indistinguível do próprio sujeito. —

Definição 5.21 (Ontologia de domínio)**.** A *ontologia de(o) domínio* é uma Ontologia objeto (Definição 5.20) específica, cujas fronteiras ontológicas são artificial e subjetivamente limitadas por um sujeito, ou por um grupo de sujeitos mediante consenso, em função de critérios particulares (como por restrição de escopo do discurso, por questões culturais, por intenções pragmáticas, etc).

Enquanto modelo conceitual, a ontologia de domínio é um discurso, no sentido de descrições ou representação da realidade, que é formalizado em uma linguagem (neste caso, a linguagem definida para Ontoprolog).

Enquanto objeto de prática, uma ontologia de domínio serve como modelo de consenso, de explicação de determinada parcela compartilhada do mundo. —

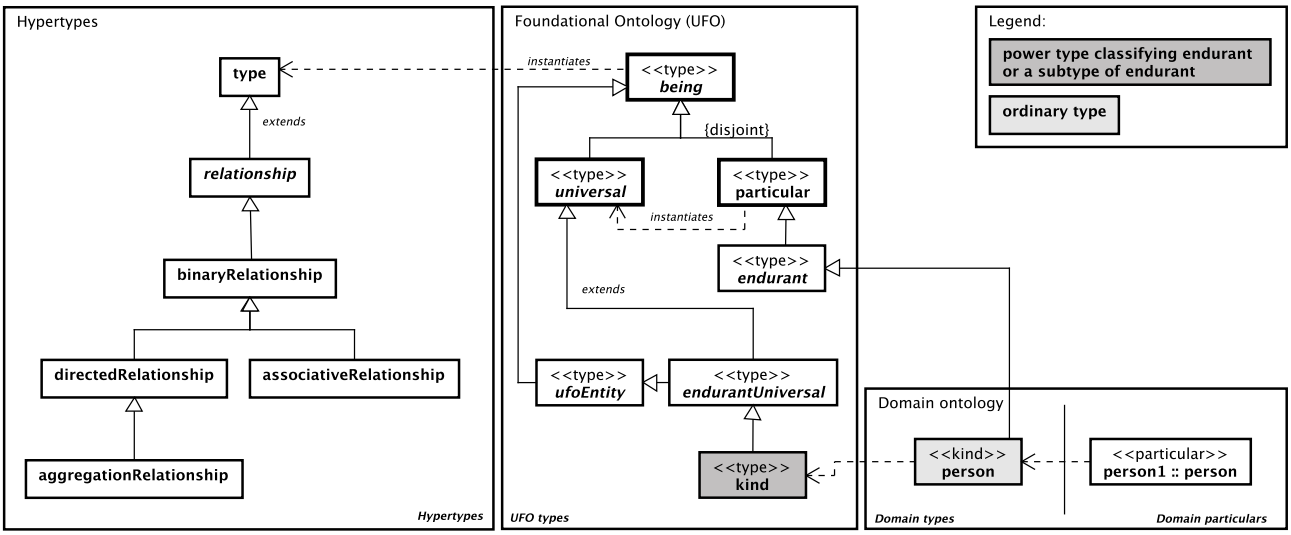
Definição 5.22 (Ontologia de referência ou de fundamentação)**.** A *ontologia de referência*, ou *ontologia de fundamentação*, é um conjunto de termos e de relações que denotam conceitos que servem como base para a construção de sistemas de explicações sobre o mundo. As ontologias de referências são potencialmente independentes de domínio. —

É importante destacar que a noção de Ontologia objeto e de Ontologia de domínio referem-se a conceitos meta-linguísticos, uma vez que não estão na própria linguagem, mas fazem parte dos fins pragmáticos de Ontoprolog. Além disso, *Ontologia objeto* e *Ontologia de domínio* não se confundem com Especificação (Definição 5.1) ou com Teoria Ontoprolog (Definição 5.2). Estas *denotam* uma Ontologia, enquanto aquelas *são* a própria Ontologia. Estas, a especificação e a teoria, podem ser chamadas de "ontologia" (com "o" minúsculo), no sentido que são representações em determinada linguagem, mas não são em si a Ontologia (com "O" maiúsculo denotando substantivação própria).

Com base nessas definições, entende-se como *Ontologia de referência* toda ontologia usada para definir outras ontologias. Isso inclui os conceitos ontológicos denotados inscritos na Metateoria de *Hypertypes*, que é um constructo lógico, de modo que do ponto de vista de uma ontologia de domínio, todos os níveis teóricos usados para defini-la são os níveis de fundamentação. Com isso em tela, a Figura 27, apresenta a visão geral de um exemplo de relacionamento entre diferentes ontologias, e diferentes níveis teóricos, passíveis de serem descritos em Ontoprolog. Novamente uma representação em UML é adotada, em que a seta tracejada e os estereótipos entre « e » denotam a relação de instância (no caso, o rótulo fora da marcação de estereótipo é o identificador da entidade que instancia a entidade identificada pelo rótulo demarcado pelos *"guillemets"*).

Na figura, destaca-se três categorias distintas de entidades: a Metateoria de *Hypertypes*, a *ontologia UFO*, que é instância da Metateoria de *Hypertypes*, e a *ontologia de domínio*, que é instância da ontologia UFO. No caso, o conceito de domínio *person*, representado pelo quadro `person`, é uma instância de um conceito da UFO chamado Kind, que é definido como uma extensão (transitiva) de `being`, que por sua vez é uma instância de `type`. Além disso, um indivíduo, identificado por `person1`, é uma instância do conceito `person`. No exemplo, tanto o indivíduo `person1`, quanto o universal `person` denotam conceitos da ontologia de domínio. O estereótipo `«particular»` e o sufixo `:: person` em `person1 :: person` denotam que o conceito identificador por `person1` participa de uma múltipla instanciação: ele tanto instancia diretamente `person`, quanto instancia

Figura 27 – Diagrama da Metateoria de *Hypertypes*, ontologia UFO e ontologia de domínio



Fonte: Os autores

Nota: O modelo contém uma simplificação das classes da UFO. A classificação completa é apresentada na Figura 28.

indiretamente `particular`, assim como, transitivamente, instancia `being`[15].

A ontologia UFO incorporada em Ontoprolog é detalhada na subseção 5.7.1. Porém, antes de se adentrar nesse tema, destaca-se o uso do padrão de design Power Type (Definição 5.7). Na figura, a entidade `kind` é definida com um *power type*, de modo que suas instâncias estendem outro conceito, no caso, `object`, que por sua vez é subsunção de `endurant`, conforme representado pela Figura 28.

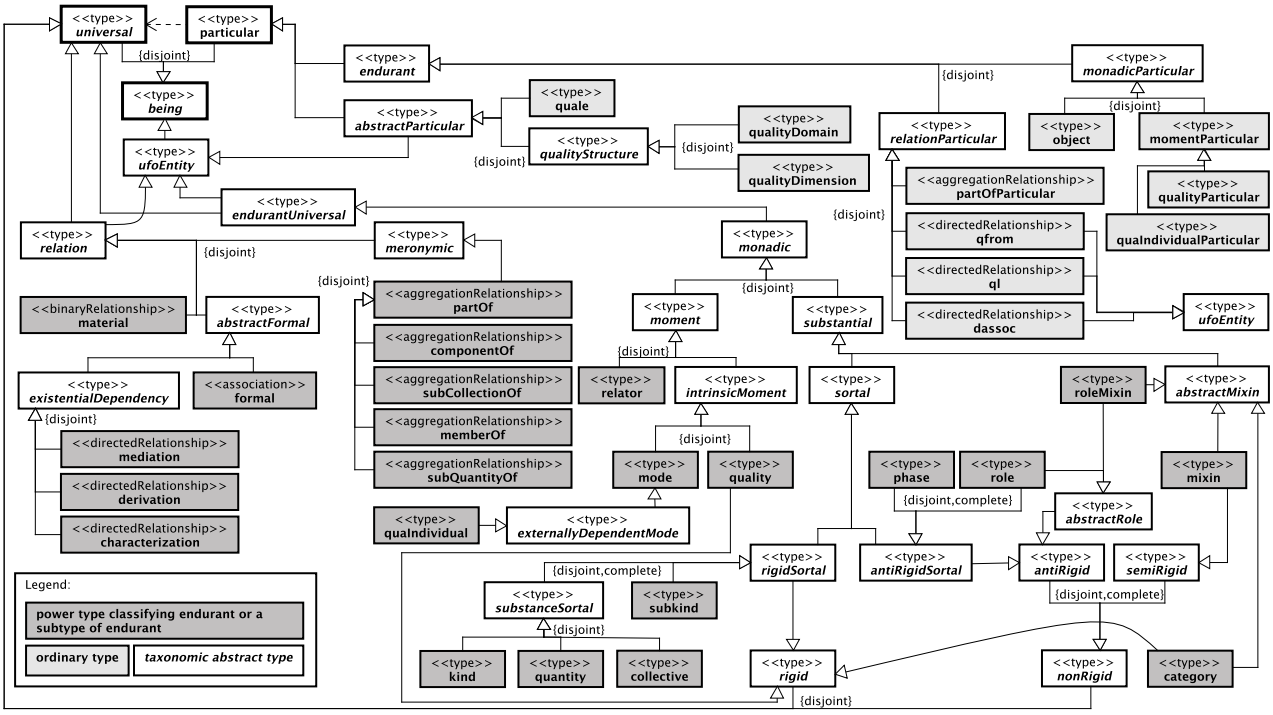
5.7.1 Metateoria da UFO definida em Ontoprolog

Como um dos resultados desta pesquisa, apresenta-se no Capítulo 8 a construção da teoria UFO em Ontoprolog, definida como instância da Metateoria de *Hypertypes*. Introduzindo esses resultados, a Figura 28 contém a lista exaustiva de conceitos da UFO tratados neste trabalho e que são incorporados como ontologia subjacente em toda especificação Ontoprolog. A semântica natural desses conceitos – no sentido de significado informal – é aquela apresentada nas publicações da UFO descritas no Capítulo 4 e no Glossário da UFO, com as adaptações apresentadas nesta seção e nos capítulos seguintes.

O conceito central descrito na Figura 28 é a entidade `being`. Trata-se do tipo mais geral do qual todos os outros ou são subsunção ou são instância. A entidade `being` denota

[15] Essas regras de instanciação são detalhadas nos capítulos posteriores, cabendo esta seção apenas descrever intuitivamente essas noções.

Figura 28 – Metateoria UFO



Fonte: Os autores

Nota: A Figura 59 do Apêndice B contém o diagrama desta figura desenhado de forma a cobrir toda a mancha gráfica da página, o que melhora a legibilidade da ilustração.

a ideia de *ser existente*, referente aos indivíduos ontológicos sobre os quais se discursa. Esse conceito geralmente é representado na literatura da UFO como Urelement ou mesmo como *entity*[16], embora em Ontoprolog opta-se por destacar o conceito de *being* como algo mais geral. Dessa forma, opta-se por adotar o termo *being* por entender ser ele mais adequado no objetivo de salientar que as teorias descritas em Ontoprolog se tratam de ontologias, ou seja, de discursos sobre os seres existentes. O conceito de `ufoEntity` representa conceitos universais e particulares abstratos, de modo que ontologias de particulares de domínio não sejam instâncias diretas de conceitos da UFO.

Isso posto, a próxima distinção realizada sobre os *being* refere-se aos conceitos *universais*, ou repetíveis, o que na UFO é chamado de Universal, e aos conceitos *particulares*, ou indivíduos, na UFO chamados de Particular. É importante destacar que embora logicamente o próprio conceito `universal` seja definido como uma instância de outro, no caso, de `type`. Ocorre que com o paradigma da meta-modelagem, em cada nível teórico atribui-se interpretações próprias de modo a se compor (ou estabelecer) um ou outro paradigma ontológico. Nesse caso, o que se deseja instituir é que embora logicamente possam

[16] Ver, por exemplo, Guizzardi e Zamborlini (2014, p. 3).

haver inúmeros níveis de instâncias, adotando-se a UFO como ontologia de fundamentação, ontologicamente apenas faz sentido discursar com dois níveis de abstração: o nível das instâncias de *conceitos universais* e o nível das instâncias de *conceitos particulares*, em que os conceitos particulares são instâncias dos conceitos universais. Com isso, admite-se que existem ao menos duas teorias diferentes em cada ontologia de domínio: uma *teoria de universais (de classes ou de tipos)*, que engloba as entidades classificadas como *universais*, e uma *teoria de particulares*, que contém as entidades *particulares*.

Enquanto do ponto de vista pragmático um arquiteto da informação infere as entidades universais a partir de características comuns de particulares, na ótica do sistema lógico de Ontoprolog as entidades particulares são instâncias de entidades universais já definidas *a priori*. Isso não implica que se advogue que uma ontologia de universais seja produzida antes da construção de uma ontologia de particulares. O que se tem, entretanto, é que todo conceito particular é, no mínimo, instância de um conceito *universal*. Ou seja, todo conceito particular é sempre instância direta ou indireta de ao menos uma entidade `universal`. O que, ao contrário do que seria natural, leva à necessidade lógica de ter uma teoria de universais antes de uma teoria de particulares.

Com isso posto, abstém-se de descrever nesta seção o significado dos demais conceitos representados na Figura 28, uma vez que encontram-se detalhados no Glossário da UFO. De todo modo, na seção 8.1 apresenta-se um detalhamento da especificação dos conceitos Momentos Intrínsecos.

5.8 Adaptações na ontologia UFO

A Figura 28 contém a taxonomia de conceitos da UFO-A construída quase como exatamente foi encontrada na literatura da área, exceto por algumas adaptações. Ao comparar a Figura 28 (e a Figura 27) com a Figura 9, que foi produzida por Guizzardi e Wagner (2010) como um fragmento quase completo que sistematiza os conceitos da UFO-A, ou ainda, ao comparar com a Figura 60 do Apêndice D, que contém um diagrama que se apresenta como mais completo em relação aos conceitos da UFO-A do que os contidos na Figura 9, observa-se algumas diferenças fundamentais. Algumas dessas diferenças, que referem-se aos ajustes lógicos realizados na UFO original para que ela pudesse ser usada como fundamento para uma linguagem lógica, são as seguintes:

a) a primeira distinção, já discutida, é a inclusão de `being` como entidade unificadora de conceitos universais, particulares e de entidades específicas das UFO, conforme descrito na subseção 5.7.1;

b) a entidade `meronymic` foi adicionada como uma generalização das entidades parte-todo. Além dela, `partOf` também foi explicitamente incluída no vocabulário, como gênero das demais entidades que tratam de meronímias na UFO. A

entidade `partOf` visa atender a definição apresentada em Part of, o que inclui o axioma Weak Supplementation, mas não restringe o domínio e o contra-domínio das entidades relacionadas. Já `meronymic` é um grande gênero, que não inclui qualquer tipo de restrição lógica;

c) a árvore de entidades não-sortais (Non Sortal) foi inteiramente reorganizada, de modo a tornar explícita entidades como `rigid`, `nonRigid`, `abstractMixin` e `abstractRole`. Essas separações permitem que regras gerais sejam aplicadas em entidades no nível hierárquico mais alto da taxonomia de conceitos, o que facilita o entendimento e otimiza o sistema de regras. Isso é evidenciado, por exemplo, no caso de `abstractRole`, que foi criado com intuito de simplificar a aplicação da restrição 3 do Quadro 19 (Mediation), p. 183;

d) a hierarquia de particulares também foi alterada, de modo a tornar explícitas certas categorias ontológicas elementares, como `object`, `momentParticular` e `relationParticular`. Nesse caso, observa-se uma distinção terminológica importante. As obras que abordam as categorias taxonômicas da UFO geralmente apresentam conceitos universais com o sufixo "Universal", enquanto os conceitos particulares são escritos sem sufixos. Em Ontoprolog adota-se uma abordagem invertida: são os particulares que possuem o sufixo "Particular", enquanto os universais não são escritos com qualquer sufixo. Isso deve-se ao fato de que em Ontoprolog foca-se no paradigma em que são definidas primeiro uma teoria de universais, e apenas depois uma ontologia de particulares. Os particulares, por sua vez, podem até serem criados automaticamente por algum algoritmo de *model checking*, ou mesmo a partir de um banco de dados existente, como de fato é realizado no exemplo apresentado pela subseção 8.4.6. Por esse motivo, entende-se que parece mais salutar evitar que se escreva sufixos nas especificações que envolvam apenas universais, que provavelmente são mais comuns. A única exceção trata-se de `endurantUniversal`. Isso ocorre porque a taxonomia dessa classe não denota conceitos endurantes, mas conceitos cujas as instâncias são endurantes;

e) os conceitos de Quality Region e de Conceptual Space são modelados como propriedades das entidades Quale e Quality Domain, respectivamente, conforme abordados na seção 8.1.

Outras limitações da atual implementação de Ontoprolog são descritas na seção 10.4.

5.9 Sobre as modalidades de Ontoprolog

Em termos de semântica de mundos possíveis de Kripke (KRIPKE, 2001), a semântica natural da UFO é apresentada por uma Lógica Modal $S5$, em que a condição

de acessibilidade entre os mundos é universal, ou seja, os mundos possuem relação de acessibilidade reflexiva, simétrica e transitiva, em que todos eles são acessíveis a partir do outro. Na apresentação da UFO, Guizzardi (2005) usa uma interpretação alética das modalidades que denotam restrições sobre modelos, assim como é comum no contexto de descrição de ontologias em pesquisas na Ciência da Computação. Porém, em trabalho recente (GUIZZARDI; ZAMBORLINI, 2014), o autor propõe uma interpretação temporal para as modalidades da UFO, mas não apresenta uma construção rigorosa das condições de acessibilidade entre os estados de tempo; entretanto, é possível presumir uma lógica com relações seriais, como as induzidas por axiomas do tipo KD (CARNIELLI; PIZZI, 2008, p. 37; 61), de modo a induzir interpretações modais como *"next"*.

Por outro lado, o interesse por modalidades lógicas neste trabalho está em formalizar discursos sobre ontologias realizados simultaneamente por sujeitos, o que induz uma interpretação epistêmica (ou doxástica) dos mundos possíveis. Por isso, Ontoprolog pode ser classificado como uma linguagem epistêmica, no sentido estrito de que a linguagem descreve conhecimentos de sujeitos sobre determinado *"intended state of affairs of the underlying conceptualization"* (GUIZZARDI; ZAMBORLINI, 2014), assim como ocorre com OntoUML para o caso em que só existe um único agente. Nesse sentido, Ontoprolog-Modal usa uma Lógica com expressividade modal, sustentada pelo arcabouço provido por MProlog, que interpreta as modalidades como contextos epistêmicos de conhecimento entre sujeitos.

Dessa forma, como a intenção deste trabalho não é investigar os aspectos modais da semântica natural da UFO, e sim aspectos formais da epistemologia de diferentes agentes sobre as ontologias descritas com base na UFO, as fórmulas modais dessa ontologia que denotam modelos na semântica interna são tratadas como fórmulas denecessitadas. Os operadores modais são ignorados e os modelos que a formulações denotam são modelos achatados em termos da semântica de Kripke, ou seja, são meros modelos clássicos sem modalidade.

Reconhece-se que essa abordagem distorce a semântica natural da UFO. Todavia, é importante destacar que essa distorção acontece somente quando especifica-se particulares do domínio, o que não é o caso para especificações somente referente a universais. Por exemplo, o conceito de Rigidez (Rigidity), transcrito abaixo, estabelece uma restrição sobre universais, no caso simbolizados por T[17].

$$Rigid(T) \coloneqq \forall x\ (\Diamond(x :: T) \rightarrow \Box(\mathcal{E}(x) \rightarrow x :: T))$$

Observa-se que uma entidade T é rígida se, e somente se, toda instância x de T que existe em algum mundo possível acessível é instância de T em todos os demais mundos em

[17] Ver observação sobre o predicado $\mathcal{E}$ na entrada Existential Dependence do Glossário da UFO e na Definição 4.19.

que x está contido. Num contexto em que não existem particulares, nunca haverá um x que instancia T, o que fará com que o conceito de rigidez se torne sempre satisfatível, uma vez que pelas regras clássicas, de uma valoração falsa no antecedente de uma implicação obtém-se uma valoração verdadeira de toda a expressão (MORTARI, 2001, p. 144).

Num contexto em que as modalidades são denecessitadas, teríamos que o conceito de rigidez seria assim definido:

$$Rigid(T) \coloneqq \forall x \ ((x :: T) \rightarrow (\mathcal{E}(x) \rightarrow x :: T))$$

O que, como era esperado, nos levaria novamente a uma fórmula satisfatível no contexto em que não existem particulares. Considerando que $\mathcal{E}(x)$ denota a proposição "x existe no mundo local", conforme a a entrada Existential Dependence do Glossário da UFO, num contexto denecessitado essa expressão poderia ser reescrita como se segue, o que é satisfeito trivialmente, por instanciação do universal:

$$Rigid(T) \coloneqq \forall x \ (x :: T \rightarrow x :: T \rightarrow x :: T)$$

Por ser trivialmente satisfatível, o conceito de rigidez não é relevante em um sistema denecessitado. Por isso, pelo fato de não ser escopo deste trabalho a implementação da semântica alética original da UFO, não se encontra nesta pesquisa a definição do conceito de rigidez proposto nessa ontologia de fundamentação. Porém, a seção 9.2 discute novamente o conceito de rigidez e apresenta uma definição baseado no sistema multi-agente $KD45_m$. Além disso, a título de sugestão de trabalho futuro, no Item c) da seção 11.3 indica-se como a semântica da UFO baseada no sistema $S5$ pode ser resgatada no âmbito do paradigma lógico empregado nesta pesquisa.

Já a integração modal epistêmica de Ontoprolog com MProlog é apresentada no Capítulo 9, realizada por meio da substituição da lógica subjacente do metaprovador do paradigma de Programação em Lógica.

5.10 Fechamento

Este capítulo apresenta tópicos introdutórios que sumarizam os principais resultados desta pesquisa, e indica como o restante deste texto está organizado. Embora não tenha sido abordado por este capítulo, destaca-se que o Glossário da UFO é um resultado da pesquisa especialmente de interesse da Ciência da Informação porque busca apresentar uma sistematização terminológica da UFO.

Em suma, Ontoprolog é apresentado de duas formas diferentes:

a) *Ontoprolog-clássico*: trata-se da definição padrão da linguagem, tendo como lógica subjacente um paradigma de Primeira Ordem clássico;

b) *Ontoprolog-modal*: extensões do *Ontoprolog-clássico* que contemplam modalidades lógicas, especialmente modalidades com interpretações referentes a multi-agentes.

Ontoprolog-clássico é escrito em uma máquina Prolog "abstrata", considerando a especificação ISO/IEC 13211 (ISO/IEC, 1995). Como extensão da versão clássica, *Ontoprolog-modal* é definido a partir de MProlog (seção 3.2), biblioteca de modalidades para Prolog proposta por Nguyen (1999).

Todo o aparato dedutivo de Ontoprolog é herdado de suas lógicas subjacentes, ou seja, o próprio Prolog ou o MProlog, no caso de modalidades. Dessa forma, os axiomas e regras de inferência de Ontoprolog são avaliados exatamente pelos métodos presentes nessas lógicas subjacentes.

Ontoprolog é concebido como uma biblioteca para o Prolog. O sufixo "Prolog" é importante porque Ontoprolog é proposto como um sistema construído sobre Prolog, tendo a Lógica Clássica como lógica subjacente. Portanto, embora trabalhos futuros possam generalizar as propostas apresentadas neste trabalho, não é objetivo apresentar resultados gerais que extrapolem o escopo de Prolog.

Ontoprolog provê uma linguagem com um conjunto de restrições fundamentais que verificam teorias a respeito de ontologias. Porém, essa linguagem é *fraca* do ponto de vista ontológico, uma vez que pressupõe apenas um conjunto pequeno de conceitos. Esses conceitos são representados pela Metateoria de *Hypertypes* (Definição 5.10), cuja distinções mais básicas estão entre as ideias aristotélicas de entidade e de propriedade. Com base na especialização das entidades e na combinação dessas com propriedades, descreve-se as relações, que por sua vez podem ser especializadas. Para compor um conjunto de restrições às instâncias dessas entidades, apresenta-se no Capítulo 6 uma série de regras de verificação de consistência dos modelos gerais. Embora essas regras sejam abrangentes e adequados para a formalização de ontologias, elas não são suficientes para contemplar o restrições aos modelos apresentadas pela UFO. Por isso, na subseção 8.3.2, outras regras complementam as primeiras nessa função de formalizar a UFO. Porém, a Metateoria de *Hypertypes* não precisa ser alterada, e cumpre seu papel de referência inicial.

Com base num paradigma fenomenológico de Arquitetura da Informação (Capítulo 2), arquitetos exercem papeis de ontologistas ao experimentar o mundo e formalizar seus discursos sobre essa experiência na sintaxe da linguagem proposta para Ontoprolog. Dessas formalizações, obtém-se um resultado que denota o discurso ontológico compartilhado por aqueles arquitetos. No contexto cíclico, essas formalizações são novamente avaliadas pelos arquitetos, que as incorporam na realidade do mundo que experimentam e, desse modo, reavaliam seus discursos até que encontrem um consenso ou um acordo. Nesse processo, Ontoprolog funciona como um instrumento para obtenção desses acordos ou consenso. No caso em que múltiplos arquitetos estão envolvidos, os diferentes tipos de

lógica providos pelo MProlog podem ser utilizados. No caso em que há apenas um único agente, pode-se usar tanto semânticas modais introspectivas ainda com MProlog, ou ainda, usar a versão clássica de Ontoprolog, em que não há modalidades lógicas explícitas.

6 Semântica formal base de Ontoprolog

Este capítulo apresenta a semântica chamada *clássica* ou *semântica sem modalidades* de Ontoprolog. A concepção de *semântica da linguagem*, cujo detalhamento ocorre neste capítulo, é apresentada pela seção 5.2. A semântica é dita *base* porque ela rege os modelos denotados pela linguagem base e por suas extensões[1].

Detalhar a definição semântica antes da sintaxe da linguagem segue a proposta de Erwig e Walkingshaw (2011), Erwig e Walkingshaw (2012), que defendem uma inversão na ordem tradicional de construção linguagens de programação ao proporem que a semântica de uma linguagem seja detalhadamente construída antes da sintaxe.

A semântica descrita neste capítulo é uma proposta suficiente para os objetivos desta pesquisa, mas permite-se ficar em aberta a aprimoramentos posteriores, especialmente quanto ao enriquecimento do rigor da formalização apresentada, bem como quanto à possibilidade de demonstrar ou incorporar propriedades específicas que tangem à complexidade computacional ou à prova de propriedades meta-lógicas, ou mesmo que tangem a aspectos técnicos adicionais da UFO. Portanto, não se afirma que a semântica formal proposta neste capítulo para Ontoprolog seja definitiva ou esteja concluída, mas que esta é uma concepção inicial adequada para os objetivos desta pesquisa. Além disso, não é intuito deste trabalho esgotar ou mesmo atentar demasiadamente aos aspectos formais. Opta-se, porém, por uma abordagem minimalista na formalização da semântica, próxima à formalização em Prolog, que visa principalmente evitar ambiguidades de interpretação, mas permite-se ser flexível e semi-formal em alguns aspectos, desde que não haja perda de clareza na intuição desejada.

Este capítulo é composto nas seguintes seções. A seção 6.1 apresenta a linguagem lógica alvo usada para formalizar a semântica da linguagem. A seção 6.2 descreve a estrutura $\mathcal{OT}$, usada como estrutura básica sobre a qual se constroem os predicados lógicos. A seção 6.3 contém o arcabouço axiomático utilizado para descrever as regras de Ontoprolog, concentrados na seção 6.4. Como encerramento, a seção 6.5 apresenta um breve fechamento do capítulo.

6.1 Linguagem alvo

Por se tratar de uma DSL interna de Prolog, a lógica subjacente natural de Ontoprolog é o mesmo fragmento da Lógica de Primeira Ordem expressável por meio de

[1] A sintaxe, e as regras semânticas podem ser ampliadas, como é realizado no Capítulo 8.

cláusulas de Horn (Definição 3.20), tendo um método de *Resolução*[2] como única regra para inferência. O fragmento lógico usado como lógica subjacente é o descrito na subseção 3.1.2, e é especialmente comprometido com o paradigma de Programação em Lógica. Por isso, a *unificação* de variáveis, conceitos meta-lógicos como *cut*, simbolizado por !, e outros recursos próprios de Programação em Lógica Clássica, também estão incluídos.

A expressão *sucede*, utilizada amplamente neste e nos capítulos seguintes, é utilizada para denotar o sucesso na instanciação de todas as variáveis em uma consulta definida realizada contra uma base de dados de um programa em Lógica. Em termos mais gerais, *sucede* significa que determinada relação predicativa *é verdadeira* em um modelo que denote completamente um programa em Lógica.

Com intuito de tornar a notação lógica mais simples de ser lida e próxima à encontrada na formalização em Prolog, além das convenções apresentadas na subseção 3.1.2, no decorrer deste capítulo outras convenções são utilizadas sempre que adequadas, conforme segue.

a) uma "regra" é denotada por um símbolo predicativo (Definição 3.17) definido por meio de cláusulas definidas (Definição 3.14) em que o antecedente não é vazio;

b) as teorias de Ontoprolog são regidas por um conjunto de "regras"[3];

c) um *relação semântica* é um Fato ou cláusula unitária (Definição 3.15), ou seja, uma Cláusula definida (Definição 3.14) com antecedente vazio;

d) o símbolo infixo $=$, lido como "unifica", é a operação de unificação da Programação em Lógica. Em $\varphi = \psi$, φ unifica com ψ (CARLSSON, 2012, p. 617);

e) similarmente, o símbolo $\neq$, lido como "não unifica", é definido como $(\varphi \neq \psi) := \neg(\varphi = \psi)$;

f) listas de termos são apresentados entre colchetes. Dessa forma, se φ é um termo predicativo, e $a_1, \ldots, a_n$ são termos, a lista $[a_1, \ldots, a_n]$ é qualificada por φ como $\varphi([a_1, \ldots a_n])$. As listas são ordenadas e um mesmo elemento pode estar presente mais de uma vez na lista;

g) conjuntos de termos são apresentados entre chaves. Dessa forma, se φ é um termo predicativo, e $a_1, \ldots, a_n$ são termos, o conjunto $\{a_1, \ldots, a_n\}$ é qualificada por φ como $\varphi(\{a_1, \ldots a_n\})$. Diferente de listas, conjuntos são tratados como conteúdo sem repetição e cuja ordem dos elementos não é relevante. O uso de conjuntos é uma abstração da linguagem aqui apresentada, uma vez que não

[2] Como por exemplo, *Selective Linear Definite* (SLD) e *Selective Linear Definite with Negation as Failure* (SLDNF).

[3] Uma regra de Ontoprolog não se confunde com a Regra de Resolução utilizada como regra de inferência da lógica subjacente.

estão presentes como elementos de primeira classe (primitivos) em Prolog, mas são definíveis com base em programas;

h) o símbolo $\in$, lido como "é membro de", é definido como: se a é um termo e δ é um conjunto $\{a_1, \ldots a_n\}$ ou uma lista $[a_1, \ldots a_n]$, então $a \in \delta$ se, e somente se, $a = a_1 \vee \ldots \vee a = a_n$. Dessa forma, $\in$ é usado como substituto ao predicado $member/2$ (CARLSSON, 2012, p. 1.008);

i) de forma similar, o símbolo $\notin$, lido como "não pertence a", é definido como $\neg \in$, ou seja: se a é um termo e δ é um conjunto $\{a_1, \ldots a_n\}$ ou uma lista $[a_1, \ldots a_n]$, então $a \notin \delta$ se, e somente se, $\neg(a = a_1) \wedge \ldots \wedge \neg(a = a_n)$;

j) o símbolo $\subseteq$, lido como "contido em", é definido como: se π_1 é um conjunto $\{a_1, \ldots, a_n\}$ ou uma lista $[a_1, \ldots, a_n]$ e π_2 é um conjunto $\{b_1, \ldots, b_n\}$ ou uma lista $[b_1, \ldots, b_n]$, $\pi_1 \subseteq \pi_2$ se, e somente se, $\forall X(X \in \pi_1 \rightarrow X \in \pi_2)$;

k) o símbolo $\subset$, lido como "contido propriamente em", é definido como: $\pi_1 \subset \pi_1 \coloneqq \pi_1 \subseteq \pi_1 \wedge \pi_1 \neq \pi_2$;

l) o predicado $bagof(Template, Goal, Bag)$, simbolizado por $bagof/3$, unifica Bag com cada sucesso alternativa de $Template$ que sucede para cada contra-exemplo existencial de $Goal$. Dessa forma, Bag é uma lista com a extensão de $Goal$ na forma de $Template$. Nesse caso, $Goal$ representa uma definição indutiva. Detalhes sobre o predicado são abordados em Carlsson (2012, p. 885);

m) o predicado $setof/3$ é similar a $bagof/3$, exceto pelo fato de Bag ser transformado em um conjunto (CARLSSON, 2012, p. 1.120).

Sempre que não houver perda de clareza, uma notação lógica simplificada das fórmulas do fragmento de Primeira Ordem em foco é usada. A simplificação é realizada conforme a subseção 3.1.2. Por exemplo,

$$\forall X \; h(X) \leftarrow \forall Y(b(X,Y) \wedge c(X,Y))$$

pode ser escrito eliminando a notação dos quantificadores como:

$$h(X) \leftarrow b(X,Y) \wedge c(X,Y)$$

Da mesma forma,

$$\forall X \; h(X) \leftarrow \forall Y((b(X,Y) \wedge c(X,Y)) \vee d(X,Y))$$

pode ser escrito como:

$$h(X) \leftarrow (b(X,Y) \wedge c(X,Y)) \vee d(X,Y)$$

e também pode ser escrito com eliminação de parênteses, uma vez que $\vee$ tem precedência sobre $\wedge$:

$$h(X) \leftarrow b(X,Y) \wedge c(X,Y) \vee d(X,Y)$$

ou ainda, pode ser escrito na *notação clausal* (Definição 3.13), que é usada, por exemplo, na seção 6.3:

$$h(X) \leftarrow b(X,Y) \wedge c(X,Y)$$
$$h(X) \leftarrow d(X,Y)$$

Em todos os casos, o símbolo $\wedge$ pode ser substituído pela vírgula (,). Nesse caso, temos a notação clausal ainda mais simplificada, e com alguma indentação opcional que visa otimizar a leitura da fórmula:

$$\begin{aligned} h(X) \leftarrow & b(X,Y), \\ & c(X,Y) \\ h(X) \leftarrow & d(X,Y) \end{aligned}$$

6.2 Estrutura semântica

A semântica de uma especificação Ontoprolog é denominada Teoria Ontoprolog (Definição 5.2), e é dada por meio da estrutura $\mathcal{OT}$:

$$\mathcal{OT} = \langle \mathcal{C}, \mathcal{G}, \mathcal{R}, \mathcal{H} \rangle$$

tal que:

a) $\mathcal{C}$ é um conjunto finito e não vazio de entidades que guardam relação ontológica com o mundo (Definição 5.3);

b) $\mathcal{G}$ é um conjunto finito e possivelmente vazio de entidades lógicas (Definição 5.5), ou seja, de entidades que não exigem compromisso ontológico com o universo descrito, mas fazem parte do universo de objetos meta-teóricos a respeito da realidade;

c) $\mathcal{R}$ é um conjunto finito e não vazio de regras positivas e negativas definidas sobre relações de $\mathcal{H}$. Essas regras regulam a noção de consequência lógica obtida a partir das relações semânticas. A regras de $\mathcal{R}$ são descritas na seção 6.3 e detalhadas na seção 6.4;

d) $\mathcal{H}$ é um conjunto finito e não vazio de *relações semânticas*, representadas por símbolos de predicados primitivos com aridade $= 1$ e de relações lógicas com aridade $= 2$ nas formas $\mathcal{C} \times \mathcal{C}$, $\mathcal{C} \times \mathcal{G}$. Tratam-se de átomos (Definição 3.6) esquemáticos que caracterizam relações lógicas entre entidades ontológicas e/ou entidades lógicas que denotam afirmações sobre os objetos envolvidos em suas estruturas e cujos símbolos predicativos e respectivos atributos são definidos

conforme seguem[4]. As relações aqui apresentadas são introduzidas na seção 5.4. A interpretação de cada relação é apresentada em negrito, de modo que o átomo que nome as relações são mnemônicos dessas interpretações:

- ***entity of the universe of discourse***: $entity(e)$, tal que:
  - $e \in \mathcal{C}$;
  - e, em $entity(e)$, é uma variável que denota a constante atribuída a cada conceito da ontologia no paradigma *Unique-Name Assumption* (subseção 3.1.4), ou seja, cada conceito distinto é representado por um e exatamente um e, e cada e distinto representa um e exatamente um conceito da ontologia, de modo que não exista dois e_1 e e_2 que representem o mesmo conceito, nem um conceito representado por dois e distintos;
  - $entity/1$ é o predicado atribuído a cada elemento do universo de discurso. Portanto, $entity/1$ denota o conjunto universo dos conceitos sobre os quais se atribui predicados, ou seja, os quais participam em relações de $\mathcal{H}$;
  - $entity(e)$ sucede exatamente para todos os elementos contidos em $\mathcal{C}$, e apenas para esses;
- ***direct instance of***: $dio(c_1, c_2)$, tal que:
  - $c_i \in \mathcal{C}$;
  - denota a relação formal "instância direta de", em que c_1 é uma instância de c_2;
  - entende-se por "instância" a noção de "membro de", em que c_1 tem a propriedade de ser membro de c_2 (Instantiation Relation);
- ***direct extension of / direct subsumption of***: $deo(c_1, c_2)$, tal que:
  - $c_i \in \mathcal{C}$;
  - denota o conceito "subsunção direta de", em que c_1 é uma subsunção de c_2;
  - a relação de subsunção, do inglês *subsumption*, é entendida como sinônimo de subordinação em que c_1 é subordinado, ou estende, c_2;
- ***direct exclusive disjunction of***: $dd(\{c_1, \ldots, c_n\})$, tal que:
  - $\{c_1, \ldots, c_n\}$ é um conjunto em que $c_i \in \mathcal{C}$;
  - denota o predicado "disjunção exclusiva direta" dos conceitos $\{c_1, \ldots, c_n\}$: os conceitos $\{c_1, \ldots, c_n\}$ são disjuntos no sentido de que não é o caso que aconteça um $c \in \mathcal{C}$ qualquer ser *instância de* c_a e de c_b, ou instância de algum c_{a1} e de algum c_{b1} que estendam transitiva, reflexiva e respectivamente c_a e c_b, para $c_a \neq c_b$ e $c_{a1} \neq c_{b1}$, sendo $\{c_a, c_b\} \subseteq \{c_1, \ldots, c_n\}$;
- ***direct complete subsumption***: $dcomplete(c_t, \{c_1, \ldots, c_n\})$, tal que:

[4] As definições que seguem visam apresentar as relações lógicas que denotam relações ontológicas entre conceitos. Definições formais das restrições lógicas que essas intuições implicam são apresentadas na seção 6.3 e na seção 6.4.

  - $c_t \in \mathcal{C}$;
  - $\{c_1, \ldots, c_n\}$ é uma lista em que $c_i \in \mathcal{C}$;
  - denota a relação "subsunção completa de", em que $\{c_1, \ldots, c_n\}$ são taxativamente todos os conceitos subordinados (ou que estendem) c_t, ou seja, não é o caso que exista um c_m que participa da relação $deo(c_m, c_t)$ e que $c_m \notin \{c_1, \ldots, c_n\}$ se ocorre $dcomplete(c_t, \{c_1, \ldots, c_n\})$;
  - cada ocorrência de $dcomplete/2$ denota uma partição conceitual das subsunções de c_t;
- ***power type***: $pt(c_1, c_2)$, tal que:
  - $c_i \in \mathcal{C}$;
  - denota que se pode inferir que instâncias de c_1 estendem c_2;
- ***direct meta-property on***: $dmo(m, c)$, tal que:
  - $c \in \mathcal{C}$;
  - $m \in \mathcal{G}$;
  - denota uma meta-propriedade, em que m é uma meta-propriedade atribuída a c;
- ***direct property on***: $dpo(p, c)$, tal que:
  - $\{p, c\} \subseteq \mathcal{C}$;
  - denota uma entidade p é uma propriedade relacionada a c;
- ***direct property value***: $dpv(at(p, c), v)$, tal que:
  - $\{p, c\} \in \mathcal{C}$;
  - $v \in \mathcal{G}$;
  - denota que o valor atribuído à propriedade p na entidade c é v;
- ***direct subset of***: $dso(c_1, c_2)$, tal que:
  - $c_i \in \mathcal{C}$;
  - denota o conceito "subconjunto direto de", em que as instâncias de c_1 é um subconjunto direto das instâncias de c_2, conforme subseção 4.4.6;
- ***direct redefinition of***: $dro(c_1, c_2)$, tal que:
  - $c_i \in \mathcal{C}$;
  - denota o conceito "redefinição direta de", em que c_1 é uma redefinição direta de c_2, conforme subseção 4.4.6;
- ***defeasible***: $defeasible(r)$, tal que:
  - $r \in \mathcal{H}$ e r diferente de $defeasible/1$;
  - denota que a relação r é retratável[5].

[5] Trata-se da estrutura de dados para um raciocínio baseado em *Default Logic*

6.3 Arcabouço axiomático da semântica da linguagem

Esta seção discute a estratégia utilizada na definição das regras presentes em $\mathcal{R}$. Há dois tipos de regras: as regras negativas escritas de modo positivo, abordadas na subseção 6.3.1 e detalhadas na subseção 6.4.3, e as regras positivas propriamente ditas, descritas na subseção 6.3.2 e detalhadas na subseção 6.4.1. A subseção 6.3.3 aborda a semântica do predicado $defeasible/1$ e a subseção 6.3.4 teça discussões a respeito de uma metáfora de *views* de banco de dados, que pode ser empregada como uma explicação alternativa à interpretação padrão de Programação em Lógica dada sobre as regras de $\mathcal{R}$.

6.3.1 Regras negativas escritas de modo positivo

Em uma Cláusula de Horn, mais precisamente em uma Cláusula definida (Definição 3.14), o consequente é formado precisamente por apenas um literal positivo (átomo). Por isso, não é possível diretamente escrever axiomas negativos – por exemplo, que denotem informalmente a expressão "não é o caso que ocorre que φ" – em uma teoria descrita em um arcabouço de Programação em Lógica, como é o caso de Ontoprolog. Por isso, conforme a linguagem apresentada na subseção 3.1.2, axiomas negativos de Ontoprolog são escritos como *regras positivas* na forma $h \leftarrow \varphi$, em que h é o consequente da implicação, chamado de *cabeça* (Definição 3.14), que consiste precisamente de um átomo (Definição 3.6), e φ é um esquema de conjunções na forma $\psi_1 \wedge \ldots \wedge \psi_n$ para $n > 0$, em que ψ_i são literais que, conforme a Definição 3.12 (Literal), são possivelmente negativos. Conforme a Definição 3.17 (Definição de Símbolo predicativo), o conjunto de todas as cláusulas com o mesmo símbolo predicativo p na cabeça é chamado de *definição* de p. Portanto, uma regra p pode ser definida como um conjunto disjunto de literais.

Conforme introduzido na seção 5.2, no caso das traduções de regras negativas em regras positivas, a *cabeça* da regra denota uma situação inconsistente do programa – no caso, da *Teoria* (Definição 5.2). A conjunção das negações das *cabeças* de regras negativas traduzidas como regras positivas – doravante identificados pelo acrônimo RNP – cujo método *Resolução* implicam em vazio ($\emptyset$), ou seja, a conjunção das negações das RNP que resultam em *True*, indicam que uma Teoria Ontoprolog é consistente com todas as regras que a regem. Ou, dito de outra forma, $\emptyset$ indica que não há contra-exemplos que desautorizem a conclusão de que o programa seja consistente com as regras.

Por exemplo, considere o axioma da Equação 6.1 escrito em FOL, que denotaria a noção de que não é o caso que exista um C que é simultaneamente instância de dois conceitos, X e Y, diferentes.

$$\forall C \; \forall X \; \forall Y (\neg dio(C, X) \wedge \neg dio(C, Y) \wedge (X \neq Y)) \tag{6.1}$$

Uma RNP equivalente à Equação 6.1 poderia ser escrita conforme a Equação 6.2,

em que o consequente $dio\_non\_unique/1$ denota o caso em que acontece de existir na teoria um C que é simultaneamente instância de dois conceitos X e Y diferentes.

$$\forall C\ dio\_non\_unique(C) \leftarrow \forall X \forall Y (dio(C, X) \wedge dio(C, Y) \wedge X \neq Y) \tag{6.2}$$

Eliminando parênteses, quantificadores, e trocando o símbolo da conjunção ($\wedge$) pela vírgula (,), conforme descrito na subseção 3.1.2, a Equação 6.3 apresenta a forma final simplificada de uma RNP.

$$\begin{aligned} dio\_non\_unique(C) \leftarrow & dio(C, X), \\ & dio(C, Y), \\ & X \neq Y \end{aligned} \tag{6.3}$$

Nesse caso, um programador poderia definir um átomo $consistent/0$, conforme a Equação 6.4[6], de modo que $consistent/0$ denote o caso em que não existe um X tal que $dio\_non\_unique(X)$ seja o caso:

$$consistent \leftarrow \neg dio\_non\_unique(\_) \tag{6.4}$$

Para um número n de RNP φ, $consistent/0$ seria definido conforme a Equação 6.5.

$$consistent \leftarrow \neg\varphi_1 \wedge \ldots \wedge \neg\varphi_n \tag{6.5}$$

Embora a solução apresentada na Equação 6.5 seja factível e suficiente para os objetivos de verificar a consistências das RNP em Teorias Ontoprolog, do ponto de vista da engenharia de programas em Lógica essa solução poderia ser criticada devido ao fator de qualidade da manutenabilidade dos programas que, nesse caso, reduzira a qualidade global do software. Isso ocorre porque, sempre que uma nova RNP fosse inserida ou removida da Teoria Ontoprolog, a definição de $consistent/0$ deveria ser atualizada para que considerasse a nova regra, ou desconsiderasse uma regra removida. Diante dessa situação indesejável, seria interessante que fosse possível tratar os predicados das RNP em uma lógica de ordem superior – possivelmente como sugere Naish (1996). Dessa forma, seria possível quantificar sobre seus termos predicativos, o que permitiria modelar um tratamento padrão para esse tipo de regra. Porém, para que ainda se permaneça no fragmento de FOL descrito na subseção 3.1.2 aderente a maioria das implementações de Prolog existentes baseados a série ISO/IEC 13211 (ISO/IEC, 1995), uma solução possível é encapsular as RNP num conjunto de cláusulas definidas de uma única cabeça, agrupadas por suas disjunções. Com esse objetivo, define-se o predicado $ngrule/n$ (de *negative global rule*, em oposição às *modal rules* descritas no Capítulo 9), em que $n \geq 1$ é a aridade de $ngrule$, conforme a Definição 6.1:

[6] Na fórmula Equação 6.4, o símbolo _ (sublinhado) é uma Variável anônima (Definição 3.2), ou seja, trata-se de uma variável existencialmente quantificada.

Definição 6.1 (Regra de validação de teoria clássica)**.** A *regra de validação clássica* é um conjunto de disjunções na forma $ngrule(R_1, V_{1_1}, \ldots, V_{1_m}) \vee \ldots \vee ngrule(R_n, V_{n_1}, \ldots, V_{n_z})$ em que R_i é uma constante que denota uma RNP e $V_{i_1}, \ldots, V_{i_m}$ e $V_{n_1}, \ldots, V_{i_z}$ são variáveis que qualificam R_i, para $n \geq 0$, $m \geq 0$, $z \geq 0$ e i variando entre 1 e n. Com essa definição, o termo predicativo *ngrule* possui aridade variável, por exemplo, a, e é identificado como $ngrule/a$, para $a \geq 1$. —

Com base na Definição 6.1, como $ngrule/n$ possui aridade variável, e como não se está num arcabouço de ordem superior, a definição de $consistent/0$ deve ser feita para cada forma possível de $ngrule/n$. Com isso em tela, $consistent/0$ poderia ser definido conforme a Equação 6.6, para o caso de $ngrule/2$ e $ngrule/3$.

$$consistent \leftarrow \neg ngrule(\_, \_), \neg ngrule(\_, \_, \_) \tag{6.6}$$

O que, de forma geral, para um n qualquer, a definição seria conforme a Equação 6.7.

$$consistent \leftarrow \neg ngrule(\_, \_), \neg ngrule(\_, \_, \_), \ldots, \neg ngrule(\_, \_, \ldots, \_) \tag{6.7}$$

Dessa forma, o exemplo da Equação 6.3 poderia ser reescrito conforme a Equação 6.8, que materializa a forma final da RNP.

$$\begin{aligned} ngrule(dio\_non\_unique, X, Y) \leftarrow & dio(C, X), \\ & dio(C, Y), \\ & X \neq Y \end{aligned} \tag{6.8}$$

6.3.1.1 Característica não-monotônica dos programas em Lógica

Há uma justificativa adicional para a existência das RNP. Ocorre que os programas em Lógica são naturalmente não-monotônicos, de modo que uma demonstração realizada no início do programa pode ser revogada (no sentido de retratada, do inglês *defeasible*) no decorrer da execução do programa.

Isso ocorre com Prolog, por exemplo, no caso de uso de termos como $assert/1$ e $retract/1$, em que o primeiro adiciona à, e o outro remove da base de fatos e de regras um fato (Definição 3.15) novo, ou existente, durante a execução do programa. Desse modo, a avaliação de uma regra do tipo $ngrule(dio\_non\_unique, X, Y)$ pode resultar em situações diferentes dependendo do programa específico em tela.

Por essa característica não monotônica, as regras de Ontoprolog não são *axiomas* clássicos propriamente ditos, no sentido de algo dado à priori que tornaria um sistema classicamente inconsistente caso ocorresse um $\varphi \wedge \neg\varphi$, para qualquer φ na forma de termos. Por esse motivo, uma nova semântica ao conceito de consistência se faz necessária. Essa semântica é apresentada de forma intuitiva pela regra do tipo $consistent/0$, descrita no

caput, em que um programa Ontoprolog é dito consistente se nenhum $ngrule/n$ for o caso para a teoria em tela durante a execução do programa, ou seja, o uma teoria Ontoprolog *é consistente em relação às ngrule/n* que o definem. Nos exemplos apresentados na seção 7.7 e na seção 8.4, o predicado `check_semantics/0` é a implementação concreta da noção de $consistent/0$.

6.3.2 Regras positivas

Axiomas e definições positivas, como as apresentadas na subseção 6.4.1.1 para $hypertypeof/2$ (Definição formal 6.18) e $type/1$ (Definição formal 6.21), por exemplo, são escritas de forma tradicional, ou seja, como regras positivas na forma $h \leftarrow \varphi$, em que h é o termo predicativo definido, e φ é o esquema de conjunções que define h.

Fechos de relações, como o caso do $extensionof/2$ (Definição formal 6.2) que é um fecho transitivo para $deo/2$, são definidos de modo a evitar-se o problema da recursão à esquerda descrito em Sterling e Shapiro (1999, p. 132). Nesses casos, um novo predicado é criado para indicar o fecho (transitivo) da relação original.

6.3.3 A semântica da relação $defeasible/1$

Conforme apresentado na seção 6.2, a relação $defeasible/1$ denota relações retratáveis de $\mathcal{H}$. Esse tipo de semântica reflete característica não monotônica (subseção 6.3.1.1) das teorias da linguagem. Porém, na versão apresentada neste trabalho, as teorias de Ontoprolog não lançam mão desse recurso no contexto das regras semânticas de $\mathcal{R}$. Ou seja, nas definições apresentadas na seção 6.4, nenhuma regra utiliza o predicado semântico $defeasible/1$.

O predicado é utilizado, no entanto, pelo procedimento de expansão das relações semânticas empregado pela função de tradução da sintaxe de Ontoprolog descrito pela subseção 7.3.3, subseção 8.2.3 e detalhadas no Apêndice F e no Apêndice G. Por exemplo, o caso da função filtro apresentado na subseção 8.2.2.5 traduz um fragmento de sentença em relações semânticas qualificadas com $defeasible/1$. Estas, por sua vez, são expandidas em relações semânticas base conforme as regras definidas no Apêndice F.

Dessa forma, o predicado $defeasible/1$ faz parte das relações semânticas de Ontoprolog, mas é mantido como um predicado chave utilizado atualmente somente pelo algoritmo de tradução mencionado e mantido na linguagem para evolução futura.

6.3.4 Metáfora das *views* de banco de dados dedutivo

O arcabouço dedutivo e o paradigma de Programação em Lógica adotado com Ontoprolog está relacionado com o paradigma usado nos banco de dados dedutivos (COLOMB, 1998). Nesse paradigma, embora outros tipos de mecanismos de inferência

sejam usados, como o caso de *magic sets* e *forward chaining*, as consultas definidas (Definição 3.18) são interpretadas como projeções, ou *views*. Dessa forma, uma *view* garante uma restrição de integridade relacional quando essa *view* é vazia. Ou, de outra forma, uma integridade relacional em um banco de dados dedutivo não é atendida quando uma *view* que garante essa integridade não é atendida.

Nessa linha, pode-se entender que a consistência semântica de teorias Ontoprolog é garantida por meio de uma única regra que é a conjunção das negações de uma série de predicados afirmativos que representam *views* em um banco de dados dedutivo. Essas *views* representam um conjunto de dados que atendem ao critério estabelecido na definição do predicado. Essencialmente, elas representam situações em que a consistência não é garantida.

Devido ao estreito relacionamento entre Programação em Lógica e Banco de Dados Dedutivos, registra-se como sugestão a trabalhos futuros (seção 11.3) a evolução desta pesquisa com intuito de implementar Ontoprolog como um banco de dados daquele tipo.

6.4 Definições e regras base de $\mathcal{R}$

As definições e regras apresentados nesta seção seguem o arcabouço lógico apresentado na seção 6.3 referentes às regras semânticas gerais dos elementos de $\mathcal{R}$ da estrutura $\mathcal{OT}$. Essas regras são gerais porque se referem exclusivamente à Metateoria de *Hypertypes* (Definição 5.10) e a restrições sobre as relações e predicados de $\mathcal{H}$ em que os demais elementos da estrutura $\mathcal{OT}$ podem figurar. Do ponto de vista de arquitetura de software, as definições e regras presentes nesta seção referem-se a uma camada mais abstrata do que os presentes na seção 8.3, que são definições e regras específicas para validação de relações de conceitos em relação à metateoria da UFO.

A expressão "Definição formal" é utilizada doravante para identificar tanto regras quanto definições que ocorrem no contexto $\mathcal{R}$ da estrutura semântica de Ontoprolog. No transcorrer do texto, as palavras "regra" e "definição" são usadas como sinônimos, uma vez que, considerando a seção 6.1, o contexto em que são utilizadas contempla informação suficiente para distinguir o uso preciso dos termos.

6.4.1 Definições positivas sobre as relações de $\mathcal{H}$

As definições desta seção tratam-se de definições positivas (subseção 6.3.2) construídas a partir das relações presentes em $\mathcal{H}$. Algumas definições são fechos transitivos, como o caso da Definição formal 6.1 ($extensionof\_irreflexive/2$), enquanto outras são definições que funcionam como lemas para outras, como Definição formal 6.8 ($disjoint\_types/1$), ou ainda, são apenas definições propriamente ditas, úteis para o usuário final da linguagem, como o caso da Definição formal 6.12 ($drel/5$).

Inicia-se com um primeiro grupo de definições referentes à relação $deo/2$:

Definição formal 6.1 ($extensionof\_irreflexive/2$)**.** Trata-se do fecho transitivo da relação $deo/2$ que denota que se um conceito $c \in \mathcal{C}$ é uma subsunção de $s_1 \in \mathcal{C}$ e s_1 é uma subsunção de s_2, então c é uma subsunção de s_2, e s_2 é um *super tipo* de c e de s_1. A relação é definida do seguinte modo. Para uma variável $T \in \mathcal{C}$ e uma variável $S \in \mathcal{C}$ que denota um *super tipo* de T, temos que:

$$extensionof\_irreflexive(T, S) \leftarrow deo(T, S)$$
$$extensionof\_irreflexive(T, S) \leftarrow deo(T, X), \quad extensionof\_irreflexive(X, S)$$

Por definição, $extensionof\_irreflexive/2$ é uma relação recursiva ascendente, transitiva e assimétrica, mas não reflexiva. Isso decorre do fato de $extensionof\_irreflexive/2$ ser definida sobre $deo/2$, que é irreflexiva e assimétrica por definição. —

Nota 6.1. A implementação concreta de *extensionof_irreflexive/2* lança mão da técnica de *memoization* para que o laço presente na definição seja protegido. Com base nisso, se em determinada teoria inconsistente com as regras da subseção 6.4.3 ocorrer um caso reflexivo ou simétrico de $deo/2$, o predicado $extensionof\_irreflexive/2$ não implicará em um *loop* infinito. Isso se dá pela definição concreta presente no Código 3.

Código 3 – Definição concreta de $extensionof\_irreflexive/2$

```
extensionof_irreflexive(T, S) :- extensionof_irreflexive(T, S).

extensionof_irreflexive(Type, _, Memo_List) :-
        util_memberchk(Type, Memo_List),
        !.
extensionof_irreflexive(Type, Super_Type, _) :-
        deo(Type, Super_Type).
extensionof_irreflexive(Type, Super_Type, Memo_List) :-
        deo(Type, X),
        extensionof_irreflexive(X, Super_Type, [Type|Memo_List]).
```

—

Definição formal 6.2 ($extensionof/2$)**.** Definido como sinônimo de *extensionof_irreflexive/2*:

$$extensionof(T, S) \leftarrow extensionof\_irreflexive(T, S)$$

—

Definição formal 6.3 ($deo\_hierarchy/2$)**.** A exemplo de *extensionof/2*, $deo\_hierarchy/2$ é definido como outro sinônimo de *extensionof_irreflexive/2*:

$$deo\_hierarchy(T, S) \leftarrow extensionof\_irreflexive(T, S)$$

—

Definição formal 6.4 ($extensionof\_reflexive/2$)**.** Trata-se do fecho reflexivo de $extensionof/2$. O fecho é realizado com base nas entidades contidas em $\mathcal{C}$, denotadas por $entity(T)$:

$$extensionof\_reflexive(T,T) \leftarrow entity(T)$$
$$extensionof\_reflexive(T,S) \leftarrow extensionof(T,S)$$

—

As definições seguintes são estabelecidas sobre a relação $dio/2$.

Definição formal 6.5 ($instanceof/2$)**.** $instanceof/2$ é uma relação definida sobre $dio/2$ e $extensionof/2$ que denota que se um conceito $c \in \mathcal{C}$ é *instância direta* de um $t \in \mathcal{C}$, e que se, de modo irreflexivo, um t é subsunção de $s \in \mathcal{C}$, então c é uma *instância* de s. Além disso, para qualquer $c \in \mathcal{C}$ que é *instância direta* de $t \in \mathcal{C}$, c é também *instância* de t. Nesse sentido, a relação $instanceof/2$ é definida como:

$$instanceof(T,M) \leftarrow dio(T,M)$$
$$instanceof(T,S) \leftarrow dio(T,M),\ extensionof(M,S)$$

—

Observar que $dio/2$ denota *instância direta*, enquanto $instanceof/2$ denota o conceito *instância* de forma genérica e recursiva ascendente com base na relação $deo/2$, devido à definição de $extensionof\_irreflexive/2$, usada na definição de $instanceof/2$.

Definição formal 6.6 ($dio\_hierarchy/2$)**.** A exemplo de $extensionof\_irreflexive/2$, $dio\_hierarchy/2$ é uma relação irreflexiva e transitiva. Trata-se do fecho transitivo de $dio/2$, de modo que $dio\_hierarchy/2$ suceda para toda a hierarquia de instâncias de determinada entidade.

$$dio\_hierarchy(T,M) \leftarrow dio(T,M)$$
$$dio\_hierarchy(T,S) \leftarrow dio(T,M),\ dio\_hierarchy(M,S)$$

—

Nota 6.2. Ainda conforme $extensionof\_irreflexive/2$, a implementação concreta de $dio\_hierarchy/2$ lança mão da técnica de *memoization*, construída de forma semelhante à descrita na Nota 6.1. —

Definição formal 6.7 ($disjoint\_types/2$)**.** $disjoint\_types/2$ é uma relação definida sobre $dd/2$ que denota que duas entidades c_1 e $c_2 \in \mathcal{C}$, para $c_1 \neq c_2$, são disjuntas se

são extensões reflexivas de dois conceitos d_1 e d_2, em que $\{d_1, d_2\} \subseteq D$ e D possui a propriedade $dd(D)$, para $d_1 \neq d_2$. Dessa forma, a relação é definida como:

$$
\begin{aligned}
disjoint\_types(T1, T1) \leftarrow & !, fail \\
disjoint\_types(T1, T2) \leftarrow & dd(DD\_List), \\
& D1 \in DD\_List, \\
& D2 \in DD\_List, \\
& D1 \neq D2, \\
& extensionof\_reflexive(T1, D1), \\
& extensionof\_reflexive(T2, D2), \\
& T1 \neq T2.
\end{aligned}
$$

Em que $!, fail$ denota que a relação $disjoint\_types/2$ não sucede sobre o caso reflexivo. Nesse caso, $member(A, B)$ sucede para cada elemento A pertencente à lista B, ou seja, para cada $A \in B$. —

Definição formal 6.8 ($disjoint\_types/1$)**.** $disjoint\_types/1$ é uma generalização da Definição formal 6.7 ($disjoint\_types/2$) em que é possível verificar se uma lista arbitrária de elementos é disjunta. A definição é dada como se segue:

$$
\begin{aligned}
disjoint\_types(List) \leftarrow & var(List), !, fail \\
disjoint\_types([\_]) \leftarrow & fail, ! \\
disjoint\_types(List) \leftarrow & combine2(X, Y, List), \\
& disjoint\_types(X, Y)
\end{aligned}
$$

O primeiro caso denota que $disjoint\_types/1$ não sucede para uma variável ($var/1$). O segundo refere-se ao caso em que a lista contém um único elemento. Nesse caso, a relação também não sucede, uma vez que um único elemento não pode ser considerado disjunto de nenhum outro. Já no último caso, $combine2/2$ sucede para cada par $\langle X, Y \rangle$ contido em $List$, de modo que $\langle X, Y \rangle$ é a combinação dada de dois em dois dos elementos da lista $List$. Dessa forma, $disjoint\_types/1$ sucede quando ao menos um par da combinação de $List$ for disjunta conforme $disjoint\_types/2$ (Definição formal 6.7). —

Definição formal 6.9 ($complete/2$)**.** Trata-se de uma normalização semântica de $dcomplete/2$. Cada ocorrência de $dcomplete(c_t, \varphi)$, para $\varphi = \{c_1, \ldots, c_n\}$, denota uma partição conceitual das subsunções de c_t. Ocorre que todo $\varphi' \subset \varphi$ é também uma partição conceitual de c_t. Dessa forma, $complete/2$ sucede apenas para todos os φ. Por isso, $dcomplete(c_t, \varphi')$ sucede para todo $\varphi' \subset \varphi$. Na regra seguinte, φ' é representado por C.

$$
\begin{aligned}
complete(S, C) \leftarrow & setof(X, dcomplete(S, X), CS), \\
& C \in CS, \\
& ([C] = CS \vee ([C] \neq CS \wedge \neg(C1 \in CS \wedge C1 \subset C)))
\end{aligned}
$$

—

Definição formal 6.10 ($property\_value/2$)**.** A relação $property\_value/2$ é uma distribuição da relação $dpv/2$ sobre $instanceof/2$ tomando como hierarquia ascendente os tipos superiores MP da propriedade $P \in \mathcal{C}$, em que $T \in \mathcal{C}$ é uma entidade e $V \in \mathcal{G}$ é uma relação lógica que representa um valor da propriedade P em T:

$$property\_value(at(MP,T),V) \leftarrow dpv(at(P,T),V), \\ instanceof(at(P,T),MP)$$

Desse modo, toda entidade T que tenha um valor atribuído a alguma propriedade P, a relação $property\_value/2$ será verdadeira para todo tipo superior do qual P é instância.

Porém, como a definição da relação em tela é defina sobre $instanceof/2$, e como $instanceof/2$ apenas sucede para um nível teórico, a semântica desta relação cobre apenas um nível de instância de propriedades, ficando em aberto a semântica de mais de dois níveis de instância de propriedades. —

Definição formal 6.11 ($propertyon/2$)**.** A relação $propertyon/2$ é a distribuição da relação $dpo/2$ sobre $extensionof/2$ e $dio\_hierarchy/2$, conforme segue:

$$\begin{aligned} propertyon(\_,\_) &\leftarrow dpo(property,\_), \\ &\quad !, fail \\ propertyon(P,T) &\leftarrow dpo(P,T) \\ propertyon(P,T) &\leftarrow dpo(P,S), \\ &\quad extensionof(T,S), \\ &\quad \neg(dpo(SP,T), extensionof(SP,P)) \\ propertyon(P,T) &\leftarrow dpo(P,M), \\ &\quad dio\_hierarchy(T,M), \\ &\quad \neg(extensionof(SP,P), dpo(SP,T)) \end{aligned}$$

—

Definição formal 6.12 ($drel/5$)**.** Trata-se de uma relação que concatena um conjunto de afirmações que denotam uma relação ontológica. $rel/5$ é definido conforme segue, em que R é uma entidade que denota uma relação binária e $domain$, $domain\_cc$, $codomain$ e $codomain\_cc$ denotam entidades instâncias de propriedades apresentadas pela Figura 26:

$$\begin{aligned} drel(R, Domain, Domain\_CC, Codomain, Codomain\_CC) \leftarrow \\ property\_value(at(domain,R), Domain), \\ property\_value(at(domain\_cc,R), Domain\_CC), \\ property\_value(at(codomain,R), Codomain), \\ property\_value(at(codomain\_cc,R), Codomain\_CC) \end{aligned}$$

—

Definição formal 6.13 ($rel/5$)**.** Trata-se do fecho transitivo das relações de subsunção reflexiva Definição formal 6.4 ($extensionof\_reflexive/2$) da hierarquia de tipos de domínios e codomínios da relação $drel/5$ (Definição formal 6.12). O fecho transitivo é limitado à existência da relação de redefinição $redefinitionof/2$ (Definição formal 6.14), que suspende a corrente transitiva.

$$
\begin{aligned}
&rel(R, Domain, Domain\_CC, Codomain, Codomain\_CC) \leftarrow \\
&\quad drel(R, Domain, Domain\_CC, Codomain, Codomain\_CC), \\
&\quad extensionof\_reflexive(D, Domain), \\
&\quad extensionof\_reflexive(C, Codomain), \\
&\quad \neg rel1(R, D, C)
\end{aligned}
$$

$$
\begin{aligned}
&rel1(R, D, \_) \leftarrow \\
&\quad redefinitionof(R1, R), \\
&\quad drel(R1, D1, \_, \_, \_), \\
&\quad extensionof\_reflexive(D, D1). \\
&rel1(R, \_, C) \leftarrow \\
&\quad redefinitionof(R1, R), \\
&\quad drel(R1, \_, \_, C1, \_), \\
&\quad extensionof\_reflexive(C, C1)
\end{aligned}
$$

—

Definição formal 6.14 ($redefinitionof/2$)**.** Trata-se da distribuição transitiva da relação de redefinição de relações ($dro/2$) sobre a subsunção irreflexiva $extensionof\_irreflexive/2$ (Definição formal 6.1).

$$
\begin{aligned}
&redefinitionof(P, S) \leftarrow \\
&\quad dio(P, T), \\
&\quad dro(T, S) \\
&redefinitionof(T, S) \leftarrow \\
&\quad dro(T, S) \\
&redefinitionof(T, SS) \leftarrow \\
&\quad dro(T, S), \\
&\quad extensionof\_irreflexive(S, SS)
\end{aligned}
$$

—

Definição formal 6.15 ($drefine/3$)**.** Trata-se de uma relação que denota a união (a disjunção) das relações $deo/2, dro/2$ e $dso/2$, uma vez que essas relações, quando definidas sobre relações ou propriedades de relações, compartilham características comuns, conforme apresentado pela subseção 6.4.3. Desse modo, a relação $drefine(R, A, B)$ denota que A é um refinamento de B pela relação formal R, em que $R \in \{dso, dro, deo\}$.

$$
\begin{aligned}
drefine(dro, at(P_a, R_a), at(P_b, R_b)) \leftarrow \\
\qquad dro(at(P_a, R_a), at(P_b, R_b)) \\
drefine(dso, at(P_a, R_a), at(P_b, R_b)) \leftarrow \\
\qquad dso(at(P_a, R_a), at(P_b, R_b)) \\
drefine(deo, at(domain, R_a), at(domain, R_b)) \leftarrow \\
\qquad deo(R_a, R_b), \\
\qquad property\_value(at(domain, R_a), \_), \\
\qquad property\_value(at(domain, R_b), \_) \\
drefine(deo, at(codomain, R_a), at(codomain, R_b)) \leftarrow \\
\qquad deo(R_a, R_b), \\
\qquad property\_value(at(codomain, R_a), \_), \\
\qquad property\_value(at(codomain, R_b), \_)
\end{aligned}
$$

No caso da relação $deo/2$, pelo fato de ela ser estabelecida entre as entidades e não sobre as propriedades, os dois últimos casos de $drefine/3$ são definidos para as propriedades *domain* e *codomain* sempre que for o caso que a relação $deo/2$ tenha sido definida sobre entidades que possuam essas propriedades. —

6.4.1.1 Níveis teóricos

Nesta seção apresenta-se definições especiais referentes a ideia de *níveis teóricos* (Definição 5.8) utilizadas nas seções seguintes.

Definição formal 6.16 ($hypertype/1$)**.** $hypertype(H)$ denota que H é uma entidade que possui o meta-atributo *hypertype*, em que $hypertype \in \mathcal{G}$. Esse meta-atributo denota que a entidade H é um *hypertype* da *teoria superior*:

$$hypertype(H) \leftarrow dmo(hypertype, H)$$

—

$top\_hypertype/1$ define a noção de *hypertype raiz*:

Definição formal 6.17 ($top\_hypertype/1$)**.** $top\_hypertype(H)$ denota que H é uma entidade que possui o meta-atributo *hypertype* e que esse H não é extensão de nenhuma

outra entidade, de modo que H seja um *hypertype raiz*.

$$top\_hypertype(H) \leftarrow dmo(hypertype, H), \\ \neg deo(H, \_)$$

—

Nota 6.3. Conforme o Apêndice A, embutido em Ontoprolog há ao menos dois *hypertypes raízes*, a saber, `type` e `property`. —

As próximas definições são auxiliares para as formalizações presentes nas seções seguintes.

Definição formal 6.18 ($hypertypeof/2$)**.** Com base em $hypertype/1$ e em $dio\_hierarchy/2$, define-se uma relação entre uma entidade E e uma entidade H na teoria de *hypertypes* na qual aquela entidade é instância direta ou indireta desta:

$$hypertypeof(E, H) \leftarrow dio\_hierarchy(E, H), \\ \neg var(H), \\ hypertype(H)$$

Nessa forma, $hypertypeof/2$ sucede para o *hypertype* imediato de determinada entidade E, independente do grau das relações de instância. —

Definição formal 6.19 ($hypertypesof/2$)**.** Semelhante à $hypertypeof/2$. Porém, $hypertypesof/2$ sucede para todos os *hypertypes* de um determinado E que estejam (os *hypertypes*) numa mesma hierarquia ligada pela relação $deo/2$.

$$hypertypeof(E, S) \leftarrow hypertypeof(E, H), \\ extensionof\_reflexive(H, S)$$

Este predicado transpõe todos os níveis de instanciação, de modo que as entidades E sejam categorizadas pelos *hypertypes* que as fundamentam. —

Definição formal 6.20 ($top\_hypertypeof/2$)**.** A exemplo de $hypertypeof/2$, trata-se de uma relação entre uma entidade E e um *hypertype raiz* H na teoria de *hypertypes* na qual aquela entidade é instância desta. Todos os níveis de instanciação são transpostos, de modo que as entidades E sejam categorizadas pelos *hypertypes* que as fundamentam.

$$hypertypeof(E, H) \leftarrow hypertypesof(E, H), \\ top\_hypertype(H)$$

Nessa forma, se $top\_hypertypeof/2$ sucede e a teoria é consistente, o predicado sucede para um único *hypertype* H, de modo que ele represente a categoria mais geral em que determinada entidade E participe como instância direta ou indireta. —

Por fim, ainda no âmbito da ideia de níveis teóricos, *type*/1 define a ideia de *type*:

Definição formal 6.21 (*type*/1)**.** Diferente de *hypertype*/1, *type*/1 é um predicado definido não sobre a relação *dmo*/2, mas a partir da relação *instanceof*/2, em que $type \in \mathcal{C}$. Trata-se da afirmação de que toda instância de *type* "*é um*" *type*:

$$type(T) \leftarrow instanceof(T, type)$$

—

6.4.2 Definições de restrições de cardinalidade

As definições presentes nesta seção referem-se às restrições de cardinalidades, que tratam-se de um tipo específico de relações lógicas.

Definição 6.2 (Restrições de cardinalidade)**.** *Restrições de cardinalidade* são informações referentes ao número de instâncias de conceitos que podem (ou devem) participar de relações entre conceitos. Tratam-se da interpretação semântica dos Indicadores de restrição de cardinalidade (Definição 7.9). —

Dessa forma, seja $\mathcal{A} \subseteq \mathcal{G}$ um conjunto finito (possivelmente vazio) de restrições de cardinalidade em uma das formas $\{\alpha, \alpha..\alpha\}$, em que $\alpha = n \in \mathbb{N}$ ou $\alpha = many$, e que *many* é uma constante e $\mathbb{N}$ é o conjunto de números naturais (inteiros não-negativos com zero incluído), em que:

a) como regra de normalização, o caso $\{\alpha\}$ deve ser interpretado como $\{\alpha..\alpha\}$, para todo α;

b) para $\{\alpha..\beta\}$, α indica a cardinalidade mínima (ou inferior), e β a cardinalidade máxima (ou superior) de elementos da relação;

c) *many* deve ser lido como "muitos", e implica que a cardinalidade é ≥ 0 (Definição formal 6.22 (*card*/1));

Nota 6.4. Embora os predicados referentes às restrições de cardinalidade estejam sobre o domínio definido como um subconjunto de $\mathcal{G}$ da estrutura $\mathcal{OT}$ – que refere-se às relações lógicas gerais – os predicados apresentados nesta seção são independentes das relações de $\mathcal{H}$, ou seja, nenhuma relação de $\mathcal{H}$ participa da definição de predicados referentes às restrições de cardinalidade. Isso significa que esses predicados podem ser usados em outros contextos que não sejam dependentes de elementos da estrutura $\mathcal{OT}$, de modo que essencialmente representem apenas funções que mapeiam seus parâmetros no domínio $\{True, False\}$. —

Definição formal 6.22 (*card*/1)**.** Uma restrição de cardinalidade pode possuir duas formas: $\{\alpha\}$, chamada *caso ou forma simples*, ou $\{\alpha..\beta\}$, chamada *caso ou forma composta*.

No segundo caso, se α e β são números inteiros, eles devem ser apresentados na ordem natural. A definição completa é esta:

$$\begin{aligned} card(C) &\leftarrow card\_base(C) \\ card(L..many) &\leftarrow card\_base(L) \\ card(L..U) &\leftarrow card\_base\_int(L), \\ &\quad\ card\_base\_int(U), \\ &\quad\ L \leq U \end{aligned}$$

em que $many$ é uma constante, e L, U, C são variáveis. As demais relações são definidas na sequência desta seção. —

Definição formal 6.23 ($card\_base/1$)**.** Trata-se do caso simples, em que a cardinalidade não é composta. São casos base um número inteiro ou a constante $many$.

$$\begin{aligned} &card\_base(many) \\ &\quad card\_base(I) \leftarrow card\_base\_int(I). \end{aligned}$$

—

Definição formal 6.24 ($card\_base\_int/1$)**.** Trata-se do caso simples em que a cardinalidade é um número inteiro positivo com zero incluído.

$$\begin{aligned} card\_base\_int(I) &\leftarrow integer(I), \\ &\quad\ I >= 0 \end{aligned}$$

em que $integer/1$ sucede para todo A se, e somente se, A é um número inteiro. —

Teorema 6.1 (Caso inválido de restrição de cardinalidade)**.** Não é uma restrição de cardinalidade válida o caso $many..\alpha$ se α é um número inteiro. ■

Demonstração. O caso $many..\alpha$ para α um número inteiro não é apresentada na definição de $card/1$ (Definição formal 6.22). □

Como predicados auxiliares, define-se os seguintes.

Definição formal 6.25 ($card\_greater\_or\_equals/2$)**.** $card\_greater\_or\_equals/2$ sucede para $card\_greater\_or\_equals(\alpha, \beta)$ se α restrige ao menos a cardinalidade β, para β uma cardinalidade simples e α uma cardinalidade simples ou composta.

$$\begin{aligned} card\_greater\_or\_equals(C, N) &\leftarrow card\_lower(C, L), \\ &\quad\ card(N..L). \end{aligned}$$

—

Definição formal 6.26 (*card_lower*/2)**.** *card_lower*/2 sucede para $card\_lower(\alpha, \beta)$ para $\alpha \in \mathcal{A}$ e $\beta \in \mathcal{A}$ e se β é a cardinalidade inferior de α. Se α está na forma simples, sucede se α e β forem idênticos.

$$
\begin{aligned}
&card\_lower(many, 0)\\
&card\_lower(L, L) \leftarrow card\_base(L),\\
&\qquad\qquad L \neq many\\
&card\_lower(L..U, L) \leftarrow card(L..U)
\end{aligned}
$$

—

Nota 6.5. A Definição formal 6.26 (*card_lower*/2) caracteriza que `many` é na verdade um sinônimo para `0 .. many`. Isso é coerente com o padrão adotado pela UML (ISO, 2012b, p. 9):

> *If the lower bound is equal to the upper bound, then an alternate notation is to use the string containing just the upper bound. For example, "1" is semantically equivalent to "1..1."*
> *A multiplicity with zero as the lower bound and an unspecified upper bound may use the alternative notation containing a single asterisk "*" instead of "0..*".*

—

Definição formal 6.27 (*card_upper*/2)**.** *card_upper*/2 sucede para $card\_upper(\alpha, \beta)$ para $\alpha \in \mathcal{A}$ e $\beta \in \mathcal{A}$ e se β é a cardinalidade superior de α. Se α está na forma simples, sucede se α e β forem idênticos.

$$
\begin{aligned}
&card\_upper(U, U) \leftarrow card\_base(U).\\
&card\_upper(L..U, U) \leftarrow card(L..U).
\end{aligned}
$$

—

Definição formal 6.28 (*card_check_single*/2)**.** *card_check_single*/2 sucede para $\alpha \in \mathcal{A}$ e $\beta \in \mathbb{N}$ quando β está entre a cardinalidade inferior e superior de α, incluindo as fronteiras:

$$
\begin{aligned}
card_check_single(C, N) \leftarrow\ & C \neq many,\\
& card\_base(C),\\
& card\_check\_single(C..C, N)\\
card\_check\_single(many, N) \leftarrow\ & card\_check\_single(0..many, N)\\
card\_check\_single(L..U, N) \leftarrow\ & card(L..N),\\
& card(N..U)
\end{aligned}
$$

—

Definição formal 6.29 ($card\_includes/2$)**.** $card\_includes/2$ é uma generalização de $card\_check\_single/2$ que sucede para $\alpha \in \mathcal{A}$ e $\beta \in \mathcal{A}$ quando as cardinalidades inferior e superior β estão entre a cardinalidade inferior de α, incluindo as fronteiras:

$$\begin{aligned} card\_includes(A, B) \leftarrow & card\_check\_single(A, B) \\ card\_includes(A, L..R) \leftarrow & card\_check\_single(A, L), \\ & card\_check\_single(A, R) \end{aligned}$$

—

Definição formal 6.30 ($card\_match/2$)**.** $card\_match/2$ sucede para todo α e todo β se $\alpha \in \mathcal{A}$ e $\beta \in \mathcal{A}$ que denotem a mesma cardinalidade, ou seja, se α e β forem idênticos, ou se um entre α e β for escrito na forma simples (e o outro na forma composta).

$$\begin{aligned} card\_match(C, C) \leftarrow & card(C) \\ card\_match(C..C, C) \leftarrow & card\_base(C) \\ card\_match(C, C..C) \leftarrow & card\_base(C) \end{aligned}$$

—

Definição formal 6.31 ($card\_prod\_single/3$)**.** $card\_prod\_single/3$, na forma do predicado $card\_prod\_single(\alpha, \beta, \gamma)$, sucede para todo α, todo β e todo γ para $\alpha, \beta, \gamma \in \mathcal{A}$, em que $card\_base/1$ sucede para α e para β, e que γ é o produto de α e β.

$$\begin{aligned} card\_prod\_single(many, \_, many) \leftarrow & ! \\ card\_prod\_single(\_, many, many) \leftarrow & ! \\ card\_prod\_single(M1, M2, R) \leftarrow & M1 \neq many, \\ & M2 \neq many, \\ & R \; is \; M1 * M2 \end{aligned}$$

Na definição, o produto da multiplicação de qualquer valor por *many* é o próprio *many*. —

Definição formal 6.32 ($card\_prod/3$)**.** Baseado em $card\_prod\_single/3$, $card\_prod/3$, na forma $card\_prod(\alpha, \beta, \gamma)$, sucede para todo α, todo β e todo γ para $\alpha, \beta, \gamma \in \mathcal{A}$, em que $card/1$ sucede para α e para β, e que γ é o produto de α e β.

$$\begin{aligned} card\_prod(Card1, Card2, RL..RU) \leftarrow & card\_lower(Card1, Card1L), \\ & card\_upper(Card1, Card1U), \\ & card\_lower(Card2, Card2L), \\ & card\_upper(Card2, Card2U), \\ & card\_prod\_single(Card1L, Card2L, RL), \\ & card\_prod\_single(Card1U, Card2U, RU) \end{aligned}$$

—

Definição formal 6.33 ($card\_sum\_single/3$)**.** $card\_sum\_single/3$, na forma do predicado $card\_sum\_single(\alpha, \beta, \gamma)$, sucede para todo α, todo β e todo γ para $\alpha, \beta, \gamma \in \mathcal{A}$, em que *card_base/1* sucede para α e para β, e que γ é a soma de α e β.

$$
\begin{aligned}
card\_sum\_single(many, \_, many) &\leftarrow ! \\
card\_sum\_single(\_, many, many) &\leftarrow ! \\
card\_sum\_single(M1, M2, R) &\leftarrow M1 \neq many, \\
&\qquad M2 \neq many, \\
&\qquad R \; is \; M1 + M2
\end{aligned}
$$

Na definição, a exemplo do que ocorre em *card_prod_single/3*, a soma de qualquer valor com *many* é o próprio *many*. —

Definição formal 6.34 ($card\_sum/3$)**.** Baseado em *card_sum_single/3*, $card\_sum/3$, na forma $card\_sum(\alpha, \beta, \gamma)$, sucede para todo α, todo β e todo γ para $\alpha, \beta, \gamma \in \mathcal{A}$, em que *card/1* sucede para α e para β, e que γ é a soma de α e β. As cardinalidades mínimas e máximas são somadas individualmente, conforme se segue:

$$
\begin{aligned}
card\_sum(Card1, Card2, RL..RU) \leftarrow & card\_lower(Card1, Card1L), \\
& card\_upper(Card1, Card1U), \\
& card\_lower(Card2, Card2L), \\
& card\_upper(Card2, Card2U), \\
& card\_sum\_single(Card1L, Card2L, RL), \\
& card\_sum\_single(Card1U, Card2U, RU)
\end{aligned}
$$

—

6.4.3 Definições bases na forma $ngrule/n$

As regras apresentadas nesta seção tratam-se de RNP (subseção 6.3.1), definidas com o termo predicativo $ngrule/n$ (Definição 6.1).

Embora na subseção 6.3.1 apresente um esquema geral para $ngrule/n$ em que n é um número inteiro positivo qualquer, por razões de simplicidade neste texto lança-se mão apenas do esquema $ngrule/1$, em que o único parâmetro de cada cláusula refere-se a uma constante existencialmente quantificada que identificada a regra, de modo que cada regra represente afirmações nomeadas, no sentido de uma sequência de *"named statements"* (LAMPORT, 2012, p. 45). Essas regras podem ser entendidos como lemas intermediários a serem provados (como falsos) e que, em conjunto, denotam a consistência de uma Teoria Ontoprolog (Definição 5.2). Na implementação de Ontoprolog, entretanto, um esquema $ngrule/3$ é usado, na forma, $ngrule(A, B, C)$, em que A é o nome da regra presente nesta seção, B é uma variável existencialmente quantificada que unifica com um ou uma lista de

indivíduos da estrutura $\mathcal{OT}$, e C é uma lista que denota uma mensagem que explica ao usuário o motivo de a regra negativa em tela ter sido disparada.

Cada definição formal desta seção *captura* um caso em que a Teoria Ontoprolog (Definição 5.2) vigente é inconsistente. Por isso, as descrições de cada um deles é sucinta e é apresentada apenas quando entender haver necessidade de ser mais claro do que a própria definição formal.

O Capítulo 8 contém outras definições de $ngrule/n$ baseadas em regras da UFO.

6.4.3.1 Regras para $hypertype/1$

Definição formal 6.35 ($hypertype\_as\_instance$)**.** Esta regra define a relação $hypertype/1$: como *hypertypes* formam a *metateoria superior*, e como os níveis de teóricos são contabilizados a partir da relação de instância, nenhuma entidade que instancia outra pode ser qualificada como *hypertype*, dessa forma:

$$\begin{aligned} ngrule(hypertype\_as\_instance) \leftarrow & hypertype(E), \\ & dio(E, \_) \end{aligned}$$

—

Definição formal 6.36 ($entity\_without\_hypertype$)**.** Esta regra avalia os casos em que há entidades na teoria que não são categorizadas por nenhum *hypertype*.

$$\begin{aligned} ngrule(entity\_without\_hypertype) \leftarrow & entity(E), \\ & \neg hypertype(E), \\ & \neg hypertypeof(E, \_) \end{aligned}$$

Devido a presença desta regra, em Ontoprolog todas as entidades de $\mathcal{C}$ são instâncias diretas ou indiretas de algum *hypertype*, exceto os próprios *hypertypes*. —

6.4.3.2 Regras para $dio/2$

Definição formal 6.37 (Regras para $dio/2$)**.** As seguintes regras definem a relação $dio/2$:

a) irreflexividade e assimetria simples:

$$\begin{aligned} ngrule(dio\_reflexive) \leftarrow & dio(C, C) \\ ngrule(dio\_symmetric) \leftarrow & dio(A, B), \\ & dio(B, A) \end{aligned}$$

b) assimetria transitiva, distribuída sobre $dio/2$ e $deo/2$:

$$\begin{aligned} ngrule(dio\_transitive\_symmetry) \leftarrow & instanceof(A, B), \\ & instanceof(B, A) \end{aligned}$$

c) a hierarquia de meta tipos deve ser respeitada nas instâncias diretas:

$$\begin{aligned} ngrule(dio\_wrong\_metatype \leftarrow & extensionof(T,S), \\ & dio(T,H), \\ & dio(S,SH), \\ & extensionof(SH,H) \end{aligned}$$

d) $dio/2$ apenas pode relacionar entidades de uma mesma categoria de *hypertype raiz*:

$$\begin{aligned} ngrule(dio\_wrong\_hypertype \leftarrow & dio(T,M), \\ & top\_hypertypeof(T,H), \\ & top\_hypertypeof(M,MH), \\ & H \neq MH \end{aligned}$$

—

6.4.3.3 Regras para $deo/2$

Definição formal 6.38 (Regras para $deo/2$)**.** As seguintes regras definem a relação $deo/2$:

a) irreflexividade e assimetria simples:

$$\begin{aligned} ngrule(deo\_reflexive) \leftarrow & deo(C,C) \\ ngrule(deo\_symmetric) \leftarrow & deo(A,B), \\ & deo(B,A) \end{aligned}$$

b) a relação $deo/2$ denota um grafo dirigido acíclico. O caso indesejado é caracterizado quando a relação transitiva $extensionof/2$ é reflexiva:

$$ngrule(deo\_circular) \leftarrow extensionof(A,A)$$

c) $deo/2$ apenas pode relacionar entidades de uma mesma categoria de *hypertype raiz*:

$$\begin{aligned} ngrule(deo\_wrong\_hypertype \leftarrow & deo(T,S), \\ & top\_hypertypeof(T,H), \\ & top\_hypertypeof(T,SH), \\ & T \neq SH \end{aligned}$$

—

6.4.3.4 Regras para $deo/2$ e $dio/2$

Definição formal 6.39 ($dio\_or\_deo\_circular$)**.** As relações $dio/2$ e $deo/2$, quando unidas, devem formam um único grafo direcionado acíclico. Dessa forma, define-se a relação $dio\_or\_deo/2$:

$$dio\_or\_deo(A, B) \leftarrow dio(A, B) \vee deo(A, B)$$

Define-se também o fecho transitivo de $dio\_or\_deo/2$, caso em que a regra $dio\_or\_deo\_transitive/2$ é transcrita para Prolog com uso de *memoization* (Nota 6.1):

$$\begin{aligned} dio\_or\_deo\_irreflexive(T, M) &\leftarrow dio\_or\_deo(T, M) \\ dio\_or\_deo\_irreflexive(T, S) &\leftarrow dio\_or\_deo(T, M), \\ &\qquad dio\_or\_deo\_irreflexive(M, S) \end{aligned}$$

Por fim, com base nessas definições, apresenta-se o caso de $ngrule/1$:

$$ngrule(dio\_or\_deo\_circular) \leftarrow dio\_or\_deo\_irreflexive(A, A)$$

—

6.4.3.5 Regras para $dd/1$

Definição formal 6.40 (Regras para $dd/1$)**.** $dd/2$ impõe restrições sobre as relações $dio/2$ e $deo/2$, de modo que nenhuma entidade pode instanciar ou estender simultaneamente dois tipos disjuntos. As seguintes regras definem a relação $dd/2$:

a) seja I a lista de entidades instanciadas por E. Nenhum par de entidades $\langle i_1, i_2 \rangle \in I$ pode ser disjunto:

$$\begin{aligned} ngrule(dio\_disjoint\_types) \leftarrow &setof(IT, dio(E, IT), I), \\ &disjoint\_types(I) \end{aligned}$$

b) seja I a lista de entidades estendidas por E. Nenhum par de entidades $\langle i_1, i_2 \rangle \in I$ pode ser disjunto:

$$\begin{aligned} ngrule(deo\_disjoint\_types) \leftarrow &setof(IT, deo(E, IT), I), \\ &disjoint\_types(I) \end{aligned}$$

—

6.4.3.6 Regras para $complete/2$

Definição formal 6.41 (*deo_complete_type*)**.** Verifica a relação "subsunção completa de", em que $\{c_1, \ldots, c_n\}$ são taxativamente todos os conceitos subordinados (ou que estendem) c_t, ou seja, não é o caso que exista um c_m que participa da relação $deo(c_m, c_t)$ e que $c_m \notin \{c_1, \ldots, c_n\}$ se ocorre $dcomplete(c_t, \{c_1, \ldots, c_n\})$:

$$\begin{aligned} ngrule(deo\_complete\_type) \leftarrow & deo(T, S), \\ & complete(S, \_), \\ & \neg(complete(S, L), T \in L) \end{aligned}$$

—

6.4.3.7 Regras para $pt/2$

Definição formal 6.42 (Regras para $pt/2$)**.** As seguintes regras definem a relação $pt/2$:

a) irreflexividade:

$$ngrule(pt\_reflexive) \leftarrow pt(T, T)$$

b) o domínio da relação deve ser uma instância da categoria de *type*:

$$\begin{aligned} ngrule(pt\_wrong\_domain) \leftarrow & pt(T, \_), \\ & \neg hypertypesof(T, type) \end{aligned}$$

c) o contra-domínio da relação deve ser uma instância da categoria de *type*:

$$\begin{aligned} ngrule(pt\_wrong\_codomain) \leftarrow & pt(\_, T), \\ & \neg hypertypesof(T, type) \end{aligned}$$

—

6.4.3.8 Regras para $dpo/2$

Definição formal 6.43 (Regras para $dpo/2$)**.** As seguintes regras regem a relação $dpo/2$:

a) o domínio da relação deve ser uma instância da categoria de *property*:

$$\begin{aligned} ngrule(dpo\_wrong\_domain) \leftarrow & pt(P, \_), \\ & \neg hypertypesof(P, property) \end{aligned}$$

b) o contra-domínio da relação deve ser uma instância da categoria de *type* ou ser um *hypertype*:

$$\begin{aligned} ngrule(dpo\_wrong\_codomain) \leftarrow & dpo(\_, T), \\ & \neg hypertype(T), \\ & \neg hypertypesof(T, type) \end{aligned}$$

—

6.4.3.9 Regras para $dpv/2$

Definição formal 6.44 ($dpv\_without\_dpo\_on\_type$)**.** A relação $dpv/2$, na forma $dpv(at(P,T),V)$, que denota o valor $V \in \mathcal{G}$ de determinada propriedade P em uma instância T, só deve ser atribuída a instâncias T de entidades M que participem da relação $dpo(P,M)$:

$$ngrule(dpv\_without\_dpo\_on\_type) \leftarrow dpv(PatT,\_), \neg(propertyon(P,M), instanceof(T,M))$$

Dessa forma, a relação $dpv/2$ denota valores de propriedades em instâncias de entidades que possuem a relação $dpo/2$. —

Definição formal 6.45 ($dpv\_dpv\_more\_than\_one\_value$)**.** A relação $dpv/2$ só pode ocorrer uma única vez para cada $at(P,T)$ em forma $dpv(at(P,T),\alpha)$, para qualquer α, de modo que apenas um único valor esteja presente para cada propriedade.

$$ngrule(dpv\_more\_than\_one\_value) \leftarrow setof(V, dpv(at(P,T),V), Vs), length(Vs,L), L > 1$$

—

6.4.3.10 Regras para $dso/2$, $dro/2$ e $deo/2$ sobre relações

Definição formal 6.46 (Regras gerais para $drefine/3$)**.** As seguintes regras regem a relação $drefine/3$:

a) irreflexividade:

$$ngrule(refinedrel\_reflexive) \leftarrow drefine(\_,T,T)$$

b) assimetria simples:

$$ngrule(refinedrel\_symmetric) \leftarrow drefine(R,T_1,T_2), drefine(R,T_2,T_1)$$

—

Definição formal 6.47 ($refinedrel\_wrong\_cc$)**.** As restrições de cardinalidade das relações refinadas devem incluir a cardinalidade das relações refinadoras, pois relação específica pode ser mais restrita do que a mais geral, mas não o contrário:

$$ngrule(refinedrel\_wrong\_cc) \leftarrow drefine(Rel\_T, Ref, Refined), refinedrel\_wrong\_cc\_get\_cc(Ref, Ref\_CC), refinedrel\_wrong\_cc\_get\_cc(Refined, Refined\_CC), \neg card\_includes(Refined\_CC, Ref\_CC)$$

Sendo que $refinedrel\_wrong\_cc\_get\_cc/2$ é apresentada na Definição formal 6.48. —

Definição formal 6.48 ($refinedrel\_wrong\_cc\_get\_cc/2$)**.** Esta relação sucede para o caso em que são definidas as propriedades *domain_cc* e *codomain_cc* quando igualmente estão presentes as relações *domain* e *codomain*, respectivamentem em uma entidade $T \in \mathcal{C}$. Dessa forma, usado no contexto de $refinedrel\_wrong\_cc$, esta relação é avaliada somente para os casos em que a relação $drefine/3$ é definida.

$$\begin{aligned} refinedrel\_wrong\_cc\_get\_cc(at(domainT), CC) \leftarrow \\ property\_value(at(domain\_cc, T), CC) \\ refinedrel\_wrong\_cc\_get\_cc(at(codomain, T), CC) \leftarrow \\ property\_value(at(codomain\_cc, T), CC) \end{aligned}$$

—

Definição formal 6.49 ($redefining\_no\_subtype$)**.** As entidades participantes como entidades redefinadoras nas relações $dro/2$ devem ser um subtipo da entidade redefinida, mas não podem ser as próprias entidades refinadas.

$$\begin{aligned} &ngrule(redefining\_no\_subtype) \leftarrow \\ &\quad dro(at(P\_Ref, Ref), at(P\_Refined, Refined)), \\ &\quad property\_value(at(P\_Ref, Ref), Type\_Ref)), \\ &\quad property\_value(at(P\_Refined, Refined), Type\_Refined)), \\ &\quad \neg extensionof(Type\_Ref, Type\_Refined) \end{aligned}$$

—

Definição formal 6.50 ($subsetting\_or\_extending\_no\_subtype$)**.** Esta regra define comportamento semelhante ao apresentado por $redefining\_no\_subtype$. Porém, utiliza-se na definição deste regra a versão reflexiva $extensionof\_reflexive/2$ da relação de subsunção.

$$\begin{aligned} &ngrule(subsetting\_or\_extending\_no\_subtype) \leftarrow \\ &\quad drefine(RT, at(P\_Ref, Ref), at(P\_Refined, Refined)), \\ &\quad (RT = dso \vee RT = deo), \\ &\quad property\_value(at(P\_Ref, Ref), Type\_Ref)), \\ &\quad property\_value(at(P\_Refined, Refined), Type\_Refined)), \\ &\quad \neg extensionof\_reflexive(Type\_Ref, Type\_Refined) \end{aligned}$$

—

Definição formal 6.51 ($refinedrel\_wrong\_intances$)**.** Esta regra avalia uma parte central da semântica natural da relação $drefine/3$, em que as instâncias das relações

redefinadoras devem incluir as instâncias das relações redefinidas.

$$
\begin{aligned}
&ngrule(refinedrel\_wrong\_intances) \leftarrow \\
&\quad drefine(\_, at(P\_Ring, Rel\_Ring), at(P\_Red, Rel\_Red)), \\
&\quad rwiiv(at(P\_Ring, Rel\_Ring), \_, I\_Type\_Ring), \\
&\quad \neg(rwiiv(at(P\_Red, Rel\_Red), \_, I\_Type\_Red), \\
&\quad extensionof\_reflexive(I\_Type\_Ring, I\_Type\_Red))
\end{aligned}
$$

Em que $rwiiv/3$ sucede para as instâncias Rel_P das entidades Rel_Type que possuem valores atribuídos às propriedades P. A definição é:

$$
\begin{aligned}
&rwiiv(at(P, Rel\_Type), Rel\_P, I\_V)) \leftarrow \\
&\qquad\qquad instanceof(Rel\_P, Rel\_Type), \\
&\qquad\qquad property\_value(at(P, Rel\_P, I\_V))
\end{aligned}
$$

—

6.4.3.11 Regras para $dio/2$ sobre relações

As regras desta seção avaliam as instâncias de conceitos que denotam relações. Trata-se da semântica das propriedades *domain*, *domain_cc*, *codomain* e *codomain_cc*, no sentido de restrições impostas aos modelos da teoria.

Definição formal 6.52 (*dio_relation_wrong_domain_or_codomain_instance*)**.** Em entidades que denotam relações, os valores das propriedades *domain* e *codomain* devem conter instâncias dos tipos indicados nas propriedades da relação instanciada. Essa característica é atendida por esta definição:

$$
\begin{aligned}
&ngrule(dio\_relation\_wrong\_domain\_or\_codomain\_instance) \leftarrow \\
&\quad drel(Rel\_I, I\_Domain, \_, I\_Codomain, \_), \\
&\quad \neg(dio(Rel\_I, H), hypertype(H)), \\
&\quad \neg(drel(Rel\_T, T\_Domain, \_, T\_Codomain, \_), \\
&\quad\quad instanceof(Rel\_I, Rel\_T), \\
&\quad\quad instanceof(I\_Domain, T\_Domain), \\
&\quad\quad instanceof(I\_Codomain, T\_Codomain))
\end{aligned}
$$

—

As regras seguintes avaliam as restrições impostas pelas propriedades *domain_cc* e *codomain_cc*. Essencialmente, essas propriedades denotam restrições sobre a quantidade de instâncias permitidas a determinada entidade.

Definição formal 6.53 (*dio_relation_wrong_domain_cc*)**.** Esta regra verifica existencialmente se a entidade que denota instância $I\_Codomain$ participa de relações instâncias do tipo Rel_M dentro do limite de M_Domain_CC ocorrências.

$$
\begin{aligned}
&ngrule(dio\_relation\_wrong\_domain\_cc) \leftarrow \\
&\quad property\_value(at(domain\_cc, Rel\_M), M\_Domain\_CC), \\
&\quad dio(Rel\_I, Rel\_M), \\
&\quad drel(Rel\_I, \_, \_, I\_Codomain, \_), \\
&\quad setof(I\_Domain, \\
&\qquad Rel\_I2\hat{}\ \\
&\qquad (dio(Rel\_I2, Rel\_M), \\
&\qquad\quad property\_value(at(domain, Rel\_I2), I\_Domain), \\
&\qquad\quad property\_value(at(codomain, Rel\_I2), I\_Codomain)), \\
&\qquad I\_Domain\_List), \\
&\quad length(I\_Domain\_List, I\_Domain\_L\_Length), \\
&\quad \neg card\_includes(M\_Domain\_CC, I\_Domain\_L\_Length)
\end{aligned}
$$

—

Definição formal 6.54 (*dio_relation_wrong_codomain_cc*)**.** Similar à definição de *dio_relation_wrong_domain_cc* (Definição formal 6.53), esta regra verifica existencialmente se a entidade que denota instância I_Domain participa de relações instâncias do tipo Rel_M dentro do limite de M_Cod_CC ocorrências.

$$
\begin{aligned}
&ngrule(dio\_relation\_wrong\_codomain\_cc) \leftarrow \\
&\quad property\_value(at(codomain\_cc, Rel\_M), M\_Cod\_CC), \\
&\quad dio(Rel\_I, Rel\_M), \\
&\quad drel(Rel\_I, I\_Domain, \_, \_, \_), \\
&\quad setof(I\_Cod, \\
&\qquad Rel\_I2\hat{}\ \\
&\qquad (dio(Rel\_I2, Rel\_M), \\
&\qquad\quad property\_value(at(domain, Rel\_I2), I\_Domain), \\
&\qquad\quad property\_value(at(codomain, Rel\_I2), I\_Cod)), \\
&\qquad I\_Cod\_List), \\
&\quad length(I\_Cod\_List, I\_Cod\_L\_Length), \\
&\quad \neg card\_includes(M\_Cod\_CC, I\_Cod\_L\_Length)
\end{aligned}
$$

—

As duas regras seguintes verificam os casos de cardinalidade obrigatória, ou seja, em que a cardinalidade é maior do que 1. Nesses casos, todos as entidades I que instanciam determinado tipo T, ou de um tipo T_n mais específico do que T, que participa como domínio ou codomínio de alguma outra entidade que denota uma relação, devem obrigatoriamente estar relacionadas com instâncias daquela relação como domínio ou codomínio, conforme a especificação do tipo da relação instanciada. Devido ao uso de *rel*/5 (Definição formal 6.13), as regras consideram hierarquia de tipos ascendentes.

Definição formal 6.55 (*particular_without_mandatory_relation_as_domain*)**.**

$$
\begin{aligned}
&ngrule(particular\_without\_mandatory\_relation\_as\_domain) \leftarrow \\
&\quad drel(Rel\_M, M\_Domain, \_, \_, M\_Codomain\_CC), \\
&\quad \neg dmo(abstract, Rel\_M), \\
&\quad card\_greater\_or\_equals(M\_Codomain\_CC, 1), \\
&\quad instanceof(I, M\_Domain), \\
&\quad \neg dmo(abstract, I), \\
&\quad \neg(rel(Rel\_I, I, \_, \_, \_), instanceof(Rel\_I, Rel\_M))
\end{aligned}
$$

—

Definição formal 6.56 (*particular_without_mandatory_relation_as_codomain*)**.**

$$
\begin{aligned}
&ngrule(particular\_without\_mandatory\_relation\_as\_codomain) \leftarrow \\
&\quad drel(Rel\_M, \_, M\_Domain\_CC, M\_Codomain, \_), \\
&\quad \neg dmo(abstract, Rel\_M), \\
&\quad card\_greater\_or\_equals(M\_Domain\_CC, 1), \\
&\quad instanceof(I, M\_Codomain), \\
&\quad \neg dmo(abstract, I), \\
&\quad \neg(rel(Rel\_I, \_, \_, I, \_), dio(Rel\_I, Rel\_M))
\end{aligned}
$$

—

6.4.4 Restrições a partir de meta-propriedades

O conjunto $\mathcal{G}$, domínio das meta-propriedades participantes de *dmo*/2, é um conjunto aberto de entidades lógicas (Definição 5.5), definido de forma intencional, de modo que qualquer constructo válido na linguagem subjacente é uma meta-propriedade válida participante da relação $dmo(l \in \mathcal{G}, c \in \mathcal{C})$. Devido a essa característica aberta, cria-se em Ontoprolog um mecanismo geral de definição de regras sobre as meta-propriedades que pode ser usado para estender regras de consistência dos modelos das teorias especificadas na linguagem. Esse mecanismo é apresentado pela Definição formal 6.57.

Definição formal 6.57 (*ncheck_for_meta_not_passing*)**.** Toda relação $dmo(M, E)$ – em que $M \in \mathcal{G}$ é uma meta-propriedade e $E \in \mathcal{C}$ é uma entidade – é uma relação consistente com as regras de Ontoprolog se não for o caso que suceda alguma regra escrita na forma `nrulemeta` M `for` E `message_`.

$$ngrule(ncheck\_for\_meta\_not\_passing) \leftarrow dmo(M, E), \texttt{nrulemeta}\ M\ \texttt{for}\ E\ \texttt{message}\ \_$$

A sintaxe padrão do constructo `nrulemeta` é dado pelo Código 4[7]. Esse constructo é definido diretamente como constructo semântico, de modo que não há uma função de tradução sintática, nos moldes das apresentadas na seção 7.3.

Código 4 – Definição EBNF da sintaxe do constructo `nrulemeta`

```
/* NEGATIVE RULES FOR META PROPERTIES */

NRULEMETA ::= 'nrulemeta' META_PROPERTY 'for' ATOM 'message' ANY_PROLOG_TERM
```

Pelo fato de se tratar de uma $ngrule/n$, a Definição formal 6.57 é incorporado em Ontoprolog e passa a reger os modelos das teorias especificadas na linguagem. Trata-se da integração entre o sistema de Ontoprolog com o universo de outros sistemas circunscritos na mesma lógica subjacente às teorias.

Com base nessa definição, um arquiteto poderia definir regras como a apresentada no Código 5, que estabelece a condição a ser verificada quando determinada entidade $Final\_T$ receber a meta-propriedade $final$. No caso, entidades com essa meta-propriedades não pode possuir subsunções. Nesse caso, o Código 5 contém uma definição que tenta provar que existe ao menos um $Final\_T$ que participe à direita numa relação $deo/2$.

Código 5 – Exemplo de uso de `nrulemeta`

```
% ncheck for meta final
nrulemeta final
      for Final_T
  message ['"', Entity, '" should not extend the final entity "', Final_T, '"'] :-
        deo(Entity, Final_T).
```

A seção 7.6 descreve como as meta-propriedades podem ser distribuídas sobre as relações de instanciação e subsunção como procedimento final de tradução das especificações em teorias Ontoprolog. O mecanismo de distribuição de meta-propriedades e o recurso de verificação de meta-propriedades é usado largamente nas definições das regras da UFO, conforme detalhado no Capítulo 8. A título de exemplo, subseção 8.4.4 discute a aplicação desse recurso para definição de restrições sobre Espaços Conceituais (Conceptual Space). O Código 32 do Apêndice A e o Código 34 do Apêndice C consistem nas definições das regras na forma `nrulemeta` que regem as teorias Ontoprolog gerais, e as específicas da UFO, respectivamente.

[7] As definições EBNF são descritas na seção 7.1 e na seção 7.2.

6.5 Fechamento

A semântica formal das "relações semânticas" apresentadas na seção 6.2, bem como das regras descritas em todo capítulo, é aquela dada pela lógica subjacente na qual estão escritas, ou seja, as chamadas "relações semânticas" são teorias cujos modelos são aqueles de suas lógicas subjacentes. No caso específico da definição da semântica apresentada neste capítulo no âmbito da Programação em Lógica Clássica, a semântica denotacional pode ser descrita pela classe de interpretações de Herbrand, conforme subseção 3.1.2. Dessa forma, conforme abordado na seção 3.3 e na seção 3.6, a semântica apresentada neste capítulo é um exemplo de *semantic embedding*, conforme proposto por Armstrong, Virding e Williams (1992).

Ontoprolog não foi projetado para que especificações sobre ontologias fossem realizadas diretamente nessas "relações semânticas" apresentadas na seção 6.2. Essas relações devem funcionar como base de fatos para que consultas definidas (Definição 3.18) sobre essa base possam ser realizadas. As entradas a essa base de fatos devem ser dadas por meio de traduções de sentenças em alto nível, que consiste no que é chamado "sintaxe de Ontoprolog", conforme descrito no Capítulo 7.

O arcabouço de regras apresentado neste capítulo é a referência para a implementação de Ontoprolog em um Prolog. Outros tipos de semântica, como aquelas baseadas em banco de dados dedutivos, ou mesmo axiomas escritos em alguma *description logic* podem ser implementados, desde que compatíveis com as regras bases apresentadas.

As regras e definições desenvolvidos são as mais importantes e que visam nortear as regras básicas a serem observadas em qualquer teoria escrita em Ontoprolog, mas não esgotam outros possivelmente necessários em uma implementação concreta. Além disso, a implementação concreta das regras apresentadas nesta seção, especialmente as RNP (subseção 6.4.3), podem lançar mão de mensagens e de explicações que ajudem o arquiteto e encontrar justificativas quando uma determinada teoria não seja considerada consistente com as regras bases. A implementação de referência de Ontoprolog é descrita no Capítulo 10.

7 Sintaxe base de Ontoprolog

Neste capítulo apresenta-se a sintaxe base da linguagem de Ontoprolog. A linguagem é proposta como declarativa textual, que consiste em frases concisas, escritas na voz ativa, e de açúcares sintáticos (subseção 1.4.4.2). A sintaxe é dita *base* porque ela pode ser estendida de modo a contemplar diferentes formas de expressão de entidades, conforme é apresentado no Capítulo 8.

A sintaxe proposta para Ontoprolog visa servir de "língua franca" entre arquitetos da informação no processo pragmático de formalização de seus discursos sobre ontologias. A linguagem é proposta para servir de instrumento principal de anotação de discursos em trabalhos colaborativos de diferentes arquitetos, e não apenas como linguagem intermediária para verificação de regras de modelos, embora ela também sirva a esse propósito.

Nesse contexto, o objetivo deste capítulo é apresentar os principais constructos sintáticos da linguagem, bem como descrever como esses constructos devem ser traduzidos em relações semânticas escritas na Linguagem alvo (seção 6.1), de modo que essas traduções possam servir de referências para a construção de consultas definidas sobre o banco de dados de fatos que denotam as anotações sintáticas. Essas relações semânticas denotam uma estrutura $\mathcal{OT}$ (seção 6.2), que é a base da semântica de Ontoprolog, conforme apresentada no Capítulo 6. A semântica de tradução de Ontoprolog é chamada de *filtro*, conforme sugere Gupta (1998, p. 152) (subseção 3.3.2). Esse filtro é responsável por mapear indutivamente cada estrutura sintática de Ontoprolog em uma ou mais relações semânticas da estrutura $\mathcal{OT}$.

Embora neste capítulo apresente-se toda sintaxe base e todas suas interpretações válidas em termos das relações semânticas, não é objetivo do capítulo ser exaustivo em relação ao mecanismo lógico da tradução propriamente dito, de modo que o que se apresenta é uma visão intuitiva e semi-formal desse processo. No entanto, a definição precisa do mecanismo de tradução de Ontoprolog é formalizado em Prolog e está disponível no código-fonte do projeto, conforme descrito pelo Capítulo 10.

E capítulo é organizado do seguinte modo: a seção 7.1 consiste em uma breve apresentação da notação EBNF e de diagramas *railroad*, usados na formalização das sintaxes neste trabalho. As definições EBNF, que representam a definição propriamente dita da sintaxe da linguagem, são apresentadas na seção 7.2. Na seção 7.3 apresenta-se a *semântica de tradução* de Ontoprolog, com a introdução da função "filtro" de tradução da sintaxe nas relações semânticas, bem como definições preliminares e algoritmos envolvidos no processo de tradução. Na seção 7.4 concentram-se as definições da tradução propriamente dita, realizada por indução na estrutura da sintaxe. A seção 7.5 complementa a seção

anterior com uma apresentação de um importante açúcar sintático, o responsável pela declaração de relações binárias. Na seção 7.6 aborda-se os processos finais de tradução, que são apresentados em Prolog no Apêndice F. A seção 7.7, por sua vez, apresenta um exemplo de especificação em Ontoprolog-Base que exercita aspectos sintáticos e semânticos desenvolvidos neste e nos capítulos anteriores. Por fim, a seção 7.8 contém um breve fechamento do capítulo.

7.1 Notação EBNF e diagramas *railroad*

A *Backus-Naur Form* (BNF) é uma meta-sintaxe usada para descrever gramáticas livres de contexto proposta na década de 1960 por Backus et al. (1960). Devido à grande aplicabilidade do modelo, diversas sugestões e melhorias foram propostas nos anos seguintes, inclusive pelos próprios autores (BACKUS et al., 1963). Porém, foi a proposta de Wirth (1977) que a *International Organization for Standardization* (ISO) tomou como base para formalizar a norma ISO/IEC 14977:1996 (ISO, 1996, p. vii), que padroniza a EBNF, chamada de *Extended BNF*. Embora trate-se de uma sintaxe normatizada pela ISO, a EBNF ainda recebeu outras extensões e adaptações sintáticas no decorrer do tempo, como é o caso da versão utilizada neste trabalho, que segue o padrão proposto pela W3C para documentos XML (W3C, 2008), e que não foi reincorporada às normas ISO.

O padrão EBNF pode ser representado concretamente por um tipo de diagrama sintático chamado diagramas *railroad*. Os diagramas *railroad* aparecem originalmente como anexos ao manual da linguagem de programação Pascal, produzido como relatório técnico por Wirth (1973), o mesmo autor da proposta da EBNF aceita pela ISO anos mais tarde[1].

As definições sintáticas realizadas neste trabalho – localizadas especialmente na seção 7.2 e na subseção 8.2.1 – são apresentadas tanto na notação EBNF da W3C quanto nos diagramas *railroad*, sendo estes construídos com o software *Railroad Diagram Generator*, escrito por *Gunther Rademacher* e disponível para acesso online em <http://bottlecaps.de/rr/ui>.

7.2 Definições EBNF

A sintaxe base de Ontoprolog é descrita por meio de uma gramática livre de contexto expressa em *Extended Backus-Naur Form* (EBNF) e por diagramas sintáticos *railroad*, conforme descrito pela seção 7.1.

Os construtores ou símbolos terminais da gramática de Ontoprolog são apresentados em inglês, mas suas descrições no corpo deste documento estão em português. Dessa forma,

[1] ISO, 1996, loc. cit.

mantém-se uma tradição na área de Ciência da Computação em se construir linguagens com palavras da linha inglesa.

O Código 6 contém a definição em EBNF de Ontoprolog. Os textos entre `/* . . . */` são comentários e são usados para agrupar as categorias sintáticas da linguagem. A subseção 7.2.1 apresenta os diagramas *railroad* equivalentes à definição EBNF em tela.

Código 6 – Definição EBNF da sintaxe base de Ontoprolog

```
ONTOLOG_SPECIFICATION ::= SENTENCE+

SENTENCE ::=
      ( DISJOINT
        | INSTANCE
        | PARTIAL_SUBSUMPTION
        | COMPLETE_SUBSUMPTION
        | PROPERTY_ASSOCIATION
        | PROPERTY_ASSIGNMENT
        | SUBSETS
        | REDEFINES
        | POWER_TYPE
        | META
      ) '.'

/* MONADIC TYPES */

DISJOINT ::=
      'disjoint' '[' ATOM (',' ATOM)* ']'

INSTANCE ::=
      ( ATOM | 'disjoint' '[' ATOM (',' ATOM )* ']' )
      '::' ( INSTANTIABLE_ENTITY | '[' ATOM (',' ATOM )* ']' )

PARTIAL_SUBSUMPTION ::=
      ( ATOM | 'disjoint'? '[' ATOM ( ',' ATOM )* ']' )
      ( '::' INSTANTIABLE_ENTITY )?
      ( 'extends' | 'extend'
      ) ( ATOM
          | '[' ATOM ( ',' ATOM )* ']' )

COMPLETE_SUBSUMPTION ::=
      ( ATOM
        | ('disjoint'? '[' ATOM ( ',' ATOM)* ']' ) )
      ( '::' INSTANTIABLE_ENTITY )?
      'cover' ATOM

/* PROPERTIES */

PROPERTY_ASSOCIATION ::=
      'property' ( ATOM | '[' ATOM (',' ATOM )* ']' )
      'on'
      ( ENTITY_WITH_PROPERTY | '[' ENTITY_WITH_PROPERTY (',' ENTITY_WITH_PROPERTY
    )* ']' )

PROPERTY_ASSIGNMENT ::=
      PROPERTY_TYPE 'at' ENTITY_IN_RELATION ':=' VALUE

VALUE ::= ANY_PROLOG_TERM

/* SUBSETS */

SUBSETS ::= PROPERTY_TYPE 'at' ENTITY_IN_RELATION 'subsets' PROPERTY_TYPE 'at'
    ENTITY_IN_RELATION

/* REDEFINES */

REDEFINES ::= PROPERTY_TYPE 'at' ENTITY_IN_RELATION 'redefines' PROPERTY_TYPE 'at'
    ENTITY_IN_RELATION

/* POWER TYPE */

POWER_TYPE ::=
      'powertype' ( ENTITY_PT | '[' ENTITY_PT (',' ENTITY_PT )* ']' )
      'classifying' ( ATOM | '[' ATOM ( ',' ATOM )* ']')

/* META PROPERTIES */
```

```
META ::=
      'meta' ( META_PROPERTY | '[' META_PROPERTY (',' META_PROPERTY)* ']' )
      'on'
      ( ( ATOM 'at' )?
        ( ( 'extensions'
          | 'transitive'? 'direct'? 'instances'
          ) 'of'
      )?
    )?
      ( ENTITY_WITH_META | '[' ENTITY_WITH_META (',' ENTITY_WITH_META )* ']' )

META_PROPERTY ::= ANY_PROLOG_TERM

/* ENTITIES */

INSTANTIABLE_ENTITY ::= ATOM

PROPERTY_TYPE ::= ATOM

ENTITY_IN_RELATION ::= ATOM

ENTITY_WITH_PROPERTY ::= ATOM

ENTITY_WITH_META ::= ENTITY_WITH_PROPERTY

ENTITY_PT ::=
      ENTITY_WITH_PROPERTY
      | ATOM 'at' ATOM

/* IMPLEMENTATION DEFINITION LEVEL */

ATOM ::=
      [a-z][a-zA-Z0-9]*
      | "'" [^']* "'"
```

7.2.1 Representação em diagramas *Railroad*

A Figura 29 contém a representação visual em diagramas *railroad* derivada da definição sintática de Ontoprolog apresentada no Código 6. As duas representações são equivalentes, embora a representação diagramática possa ser mais intuitiva em algumas situações. Por isso ela é utilizada no restante deste capítulo, em detrimento à versão clássica em EBNF.

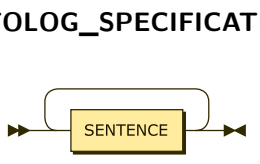
Na figura há três tipos de quadros:

a) *retângulos com cantos circulares*: tratam-se de *construtores terminais rígidos*, *palavras* ou *símbolos reservados*, que essencialmente são símbolos terminais justapostos definidos como os construtores léxicos básicos da linguagem;

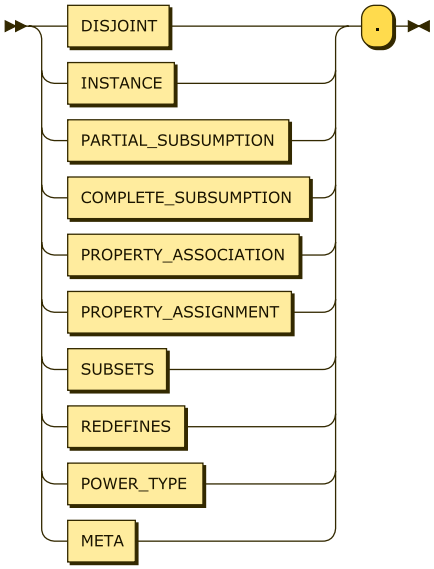
b) *retângulos com cantos retos*: referência a definições não terminais;

c) *hexágonos*: expressões regulares sobre conjuntos de caracteres.

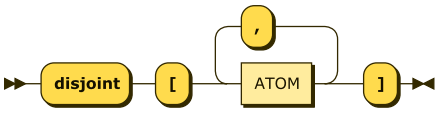
Figura 29 – Diagramas *railroad* da sintaxe de Ontoprolog



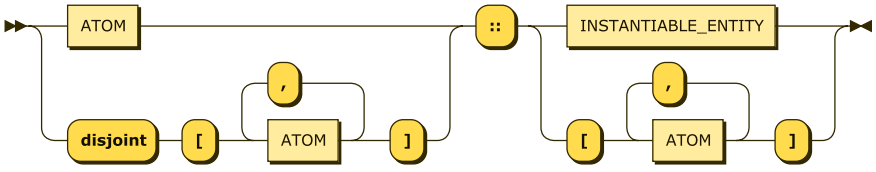
SENTENCE:



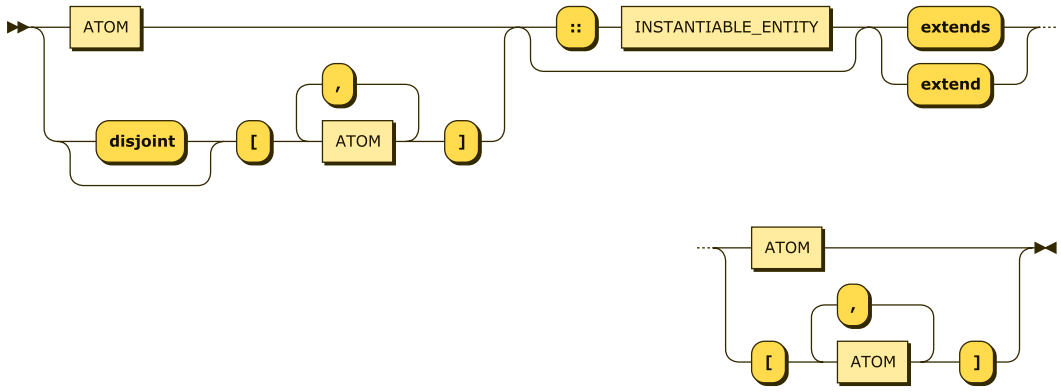
DISJOINT:



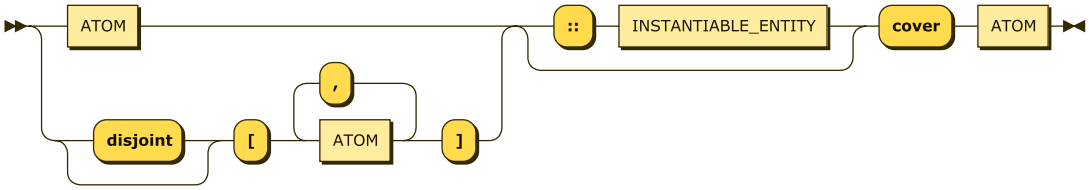
INSTANCE:



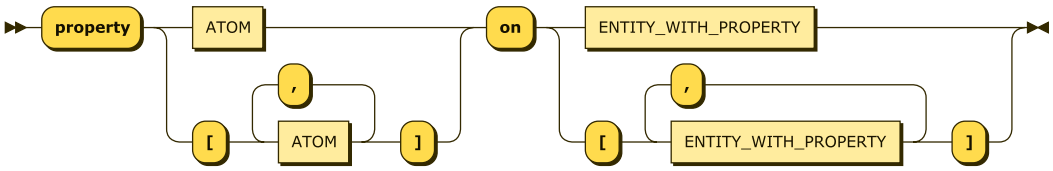
PARTIAL_SUBSUMPTION:



COMPLETE_SUBSUMPTION:



PROPERTY_ASSOCIATION:



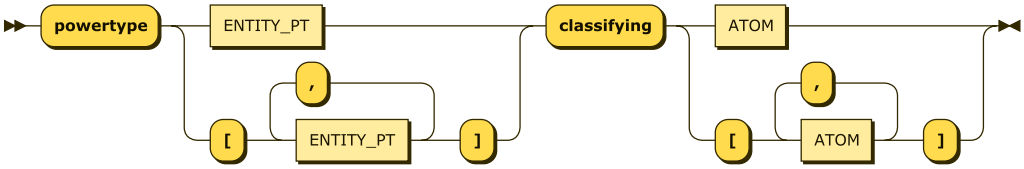
PROPERTY_ASSIGNMENT:

VALUE:

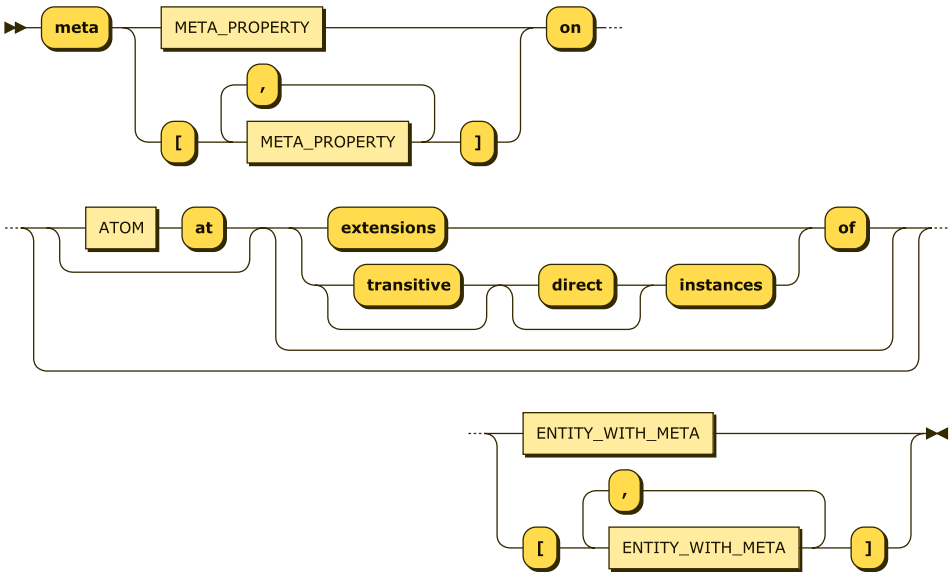
SUBSETS:

REDEFINES:

POWER_TYPE:



META:



META_PROPERTY:

INSTANTIABLE_ENTITY:

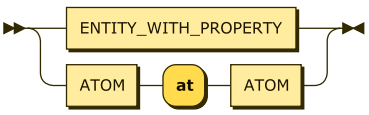
PROPERTY_TYPE:

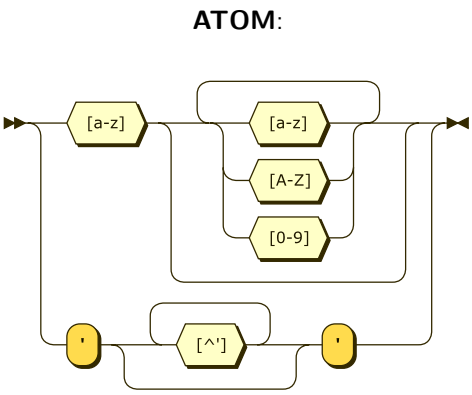
ENTITY_IN_RELATION:

ENTITY_WITH_PROPERTY:

ENTITY_WITH_META:

ENTITY_PT:





7.3 Filtro de tradução da sintaxe

Nesta seção descreve-se a tradução sintática de Ontoprolog nos predicados e relações lógicas escritos na *linguagem alvo* (seção 6.1). A abordagem adotada nesta seção segue as diretrizes daquela adotada por Decker (1998) (subseção 3.3.2). Trata-se, essencialmente, de uma abordagem orientada pela sintaxe, em que a semântica é definida por indução sobre a estrutura da sintaxe da linguagem. Ou seja, por meio de regras de produção, atribui-se semântica a cada possibilidade de combinação dos constructos sintáticos. A tradução apresentada nesta seção é um tipo de *semântica denotacional*, criada com base em Nielson e Nielson (2007, p. 91), no sentido de que "há uma cláusula semântica para cada categoria sintática básica" e que "para cada método de construção de elementos compostos (na categoria sintática) há uma cláusula semântica definida em termos da função semântica aplicada aos constituintes imediatos dos elementos compostos." Também é possível entender a tradução apresentada como uma "semântica por tradução", no sentido que a semântica das sentenças sintáticas de Ontoprolog é escrita em sentenças do fragmento de fórmulas de Primeira Ordem utilizado pelo Prolog, na forma de predicados e relações conforme expostos no Capítulo 6.

Embora diferentes abordagens para se atribuir esta etapa da semântica de Ontoprolog tenham sido consideradas, essa abordagem foi escolhida devido a sua simplicidade, clareza e alinhamento com a proposta geral deste trabalho. Conforme mencionado na introdução deste capítulo, esta seção é exaustiva na definição das interpretações semânticas válidas para cada sentença sintática. Porém, ela não descreve formalmente todo o mecanismo a ser adotado para a efetiva interpretação da sintaxe base de Ontoprolog. De toda forma, a introdução da subseção 7.3.1 apresenta uma intuição de como o processo é implementado em Ontoprolog.

7.3.1 Função filtro

A Função filtro, responsável pela tradução da sintaxe de Ontoprolog, é definida conforme segue:

Definição 7.1 (Função filtro)**.** A *função filtro*[2], representada por $\Longrightarrow$, é uma *função semântica* (NIELSON; NIELSON, 2007, p. 91) que, dado um contexto Δ, mapeia uma sentença ou um fragmento de sentença sintática α em um conjunto finito e não vazio β, que consiste em ao menos um caso de: zero ou mais relações semânticas; e zero ou mais sentenças ou fragmentos sintáticos diferentes de α; ou uma constante $\perp$, que denota que a função não é definida para a entrada α, de modo que a *função filtro* seja uma *função total*.

A função $\Longrightarrow$ é definida por casos, em que cada caso α é uma sentença ou um fragmento de sentença delimitados por $[\![$ e $]\!]$ obtidos por indução sobre a estrutura das sentenças da linguagem extraída da seção 7.2, conforme apresentado na subseção 7.3.3. Uma sentença é uma instância da gramática EBNF descrita na já mencionada seção 7.2. Um fragmento de sentença consiste em concatenações de relações semânticas com partes de sentenças sintáticas. Por exemplo, `a :: b` é uma sentença, uma vez que é uma instância completa da EBNF descrita na seção 7.2. Já $dio([\![\mathsf{a}]\!]_t, [\![\mathsf{b}]\!]_t)$ é um fragmento de sentença, porque representa um estágio intermediário entre uma relação semântica e uma sentença. Por sua vez, $dio(a, b)$ é uma relação semântica, porque representa o estado final de uma relação semântica, uma vez que não contém nenhum fragmento delimitado por $[\![$ e $]\!]$. —

A definição dos casos α de $\Longrightarrow$ é apresentada na seção 7.4. Nessas definições, os símbolos iniciados por letras maiúsculas representam variáveis para Construtor de identificador de entidade (Definição 7.5) se estiverem circunscritas por $[\![$ e $]\!]$, e as letras gregas representam *esquemas sintáticos*[3], que agrupam um ou mais operadores e respectivos construtores de identificadores de entidade.

Teorema 7.1 (Adequação da função filtro $\Longrightarrow$)**.** A definição da função $\Longrightarrow$ é adequada à tradução da gramática EBNF definida na seção 7.2. ■

Demonstração. Por construção, em que as definições apresentadas na seção 7.4 cobrem todas as combinações possíveis que geram instâncias válidas da gramática EBNF definida na seção 7.2. □

Considerando completa a definição de casos de $\Longrightarrow$ em relação às definições sintáticas da linguagem descrita na seção 7.2 (Teorema 7.1), toda *especificação* composta por sentenças bem formadas (Definição 7.8) denota uma Teoria Ontoprolog (Definição 5.2).

[2] Também chamada de *função de interpretação*, e lida como "denota". O nome *função filtro* é extraído de Gupta (1998, p. 152).

[3] Os esquemas sintáticos são introduzidos na subseção 7.4.2.

As subseções seguintes apresentam a definição da Função filtro por meio da indução na estrutura sintática de Ontoprolog, conforme apresentado na seção 7.2.

7.3.2 O contexto Δ da função filtro

O contexto Δ, abordado pela subseção 7.3.1, é definido como:

Definição 7.2 (Contexto Δ)**.** O contexto Δ é um conjunto de relações semânticas tomadas *a priori* pela função $\Longrightarrow$ e disponível no escopo da função. —

Os predicados contidos na Equação 7.1 e na Equação 7.2, apresentados na anotação descrita na seção 6.1, são válidos para instâncias de relações do contexto Δ:

$$\begin{aligned}\forall T\ \forall S\ \forall X \in \Delta(\quad & extensionof\_irreflexive(T,S) \leftarrow deo(T,S) \\ & extensionof\_irreflexive(T,S) \leftarrow deo(T,X), \\ & \qquad extensionof\_irreflexive(X,S) \quad)\end{aligned} \tag{7.1}$$

$$\begin{aligned}\forall T\ \forall S\ \forall M \in \Delta(\quad & instanceof(T,M) \leftarrow dio(T,M) \\ & instanceof(T,S) \leftarrow dio(T,M), \\ & \qquad extensionof\_irreflexive(M,S) \quad)\end{aligned} \tag{7.2}$$

Esses predicados são usados por casos de $\Longrightarrow$ que dependem do contexto Δ, como os descritos na subseção 8.2.2.9 e na subseção 8.2.2.10.

7.3.3 Algoritmo de controle da tradução

Um algoritmo Σ que traduz um conjunto de sentenças sintáticas (ou seja, uma Especificação Ontoprolog) e produz um conjunto de relações semânticas (seção 6.2) é um algoritmo de ponto fixo que aplica $\Longrightarrow$ recursivamente a cada sentença ou fragmento de sentença α do conjunto de entrada e a cada sentença ou fragmento de sentença da imagem β da função. A cada iteração n do algoritmo (ou seja, a cada aplicação da função), o conjunto Δ é incrementado com as relações semânticas obtidas da imagem β da iteração anterior. O algoritmo é aplicado recursivamente até que uma determinada produção β_n seja a mesma de β_{n+1}, ou em algum β_i ocorra um $\perp$, caso em que toda a árvore de aplicações do algoritmo denota um erro.

Outro processo controla a tradução de sentenças de um programa Prolog em relações semânticas e faz uso de Σ do seguinte modo:

a) Inicialmente o algoritmo identifica as sentenças Prolog cujo predicado mais externo seja um Operador principal (Definição 7.4);

b) O algoritmo Σ é aplicado sobre o conjunto dessas sentenças e um contexto inicial Δ de modo que a saída β de Σ seja tratada da seguinte forma:

   – Caso em β exista um $\perp$, o algoritmo controlador termina com erro;

   – Caso contrário, como a função $\Longrightarrow$ é completa, β conterá as relações semânticas denotadas pelas sentenças do conjunto inicial. Um procedimento final de expansão, denominado *semantic_expansion*/1, é aplicado sobre β de modo a se obter o conjunto final de relações semânticas de $\mathcal{H}$ da estrutura $\mathcal{OT}$, que será a união de β com as instâncias do universo de Herbrand dos predicados resultantes das expansões. Dessa forma, as relações do conjunto $\mathcal{H}$, expandidas da saída β, são todas e somente essas as fórmulas semânticas que fazem parte dos modelos de uma Teoria Ontoprolog. As expansões das relações semânticas são descritas na seção 7.6 e detalhadas no Apêndice F.

7.3.4 Definições preliminares

As definições apresentadas nesta subseção são utilizadas nas seções seguintes.

Definição 7.3 (Operador)**.** Os *operadores* são *construtores terminais rígidos* (subseção 7.2.1) que relacionam um ou mais Construtor de identificador de entidade (Definição 7.5) ou outros operadores. A lista completa de operadores de Ontoprolog é apresentada no Apêndice E. —

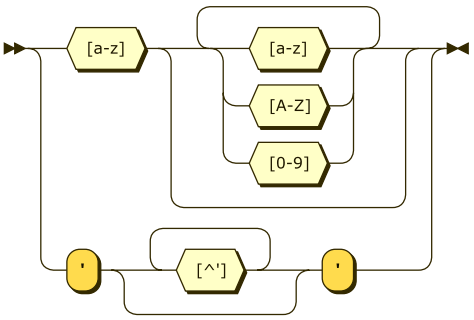
Definição 7.4 (Operador principal)**.** *Operador principal* é o Operador que identifica uma categoria de Sentença. A Tabela 6 contém a lista de operadores principais. Cada sentença possui um e exatamente um operador principal. —

Definição 7.5 (Construtor de identificador de entidade)**.** Um *construtor de identificador de entidade* é a sequência de caracteres que representa sintaticamente um nome (ou um identificador) de uma entidade. Um construtor de identificador de entidade é representado por $[\![\mathsf{T}]\!]_t$, sendo que do ponto de vista sintático, $[\![\mathsf{T}]\!]_t$ é um conjunto de caracteres compostos conforme a Figura 30 que não coincida com um Operador (Definição 7.3), nem com algum outro *construtor terminal rígido* (subseção 7.2.1). T é a denotação de $[\![\mathsf{T}]\!]_t$ no mundo semântico. Dessa forma, temos que:

$[\![\mathsf{T}]\!]_t \Longrightarrow \mathsf{T}$

se $atom(T)$ sucede para qualquer T, sendo que $atom/1$ sucede se o parâmetro é qualificado como átomo conforme a Figura 30 e se T não coincida com um Operador (Definição 7.3), nem com algum outro *construtor terminal rígido* (subseção 7.2.1).

Figura 30 – Diagrama *railroad* de Construtor de identificador de entidade



Fonte: Os autores

A definição seguinte refere-se ao caso em que a sintaxe agrupa os construtores de identificadores de entidades em listas. Nesse caso, a semântica de uma lista de entidades é a conjunção da semântica individual de cada entidade, sse cada T_i for uma entidade:

$$[\![\texttt{[} \mathsf{T}_1\texttt{, . . . , } \mathsf{T}_n \texttt{]}]\!]_{tl} \Longrightarrow \texttt{\{ } [\![\mathsf{T}_1]\!]_t\texttt{, . . . , } [\![\mathsf{T}_n]\!]_t \texttt{ \}}$$

Definição 7.6 (Função *entity*)**.** A função *entity*, representada por $[\![\alpha]\!]_{entity}$ denota o predicado $entity/1$, conforme segue:

$$[\![\mathsf{T}]\!]_{entity} \Longrightarrow entity([\![\mathsf{T}]\!]_{et})$$

$$[\![\texttt{[}\mathsf{T}_1\texttt{, . . . , }\mathsf{T}_n\texttt{]}]\!]_{entity}$$

$$\Longrightarrow$$

$$[\![\mathsf{T}_1]\!]_{entity}$$

$$\vdots$$

$$[\![\mathsf{T}_n]\!]_{entity}$$

quando $[\![\mathsf{T}_i]\!]_{entity}$ sucede para todos os $1 \leq i \leq n$.

Em que $[\![\mathsf{T}]\!]_{et}$ é definido conforme segue:

$[\![\mathsf{T}]\!]_{et} \Longrightarrow \mathsf{T}$

Se

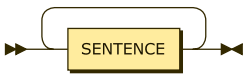
$$\begin{cases} [\![\mathsf{T}]\!]_t \text{ sucede, ou} \\ T = \texttt{P at R}, \text{ quando } [\![\mathsf{P}]\!]_t \text{ e } [\![\mathsf{R}]\!]_t \text{ sucedem} \end{cases}$$

A subseção 8.2.2.1 apresenta uma extensão da Definição 7.6 de modo a comportar Qua Individuals no contexto de ontologias UFO.

7.3.5 Especificação e Sentença

Definição 7.7 (Especificação Ontoprolog)**.** *Especificação* ou *Especificação Ontoprolog* é o conjunto não vazio de *sentenças* (Definição 7.8), conforme ilustrado pela Figura 31.

Figura 31 – Diagrama *railroad* de Especificação Ontoprolog



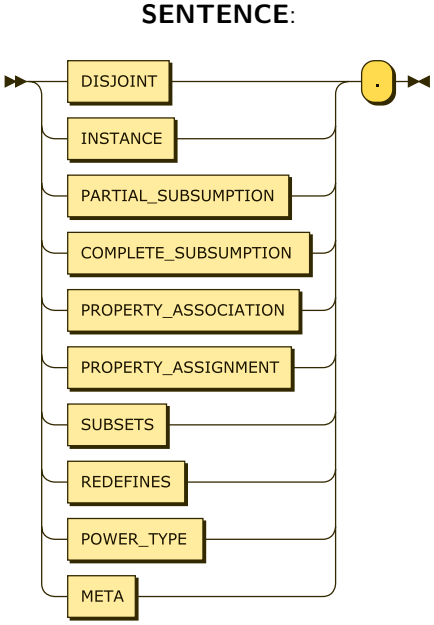
Fonte: Os autores

Definição 7.8 (Sentença)**.** Um conjunto de caracteres é uma *sentença Ontoprolog* (frase, sentença sintática ou sentença bem formada) se, e somente se, contém um Operador principal e são ordenados conforme as regras sintáticas definidas na seção 7.2. As sentenças são finalizadas com um ponto (`.`).

A Figura 32 contém a definição sintática de *grupos de sentenças Ontoprolog* representados por um diagrama *railroad*. Cada caso da disjunção indicada na figura refere-se a um *grupo de sentenças*, que agrega as sentenças pela *finalidade semântica principal* que denotam. Por exemplo, a finalidade principal de `INSTANCE` é declarar instâncias de *entidades* (subseção 7.4.4); a finalidade principal de `DISJOINT` é a declaração de disjunções entre instâncias de entidades (subseção 7.4.1). Tratam-se de grupo de sentenças porque, devido aos recursos de açúcares sintáticos[4], agrupam sentenças de diferentes *categorias*. Uma *categoria de sentença* Ontoprolog é identificada por um Operador principal (Definição 7.4).

[4] Um exemplo de açúcar sintático é o caso de uma única sentença que indica simultaneamente que uma lista de entidades são disjuntas e são instância de outra.

Figura 32 – Diagrama *railroad* de Sentença



Fonte: Os autores

Embora um grupo de sentenças possam conter diferentes categorias de sentenças, e como cada categoria de sentença possui um operador principal, é necessário identificar, num grupo de sentenças, qual operador entre elas é o operador principal. Isso ocorre da seguinte forma. Nos casos de composição de diferentes categoria de sentenças, para que seja possível identificar qual é o operador principal, os operadores são providos de prioridades. Desse modo, o operador com maior prioridade numa sentença é o principal. A combinação de dois ou mais operadores com a mesma prioridade torna a sentença um caso de possível colapso de múltiplos operadores principais. Nesse caso, a semântica é desambiguizada pela ordem em que os operadores aparecem e pela associativade atribuída a cada um. Porém, saliente-se que mesmo nesses casos, há apenas um único operador principal.

A lista de operadores principais[5] está presente na Tabela 6[6], que lista cada um dos operadores principais, apresentados por ordem de prioridade, sendo que o primeiro é o de maior, e o último, o de menor prioridade. Em cada linha, apresenta-se também o significado natural da sentença – que pode ser usada para leitura oral das sentenças – as relações semânticas que cada operador sintático pode denotar[7] e as subseções da seção 7.4 que abordam os detalhes das funções de filtro sintático do grupo de sentenças

[5] Os operadores principais podem exigir que outros operadores sejam utilizados para que a sintaxe da sentença esteja correta. Esses operadores acessórios são utilizados para verificar a adequação das sintaxes concretas em relação às definições contidas na seção 7.2.

[6] Há operadores sintáticos acessórios ou dependentes dos principais, que não estão listados na tabela.

[7] Além das relações principais, os construtores sintáticos também podem denotar outras relações por meio de açúcares sintáticos.

que necessitam do operador principal indicado. Essas seções contém a definição por *grupo de sentença*. Conforme a definição de *entity*/1 (seção 6.2), todos os constructos sintáticos derivam uma ou mais relações *entity*(e), uma para cada entidade e contida nas relações semânticas. A lista completa de operadores é apresentada pelo Apêndice E.

Tabela 6 – Operadores principais da sintaxe base de Ontoprolog

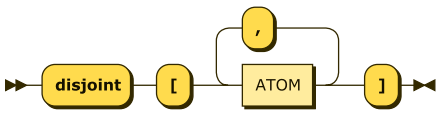
Operador principal	Tipo	Leitura natural	Relações semânticas	Seções (grupo de sentenças)
`on`	infixo, associado à direita	φ *é atribuído a* ψ	*dpo*/2 *dmo*/2	7.4.5 7.4.7
`classifying`	infixo, associado à direita	φ *é um Power Type que classifica* ψ	*pt*/2	7.4.8
`:=`	infixo, assoc. à direita	φ *cujo valor é* ψ	*dpv*/2 *dio*/2 *dpo*/2	7.4.6
`redefines`	infixo, assoc. à esquerda	φ *é uma redefinição de* ψ	*dro*/2 *dio*/2 *dpo*/2	7.4.10
`subsets`	infixo, assoc. à esquerda	φ *é um subconjunto de* ψ	*dso*/2 *dio*/2 *dpo*/2	7.4.9
`extends` `extend`	infixo, assoc. à esquerda	φ *estende* ou *é uma subsunção (parcial) de* ou *é subordinado (parcialmente) a* ψ	*deo*/2 *dio*/2 *dd*/1	7.4.2 7.5.2
`cover`	infixo, assoc. à esquerda	φ *estende completamente* ou *é uma subsunção completa de* ou *é toda subordinação de* ψ	*dcomplete*/1 *deo*/2 *dio*/2 *dd*/1	7.4.3
`::`	infixo, assoc. à direita	φ *é instância de* ou *é particular de* ou *exemplifica* ψ	*dio*/2 *dd*/1	7.4.4 7.4.2 7.4.3 7.5.2
`disjoint`	prefixado associativo	*todos de* φ *são disjuntos*	*dd*/1	7.4.1 7.4.4 7.4.2 7.4.3

Fonte: Produzido pelos autores.

7.4 Indução na estrutura da sintaxe de sentenças

As subseções desta seção apresentam a definição dos casos de $\Longrightarrow$ com base em cada um dos grupos de sentenças indicados na Figura 32 (Diagrama *railroad* de Sentença). Tratam-se de interpretações semântica de cada uma das formas de sentenças bem formadas em acordo com as regras apresentadas na seção 7.2[8]. Em cada subseção, a título de conveniência na verificação da totalidade da definição do filtro – ou seja, da tradução para cada combinação possível dos operadores sintáticos – acrescenta-se uma representação EBNF na forma de diagrama *railroad* extraídos da subseção 7.2.1[9].

7.4.1 Disjunção de conceitos



Trata-se da sentença cujo operador principal denota que uma lista de conceitos são disjuntos. O operador é traduzido no predicado semântico $dd/1$:

$[\![\texttt{disjoint [T}_1\texttt{, . . . , T}_n\texttt{]}]\!]$

$\Longrightarrow$

$dd([\![\texttt{[T}_1\texttt{, . . . , T}_n\texttt{]}]\!]_{tl})$

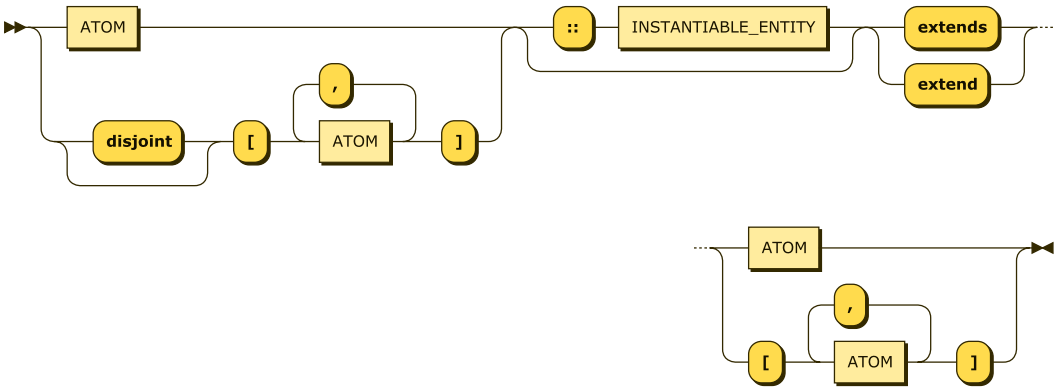
$[\![\texttt{[T}_1\texttt{, . . . , T}_n\texttt{]}]\!]_{entity}$

[8] Os casos mal formados devem ser tratados na implementação específica com mensagens de erro ou outro tipo de destaque.

[9] Todas as figuras de diagramas EBNF nesta e nas demais seções foram extraídas da subseção 7.2.1.

7.4.2 Subsunção parcial

PARTIAL_SUBSUMPTION:



Na análise desta sintaxe, introduz-se a noção de *esquema sintático*, que consiste na captura de um ou mais grupos de operadores sintáticos. Os esquemas são representados por uma variável designada por uma letra grega e, a exemplo do paradigma usado para a função filtro em si, os esquemas são analisados por casos.

A sintaxe da subsunção parcial é utilizada com diferentes açúcares sintáticos, e é combinada com outros operadores de modo a otimizar a pragmática da linguagem. Devido à prioridade dos operadores `extends` e `extend` (Tabela 6), o esquema padrão da relação de subsunção parcial pode ser apresentado como:

$$[\![\varphi \texttt{ extends } \psi]\!]$$

ou como:

$$[\![\varphi \texttt{ extend } \psi]\!]$$

em que φ são os construtores à esquerda do operador principal, e ψ os à direita.

Na sintaxe apresentada, destacam-se que dois operadores podem ser usados como sinônimos[10] nas sentenças que denotam subsunção parcial: `extends` e `extend`. Observa-se que não se trata de um caso em que existem dois operadores principais num mesmo grupo de sentenças, mas sim de um caso em que um ou outro operador pode ser usado para denotar o mesmo conjunto de predicados e relações semânticas. Por isso, como são sinônimos, obtém-se esta tradução:

[10] Os dois operadores foram incluídos na linguagem como sinônimos para que se possa produzir frases gramaticalmente bem formadas na língua inglesa. Desse modo, se à esquerda do operador estiver presente apenas um único termo `T`, o usuário da linguagem poderia escrever `T extends S`, enquanto, se houver mais de um termo à esquerda, seria natural escrever a frase conjugando o verbo ativo na terceira pessoa do plural: `[`$\texttt{T}_1$`, `$\texttt{T}_2$`] extend S`.

$[\![\varphi$ `extend` $\psi\ ]\!]$

$\Longrightarrow$

$[\![\varphi$ `extends` $\psi]\!]$

Dessa forma, apresenta-se apenas a tradução com base no operador `extends`.

Com base no esquema apresentado, a tradução deste grupo sintático pode ser dada pela análise de casos de φ e ψ. Desse modo, os seguintes são os casos de açúcares sintáticos que combinam o operador principal com a declaração de disjunção (subseção 7.4.1) e de instância de entidade (subseção 7.4.4):

$[\![\varphi$ `extends` $\psi\ ]\!]$

$\Longrightarrow$

$[\![\varphi]\!]$
$[\![\alpha$ `extends` $\psi]\!]$

se, para qualquer ψ:

$$\varphi = \begin{cases} \texttt{T :: W} & \text{, então } \alpha = \texttt{T} \\ \texttt{[T}_1\texttt{, . . . , T}_n\texttt{] :: W} & \text{, então } \alpha = \texttt{[T}_1\texttt{, . . . , T}_n\texttt{]} \\ \texttt{disjoint [T}_1\texttt{, . . . , T}_n\texttt{]} & \text{, então } \alpha = \texttt{[T}_1\texttt{, . . . , T}_n\texttt{]} \\ \texttt{disjoint [T}_1\texttt{, . . . , T}_n\texttt{] :: W} & \text{, então } \alpha = \texttt{[T}_1\texttt{, . . . , T}_n\texttt{]} \end{cases}$$

Os casos padrões são os que possuem um único conceito ou uma lista de conceitos em ambos os lados do operador principal. Esses casos que são traduzidos como:

$[\![\texttt{T}$ `extends` $\texttt{S}\ ]\!]$

$\Longrightarrow$

$deo([\![\texttt{T}]\!]_t, [\![\texttt{S}]\!]_t)$
$[\![\texttt{[T, S]}]\!]_{entity}$

se $[\![\texttt{T}]\!]_t \neq [\![\texttt{S}]\!]_t$

$[\![\texttt{[T}_1\texttt{, . . . , T}_n\texttt{]}$ `extends` $\texttt{S}\ ]\!]$

$\Longrightarrow$

$[\![\texttt{T}_1\ \texttt{extends}\ \texttt{S}]\!]$

$\vdots$

$[\![\texttt{T}_n\ \texttt{extends}\ \texttt{S}]\!]$

quando $[\![\texttt{T}_i]\!]_t$ sucede para todos os $1 \leq i \leq n$.

$[\![\texttt{T}\ \texttt{extends}\ [\texttt{S}_1, \ldots, \texttt{S}_n]\ ]\!]$

$\Longrightarrow$

$[\![\texttt{T}\ \texttt{extends}\ \texttt{S}_1]\!]$

$\vdots$

$[\![\texttt{T}\ \texttt{extends}\ \texttt{S}_n]\!]$

quando $[\![\texttt{S}_i]\!]_t$ sucede para todos os $1 \leq i \leq n$.

$[\![[\texttt{T}_1, \ldots, \texttt{T}_n]\ \texttt{extends}\ [\texttt{S}_1, \ldots, \texttt{S}_m]]\!]$

$\Longrightarrow$

$[\![\texttt{T}_1\ \texttt{extends}\ \texttt{S}_1]\!]$

$\vdots$

$[\![\texttt{T}_n\ \texttt{extends}\ \texttt{S}_1]\!]$

$\vdots$

$[\![\texttt{T}_1\ \texttt{extends}\ \texttt{S}_m]\!]$

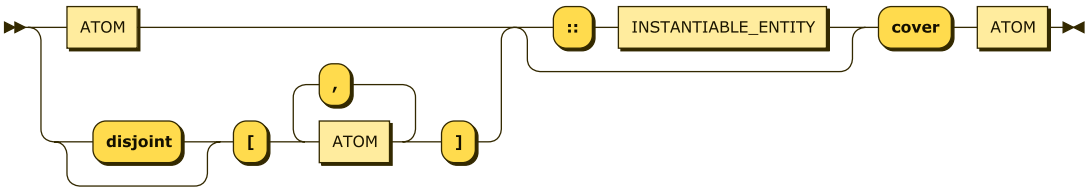
$\vdots$

$[\![\texttt{T}_n\ \texttt{extends}\ \texttt{S}_m]\!]$

quando $[\![\texttt{T}_i]\!]_t$ sucede para todos os $1 \leq i \leq n$ e $[\![\texttt{S}_j]\!]_t$ sucede para todos os $1 \leq j \leq m$.

7.4.3 Subsunção completa

COMPLETE_SUBSUMPTION:



A exemplo da Subsunção parcial (subseção 7.4.2), a sintaxe da subsunção completa (ou total) é traduzida por meio de esquemas. Dessa forma, tem-se:

$[\![\varphi \texttt{ cover T}]\!]$

em que φ são construtores à esquerda do operador principal e `T` é o construtor à direita. Um símbolo grego não é utilizado à direita porque não se trata de um grupo de operadores, e sim de um construtor de conceito diretamente.

Os casos de açúcares sintáticos que combinam a subsunção completa com disjunção (subseção 7.4.1) e com instância de entidade (subseção 7.4.4) são tratados antes do caso padrão:

$[\![\varphi \texttt{ cover T}]\!]$

$\Longrightarrow$

$[\![\varphi]\!]$

$[\![\alpha \texttt{ cover T}]\!]$

se:

$$\varphi = \begin{cases} \texttt{[T}_1\texttt{, . . . , T}_n\texttt{] :: W} & \text{, então } \alpha = \texttt{[T}_1\texttt{, . . . , T}_n\texttt{]} \\ \texttt{disjoint [T}_1\texttt{, . . . , T}_n\texttt{]} & \text{, então } \alpha = \texttt{[T}_1\texttt{, . . . , T}_n\texttt{]} \\ \texttt{disjoint [T}_1\texttt{, . . . , T}_n\texttt{] :: W} & \text{, então } \alpha = \texttt{[T}_1\texttt{, . . . , T}_n\texttt{]} \end{cases}$$

Por fim, o caso padrão é o que possui uma lista à esquerda do operador principal, e um único termo à direita:

$[\![\texttt{[T}_1\texttt{, . . . , T}_n\texttt{] cover S }]\!]$

$\Longrightarrow$

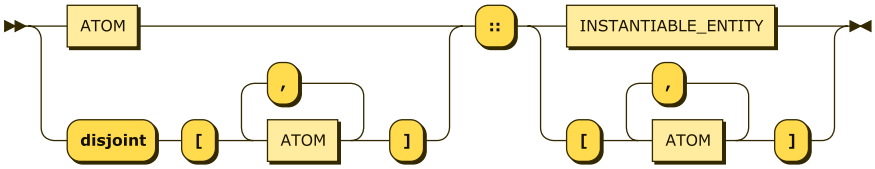
$dcomplete(\llbracket \texttt{S} \rrbracket_t, \llbracket \texttt{[T}_1\texttt{, . . . T}_n\texttt{]} \rrbracket_{tl})$

$\llbracket \texttt{[T}_1\texttt{, . . . , T}_n\texttt{] extend S} \rrbracket$

Por essa definição, o operador `cover` denota o mesmo que os operadores `extends` e `extend`, mas também acrescenta o predicado *dcomplete*/2, que indica que a lista em conceitos em tela são completos.

7.4.4 Instância

INSTANCE:



INSTANTIABLE_ENTITY:

Trata-se da sintaxe para expressão de instâncias. O símbolo `::` é preferido ao invés de uma palavra na língua inglesa por ser este símbolo comum na literatura da UFO e de UML, de modo que é natural para os especialistas e pode igualmente ser oralizado conforme indicação da Tabela 6.

A construção padrão é a que denota que uma única entidade `P` é instância de um único *type* `T`:

$\llbracket \texttt{P :: T} \rrbracket$

$\Longrightarrow$

$dio(\llbracket \texttt{P} \rrbracket_t, \llbracket \texttt{T} \rrbracket_t)$

$\llbracket \texttt{[P,T]} \rrbracket_{entity}$

se $P \neq T$

Outro caso é o que denota que um conjunto de conceitos são, todos eles, instância de um único *type* T:

$$[\![\texttt{[P}_1\texttt{, . . . , P}_n\texttt{] :: T}]\!]$$

$$\Longrightarrow$$

$$[\![\texttt{P}_1\ \texttt{::}\ \texttt{T}]\!]$$

$$\vdots$$

$$[\![\texttt{P}_n\ \texttt{::}\ \texttt{T}]\!]$$

Ou então, quando um único P é instância de múltiplos T, caso que denota a *múltipla instanciação*, ou seja, quando um conceito P é instância simultânea direta de mais do que um único tipo:

$$[\![\texttt{P :: [T}_1\texttt{, . . . , T}_n\texttt{]}]\!]$$

$$\Longrightarrow$$

$$[\![\texttt{P}\ \texttt{::}\ \texttt{T}_1]\!]$$

$$\vdots$$

$$[\![\texttt{P}\ \texttt{::}\ \texttt{T}_n]\!]$$

Como última combinação possível das listas, apresenta-se a produção para o caso em que há listas em cada lado do operador :: em tela:

$$[\![\texttt{[P}_1\texttt{, . . . , P}_n\texttt{] :: [T}_1\texttt{, . . . , T}_n\texttt{]}]\!]$$

$$\Longrightarrow$$

$$[\![\texttt{P}_1\ \texttt{::}\ \texttt{T}_1]\!]$$

$$\vdots$$

$$[\![\texttt{P}_n\ \texttt{::}\ \texttt{T}_1]\!] \qquad \vdots$$

$$[\![\texttt{P}_1\ \texttt{::}\ \texttt{T}_m]\!]$$

$$\vdots$$

$$[\![\texttt{P}_n\ \texttt{::}\ \texttt{T}_m]\!]$$

Por fim, acrescenta-se a definição do açúcar sintático que combina o operador `disjoint`, em que α e β estão, respectivamente, nas formas precedentes antes e depois do operador `::` para instância:

$$[\![\texttt{disjoint}\ \alpha\ \texttt{::}\ \beta]\!]$$

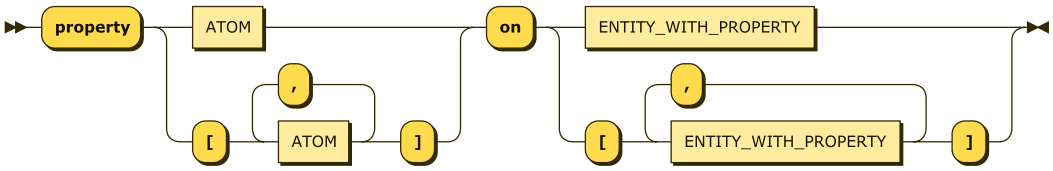
$\Longrightarrow$

$[\![\texttt{disjoint}\ \alpha]\!]$

$[\![\alpha\ \texttt{::}\ \beta]\!]$

7.4.5 Associação de propriedade

PROPERTY_ASSOCIATION:



ENTITY_WITH_PROPERTY:

Este constructo sintático deve ser usado para associar propriedades a entidades da teoria. Nos casos definidos nesta seção, assume-se a existência de uma entidade chamada `property`, que deve estar presente na Metateoria de *Hypertypes*.

A tradução é realizada da seguinte forma:

$[\![\texttt{property P on T}]\!]$

$\Longrightarrow$

$dpo([\![\mathsf{P}]\!]_t, [\![\mathsf{T}]\!]_t)$

$defeasible(dio([\![\mathsf{P}]\!]_t, property))$

$[\![\texttt{[P, T]}]\!]_{entity}$

se $[\![\mathsf{P}]\!]_t \neq [\![\mathsf{T}]\!]_t$

$[\![\texttt{property [P}_1\texttt{, . . . , P}_n\texttt{] on T}\ ]\!]$

$\Longrightarrow$

$[\![\texttt{property P}_1\ \texttt{on T}]\!]$

$\vdots$

$[\![\texttt{property P}_n\ \texttt{on T}]\!]$

quando $[\![\texttt{P}_i]\!]_t$ sucede para todos os $1 \leq i \leq n$.

$[\![\texttt{property P on [T}_1\texttt{, . . . , T}_n\texttt{] }]\!]$

$\Longrightarrow$

$[\![\texttt{property P on T}_1]\!]$

$\vdots$

$[\![\texttt{property P on T}_n]\!]$

quando $[\![\texttt{T}_i]\!]_t$ sucede para todos os $1 \leq i \leq n$.

$[\![\texttt{property [P}_1\texttt{, . . . , P}_n\texttt{] on [T}_1\texttt{, . . . ,T}_m\texttt{]}]\!]$

$\Longrightarrow$

$[\![\texttt{property P}_1\ \texttt{on T}_1]\!]$

$\vdots$

$[\![\texttt{property P}_n\ \texttt{on T}_1]\!]$

$\vdots$

$[\![\texttt{property P}_1\ \texttt{on T}_m]\!]$

$\vdots$

$[\![\texttt{property P}_n\ \texttt{on T}_m]\!]$

quando $[\![\texttt{P}_i]\!]_t$ sucede para todos os $1 \leq i \leq n$ e $[\![\texttt{T}_j]\!]_t$ sucede para todos os $1 \leq j \leq m$.

Destaca-se nesta definição o uso do predicado $defeasible/1$. Conforme discutido na subseção 6.3.3, o predicado é usado para raciocínio retratável a respeito da relação escopo do predicado. No caso específico desta definição, a semântica atribuída à $defeasible(dio([\![\texttt{P}]\!]_t, property))$, dada pelo código presente no Apêndice F e discutida na

seção 7.6, é esta: no caso de não haver outra informação a respeito do tipo instanciado pela entidade $[\![\texttt{P}]\!]_t$, o sistema assumirá que se trata de *property*. Logo, ao declarar associação de propriedades a determinada entidade, o sistema assumirá o tipo daquela propriedade como sendo uma instância do *hypertype property*.

7.4.6 Atribuição de valor de propriedade

PROPERTY_ASSIGNMENT:

PROPERTY_TYPE:

ENTITY_IN_RELATION:

Este constructo sintático deve ser usado para atribuir valores à instâncias de propriedades de entidades. Trata-se de um constructo que denota uma série de afirmações em termos semânticos: as instâncias de propriedades são definidas em Ontoprolog por meio de uma relação especial, chamada `at/2`, que é transliterada do nível sintático para o nível semântico exatamente com o mesmo predicado. Em `P at T`, uma propriedade P atribuída à entidade T é instância da propriedade P, que por sua vez *deve ser* instância de `property` (conforme definições presentes na subseção 6.4.3.8). Dessa forma, Ontoprolog trata a hierarquia de instâncias de propriedades como elementos primitivos (de primeira classe) da linguagem.

A interpretação do constructo sintático para atribuição de valores à propriedades de instâncias é dado pela definição:

$$[\![\texttt{P at T := V}]\!]$$

$\Longrightarrow$

$dpv(at(\llbracket \texttt{P} \rrbracket_t, \llbracket \texttt{T} \rrbracket_t), V)$

$dio(at(\llbracket \texttt{P} \rrbracket_t, \llbracket \texttt{T} \rrbracket_t), \llbracket \texttt{P} \rrbracket_t)$

$\llbracket \texttt{property P on T} \rrbracket$

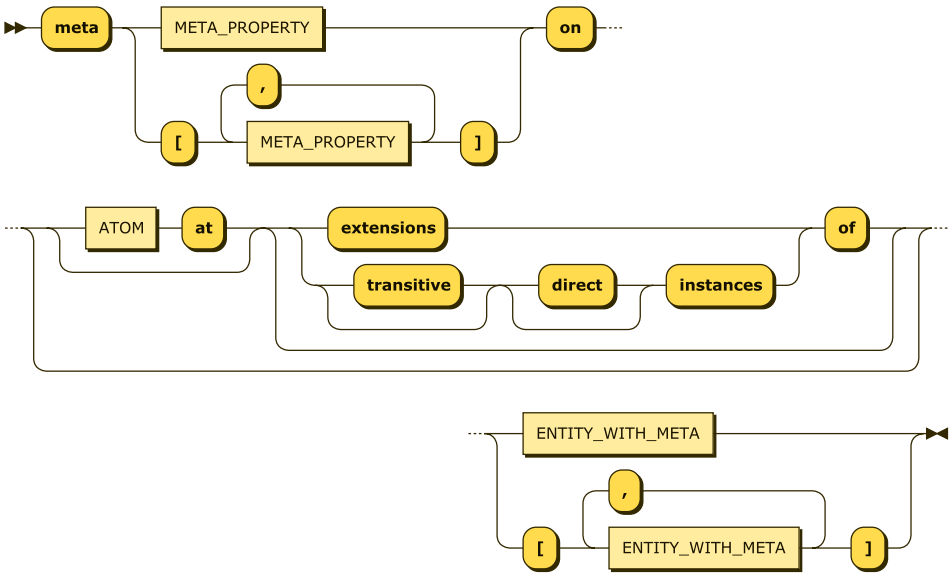
$\llbracket \llbracket \texttt{P} \rrbracket_t \texttt{ at } \llbracket \texttt{T} \rrbracket_t \rrbracket_{entity}$

O valor `V` é uma Entidade lógica (Definição 5.5) transliterada do nível sintático para o mundo semântico (V) sem nenhuma interpretação adicional, de modo que exige-se que `V` seja um termo válido em Prolog e formado por operadores com prioridade inferior e com associatividade compatível com a definição do operador `:=/2` (Apêndice E).

Ao definir um valor para uma propriedade em uma instância, infere-se de modo não retratável que a entidade está associada àquela instância (subseção 7.4.5).

7.4.7 Meta-propriedades

META:



META_PROPERTY:

ENTITY_WITH_META:

Este constructo sintático deve ser usado para atribuir meta-propriedades a entidades da teoria. Uma característica especial desse construtor sintático é o fato de os constructos entre `meta` e `on` serem traduzidos para o universo semântico exatamente como estão sintaticamente. A exemplo do que é apresentado na Definição 7.5 e discutido na subseção 7.4.6, isso não implica que não haja tradução, mas apenas que uma vez interpretados pela função filtro, suas construções estão no universo semântico, e não mais sintáticos.

A tradução da atribuição de meta-propriedades lança mão de esquemas sintáticos e é realizada da seguinte forma:

$[\![$`meta M on` φ `T`$]\!]$

$\Longrightarrow$

$dmo(M, \varphi\ [\![\mathtt{T}]\!]_t)$

$[\![\mathtt{T}]\!]_{entity}$

se M é qualquer termo Prolog cujas prioridades dos operadores sejam inferiores e compatíveis com as regras de associação dos operadores `meta` e `on`, conforme definidos no Apêndice E, e se um dos casos abaixo é atendido:

$$\varphi = \begin{cases} \gamma\ \texttt{extensions of} \\ \gamma\ \texttt{instances of} \\ \gamma\ \texttt{direct instances of} \\ \gamma\ \texttt{transitive instances of} \\ \gamma\ \texttt{transitive direct instances of} \\ \gamma \end{cases}$$

quando γ é vazio ou $\gamma =$ `A at`, se $[\![\mathtt{A}]\!]_t$ sucede.

$[\![$`meta [M`$_1$`, . . . , M`$_n$`] on` φ `T` $]\!]$

$\Longrightarrow$

$[\![$`meta M`$_1$ `on` φ `T`$]\!]$

$\vdots$

$[\![$`meta M`$_n$ `on` φ `T`$]\!]$

quando $[\![\mathtt{M}_i]\!]_t$ sucede para todos os $1 \leq i \leq n$, em que φ é definido conforme o primeiro caso.

$[\![$`meta M on` φ `[`$\mathtt{T}_1$`, . . . ,` $\mathtt{T}_n$`]` $]\!]$

$\Longrightarrow$

$[\![$`meta M on` φ $\mathtt{T}_1]\!]$

$\vdots$

$[\![$`meta M on` φ $\mathtt{T}_n]\!]$

quando $[\![\mathtt{T}_i]\!]_t$ sucede para todos os $1 \leq i \leq n$, em que φ é definido conforme o primeiro caso.

$[\![$`meta [`$\mathtt{P}_1$`, . . . ,` $\mathtt{P}_n$`] on` φ `[`$\mathtt{T}_1$`,. . . ,` $\mathtt{T}_m$`]`$]\!]$

$\Longrightarrow$

$[\![$`meta` $\mathtt{M}_1$ `on` φ $\mathtt{T}_1]\!]$

$\vdots$

$[\![$`meta` $\mathtt{M}_n$ `on` φ $\mathtt{T}_1]\!]$

$\vdots$

$[\![$`meta` $\mathtt{M}_1$ `on` φ $\mathtt{T}_m]\!]$

$\vdots$

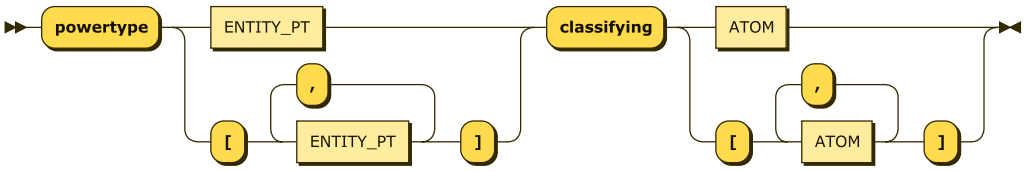
$[\![$`meta` $\mathtt{M}_n$ `on` φ $\mathtt{T}_m]\!]$

quando $[\![\mathtt{M}_i]\!]_t$ sucede para todos os $1 \leq i \leq n$ e $[\![\mathtt{T}_j]\!]_t$ sucede para todos os $1 \leq j \leq m$, em que φ é definido conforme o primeiro caso.

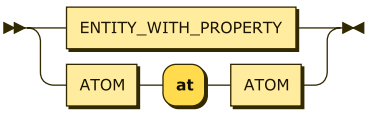
Ou seja, pela definição sintática apresentada nesta seção, os constructos `extensions of`, `instances of`, `direct instances of`, `transitive instances of` e `transitive direct instances of` são transliterados do universo sintático para o universo semântico. A semântica específica atribuída a esses constructos é descrita na seção 7.6. Essencialmente, esses constructos denotam o modo com as metas-propriedades indicadas são distribuídas sobre as relações de instanciação ($dio/2$) e subsunção ($deo/2$).

7.4.8 Power types

POWER_TYPE:



ENTITY_PT:



Este constructo sintático deve ser usado para expressar que instâncias de uma determinada entidade serão subsunção de outra entidade. A sintaxe é interpretada do seguinte modo:

$[\![$`powertype P classifying T`$]\!]$

$\Longrightarrow$

$pt([\![\texttt{P}]\!]_{pt}, [\![\texttt{T}]\!]_{t})$

$[\![$`[P, T]`$]\!]_{entity}$

se $[\![\texttt{P}]\!]_{pt} \neq [\![\texttt{T}]\!]_{t}$

$[\![$`powertype [`$\texttt{P}_1$`, ..., `$\texttt{P}_n$`] classifying T`$]\!]$

$\Longrightarrow$

$[\![$`powertype `$\texttt{P}_1$` classifying T`$]\!]$

$\vdots$

$[\![$`powertype `$\texttt{P}_n$` classifying T`$]\!]$

quando $[\![\texttt{P}_i]\!]_{pt}$ sucede para todos os $1 \leq i \leq n$.

$[\![$`powertype P classifying [`T_1`, . . . ,`T_n`] `$]\!]$

$\Longrightarrow$

$[\![$`powertype P classifying `$\mathsf{T}_1$$]\!]$

$\vdots$

$[\![$`powertype P classifying `$\mathsf{T}_n$$]\!]$

quando $[\![\mathsf{T}_i]\!]_t$ sucede para todos os $1 \leq i \leq n$.

$[\![$`powertype [`P_1`, . . . ,`P_n`] classifying [`T_1`, . . . ,`
T_m`]`$]\!]$

$\Longrightarrow$

$[\![$`powertype `P_1` classifying `$\mathsf{T}_1$$]\!]$

$\vdots$

$[\![$`powertype `P_n` classifying `$\mathsf{T}_1$$]\!]$

$\vdots$

$[\![$`powertype `P_1` classifying `$\mathsf{T}_m$$]\!]$

$\vdots$

$[\![$`powertype `P_n` classifying `$\mathsf{T}_m$$]\!]$

quando $[\![\mathsf{P}_i]\!]_{pt}$ sucede para todos os $1 \leq i \leq n$ e $[\![\mathsf{T}_j]\!]_t$ sucede para todos os $1 \leq j \leq m$.

Por fim, a definição de $[\![\alpha]\!]_{pt}$:

$[\![\alpha]\!]_{pt} \Longrightarrow \beta$

se:

$$\alpha = \begin{cases} \mathsf{T} & \text{, então } \beta = [\![\mathsf{T}]\!]_t \\ \mathsf{P\ at\ T} & \text{, então } \beta = at([\![\mathsf{P}]\!]_t, [\![\mathsf{T}]\!]_t) \end{cases}$$

A atribuição da semântica da relação de Power Type é descrita na seção 7.6.

7.4.9 Subconjunto de relação

SUBSETS:

Esta sintaxe denota o predicado $dso/2$, usado para qualificar relações de subconjunto entre conceitos. O uso padrão da sintaxe é definido para que `codomain` e `domain` estejam presentes como propriedades `P`.

$[\![\mathtt{P}_a \texttt{ at } \mathtt{T}_a \texttt{ subsets } \mathtt{P}_b \texttt{ at } \mathtt{T}_b]\!]$

$\Longrightarrow$

$dso(at([\![\mathtt{P}_a]\!]_t, [\![\mathtt{T}_a]\!]_t), at([\![\mathtt{P}_b]\!]_t, [\![\mathtt{T}_b]\!]_t))$

$[\![[\![\mathtt{P}_a]\!]_t \texttt{ at } [\![\mathtt{T}_a]\!]_t]\!]_{entity}$

$dio(at([\![\mathtt{P}_a]\!]_t, [\![\mathtt{T}_a]\!]_t), [\![\mathtt{P}_a]\!]_t)$

$[\![[\![\mathtt{P}_b]\!]_t \texttt{ at } [\![\mathtt{T}_b]\!]_t]\!]_{entity}$

$dio(at([\![\mathtt{P}_b]\!]_t, [\![\mathtt{T}_b]\!]_t), [\![\mathtt{P}_b]\!]_t)$

$[\![\texttt{property } \mathtt{P}_a \texttt{ on } \mathtt{T}_a]\!]$

$[\![\texttt{property } \mathtt{P}_b \texttt{ on } \mathtt{T}_b]\!]$

7.4.10 Redefinição de relação

REDEFINES:

Esta sintaxe é denota o predicado $dro/2$ que, a exemplo da subseção 7.4.9, usado é para restringir relações entre conceitos por meio de *redefinição*. Novamente, no uso padrão da sintaxe `codomain` e `domain` são usados como a propriedade `P`.

$[\![\mathtt{P}_a \texttt{ at } \mathtt{T}_a \texttt{ redefines } \mathtt{P}_b \texttt{ at } \mathtt{T}_b]\!]$

$\Longrightarrow$

$dro(at(\llbracket \mathsf{P}_a \rrbracket_t, \llbracket \mathsf{T}_a \rrbracket_t), at(\llbracket \mathsf{P}_b \rrbracket_t, \llbracket \mathsf{T}_b \rrbracket_t))$

$\llbracket \llbracket \mathsf{P}_a \rrbracket_t \texttt{ at } \llbracket \mathsf{T}_a \rrbracket_t \rrbracket_{entity}$

$dio(at(\llbracket \mathsf{P}_a \rrbracket_t, \llbracket \mathsf{T}_a \rrbracket_t), \llbracket \mathsf{P}_a \rrbracket_t)$

$\llbracket \llbracket \mathsf{P}_b \rrbracket_t \texttt{ at } \llbracket \mathsf{T}_b \rrbracket_t \rrbracket_{entity}$

$dio(at(\llbracket \mathsf{P}_b \rrbracket_t, \llbracket \mathsf{T}_b \rrbracket_t), \llbracket \mathsf{P}_b \rrbracket_t)$

$\llbracket \texttt{property } \mathsf{P}_a \texttt{ on } \mathsf{T}_a \rrbracket$

$\llbracket \texttt{property } \mathsf{P}_b \texttt{ on } \mathsf{T}_b \rrbracket$

7.5 Açúcar sintático para relações binárias

Com base nas regras da tradução da sintaxe base de Ontoprolog descritas na seção 7.4, esta seção apresenta um exemplo de como as regras podem ser combinadas para a criação de novos açúcares sintáticos da linguagem (subseção 1.4.4.2). Especialmente, apresenta-se a definição de um conjunto de açúcares que simplificam a especificação de relações binárias. As relações binárias são essencialmente entidades que instanciam (um dos subtipos de) `binaryRelationship` (seção 5.6).

Conforme a definição da Metateoria de *Hypertypes* (Definição 5.10) (Apêndice A), o que caracteriza uma relação binária (identificada por `binaryRelationship`) é a atribuição das quatro propriedades instanciadas diretamente pela Metateoria de *Hypertypes*, a saber, `domain`, `codomain`, `domain_cc` e `codomain_cc`. Com base nisso, acrescenta-se à Tabela 6 (Operadores principais da sintaxe base de Ontoprolog) a linha contida na Tabela 7.

Tabela 7 – Complemento aos operadores principais da sintaxe base de Ontoprolog: Relações binárias

Operador principal	Tipo	Leitura natural	Relações semânticas	Seções (grupo de sentenças)
`relates`	infixo, assoc. à esquerda	φ *relaciona* ψ_a *e/a/parte de* ψ_b	$dpo/2$ $dpv/2$ $dio/2$ $deo/2$ $dmo/2$	subseção 7.5.2

Fonte: Produzido pelos autores.

7.5.1 Definição da sintaxe EBNF

O Código 7 contém a especificação EBNF do açúcar sintático referente à declaração das relações binárias. A definição de SENTENCE, apresentada na subseção 7.3.5, fica acrescida da definição de SENTENCE desta seção, de modo que o conteúdo do Código 7 seja uma extensão ao conteúdo do Código 6.

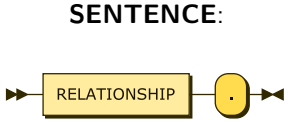
Código 7 – Definição EBNF da sintaxe especial para relações binárias de Ontoprolog

```
/* RELATIONAL TYPES */

SENTENCE ::=
      RELATIONSHIP '.'

RELATIONSHIP ::=
      ATOM '::' ('nonshared')? ATOM
      ( 'extends' ( ATOM | '[' ATOM ( ',' ATOM )* ']') )?
      'relates' RELATED ('and' | 'to' | 'partof') RELATED

RELATED ::=
      (CARDINALITY_CONSTRAINT ('instance' | 'instances')? 'of')? ENTITY_IN_RELATION

CARDINALITY_CONSTRAINT ::=
      INTEGER | 'many'
      | ('many' | INTEGER ) '..' ('many' | INTEGER )

INTEGER ::= [0-9]*
```

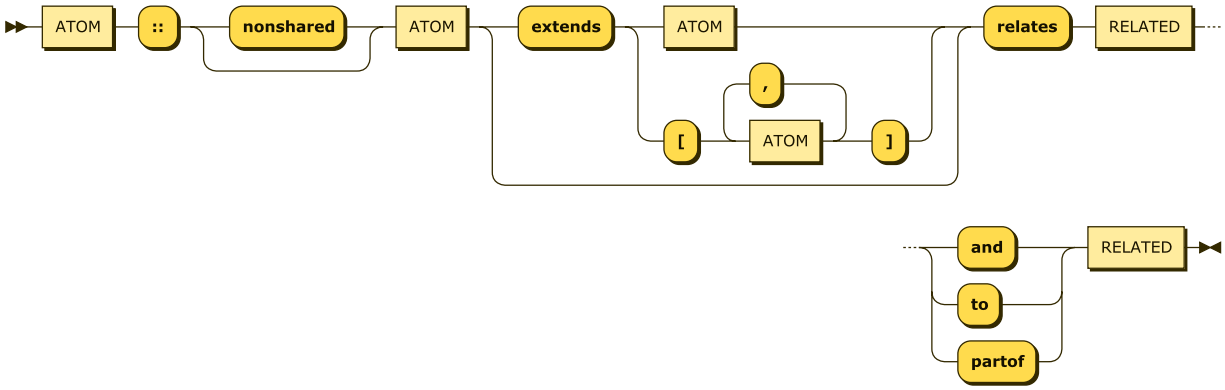
7.5.2 Filtro de tradução da sintaxe especial para relações binárias

A exemplo da seção 7.4, esta seção acrescenta casos à definição de $\Longrightarrow$ (subseção 7.3.1) que contemplam a tradução da especificação sintática da subseção 7.5.1. Da mesma forma, a exemplo da Figura 29, a Figura 33 contém a representação em diagramas *railroad* da EBNF em tela.

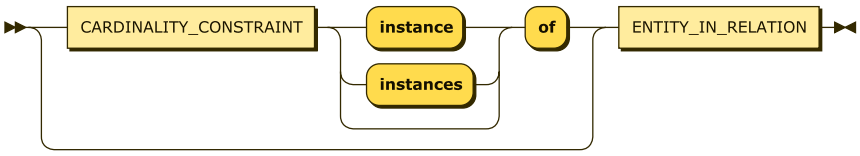
Figura 33 – Diagramas *railroad* da sintaxe especial para relações binárias



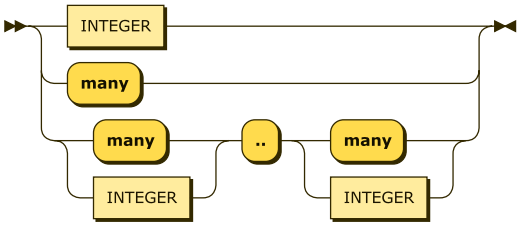
RELATIONSHIP:



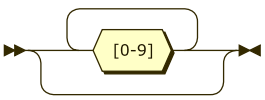
RELATED:



CARDINALITY_CONSTRAINT:



INTEGER:



O operador principal a ser analisado é o operador `relates`.

A sintaxe de restrições de cardinalidade apresentadas nesta seção, conforme a Definição 7.9, refere-se às restrições a serem observadas pelas instâncias das relações, conforme é definido pelas regras apresentadas pela subseção 6.4.3.10.

Definição 7.9 (Indicadores de restrição de cardinalidade)**.** Os *indicadores de restrição de cardinalidade* são construtores léxicos que aceitam um número inteiro ou construtor terminal `many`, ou ainda qualquer combinação deles separados pelo operador `..` (pontos duplos seguidos), conforme a representação CARDINALITY_CONSTRAINT da Figura 33. Eles visam representar configurações possíveis entre instâncias de entidades às quais eles estão relacionados, conforme é exposto na Definição 6.2 (Restrições de cardinalidade). —

Utilizando a abordagem por esquemas introduzida na subseção 7.4.2, a tradução é iniciada com a normalização das opções de `and`, `to` e `partof`, que são propostas sintaticamente apenas com intuito de melhorar a legibilidade das relações para os casos de associação simétrica, direcionada e de meronímia, respectivamente. Porém, esses diferentes constructos sintáticos não apresentam semântica específica, e são tratados como idênticos[11]. Com base nisso, têm-se as primeiras definições, em que γ é um esquema sintático:

$[\![\gamma$ `relates` α `to` $\beta]\!]$

$\Longrightarrow$

$[\![\gamma$ `relates` α `and` $\beta]\!]$

$[\![\gamma$ `relates` α `partof` $\beta]\!]$

$\Longrightarrow$

$[\![\gamma$ `relates` α `and` $\beta]\!]$

Uma vez realizada a normalização inicial, avança-se sobre o caso com construtor opcional que denota relação *deo*/2. Na sintaxe subsequente, γ, α e β representam um dos casos contidos na Figura 33.

$[\![$`T ::` γ `extends S relates` α `and` $\beta]\!]$

$\Longrightarrow$

$[\![$`T extends S`$]\!]$

$[\![$`T ::` γ `relates` α `and` $\beta]\!]$

[11] A diferenciação semântica entre as três categorias de relações apresentadas está na definição das entidades a partir da Metateoria de *Hypertypes*, em que `and` é usado para indicar relações associativas (`associativeRelationship`), `to` é usado para indicar relações direcionais (`directedRelationship`) e `partof` é usado para indicar relações de agregação (`aggregationRelationship`).

$[\![$ T :: γ extends [S_1, . . . , S_n] relates α and β $]\!]$

$\Longrightarrow$

$[\![$ T extends [S_1, . . . , S_n] $]\!]$

$[\![$ T :: γ relates α and β $]\!]$

Um último caso opcional é o que permite indicar uma meta-propriedade específica, no caso, a propriedade nonshared:

$[\![$ T :: nonshared M relates α and β $]\!]$

$\Longrightarrow$

$[\![$ meta nonshared on codomain at T $]\!]$

$[\![$ T :: M relates α and β $]\!]$

Uma vez os casos opcionais analisados, o caso padrão é traduzido como:

$[\![$ T :: M relates α and β $]\!]$

$\Longrightarrow$

$[\![$ T :: M $]\!]$

$[\![$ domain at T := $[\![\alpha]\!]_{twc}$ $]\!]$

$[\![$ codomain at T := $[\![\beta]\!]_{twc}$ $]\!]$

$[\![$ domain_cc at T := $[\![\alpha]\!]_{cc}$ $]\!]$

$[\![$ codomain_cc at T := $[\![\beta]\!]_{cc}$ $]\!]$

$[\![$ [$[\![\alpha]\!]_{twc}$, $[\![\beta]\!]_{twc}$] $]\!]_{entity}$

Tal que $[\![\varphi]\!]_{twc}$ e $[\![\varphi]\!]_{cc}$, para $\varphi = \alpha$ ou $\varphi = \beta$, são definidos conforme seguem.

As definições de $[\![\varphi]\!]_{twc}$ são:

$[\![$ T $]\!]_{twc} \Longrightarrow [\![$ T $]\!]_t$

$[\![\psi$ T $]\!]_{twc} \Longrightarrow [\![$ T $]\!]_t$

se $[\![\psi]\!]_{cc}$ sucede.

Por fim, as definições de $[\![\varphi]\!]_{cc}$, que denotam informações quanto à restrição de cardinalidade das instâncias da relação:

$[\![\mathtt{T}]\!]_{cc}$ [a] $\Longrightarrow 1$

[a] Restrição padrão de cardinalidade.

$[\![\psi \texttt{ of T}]\!]_{cc} \Longrightarrow \psi$

se $\psi \in \{many, \mathbb{N}, many..many, \mathbb{N}..\mathbb{N}, many..\mathbb{N}, \mathbb{N}..many\}$

$[\![\psi \texttt{ instance of T}]\!]_{cc} \Longrightarrow [\![\psi \texttt{ of T}]\!]_{cc}$

$[\![\psi \texttt{ instances of T}]\!]_{cc} \Longrightarrow [\![\psi \texttt{ of T}]\!]_{cc}$

Nota 7.1 (Operador sintático `..` mantido na representação semântica). Pelas definições apresentas, na forma semântica, o operador `..`, a constante `many` e os números inteiros ($\mathbb{N}$) são traduzidos para equivalentes semânticos que são idênticos aos presentes na sintaxe. Porém, embora a representação seja idêntica, essencialmente tratam-se de elementos diferentes, devido à tradução. —

7.6 Expansões das relações semânticas

Como procedimento final do *Algoritmo de controle da tradução*, apresentado na subseção 7.3.3, a saída sem erros do algoritmo Σ, representada por um conjunto β de relações semânticas, é usado como base para que novas relações semânticas sejam deduzidas daquelas produzidas pela denotação imediata das sentenças sintáticas. Dessa forma, o conjunto $\mathcal{H}$ da estrutura $\mathcal{OT}$ (seção 6.2) será a união de β com as instâncias do universo de Herbrand dos predicados resultantes das expansões.

Um dos objetivos dessas expansões é realizar derivações custosas do ponto de vista de tempo e espaço computacional, no âmbito da fronteira entre tradução sintática e interpretação semântica da linguagem.

Opta-se por apresentar no Apêndice F o detalhamento das definições de $semantic\_expansion/1$ em notação de implementação em Prolog, ao invés de apresentar definições neste texto – como é feito na seção 6.4, por exemplo. Isso é realizado devido ao entendimento de que o código apresentado está próximo o suficiente da abordagem adotada neste texto. Entretanto, com uma explicação semi-formal, a expansão discutida nesta seção consiste em:

a) resolver a definição retratável $defeasible(dio(P, property))$ de β, a respeito da instância de propriedade apresentada na subseção 7.4.5, em que o sistema efetivamente adiciona $dio(P, property)$ à $\mathcal{H}$ caso β não contenha outra relação $dio(P, E)$ para qualquer E;

b) atribuir a semântica da padrão Power Type (Definição 5.7) – conforme definido na subseção 7.4.8, cuja definição sintática é apresentada na subseção 7.4.8 – com a distribuição da relação de subsunção sobre instâncias de entidades participantes do relação $pt/2$. O algoritmo adiciona à $\mathcal{H}$ instâncias de relações $deo(P, W)$ para cada ocorrência simultânea de $dio(P, M)$ e $pt(M, W)$ em β;

c) distribuir as meta-propriedades por meio da interpretação dos operadores `extensions of`, `instances of`, `direct instances of`, `transitive instances of` e `transitive direct instances of`, incorporados do universo sintático para o universo semântico, conforme descrito na subseção 7.4.7. A distribuição visa compartilhar as meta-propriedades sobre instâncias (`instances of`) ou sobre extensões (`extensions of`) de entidades de $\mathcal{C}$, quantificadas universalmente, em cada $dmo/2$ de $\mathcal{H}$ que contenha algum dos operadores de distribuição. Os casos possíveis de distribuição e a semântica atribuída a cada combinação é definida da seguinte forma, em que M é uma meta-propriedade, E é uma entidade, P é uma entidade que instancia o *hypertype property*, $\{E, P, R, S\} \in \mathcal{C}$, $M \in \mathcal{G}$, $deo\_hierarchy$ é apresentado na Definição formal 6.3 e $dio\_hierarchy/2$ na Definição formal 6.6. Para:

Entidades:

- $dmo(M,$ `extensions of` $E)$: acrescenta $dmo(M, R)$ à $\mathcal{H}$, em que:
  - R sucede para todo $deo\_hierarchy(R, E)$;
- $dmo(M,$ `instances of` $E)$: acrescenta $dmo(M, R)$ à $\mathcal{H}$, em que:
  - R sucede para $dio(R, E)$; ou
  - R sucede para $deo\_hierarchy(S, E) \wedge dio(R, S)$;
- $dmo(M,$ `direct instances of` $E)$: acrescenta $dmo(M, R)$ à $\mathcal{H}$, em que:
  - R sucede para $dio(R, E)$;
- $dmo(M,$ `transitive direct instances of` $E)$: acrescenta $dmo(M, R)$ à $\mathcal{H}$, em que:
  - R sucede para $dio\_hierarchy(R, E)$;
- $dmo(M,$ `transitive instances of` $E)$: acrescenta $dmo(M, R)$ à $\mathcal{H}$, em que:
  - R sucede para $dio\_hierarchy(R, E)$; ou
  - R sucede para $deo\_hierarchy(S, E) \wedge dio(R, S)$.

Propriedades:

- $dmo(M, P$ `at extensions of` $E)$: $dmo(M, at(P, R))$ à $\mathcal{H}$, em que:
  - R sucede para todo $deo\_hierarchy(R, E)$;
- $dmo(M, P$ `at instances of` $E)$: acrescenta $dmo(M, at(P, R))$ à $\mathcal{H}$, em que:
  - R sucede para $dio(R, E)$; ou
  - R sucede para $deo\_hierarchy(S, E) \wedge dio(R, S)$;

- $dmo(M, P$ `at direct instances of` $E)$: acrescenta $dmo(M, at(P, R))$ à $\mathcal{H}$, em que:
  - R sucede para $dio(R, E)$;
- $dmo(M, P$ `at transitive direct instances of` $E)$: acrescenta $dmo(M, at(P, R))$ à $\mathcal{H}$, em que:
  - R sucede para $dio\_hierarchy(R, E)$;
- $dmo(M, P$ `at transitive instances of` $E)$: acrescenta $dmo(M, at(P, R))$ à $\mathcal{H}$, em que:
  - R sucede para $dio\_hierarchy(R, E)$; ou
  - R sucede para $deo\_hierarchy(S, E) \wedge dio(R, S)$.

Por exemplo, uma sentença como `meta abstract on instances of sentavel.` denota que a propriedade `abstract` seja atribuída a toda instância da entidade `sentavel`;

d) criar duas instâncias do predicado $dd/1$, na forma $dd(D)$:

- uma que em D figuram todas as propriedades (entidades instâncias transitivas de *property*);
- e outra que em D constam a lista de todos os hypertypes (entidades que possua a meta-propriedade *hypertype*)

Trata-se de uma disjunção generalizada dessas entidades, de modo que nenhuma entidade pode instanciar simultaneamente duas propriedades nem dois *hypertypes*.

7.7 Exemplo de especificação em Ontoprolog-Base

O Código 8 contém exemplo de especificação Ontoprolog construída utilizando a sintaxe base desenvolvida neste capítulo. Trata-se de uma especificação de teoria construída como instância direta da Metateoria de *Hypertypes*. No exemplo, duas propriedades, `nome` e `data_morte`, são criadas como instâncias do *hypertype* `slotProperty`. Uma hierarquia de entidades é definida no âmbito de uma teoria de tipos: as entidades `criatura`, `pessoa`, `vivo` e `morto` são definidas como instâncias de `entity`, e relacionadas formalmente entre si por meio da relação subsunção. Além disso, `vivo` e `morto` são definidas como entidades disjuntas. As propriedades `nome` e `data_morte` são atribuídas, respectivamente, à `pessoa` e à `morto`. Com base nessa construção, define-se as noções de `pessoa_viva`, como sendo uma entidade que estende características de `pessoa` e de `vivo`, e de `pessoa_morta`, como extensão de `pessoa` e de `morto`. Por fim, define-se uma teoria de indivíduos, em que `lauro` é instância de `pessoa_viva`; e que `sinatra` é instância simultânea de `pessoa_morta` e de `vivo`. Um valor é atribuído à propriedade `nome` em `lauro`.

Código 8 – Especificação "Sinatra" com base na Metateoria de *Hypertypes*

```
:- include('../../../ontoprolog/ontoprolog').

% properties
[nome, data_morte] :: slotProperty.

% type theory
criatura :: type.
pessoa :: type extends criatura.

disjoint [vivo, morto] :: type extends criatura.

property nome on pessoa.
property data_morte on morto.

pessoa_viva :: type extends [pessoa, vivo].
pessoa_morta :: type extends [pessoa, morto].

% individual theory
lauro :: pessoa_viva.
nome at lauro := 'lauro cesar'.

sinatra :: [pessoa_morta, vivo].

:- otp_compile,
   check_semantics.
```

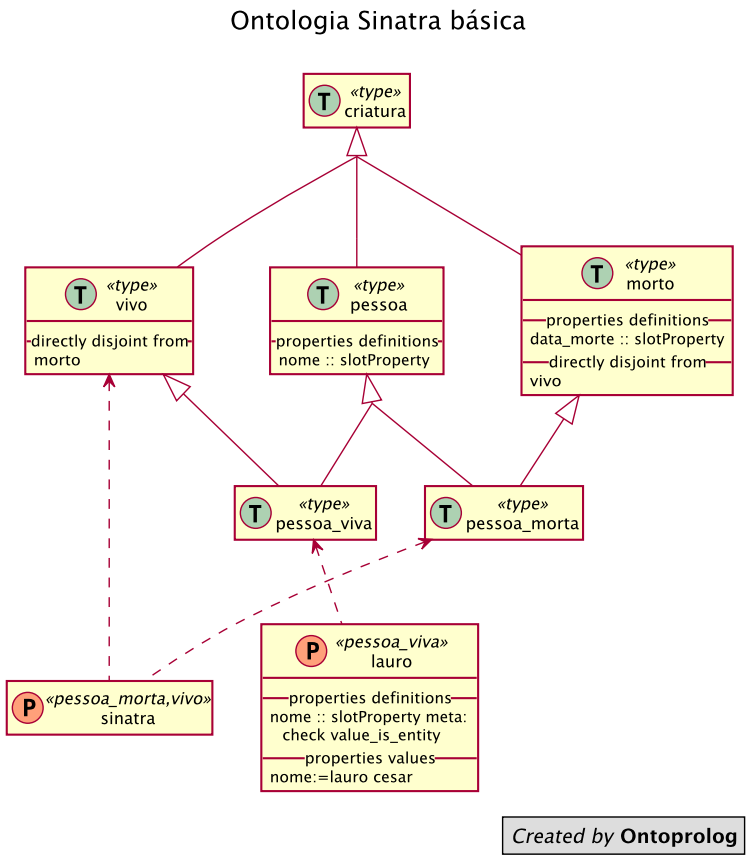
A Figura 34 contém uma ilustração em diagramas OntoUML dessa especificação. Os diagramas são criados conforme tradução descrita na seção 10.3, e constam nesta seção apenas com intuito de ilustrar a especificação textual.

O Código 9 contém a saída da avaliação semântica do Código 8. O predicado `opt_compile/0` consiste na execução do filtro de tradução (seção 7.3) e na produção das cláusulas que denotam as relações semânticas. O predicado `check_semantics/0`, por sua vez, aplica as RNP na Teoria produzida pelo filtro e verifica a noção de consistência do modelo conceitual. Como era esperado, o mecanismo de avaliação da Teoria denotada pela Especificação verifica que há uma inconsistência a respeito do conceito `sinatra`, capturado pela regra `dio_disjoint_types` (Definição formal 6.40). No caso, `sinatra` não pode instanciar simultaneamente os tipos `pessoa_morta` e `vivo`, por esses serem conceitos disjuntos. O resultado demonstra o mecanismo que define a noção de consistência no âmbito das teorias de Ontoprolog, conforme a seção 5.2.

Código 9 – Fragmento da saída da avaliação semântica do Código 8

```
Getting candidate sentences...
Starting Ontoprolog compilation...
Retracting current semantic terms...
Checking syntax of Ontoprolog candidate sentences...
Compiling sentences ...
  iteration 1...
All sentences are syntactically well-formed.

Asserting semantic terms produced by sentence decoding... 671
Asserting semantic expansions helpers...
   asserting semantic_expansion_dio_hierarchy/2...197
   asserting semantic_expansion_deo_hierarchy/2...375
Asserting semantic expansions...733
Asserting extensionof_irreflexive/2 relations...375
Asserting instanceof/2 relations...199
Asserting drefine/3 relations...8
Asserting propertyon/2 relations...80

Compilation of the Ontoprolog specification was successful.
```

Figura 34 – Diagramação da teoria especificada no Código 8



Fonte: Os autores

Nota: Diagrama criado conforme tradução descrita na seção 10.3

```
Checking semantics using negative rules of the current Ontoprolog theories...
-------------------------------------------------
Checking ngrule/3...
 sinatra:
   -> "sinatra" instantiates disjoint types "[pessoa_morta,vivo]".
    (dio_disjoint_types)

-------------------------------------------------
```

Com base na Especificação e na Teoria por ela denotada, é possível construir programas que utilizem ou forneçam informações a respeito de ontologias. Aplicações no âmbito de sistemas especialistas podem utilizar essa estrutura ontológica de descrição como insumo a decisões e avaliações de regras significativas em contextos específicos.

A seção 8.4 consiste em discussões e apresenta outros exemplos de especificações de ontologias e de aplicações de Ontoprolog. Além dessa seção, a subseção 8.4.3 contém o Código 11 que consiste na especificação de uma teoria a respeito das mesmas entidades contidas no Código 8, porém, construída com base na ontologia UFO, e não diretamente Metateoria de *Hypertypes*.

7.8 Fechamento

Pela definição sintática apresentada neste capítulo, em Ontoprolog não é possível afirmar a existência de qualquer entidade c sem mencionar essa entidade em alguma relação. Isso ocorre porque não há constructo que seja traduzido apenas na relação $entity/1$. Essa característica está alinhada com o objetivo primário de permitir descrever entidades apenas por meio de relações que essas entidades participam com outras. Além disso, devido ao paradigma de metamodelagem, é necessário que toda entidade participe como primeiro elemento na relação $dio/2$, na forma $dio(c, t)$, em que t é o tipo (ou a classe) de c ou que c seja marcado com a meta-propriedade *hypertype*, conforme característica induzida pela Definição formal 6.36 (*entity_without_hypertype*). Ou seja, sintaticamente, é necessário haver ao menos uma sentença na forma `c :: t`, ou na forma `meta hypertype on c`, conforme especificação da Metateoria de *Hypertypes* no Apêndice A.

Ontoprolog não pode ser considerada um DSL em sentido estrito, porque não é construída para um domínio específico, uma vez que é potencialmente útil para representação do conhecimento em diferentes domínios. Porém, ela é construída para um propósito específico, ao contrário de linguagens de propósitos gerais, como é o caso de C, Java, Haskell e Prolog. No caso, o propósito de Ontoprolog é o de descrever ontologias de forma bem-fundamentada. De toda forma, ela pode ser compreendida como uma DSL no sentido de que é usada num domínio específico do trabalho dos arquitetos da informação, ou de ontologistas, que usam alguma ontologia de fundamentação como ontologia subjacente aos seus discursos.

8 Especificação da UFO em Ontoprolog

Este capítulo descreve a instanciação e a extensão do arcabouço desenvolvido nos capítulos anteriores de modo que a UFO seja definida como uma teoria em Ontoprolog com base nas definições e nas regras descritas no Capítulo 4 e no Glossário da UFO.

Trata-se de uma instanciação porque a Metateoria de *Hypertypes* (Definição 5.10) é usada como base para que os conceitos da UFO contidos na Figura 59 (Apêndice B) figurem como vocabulário da linguagem disponível a arquitetos da informação que utilizem o arcabouço instrumental desenvolvido nesta pesquisa. Também trata-se de uma extensão porque a sintaxe base descrita no Capítulo 7, e as RNP do Capítulo 6, são ampliados de modo a comportar novos açúcares sintáticos que otimizam a experiência de uso de Ontoprolog e de UFO, bem como novas regras de inferência e de restrições de validade de modelos.

Com esses objetivos, o restante deste capítulo é organizado de forma a descrever como a UFO pode é definida em Ontoprolog, e como o arcabouço é estendido: a seção 8.1 detalha a apresentação da especificação da UFO em Ontoprolog, introduzida na subseção 5.7.1, com foco na especificação dos Momentos Intrínsecos. O objetivo da seção é descrever parcialmente como a instanciação do arcabouço de Ontoprolog é realizado, e porque é adequada a extensão da sintaxe base. Na seção 8.2 apresenta-se a extensão sintática da linguagem proposta no Capítulo 7 e na seção 8.3 apresenta-se as definições axiomáticas específicas da UFO. Já a seção 8.4 consiste em discussões quanto às características da UFO presentes na semântica apresentada neste capítulo, com foco em alguns exemplos de uso do arcabouço. Destaca-se o exemplo estendido contido na subseção 8.4.8, que consiste em uma redefinição em Ontoprolog do modelo contido na tese de doutoramento do autor da UFO. Como de praxe, a última seção, seção 8.5, contém um breve fechamento do capítulo.

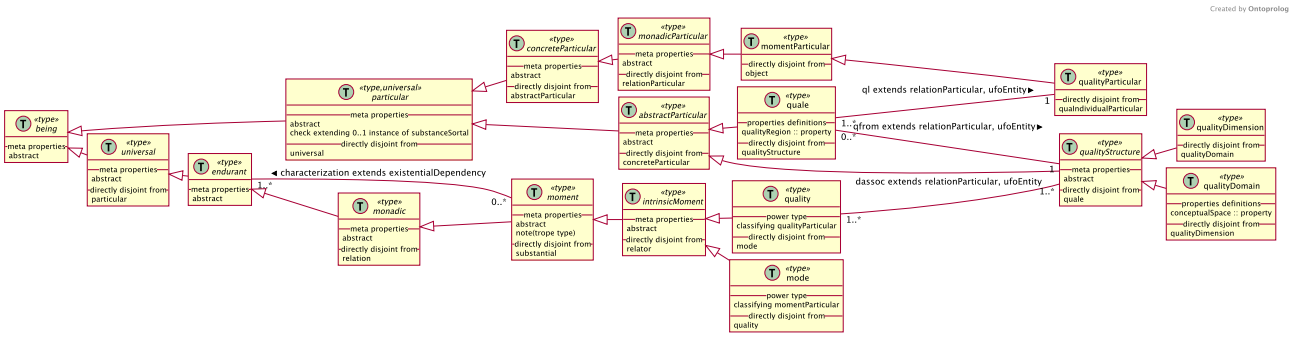
8.1 Definição de Momentos Intrínsecos

A especificação da UFO em Ontoprolog é realizada por um processo sistemático de interpretação dos conceitos do Glossário da UFO e das regras presentes nos quadros da subseção 4.5.1. A especificação apresentada no Apêndice C é realizada na sintaxe proposta no Capítulo 7 considerando modelos produzidos no Capítulo 6. Esta seção descreve uma parte dessa especificação, com o objetivo de descrever parcialmente como a instanciação do arcabouço de Ontoprolog é realizado para os Momentos Intrínsecos (Intrinsic Moment). A escolha da apresentação dos Momentos Intrínsecos é justificada por representarem parcela significativa da especificação da UFO em Ontoprolog. De toda forma, uma discussão a respeito dos Momentos extrínsecos (Extrinsic Moment), complementar a que é apresentada

nesta seção, é realizada na subseção 8.4.7.

A Figura 35 contém ilustração em UML do fragmento da especificação em Ontoprolog da UFO-A referente aos Momentos Intrínsecos (Intrinsic Moment). A figura é produzida com base na tradução descrita na seção 10.3.

Figura 35 – Intrinsic Moments implement in Ontoprolog



Fonte: Os autores

Instâncias diretas ou indiretas de `universal` são chamadas conceitos universais, enquanto instâncias diretas ou indiretas de `particular` são chamadas conceitos particulares. No âmbito da UFO-A, instâncias de conceitos particulares, ou seja, instâncias de instâncias de `particular`, não são autorizados, pois não possuem significado ontológico nessa teoria.

As entidades instanciáveis que subsumem `universal` são *power types* (Definição 5.7) que classificam um conceito que subsume `particular`. Logo, os conceitos particulares são formalmente definidos como instâncias de conceitos universais. Por exemplo, a entidade `quality` é definida como um *power type* que classifica o conceito `qualityParticular` (ver a seção *power type* no retângulo que representa o conceito em foco). Logo, toda instância de `quality` será assumida em Ontoprolog como subsunção de `qualityParticular`. De forma intuitiva, instâncias de `quality` são conceitos universais[1], uma vez que são instância (indireta) de `universal`. As instâncias de Quality, por sua vez, são os Quality Universal. Instâncias dos Quality Universal são particulares, uma vez que serão instâncias (indiretas) do conceito `qualityParticular` (que é subsunção de `particular`), assumido pela instância do *power type*.

Os `moment`, dos quais `quality` e `mode` são subsunção, participam da relação `characterization`, que vincula, no nível de universais, o conceito de `moment` a um ou mais `endurant`. Instâncias da relação `characterization` são entidades relacionais universais, que denotam o conceito de Caracterização entre Momentos Universais e outros Endurantes. Já instâncias de um Caracterização universal é chamada de Inherence, e denota a relação que ocorre entre dois particulares: um Moment particular e outro Endurante particular.

[1] A "forma intuitiva", no caso, pressupõe a existência de um fecho transitivo da relação de subsunção, simbolizado pela seta aberta, que aponta para o conceito mais geral.

Nesse caso, uma regra específica[2] de tradução do operador sintático que denota caracterização garante que um Moment particular não participe mais de uma vez de uma relação Inherence, ou seja, que um moment particular não seja dependente de mais do que um Endurante.

Lendo a Figura da direita para a esquerda, os quadros `qualityDimension` e `qualityDomain` são subsunções de `qualityStructure`, conforme é comumente apresentado na literatura sobre a UFO (Quality Structure, Quality Domain, Quality Dimension). A distinção lógica entre os dois conceitos refere-se ao fato de que os `qualityDomain` possuem uma propriedade chamada `conceptualSpace`, que contém a estrutura lógica do Espaço de qualidade composto por *dimensões* que ele denota. Nesse caso, o conceito de espaço de qualidade é tratado de forma livre, de modo que qualquer representação lógica é aceitável como instância dessa propriedade, embora ontologicamente ela deva ser interpretada conforme apresentado na subseção 8.4.4. A presença da propriedade `conceptualSpace` em `qualityDomain`, e não em `qualityStructure`, é uma simplificação conceitual que visa indicar que os Quality Domains são compostos por Espaços Conceituais que devem ser descritos por meios lógicos, enquanto nos Quality Dimensions o Espaço Conceitual já é delimitado como unidimensional e não-composto, por isso, prescinde de ser detalhado.

Os `qualityStructure` e os `quale` são subsunções de `abstractParticular`. Os primeiros mantém uma Relação de Associação (`dassoc`) com um ou mais universais de `quality`. Essa relação denota que Quality Universals estarão associados a um ou mais instâncias de Quality Structure. Os `qualityParticular`, que denotam as instâncias de Quality Universals, por outro lado, serão relacionados via relação `ql` (*quale of a quality*) a um ou mais `quale`, e esses serão vinculados, pela relação `qfrom`, a exatamente um `qualityStructure`. A relação `qfrom` indica qual é a `qualityStructure` da qual a propriedade `qualityRegion` é um um valor (ou uma observação aproximada). Ou seja, um `quale` só pode ser referente a um `qualityStructure`, embora o `quality` possa ter como valor diferentes `quale`.

A modelagem de Quality Region como uma propriedade de Quale é uma abstração da modelagem detalhada das Quality Region da UFO proposta por Albuquerque e Guizzardi (2013). No caso, a propriedade indica o valor aproximado (a aproximação) do Qualia de um Quality específico, "para o propósito de referência e comunicação", em respeito ao Quality Region denotado pelo Quality Space do Quality Structure associado ao Quality, conforme é discutido na subseção 4.4.4. Por isso, o valor da propriedade `qualityRegion` de `quale` contém a representação léxica da observação específica da qualidade em relação a um `qualityStructure`.

É interessante observar que instâncias de `qualityStructure` e de `quale`, como subsunção de `abstractParticular`, não são conceitos universais, mas particulares que

[2] A regra é descrita na subseção 8.2.2.8.

não possuem contrapartida universal nessa formalização da UFO.

Embora as relações de instanciação de entidades monádicas e de entidades diádicas possam ser expressas normalmente em Ontoprolog, é conveniente que existam formas sintaticamente naturais de expressar as relações complexas que existem entre os Momentos Intrínsecos, especialmente as relações entre `qualityStructure`, `quality` e `quale`. Por isso, a definição da sintaxe de Ontoprolog apresentada no Capítulo 8 é estendida de modo que se possa otimizar a linguagem a ser usada durante a formalização de discursos sobre ontologias de domínio com base na UFO em Ontoprolog. A extensão visa prover açúcares sintáticos referentes à expressão das relações de Characterization, Inherence, Quality, Conceptual Space, Quale, além de outras entidades de conceitos correlatos, conforme abordado na seção 8.2.

8.2 Otimização da sintaxe

A definição da sintaxe de Ontoprolog apresentada no Capítulo 7 é estendida de modo que se possa otimizar a linguagem a ser usada pelos arquitetos da informação durante a formalização dos discursos sobre a ontologia de domínio. A extensão visa prover açúcares sintáticos referentes a três categorias específicas expressões baseadas na UFO:

a) expressão das relações de Caracterização (Characterization) e de Inerência (Inherence), e consequentemente, conceitos como Bearer of a Moment e Exemplification;

b) expressão de Qualidades (Quality), Espaços Conceituais (Conceptual Space), Quales e outras entidades de conceitos correlatos;

c) expressão de Relators, Qua Individuals e relações parte-todo envolvidas.

A extensão apresentada não é essencial, exceto no caso da redefinição da Função *entity* (Definição 7.6), que passa a reconhecer como entidades constructos que denotem Qua Individuals, na forma `E qua R in O`, em que `E` denota uma entidade, `R` uma instância de Role (Quadro 7) e `O` de um Relator (Quadro 18), todos entidades de $\mathcal{C}$ da estrutura $\mathcal{OT}$ (seção 6.2). Dessa forma, uma cláusula $in(O, qua(E, R))$ denota uma entidade (qualificada em $entity/1$), igualmente pertencente à $\mathcal{C}$. A redefinição de Função *entity* é detalhada na subseção 8.2.2.1. À parte dessa ampliação à Função *entity*, todas as demais definições são meros açúcares sintáticos embutidos na linguagem base, que visam otimizar a experiência dos arquitetos, e também demonstrar o modo como a ferramenta construída nos capítulos anteriores pode ser estendida.

8.2.1 Extensão das definições em EBNF

A exemplo do que é apresentado na seção 7.2, o Código 10 contém a definição em EBNF da extensão à linguagem base de Ontoprolog. As definições em *railroad* da sintaxe são apresentas na subseção 8.2.2 em conjunto com as definições do *filtro de tradução*. O conteúdo do Código 10 é uma extensão ao conteúdo do Código 6.

Código 10 – Definição EBNF da sintaxe estendida de Ontoprolog para UFO

```
/* Changes on base syntax */

SENTENCE ::=
      ( ENTITY_QUALITY_ASSOC
        | QUALE_PARTICULAR
        | MONADIC_PARTICULAR
        | CHARACTERIZATION
        | INHERENCE
        | RELATOR_TYPE_BINARY_RELATION
        | RELATOR_PARTICULAR_BINARY_RELATION
      ) '.'

/* ENTITIES */

INSTANTIABLE_ENTITY ::=
      ATOM
      | 'qualityDomain' '=>' 'space' QUALITY_SPACE
      | 'quality' QUALITY_ASSOC

ENTITY_WITH_PROPERTY ::=
      ATOM
      | ATOM 'qua' ATOM
      | ATOM 'qua' ATOM 'in' ATOM

ENTITY_IN_RELATION ::=
      ATOM
      | ATOM 'qua' ATOM 'in' ATOM

/* QUALITY RELATED */

ENTITY_QUALITY_ASSOC ::=
      ATOM QUALITY_ASSOC

QUALITY_ASSOC ::=
      '=>'
      ( ATOM
        | ATOM '::' ('qualityDimension' | 'qualityDomain' '=>' 'space'
    QUALITY_SPACE)
        | 'space' QUALITY_SPACE )

QUALITY_SPACE ::=
      '[' FUNCTION_NAME ( ',' (FUNCTION_NAME | FUNCTION_NAME '=>' QUALITY_SPACE) )*
    ']'

FUNCTION_NAME ::= ATOM

/* MONADIC PARTICULARS */

QUALE_PARTICULAR ::=
      ATOM '::' 'quale' QUALE_VALUE 'from' ATOM

QUALE_VALUE ::=
      ATOMIC_OR_ATOMIC_LIST
      | '[' FUNCTION_NAME '<=' QUALE_VALUE  ( ',' FUNCTION_NAME '<=' QUALE_VALUE )*
    ']'

ATOMIC_OR_ATOMIC_LIST ::=
      ATOM
      | '[ ]'
      | '[' ATOMIC_OR_LIST ']'

MONADIC_PARTICULAR ::=
      ( ATOM | '[' ATOM (',' ATOM )* ']' )
      '::' ( ATOM | '[' ATOM (',' ATOM )* ']' ) WITH_QUALITY_STRUCTURE_QUALE

WITH_QUALITY_STRUCTURE_QUALE ::=
      '<=' ( ATOM | 'quale' QUALE_VALUE ('from' ATOM)? )
```

```
/* BEARER */

CHARACTERIZATION ::=
      (BEARING_TYPE | '[' BEARING_TYPE (',' BEARING_TYPE)* ']')
      ('characterizes' | 'characterize')
      ( CHARACTERIZED_TYPE
        | '[' CHARACTERIZED_TYPE (',' CHARACTERIZED_TYPE)* ']'
      )

CHARACTERIZED_TYPE ::=
      ATOM ( 'qua' ATOM 'in' ATOM
                  | '::' ( INSTANTIABLE_ENTITY | '[' ATOM (',' ATOM )* ']' )
                  | QUALITY_ASSOC
             )?

BEARING_TYPE ::=
      (CARDINALITY_CONSTRAINT ('instance' | 'instances')? 'of')?
      'immutable'?
      ATOM ( ( ('::' 'quality')? QUALITY_ASSOC )
                  | ('::' 'mode')
                  | ('qua' ATOM 'in' ATOM
                  )
              )?

INHERENCE ::=
      ( INTRINSIC_MOMENT_PARTICULAR | '[' INTRINSIC_MOMENT_PARTICULAR ( ','
    INTRINSIC_MOMENT_PARTICULAR )* ']' )
      ( 'inheres' | 'inheresin' | 'inhere' | 'inherein')
      ATOM ( 'qua' ATOM 'in' ATOM
                | '::' (ATOM | '[' ATOM (',' ATOM)* ']') WITH_QUALITY_STRUCTURE_QUALE
             )?

INTRINSIC_MOMENT_PARTICULAR ::=
      ( ATOM ('qua' ATOM 'in' ATOM)?
        | ( ( ATOM | '[' ATOM (',' ATOM )* ']' )
            '::' ( ATOM | '[' ATOM (',' ATOM )* ']' )
    WITH_QUALITY_STRUCTURE_QUALE?
          )
      )

/* RELATOR TYPE */

RELATOR_TYPE_BINARY_RELATION ::=
      ATOM  '::' 'relator' ('extends' ATOM)? 'mediates'
      RELATOR_TYPE_MEDIATED 'and' RELATOR_TYPE_MEDIATED
      ('deriving' ATOM)?

RELATOR_TYPE_MEDIATED ::=
      (CARDINALITY_CONSTRAINT ('instance' | 'instances')? 'of')? ATOM ('bearing'
    CARDINALITY_CONSTRAINT ('instance' | 'instances')? 'of' 'this')?

/* RELATOR PARTICULAR */

RELATOR_PARTICULAR_BINARY_RELATION ::=
      ATOM  '::' ATOM
      'mediates'
            (ATOM | ('[' ATOM (',' ATOM)* ']')) 'qua' ATOM
            'and' (ATOM | ('[' ATOM (',' ATOM)* ']')) 'qua' ATOM
      ('deriving' ATOM)?
```

8.2.2 Extensão do filtro de tradução

A exemplo do que é realizado na seção 7.3 para a sintaxe base, esta seção apresenta a tradução da sintaxe estendida de Ontoprolog em relações semânticas da estrutura $\mathcal{OT}$.

8.2.2.1 Extensão da Função *entity* (Definição 7.6)

Para comportar a estrutura de Qua Individuals, a Função *entity* (Definição 7.6) é estendida para que $[\![\mathsf{T}]\!]_{et}$ também suceda no seguinte caso, em que O denota uma entidade que é instância (ou instância de uma instância) de um Relator, E é uma entidade e R é

uma instância (ou instância de uma instância) de um Role:

$$[\![\texttt{E qua R in O}]\!]_{et} \Longrightarrow in([\![\texttt{O}]\!]_t, qua([\![\texttt{E}]\!]_t, [\![\texttt{R}]\!]_t))$$

Uma forma usada para denotar Qua Individuals instâncias de Universal, independentemente de qual Relator que participam, para que sirvam como gêneros de Qua Individuals específicos de determinados tipos de Relator, é definida como:

$$[\![\texttt{E qua R}]\!]_{et} \Longrightarrow qua([\![\texttt{E}]\!]_t, [\![\texttt{R}]\!]_t)$$

Com a definição, Qua Individuals são tratados como entidades de primeira classe, em que $in/2$ e $qua/2$ reificam a especificação da entidade em cláusulas no universo semântico. Por isso, $in/2$ e $qua/2$ são membros de $\mathcal{C}$ da estrutura $\mathcal{OT}$, embora não sejam *relações semânticas* no sentido apresentado no Capítulo 6. Essa forma é usada na subseção 8.2.2.10

8.2.2.2 Operadores principais da sintaxe estendida

A Tabela 8 contém a lista de operadores principais da sintaxe estendida de Ontoprolog. Esta tabela complementa a Tabela 6 e a Tabela 7. No entanto, a coluna "Predicados/relações", presentes naquelas tabelas, está ausente nesta, porque todos os constructos sintáticos são meras composições de um ou mais dos constructos bases.

Tabela 8 – Operadores principais da sintaxe estendida

Operador principal	Tipo	Leitura natural	Seções (grupo de sentenças)
`deriving`	infixo, assoc. à direita	φ *media e deriva* ψ	8.2.2.10 8.2.2.11
`mediates`	infixo, assoc. à direita	φ *media* ψ	8.2.2.10 8.2.2.11
`characterizes` `characterize`	infixo, assoc. à esquerda	φ *possui/possuem* ψ	8.2.2.8

`inheres`
`inheresin`
`inhere`
`inherein`
	infixo, assoc. à esquerda	φ *possui/possuem* ψ	8.2.2.9

`=>`[3]	infixo, assoc. à direita	φ *associado à estrutura de qualidade* ou *associado ao espaço conceitual* ψ	8.2.2.5

Fonte: Produzido pelos autores.

[3] `=>` e `::` possuem a mesma prioridade. Portanto, a associatividade dos operadores é utilizada para desambiguizar os referentes com base na ordem em que aparecem na sentença.

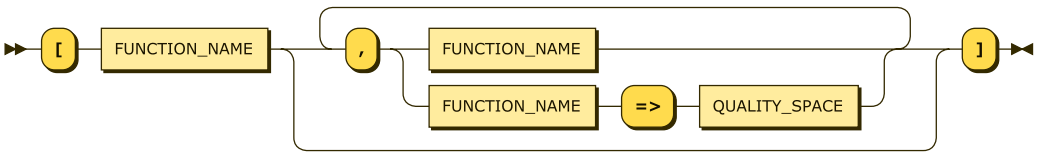
8.2.2.3 Espaços de qualidade

Definição 8.1 (Espaço conceitual)**.** Os espaços conceituais, ou espaços de qualidade, são estruturas abstratas e formais, compostas por Quality Structures e por restrições formais entre seus elementos. A sintaxe de Conceptual Space é definida conforme a Figura 36.

$[\![\mathsf{S}]\!]_{qs} \Longrightarrow \mathsf{S}$

se S possui a forma representada na Figura 36, em que a definição de `FUNCTION_NAME` é apresentada na Definição 8.2.

Figura 36 – Diagrama *railroad* de Espaço conceitual de qualidade

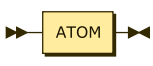


Fonte: Os autores

Definição 8.2 (Construtor de nome de função de atribuição)**.** O construtor de nome de função de atribuição, apresentado pelo diagrama EBNF da Figura 37, é definido com $[\![\mathsf{F}]\!]_t$:

$[\![\mathsf{F}]\!]_f \Longrightarrow [\![\mathsf{F}]\!]_t$

Figura 37 – Diagrama *railroad* de Nomes de funções



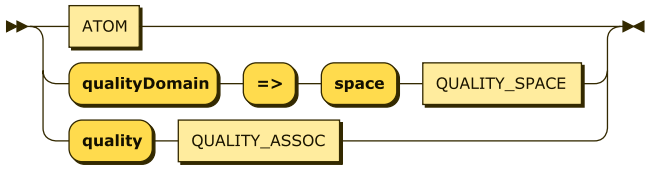
Fonte: Os autores

As definições de restrições sobre os espaços de qualidade são descritas pelas subseção 8.4.4.

8.2.2.4 Redefinição de INSTANTIABLE_ENTITY

Essa definição visa embutir em Ontoprolog a possibilidade de declaração de Espaços Conceituais (Conceptual Space) relacionados a uma Estrutura de Qualidade (Quality Structure), no caso, `qualityDomain` (Quality Domain). A definição de `INSTANTIABLE_ENTITY`, apresentada na subseção 7.4.4, é expandida e passa a contemplar além do caso base `ATOM`, os casos seguintes, em que `QUALITY_ASSOC` é definido na subseção 8.2.2.5:

INSTANTIABLE_ENTITY:



A semântica dessa sintaxe é dada conforme segue, em que a sintaxe é transformada em duas sentenças: a primeira que denota a instância e a outra que denota o valor da propriedade `conceptualSpace`. O construtor `space` é usado para denotar o Conceptual Space relacionado à Quality Structure.

$$[\![\texttt{T :: qualityDomain => space S}]\!]$$

$$\Longrightarrow$$

$$[\![\texttt{T :: qualityDomain}]\!]$$

$$[\![\texttt{conceptualSpace at T := } [\![\texttt{S}]\!]_{qs}]\!]$$

Além do caso da declaração explícita de instância de `qualityDomain`, o outro caso de açúcar sintático é um caso geral que permite diferentes formas de declaração da Relação de Associação (Association Relation) entre uma Qualidade (Quality) e uma Quality Structure. A semântica desta sintaxe é analisada na subseção 8.2.2.5. O caso tratado nesta seção é o açúcar sintático que permite a declaração simultânea de instâncias de Qualities de associações com estruturas de qualidade.

$$[\![\texttt{T :: quality => S}]\!]$$

$$\Longrightarrow$$

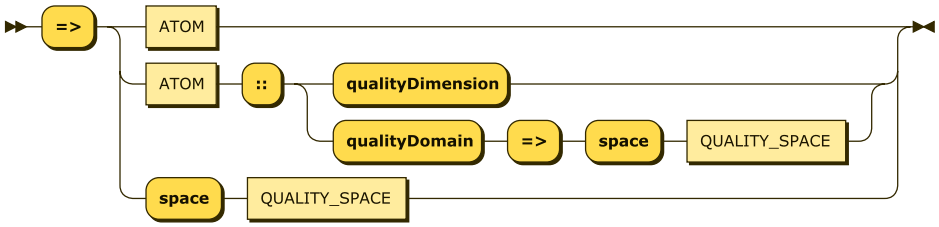
$$[\![\texttt{T :: quality}]\!]$$

$$[\![\texttt{T => S}]\!]$$

8.2.2.5 Associação de estrutura de qualidade

ENTITY_QUALITY_ASSOC:

QUALITY_ASSOC:



Trata-se de uma sintaxe especial que denota que uma entidade `T` está associada a uma Estrutura de Qualidade (Quality Structure) D. As Estruturas de Qualidade são representadas por instâncias de `qualityDimension` (Quality Dimension) ou de `qualityDomain` (Quality Domain). Porém, no âmbito da semântica apresentada nesta seção, nenhuma exigência sobre D ser ou não instância de uma estrutura de qualidade é verificada.

Sentenças deste tipo podem conter dois operadores com a mesma prioridade, uma vez que `::` e `=>` possuem a mesma prioridade de precedência (Tabela 8). Nesse caso, é necessário utilizar a ordem em que os operadores aparecem na sentença para identificar as relações denotadas. A associatividade à direita dos operadores garante que o operador mais à esquerda, ou seja, o que primeiro aparece, seja o operador principal da sentença.

O caso padrão é o que relaciona uma Quality Q a uma Quality Structure D. Se nada mais for dito a respeito dos tipos que Q e de D instanciam, o sistema assume que se tratam de instâncias de `quality` e `qualityDomain`, respectivamente. Trata-se de inferência retratável, realizada por raciocínio com regras padrão, conforme a subseção 8.2.3.

$[\![$`Q => D`$]\!]$

$\Longrightarrow$

$[\![$`A :: dassoc relates Q and D`$]\!]$

$[\![$`meta abstract on A`$]\!]$

$defeasible(dio([\![\texttt{Q}]\!]_t, quality))$

$defeasible(dio([\![\texttt{D}]\!]_t, qualityDomain))$

em que `A` é a concatenação de $[\![\texttt{Q}]\!]_t$, "_assoc_" e de $[\![\texttt{D}]\!]_t$.

Um caso de açúcar sintático é o que permite a declaração de uma instância de uma estrutura de qualidade (`qualityDimension` ou `qualityDomain`) à direita do operador `=>`. Nesse caso, a semântica é dada como:

$[\![$`T => D :: qualityDimension`$]\!]$

$\Longrightarrow$

$[\![$`T => D`$]\!]$

$[\![$`D :: qualityDimension`$]\!]$

$[\![$`T => D :: qualityDomain => space S`$]\!]$

$\Longrightarrow$

$[\![$`T => D`$]\!]$

$[\![$`D :: qualityDomain => space S`$]\!]$

Pelas regras da UFO (Quadro 13), as Qualities estão sempre relacionadas a pelo menos uma Quality Structure; e os Quality Domains sempre estão relacionados a exatamente um Conceptual Space (Quadro 15) e a pelo menos uma Quality. Dessa forma, assume-se que sempre que uma `quality` estiver associada a um `qualityDomain`, ela está indiretamente associada a um Conceptual Space. Com isso, o caso seguinte permite a omissão da declaração do `qualityDomain` de modo que ele seja existencialmente derivado da declaração de uma associação de Estrutura de Qualidade. Ou seja, trata-se de um caso de açúcar sintático que permite que a Estrutura de Qualidade seja omitida no momento da declaração da associação da Quality. Nesse caso, uma instância de `qualityDomain` é automaticamente deduzida, devido às regras semânticas de Association Relation, estabelecidas no Quadro 13. Opta-se por adicionar o açúcar sintático na declaração de `quality`, e não de `qualityDomain`, porque é mais natural afirmar diretamente qual é o Conceptual Space vinculado a uma Quality do que afirmar qual a Quality vinculado a uma Quality Structure (no caso, `qualityDomain`). Os casos seguintes incorporam essa escolha e descrevem a tradução dos constructos sintáticos desta seção.

$[\![$`T => space S`$]\!]$

$\Longrightarrow$

$[\![[\![$`T`$]\!]_{pdid}$ `:: qualityDomain => space S`$]\!]$

$[\![$`T =>` $[\![$`T`$]\!]_{pdid}$$]\!]$

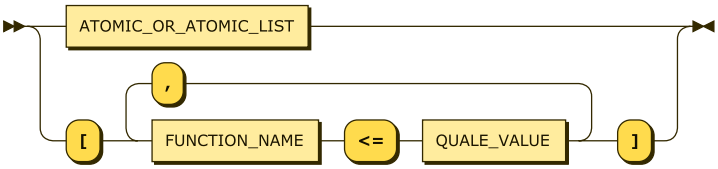
Em que $[\![$`T`$]\!]_{pdid}$ é definido conforme segue:

$[\![\mathtt{T}]\!]_{pdid} \Longrightarrow \text{concat}([\![\mathtt{T}]\!]_t, \text{“}\_qs\_qdomain\text{”})$, em que o nome de T é concatenado com “_qs_domain”, de modo a criar um novo identificador.

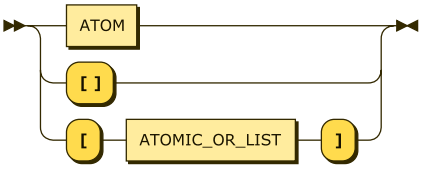
8.2.2.6 Instância de Quale

QUALE_PARTICULAR:

QUALE_VALUE:



ATOMIC_OR_ATOMIC_LIST:



Trata-se da sintaxe específica para a declaração de instâncias de Quales que são ontologicamente interpretados como regiões do Espaço Conceitual (Conceptual Space) denotado por uma Estrutura de Qualidade (Quality Structure).

A semântica da sintaxe é dada por:

$[\![\mathtt{Q :: quale\ V\ from\ D}]\!]$

$\Longrightarrow$

$[\![\mathtt{Q :: quale}]\!]$

$[\![\mathtt{QF :: qfrom\ relates\ Q\ and\ D}]\!]$

$[\![\mathtt{meta\ abstract\ on\ QF}]\!]$

$[\![\mathtt{qualityRegion\ at\ Q := V}]\!]$

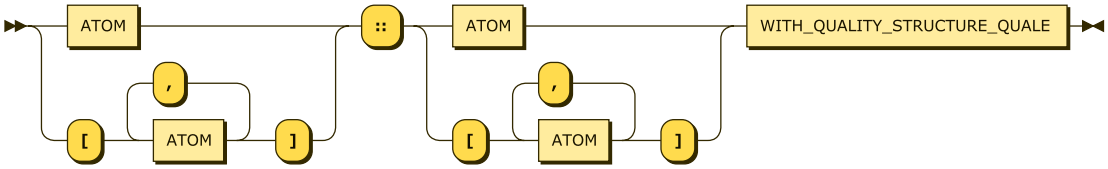
em que QF é a concatenação de $[\![\mathtt{Q}]\!]_t$, “_qfrom_” e de $[\![\mathtt{D}]\!]_t$.

Essa sintaxe denota que uma instância de Quale `Q` aponta para a região `V` de uma Quality Structure D.

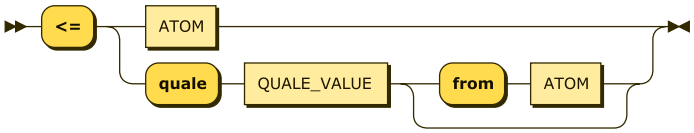
Uma implementação concreta desta sintaxe poderia avaliar a estrutura do constructo `QUALE_VALUE`, mas isso não é apresentado nesta seção, de modo que `V` é assumido como uma relação lógica qualquer.

8.2.2.7 Instância de tipo monádico

MONADIC_PARTICULAR:



WITH_QUALITY_STRUCTURE_QUALE:



Sentenças desta categoria estendem a definição de instância de tipo monádico apresentada na subseção 7.4.4 e na subseção 8.2.2.4 com a possibilidade de declarar, numa única sentença, instâncias de Quales (subseção 8.2.2.6). Tratam-se de casos em que o operador `<=` está presente. Nesses casos, as sentenças possuem uma das formas seguintes, em que γ denota uma entidade T ou uma lista de entidades T, para todo ψ conforme as próximas definições:

$[\![\texttt{P :: } \gamma \texttt{ <= } \psi]\!]$

ou

$[\![\texttt{[P}_1\texttt{, . . . , P}_n\texttt{] :: } \gamma \texttt{ <= } \psi]\!]$

Dessa forma, a semântica dada às sentenças é definida do seguinte modo. Inicialmente, elimina-se o caso da lista à esquerda, expandindo-o em uma sequência de sentenças sem listas:

$[\![\texttt{[P}_1\texttt{, . . . , P}_n\texttt{] :: } \gamma \texttt{ <= } \psi]\!]$

$\Longrightarrow$

$[\![\texttt{P}_1\ \texttt{::}\ \gamma\ \texttt{<=}\ \psi]\!]$

$\vdots$

$[\![\texttt{P}_n\ \texttt{::}\ \gamma\ \texttt{<=}\ \psi]\!]$

quando $[\![\texttt{P}_i]\!]_t$ sucede para todos os $1 \leq i \leq n$.

Com isso, aborda-se as formas possíveis que seguem o operador `<=`. O primeiro caso denota que a instância `O` de um Quale (subseção 8.2.2.6) contém como propriedade a região do Conceptual Space no qual a instância `Q` de uma Quality está associada. Nesse caso, `O` é o *Quale* de uma qualidade (Quale of a quality) de `Q`:

$[\![\texttt{Q}\ \texttt{::}\ \gamma\ \texttt{<=}\ \texttt{O}]\!]$

$\Longrightarrow$

$[\![\texttt{Q}\ \texttt{::}\ \gamma]\!]$

$[\![\texttt{QL :: ql relates O and Q}]\!]$

$[\![\texttt{meta abstract on QL}]\!]$

em que `QL` é a concatenação de $[\![\texttt{O}]\!]_t$, "_ql_" e de $[\![\texttt{Q}]\!]_t$.

O segundo caso é um açúcar sintático que permite que os valores de Quales sejam declarados diretamente, de modo que a instância do Quale seja inferida automaticamente. Nessa forma, há dois novos casos a serem avaliados. O primeiro deles é aquele em que se conhece a Quality Structure `D` origem do valor `V` do Quale `O` a ser criado:

$[\![\texttt{Q}\ \texttt{::}\ \gamma\ \texttt{<=}\ \texttt{quale V from D}]\!]$

$\Longrightarrow$

$[\![\texttt{Q}\ \texttt{::}\ \gamma\ \texttt{<=}\ \texttt{O}]\!]$

$[\![\texttt{O :: quale V from D}]\!]$

em que `O` é a concatenação de $[\![\texttt{Q}]\!]_t$, "_quale".

Já o outro é aquele em que a Quality Structure origem do valor `V` do Quale `O` a ser criado não é informada. Nesse caso, as regras semânticas tentarão inferir a Quality Structure associada à Quality `Q` dessa sintaxe. Porém, isso nem sempre será possível, uma vez que pode acontecer de `Q` estar associado a mais do que uma única Quality Structure (subseção 8.2.2.5). Essa questão é tratada pelo sistema descrito no Apêndice G.

⟦Q :: γ <= quale V⟧

$\Longrightarrow$

⟦Q :: γ <= O⟧

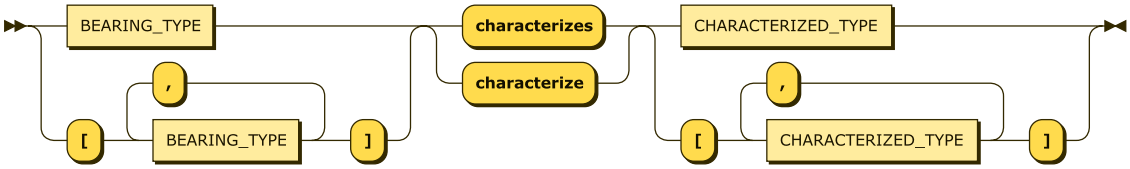
⟦O :: quale⟧

⟦qualityRegion at O := V⟧

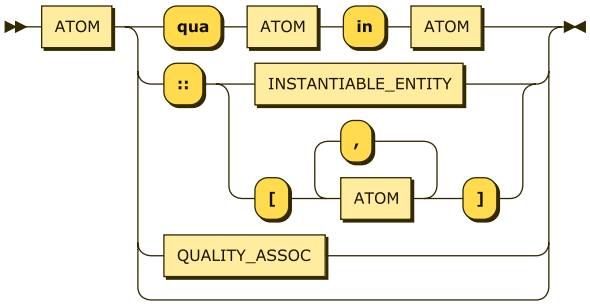
em que O é a concatenação de $[\![Q]\!]_t$, "_quale".

8.2.2.8 Caracterização

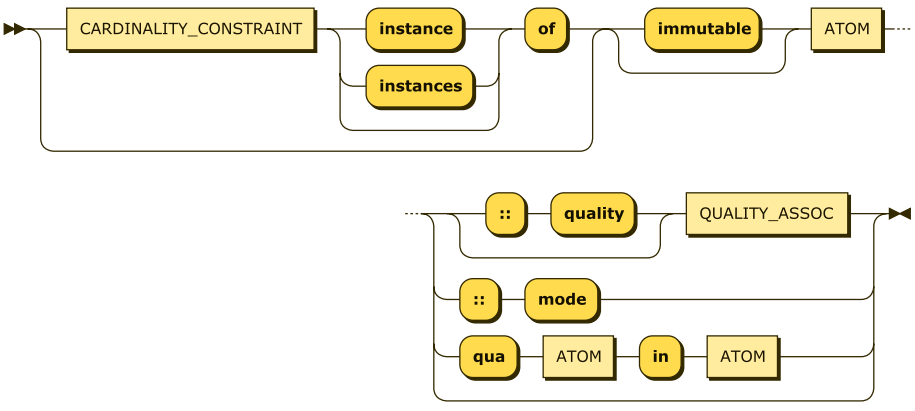
CHARACTERIZATION:



CHARACTERIZED_TYPE:



BEARING_TYPE:



Esta sintaxe visa explicitar a relação formal Caracterização (Characterization), em que determinado tipo é caracterizado por outro tipo Momento (Moment). O operador principal desta categoria sintática é o `characterizes` – e seu sinônimo `characterize`. A denotação semântica principal representa instâncias da entidade `characterization` (Apêndice C).

Uma característica desse constructo sintático é a riqueza de açúcares sintáticos, especialmente devido à combinação da relação de Caracterização com as relações de instância apresentadas na subseção 8.2.2.7.

A exemplo da subseção 7.4.2, a tradução apresentada nesta seção lança mão de esquemas sintáticos.

Devido à prioridade dos operadores `characterizes` e `characterize` (Tabela 8), o esquema padrão da relação em tela pode ser normalizado para apenas um dos operadores, em que φ são os construtores à esquerda do operador principal, e ψ os à direita:

$$[\![\varphi \texttt{ characterize } \psi\ ]\!]$$

$$\Longrightarrow$$

$$[\![\varphi \texttt{ characterizes } \psi]\!]$$

Com isso, apenas o caso `characterizes` é tratado. Inicialmente, elimina-se as listas à direita e à esquerda do operador principal:

$$[\![\texttt{[}\varphi_1\texttt{, . . . , }\varphi_n\texttt{] characterizes } \psi\ ]\!]$$

$$\Longrightarrow$$

$$[\![\varphi_1 \texttt{ characterizes } \psi]\!]$$

$$\vdots$$

$$[\![\varphi_n \texttt{ characterizes } \psi]\!]$$

$$[\![\varphi \texttt{ characterizes [}\psi_1\texttt{, . . . , }\psi_n\texttt{] }]\!]$$

$$\Longrightarrow$$

$$[\![\varphi \texttt{ characterizes } \psi_1]\!]$$

$$\vdots$$

$$[\![\varphi \texttt{ characterizes } \psi_n]\!]$$

$[\![[\varphi_1, \ldots, \varphi_n]$ characterizes $[\psi_1, \ldots, \psi_m]]\!]$

$\Longrightarrow$

$[\![\varphi_1$ characterizes $\psi_1]\!]$

$\vdots$

$[\![\varphi_n$ characterizes $\psi_1]\!]$

$\vdots$

$[\![\varphi_1$ characterizes $\psi_m]\!]$

$\vdots$

$[\![\varphi_n$ characterizes $\psi_m]\!]$

Uma vez as listas eliminadas, analisa-se os casos individuais, ou seja, os casos em que não há listas à direita ou à esquerda do operador principal, iniciando-se pelos casos de composição de operadores de instância e Estrutura de Qualidade à direita do operador principal.

$[\![\varphi$ characterizes $\psi]\!]$

$\Longrightarrow$

$[\![\alpha]\!]$

$[\![\varphi$ characterizes $\beta]\!]$

se, para qualquer φ, :

$$\psi = \begin{cases} \texttt{P :: } \gamma & \text{, então } \alpha = \texttt{P :: } \gamma \text{ e } \beta = \texttt{P}, \\ & \text{para } \gamma \text{ como uma lista de ATOM} \\ & \text{ou um caso de INSTANTIABLE\_ENTITY (subseção 8.2.2.4)} \\ \texttt{P => } \pi & \text{, então } \alpha = \texttt{P => } \pi \text{ e } \beta = \texttt{P}, \\ & \text{para } \pi \text{ como um caso de QUALITY\_ASSOC (subseção 8.2.2.5)} \end{cases}$$

Uma vez tratados os casos à direita do operador principal, analisa-se os casos de composição de operadores à esquerda referentes à relação de instância abordada na subseção 7.4.4 e o caso de Estrutura de Qualidade (Quality Structure), este descrito inicialmente na subseção 8.2.2.5:

$[\![\varphi$ `characterizes P`$]\!]$

$\Longrightarrow$

$[\![\alpha]\!]$
$[\![\beta$ `characterizes P`$]\!]$

se, para qualquer γ:

$$\varphi = \begin{cases} \gamma\ \texttt{P :: quality} & \text{, então } \alpha = \texttt{P :: quality} \text{ e } \beta = \gamma\ \texttt{P} \\ \gamma\ \texttt{P :: quality => } \pi & \text{, então } \alpha = \texttt{P :: quality => } \pi \text{ e } \beta = \gamma\ \texttt{P}, \\ & \text{para } \pi \text{ como um caso de} \\ & \texttt{QUALITY\_ASSOC} \text{ (subseção 8.2.2.5)} \\ \gamma\ \texttt{P :: mode} & \text{, então } \alpha = \texttt{P :: mode} \text{ e } \beta = \gamma\ \texttt{P} \end{cases}$$

Tratados os casos de açúcares sintáticos que combinam a relação de instância, normaliza-se as combinações de operadores que denotam cardinalidade da relação. A regra de tradução impõe que se nenhuma restrição de cardinalidade for informada, assume-se que se trata de 1 e apenas 1 instância da entidade relacionada:

$[\![\varphi$ `characterizes P`$]\!]$

$\Longrightarrow$

$[\![\alpha$ `characterizes P`$]\!]$

se, para π conforme a Definição 7.9 (Indicadores de restrição de cardinalidade):

$$\varphi = \begin{cases} \texttt{P} & \text{, então } \alpha = \texttt{1 instance of P} \\ \texttt{immutable P} & \text{, então } \alpha = \texttt{1 instance of immutable P} \\ \pi\ \texttt{instances of P} & \text{, então } \alpha = \pi\ \texttt{instance of P} \\ \pi\ \texttt{instances of immutable P} & \text{, então } \alpha = \pi\ \texttt{instance of immutable P} \end{cases}$$

Uma vez os casos de restrição de cardinalidade normalizados, apresenta-se a definição referente à declaração da meta-propriedade `immutable`, que denota o conceito de Parte Imutável (Immutable Part):

$[\![\pi$ `instance of immutable P characterizes T` $]\!]$

$\Longrightarrow$

$[\![$`meta immutable on C`$]\!]$

$[\![\pi$ `instance of P characterizes T` $]\!]$

em que `C` é a concatenação de $[\![$`P`$]\!]_{cqua}$, "_charac_" e de $[\![$`T`$]\!]_t$, em que $[\![$`P`$]\!]_{cqua}$ é definido conforme segue.

$[\![$`T`$]\!]_{cqua} \Longrightarrow [\![$`T`$]\!]_t$

$[\![$`T qua R on O`$]\!]_{cqua} \Longrightarrow$ `R`

em que `R` é a concatenação de $[\![$`T`$]\!]_t$, "_qua_", $[\![$`R`$]\!]_t$, "_in_" e $[\![$`O`$]\!]_t$.

Por fim, apresenta-se a definição final para o caso normalizado:

$[\![\pi$ `instance of P characterizes T` $]\!]$

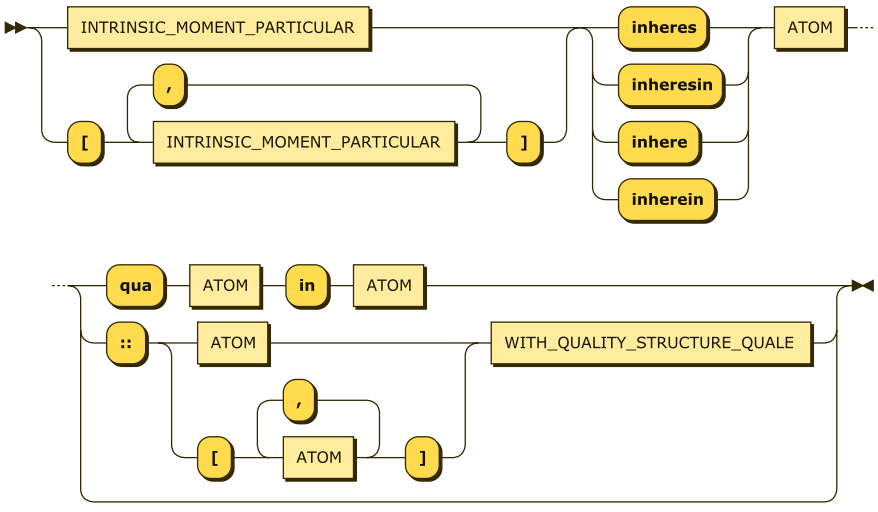
$\Longrightarrow$

$[\![$`C :: characterization relates` π `instances of P and 1 instance of T`$]\!]$

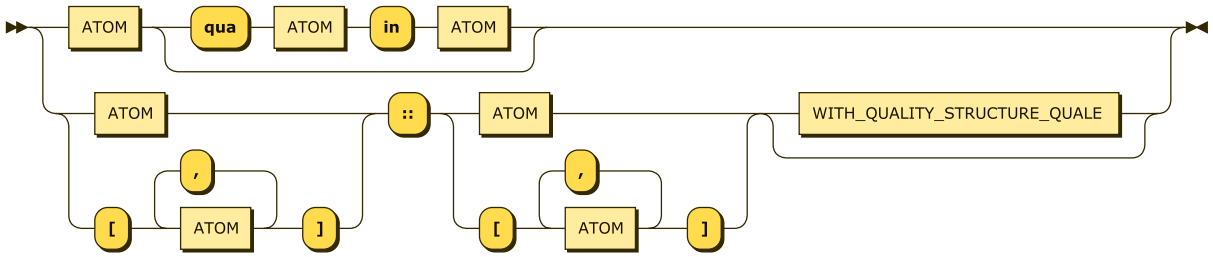
em que `C` é definido conforme já apresentado.

8.2.2.9 Instância de caracterização

INHERENCE:



INTRINSIC_MOMENT_PARTICULAR:



Os constructos `inheres`, `inheresin`, `inhere` e `inherein` são todos equivalentes e denotam instâncias da relação de caracterização detalhada na subseção 8.2.2.8. A tradução desta sintaxe usa o contexto Δ descrito na subseção 7.3.2. A tradução é definida da seguinte forma.

Inicialmente, normaliza-se os diferentes operadores sinônimos, reduzindo todos ao operador `inheres`:

$[\![\varphi \texttt{ inheresin } \psi]\!]$

$\Longrightarrow$

$[\![\varphi \texttt{ inheres } \psi]\!]$

$[\![\varphi \texttt{ inherein } \psi]\!]$

$\Longrightarrow$

$[\![\varphi \texttt{ inheres } \psi]\!]$

$[\![\varphi \texttt{ inhere } \psi]\!]$

$\Longrightarrow$

$[\![\varphi \texttt{ inheres } \psi]\!]$

Na sequência, avalia-se o caso da lista à esquerda, em que a lista é traduzida para uma sequência de sentenças com entidades unitárias à esquerda do predicado principal. À direita do operador mantém-se a combinação clausal ψ original, presente na sentença transcrita na mesma posição da sentença inicial:

$[\![$ `[`φ_1`, . . . ,` φ_n`]` `inheres` ψ $]\!]$

$\Longrightarrow$

$[\![\varphi_1$ `inheres` $\psi]\!]$

$\vdots$

$[\![\varphi_n$ `inheres` $\psi]\!]$

Em seguida, elimina-se o caso de açúcar sintático da declaração de instância à direita, para qualquer φ à esquerda:

$[\![\varphi$ `inheres T ::` ψ $]\!]$

$\Longrightarrow$

$[\![\varphi$ `inheres T`$]\!]$

$[\![$`T ::` $\psi]\!]$

Se ψ é `ATOM`, uma lista de `ATOM`, ou qualquer desses casos seguido da combinação sintática `WITH_QUALITY_STRUCTURE_QUALE` (subseção 8.2.2.7).

Na sequência, resolve-se o caso de açúcar sintático da declaração de instância à esquerda:

$[\![\varphi$ `inheres T` $]\!]$

$\Longrightarrow$

$[\![\alpha]\!]$

$[\![\beta]\!]$

se, para γ como `ATOM` ou uma lista de `ATOM`, ou qualquer desses casos seguidos de

WITH_QUALITY_STRUCTURE_QUALE (subseção 8.2.2.7):

$$\varphi = \begin{cases} \texttt{A :: } \gamma & \text{, então:} \\ & \alpha = \texttt{A :: } \gamma \\ & \beta = \texttt{A inheres T} \\ \texttt{[A}_1\texttt{, . . . , A}_n\texttt{] :: } \gamma & \text{, então:} \\ & \alpha = \texttt{[A}_1\texttt{, . . . , A}_n\texttt{] :: } \gamma \text{ e} \\ & \beta = \begin{cases} \texttt{A}_1 \texttt{ inheres T} \\ \vdots \\ \texttt{A}_n \texttt{ inheres T} \end{cases} \end{cases}$$

Por fim, apresenta-se a tradução do caso padrão:

⟦ A inheres B⟧

$\Longrightarrow$

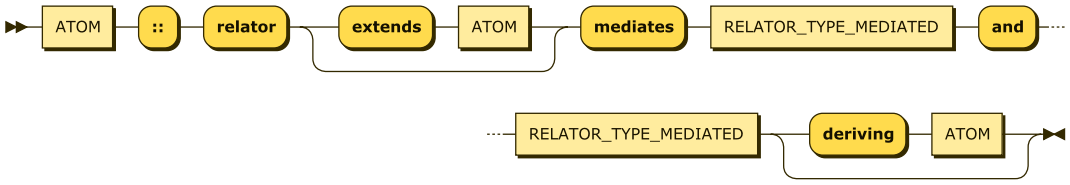
⟦I :: C relates A to B⟧

em que I é a concatenação de $[\![\texttt{A}]\!]_{cqua}$, "_inheresin_" e de $[\![\texttt{B}]\!]_{cqua}$, conforme a definição apresentada na subseção 8.2.2.8; e C refere-se a uma relação de caracterização presente no contexto, conforme a seguinte condição:

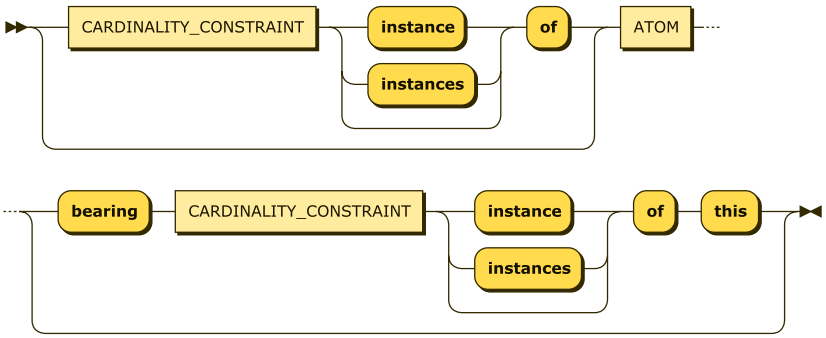
$$\begin{aligned} \exists\, C\ & (dio(C, characterization) \in \Delta \\ & \wedge\, dpv(at(domain, C), DT) \in \Delta \\ & \wedge\, dpv(at(codomain, C), OT) \in \Delta \\ & \wedge\, dio(DI, DT) \in \Delta \\ & \wedge\, dio(OI, OT) \in \Delta \\ & \wedge\, \neg(cextensionof\_irreflexive(S, C) \in \Delta \\ & \quad \wedge\, dpv(at(domain, S), DT) \in \Delta \\ & \quad \wedge\, dpv(at(codomain, S), OT) \in \Delta)) \end{aligned} \tag{8.1}$$

8.2.2.10 Relator universal

RELATOR_TYPE_BINARY_RELATION:



RELATOR_TYPE_MEDIATED:



Considerando apenas as sentenças bases de Ontoprolog, a expressão de Relators exigiria que diversas afirmações fossem realizadas, de modo a se descrever cada Relator conforme apresentado pela UFO. Porém, como Relators referem-se a contribuições importantes na área de modelagem conceitual, especialmente devido à análise do problema da contagem (subseção 4.4.7), esta sintaxe visa incorporar em Ontoprolog um método direto para formalização desses conceitos.

Com intuito de atingir certa simplicidade na apresentação, apenas a declaração de Relators que envolvem uma relação material entre duas entidades distintas são expostas. No entanto, uma sintaxe para Relators que envolvam mais do que duas entidades pode ser definida com base na estratégia apresentada nesta seção.

Os operadores principais `mediates` e `deriving` são usados para identificar sentenças que envolvam a declaração de universais de Relators. A sintaxe referente à declaração de instâncias de tipos desse conceito é abordada na subseção 8.2.2.11.

Resolve-se, inicialmente, os casos de açúcar sintático, de modo a se obter um caso normalizado. Os primeiros casos são os que o nome da relação material que relaciona as entidades participantes é omitido. Nesses casos, uma relação material é existencialmente derivada.

⟦R :: relator mediates φ and ψ⟧

$\Longrightarrow$

⟦R :: relator mediates φ and ψ deriving M⟧

Em que ψ não contém o operador `deriving`; e o identificador `M` é a concatenação de $[\![\texttt{A}]\!]_t$ e "_derived_material".

⟦R :: relator extends E mediates φ and ψ⟧

$\Longrightarrow$

⟦R :: relator extends E mediates φ and ψ deriving M⟧

Em que ψ não contém o operador `deriving`; e o identificador `M` é a concatenação de $[\![\texttt{A}]\!]_t$ e "_derived_material".

Na sequência, resolve-se o caso da declaração da relação de subsunção (subseção 7.4.2).

⟦R :: relator extends E mediates φ and ψ⟧

$\Longrightarrow$

⟦R extends E⟧
⟦R :: relator mediates φ and ψ deriving M⟧

Seguindo com a tradução, conforme as regras EBNF, os esquemas φ e ψ denotam um mesmo padrão, `RELATOR_TYPE_MEDIATED`. Por seguir a mesma estratégia dos casos anteriores, o processo de normalização não é apresentado, de modo que considera-se o seguinte o caso normal, em que π_X está na forma apresentada pela Definição 7.9 (Indicadores de restrição de cardinalidade), para $X \in \{a1, a2, b1, b2\}$:

⟦R :: relator mediates
π_{ro_1} of Role_1 bearing π_{re_1} of this
and π_{ro_2} of Role_2 bearing π_{re_2} of this
deriving M⟧

$\Longrightarrow$

$dio(in(\llbracket \mathtt{R} \rrbracket_t, qua(\llbracket \mathtt{RgSort}_1 \rrbracket_t, \llbracket \mathtt{Role}_1 \rrbracket_t)), quaIndividual)$

$dio(qua(\llbracket \mathtt{RgSort}_1 \rrbracket_t, \llbracket \mathtt{Role}_1 \rrbracket_t), quaIndividual)$

$deo(in(\llbracket \mathtt{R} \rrbracket_t, qua(\llbracket \mathtt{RgSort}_1 \rrbracket_t, \llbracket \mathtt{Role}_1 \rrbracket_t)), qua(\llbracket \mathtt{RgSort}_1 \rrbracket_t, \llbracket \mathtt{Role}_1 \rrbracket_t))$

$dio(in(\llbracket \mathtt{R} \rrbracket_t, qua(\llbracket \mathtt{RgSort}_2 \rrbracket_t, \llbracket \mathtt{Role}_2 \rrbracket_t)), quaIndividual)$

$dio(qua(\llbracket \mathtt{RgSort}_2 \rrbracket_t, \llbracket \mathtt{Role}_2 \rrbracket_t), quaIndividual)$

$deo(in(\llbracket \mathtt{R} \rrbracket_t, qua(\llbracket \mathtt{RgSort}_2 \rrbracket_t, \llbracket \mathtt{Role}_2 \rrbracket_t)), qua(\llbracket \mathtt{RgSort}_2 \rrbracket_t, \llbracket \mathtt{Role}_2 \rrbracket_t))$

$\llbracket \mathtt{RgSort}_1 \text{ qua } \mathtt{Role}_1 \rrbracket_{et}$

$\llbracket \mathtt{RgSort}_1 \text{ qua } \mathtt{Role}_1 \text{ in } \mathtt{R} \rrbracket_{et}$

$\llbracket \mathtt{RgSort}_2 \text{ qua } \mathtt{Role}_2 \rrbracket_{et}$

$\llbracket \mathtt{RgSort}_2 \text{ qua } \mathtt{Role}_2 \text{ in } \mathtt{R} \rrbracket_{et}$

$\llbracket \mathtt{R} :: \mathtt{relator} \rrbracket$

$\llbracket \mathtt{meta\ abstract\ on\ [RgSort}_1 \text{ qua } \mathtt{Role}_1, \mathtt{RgSort}_2 \text{ qua } \mathtt{Role}_2 \mathtt{]} \rrbracket$

$\llbracket \mathtt{Me}_1 :: \mathtt{mediation\ relates}\ \pi_{re_1}\ \mathtt{of\ R\ and}\ \pi_{ro_1}\ \mathtt{of}\ \mathtt{Role}_1 \rrbracket$

$\llbracket \mathtt{Me}_2 :: \mathtt{mediation\ relates}\ \pi_{re_2}\ \mathtt{of\ R\ and}\ \pi_{ro_2}\ \mathtt{of}\ \mathtt{Role}_2 \rrbracket$

$\llbracket \mathtt{M} :: \mathtt{material\ relates}\ \pi_{m_1}\ \mathtt{of}\ \mathtt{Role}_1\ \mathtt{and}\ \pi_{m_2}\ \mathtt{of}\ \mathtt{Role}_2 \rrbracket$

$\llbracket \mathtt{D} :: \mathtt{derivation\ relates\ 1\ of\ R\ and}\ \pi_d\ \mathtt{of\ M} \rrbracket$

$\llbracket \mathtt{RPT}_1 :: \mathtt{partOf\ relates}\ \pi_{ro_1}\ \mathtt{of}\ \mathtt{RgSort}_1 \text{ qua } \mathtt{Role}_1 \text{ in R partof } \pi_{re_1} \text{ of R} \rrbracket$

$\llbracket \mathtt{RPT}_2 :: \mathtt{partOf\ relates}\ \pi_{ro_2}\ \mathtt{of}\ \mathtt{RgSort}_2 \text{ qua } \mathtt{Role}_2 \text{ in R partof } \pi_{re_2} \text{ of R} \rrbracket$

$\llbracket \mathtt{meta\ [abstract, final]\ on\ [RPT}_1, \mathtt{RPT}_2 \mathtt{]} \rrbracket$

$\llbracket \mathtt{RgSort}_1 \text{ qua } \mathtt{Role}_1 \text{ in R characterizes } \mathtt{RgSort}_1 \rrbracket$

$\llbracket \mathtt{RgSort}_2 \text{ qua } \mathtt{Role}_2 \text{ in R characterizes } \mathtt{RgSort}_2 \rrbracket$

$\llbracket \mathtt{CT} :: \mathtt{collective} \rrbracket$

$\llbracket \mathtt{RPC}_1 :: \mathtt{componentOf\ relates}\ \pi_{ro_1}\ \mathtt{of}\ \mathtt{Role}_1\ \mathtt{partof}\ \pi_{re_1}\ \mathtt{of\ CT} \rrbracket$

$\llbracket \mathtt{RPC}_2 :: \mathtt{componentOf\ relates}\ \pi_{ro_2}\ \mathtt{of}\ \mathtt{Role}_2\ \mathtt{partof}\ \pi_{re_2}\ \mathtt{of\ CT} \rrbracket$

$\llbracket \mathtt{R\ characterizes\ CT} \rrbracket$

$\llbracket \mathtt{meta\ [abstract, final]\ on\ [RPC}_1, \mathtt{RPC}_2 \mathtt{]} \rrbracket$

Em que $R \neq Role_1 \neq Role_2 \neq M$, as variáveis ψ_X para $X \in \{re_1, re_2, ro_1, ro_2, m_1, m_2, d\}$, estão na forma apresentada pela Definição 7.9 (Indicadores de restrição de cardinalidade), e ψ_{m_1}, ψ_{m_2}, ψ_d e as demais variáveis presentes na tradução apresentadas são definidas do seguinte modo:

– $\mathtt{RgSort}_X$, para $X = \{1, 2\}$, denota o primeiro tipo Rigid Sortal instanciado por

`Role`$_X$[a,b], conforme a seguinte restrição:

$$\begin{aligned} \exists\, RS\ (&dio(Role_X, role) \in \Delta \\ &\wedge extensionof\_irreflexive(R, RgSort_X) \in \Delta \\ &\wedge instanceof(RgSort_X, rigidSortal) \in \Delta \wedge ! \end{aligned} \tag{8.2}$$

- $\pi_{m_1} \coloneqq X$, em $card\_prod(\pi_{re_2}, \pi_{ro_1}, X)$, baseado em *card_prod/3* (Definição formal 6.32);
- $\pi_{m_2} \coloneqq X$, em $card\_prod(\pi_{re_1}, \pi_{ro_2}, X)$;
- $\psi_d \coloneqq X$, em $card\_prod(\pi_{ro_1}, \pi_{ro_2}, X)$;
- `Me`$_1$ é a concatenação de $[\![\texttt{R}]\!]_t$, "_med_" e $[\![\texttt{Role\_1}]\!]_t$;
- `Me`$_2$ é a concatenação de $[\![\texttt{R}]\!]_t$, "_med_" e $[\![\texttt{Role\_2}]\!]_t$;
- `RPT`$_1$ é a concatenação de $[\![\texttt{Role}_1]\!]_t$, "_partof_" e $[\![\texttt{R}]\!]_t$;
- `RPT`$_2$ é a concatenação de $[\![\texttt{Role}_2]\!]_t$, "_partof_" e $[\![\texttt{R}]\!]_t$;
- `CT` é a concatenação de $[\![\texttt{R}]\!]_t$, "_collective";
- `RPC`$_1$ é a concatenação de $[\![\texttt{Role}_1]\!]_t$, "_partof_" e `CT`;
- `RPC`$_2$ é a concatenação de $[\![\texttt{Role}_2]\!]_t$, "_partof_" e `CT`;

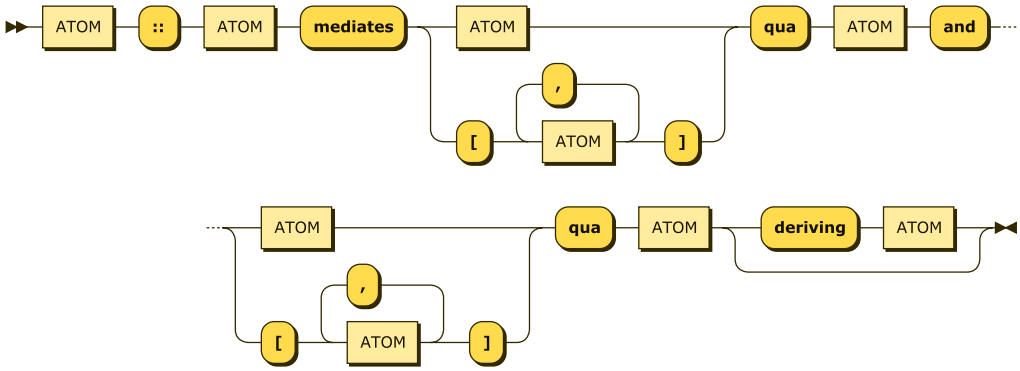
[a] A expressão apresentada na Equação 8.2 é um esboço da regra implementada em Prolog.
[b] É necessário que a sintaxe apresentada nesta seção seja aperfeiçoada para que seja possível expressar qual tipo rígido na hierarquia de tipos instanciados por Role$_X$ se deseja considerar no relacionamento em tela. No modo como apresentado neste texto, a restrição é satisfeita pelo primeiro tipo arbitrário na lista de tipos instanciados pela entidade.

Os Relators são construídos em Ontoprolog com base nas regras contidas no Glossário da UFO. Conforme apresentado na entrada correspondente, entidades do esquema `RigidSortal qua Role in Relator`, instâncias de Qua Individual, são criadas de modo que denotem existencialmente a entidade `RigidSortal` (do tipo Rigid Sortal) no papel (Role) de `Role` em um Relator `Relator`. Esses indivíduos são partes do Relator, via Part of e caracterizam (Characterization) a entidade identificada como `RigidSortal`. Por ser uma instância de Moment, `Relator` deve caracterizar alguma outra entidade, de modo que as instâncias do `Relator` sejam inerentes (Inherence) a instâncias da outra entidade. Por isso, a entidade `CT`, instância de Collective, é criada como um Whole que agrega os Roles participantes da relação em tela via uma relação Components of. Como era de se esperar pela definição de Relator, uma relação derivada (Derivation Relation) é criada de modo a relacionar `Relator` e a relação material (Material Relation) `M`.

A subseção 8.4.7 apresenta discussões e exemplo de uso da sintaxe desenvolvida nesta seção.

8.2.2.11 Relator particular

RELATOR_PARTICULAR_BINARY_RELATION:



A declaração de instâncias de universais de Relator segue o mesmo esquema apresentado na subseção 8.2.2.10, e por isso detalhes são intencionalmente omitidos nesta seção.

Na tradução da sintaxe, o contexto Δ (subseção 7.3.2) é consultado para identificar as instâncias de relações do tipo Mediation, Characterization e Derivation Relation, que definem uma entidade que denota uma instância de Relator. Já as relações instâncias de Part of e de Component of não são instâncias no âmbito de particulares, conforme justificativa apresentada na subseção 8.4.7. Ao invés disso, relações instâncias diretamente de `partOfParticular` são usadas para vincular as instâncias particulares das entidades relacionadas via Part of e Component of. Dessa forma, o axioma Weak Supplementation é atendido.

Detalhes podem ser consultados diretamente na definição em Prolog da tradução de Relator particular (Capítulo 10).

8.2.3 Extensão das expansões das relações semânticas

O algoritmo de expansão das relações semânticas do filtro de tradução, abordado pela subseção 7.3.3, descrito na seção 7.6 e detalhado no Apêndice F, é também expandido para o caso do aperfeiçoamento que envolvem a especificação de ontologias com base na UFO. A exemplo do que ocorre na seção 7.6, opta-se por apresentar apenas a implementação em Prolog dessas expansões. Dessa forma, a implementação em tela está contida no Apêndice G.

Essencialmente, a expansão específica para a UFO consiste em:

a) decidir sobre relações predicadas com $defeasible/1$ (subseção 6.3.3), em que se atribui prioridades para a relação $dio/2$. No caso, se mais de uma afirmação retratável estiver disponível acerca dessa relação, opta-se por tornar definitivas as

instâncias de `quality` e `qualityDimension` (subseção 8.2.2.5), respectivamente. Por outro lado, se nada for afirmado sobre a relação de instância de determinada entidade e em que $entity(e)$ suceda, esta expansão passa a afirmar que e participa da relação $dio(e, subkind)$. Isso visa atender a restrição 1 do Quadro 5 (Subkind), p. 171 e a definição de tradução sintática descrita na subseção 8.2.2.5;

b) distribuir meta-propriedades deriváveis de outras meta-propriedades, como as estabelecidas pela restrição 2 do Quadro 24 (Meronymic), p. 187 e pela restrição 4 do Quadro 28 (MemberOf), p. 191, entre outras regras detalhadas no Apêndice F;

c) derivar entidades relacionais referentes à instância da entidade $qfrom$, usada para vincular instâncias de Quale à instância de Quality, devido ao açúcar sintático descrito na subseção 8.2.2.6.

8.3 Definições e regras específicas da UFO

Além dos aperfeiçoamentos sintáticos descritos na seção 8.2, esta seção apresenta extensões às definições e regras de $\mathcal{R}$ da estrutura $\mathcal{OT}$ presentes na seção 6.4. A fim de racionalizar a parte textual, apresenta-se as definições em apenas duas categorias, uma referente a regras positivas, que essencialmente são definições de conceitos com base em relações semânticas, e outra referente a regras RNP (subseção 6.4.3). Cada uma dessas categorias está contida em uma das seções seguintes, respectivamente.

8.3.1 Definições positivas

As definições formais apresentados nesta seção visam enriquecer o poder dedutivo das teorias escritas em Ontoprolog com definições específicas da UFO. O objetivo é prover ao arquiteto da informação, usuário da linguagem, uma série de instrumentos de modo que ele possa não só descrever o discurso ontológico, mas também consultar a base de afirmações desse discurso.

Apenas um pequeno exemplo com quatro predicados possíveis de serem definidos em Ontoprolog para a UFO são apresentados nesta seção. Várias outras definições estão presentes no código-fonte do projeto (Capítulo 10). De toda forma, uma biblioteca desses predicados podem ser desenvolvidos e incorporados a futuras versões da ferramenta.

Definição formal 8.1 ($functional\_complex/1$)**.** Este predicado segue a definição de Complexo Funcional (Functional Complex), de modo que $functional\_complex(F)$ suceda sempre que F for uma instância de Kind, uma subsunção de uma entidade que é instância de Kind, uma instância de uma entidade que seja um Functional Complex ou ainda,

quando F se tratar de uma instância de Mixin que não possua uma subsunção instância nem de Quantity, nem de Collective.

$$
\begin{aligned}
functional\_complex(F) \leftarrow & instanceof(F, kind) \\
functional\_complex(F) \leftarrow & extensionof(F, S), \\
& instanceof(S, kind) \\
functional\_complex(F) \leftarrow & dio(F, T), \\
& functional\_complex(T) \\
functional\_complex(F) \leftarrow & instanceof(F, abstractMixin), \\
& extensionof\_reflexive(S, F), \\
& \neg dio(S, quantity), \\
& \neg dio(S, collective)
\end{aligned}
$$

—

Sintaticamente, a relação Characterization é definida conforme a tradução apresentada na subseção 8.2.2.8. Porém, conforme se observa, o construtor `characterizes` é traduzido em diversas relações semânticas que, juntas, compõem o significado da Caracterização no âmbito de Ontoprolog. Dessa forma, é útil que a simplicidade que se obteve com uma sentença direta no âmbito sintático pode ser resgatada no mundo semântico com uma definição simples, como a apresentada pela Definição formal 8.2.

Definição formal 8.2 ($dcharacterization/2$)**.** Define a relação $dcharacterization(A, B)$, que denota que a entidade A *characteriza* a entidade B diretamente:

$$
\begin{aligned}
dcharacterization(A, B) \leftarrow & dio(R, characterization), \\
& drel(R, A, \_, B, \_)
\end{aligned}
$$

—

O conceito de Inerência (Inherence) é a contrapartida da relação Characterization no âmbito de particulares. Com base no arcabouço construído até este ponto, o conceito pode ser definido do seguinte modo:

Definição formal 8.3 ($inherence/2$)**.** Uma entidade A é inerente a uma entidade B quando A é a instância de uma entidade que caracteriza a instância da entidade que B é instância.

$$
\begin{aligned}
inherence(A, B) \leftarrow & dio(C, characterization), \\
& dio(I, C), \\
& drel(I, A, \_, B, \_)
\end{aligned}
$$

—

Por fim, o conceito de Portador (Bearer) (Bearer of a Moment), em que uma entidade A é a detentora de uma entidade B que lhe é inerente ou que lhe caracteriza, pode generalizar os predicados *inherence*/2 e *dcharacterization*/2 no que tange ao nível teórico. Dessa forma, uma relação *dbo*/2 pode ser definida com essa característica:

Definição formal 8.4 (*dbo*/2)**.** *dbo*/2, lido como *direct bearer of*, generaliza os predicados *inherence*/2 e *dcharacterization*/2 e implementa o conceito de Bearer of a Moment:

$$dbo(A, B) \leftarrow dcharacterization(B, A)$$
$$dbo(A, B) \leftarrow inherence(B, A)$$

—

8.3.2 Extensões de definições na forma $ngrule/n$

As regras presentes nesta seção são RNP escritos na norma $ngrule/n$. Tratam-se de definições formais que complementam as regras detalhadas na subseção 4.5.1, e envolvem especialmente o Quadro 25 (ComponentOf) e o Quadro 28 (MemberOf).

Definição formal 8.5 (*component_relating_non_fcomplex_types*)**.** Verifica a restrição 1 do Quadro 25 (ComponentOf), p. 188 para universais, em que ao menos um entre a parte P ou o todo W devem se referir a um Complexo Funcional (Functional Complex).

$$\begin{aligned} ngrule(component\_relating\_non\_fcomplex\_types) \leftarrow \\ & drel(R, P, \_, W, \_), \\ & instanceof(R, componentOf), \\ & \neg once((functional\_complex(P) \\ & \quad \vee functional\_complex(W))) \end{aligned}$$

—

Definição formal 8.6 (*component_relating_non_fcomplex_particular*)**.** Esta regra complementa a Definição formal 8.5 para o caso dos particulares, uma vez que sucede sempre que uma parte P e um todo W de uma relação instância de uma instância de Component of não se referirem ambos a um Functional Complex.

$$\begin{aligned} ngrule(component\_relating\_non\_fcomplex\_particular) \leftarrow \\ & drel(R, \_, \_, \_, \_), \\ & instanceof(R, componentOf), \\ & instanceof(RP, R), \\ & drel(RP, P, \_, W, \_), \\ & \neg once((functional\_complex(P) \\ & \quad \vee functional\_complex(W))) \end{aligned}$$

—

Definição formal 8.7 (*memberOf_non_fcomplex_type_as_part*)**.** Esta regra verifica a restrição 2 do Quadro 28 (MemberOf), p. 191, em que uma parte *Part* deveria ser um Functional Complex no âmbito de universais.

$$\begin{aligned} ngrule(cmemberOf\_non\_fcomplex\_type\_as\_part) \leftarrow \\ drel(R, Part, \_, \_, \_), \\ instanceof(R, memberOf), \\ \neg instanceof(Part, collective), \\ \neg functional\_complex(Part) \end{aligned}$$

—

Definição formal 8.8 (*memberOf_non_fcomplex_particular_as_part*)**.** Esta regra complementa a Definição formal 8.7 no caso de a parte *Part*, no âmbito de instância de particulares, não ser um Functional Complex.

$$\begin{aligned} ngrule(memberOf\_non\_fcomplex\_particular\_as\_part) \leftarrow \\ drel(R, \_, \_, \_, \_), \\ instanceof(R, memberOf), \\ instanceof(RP, R), \\ drel(RP, Part, \_, \_, \_), \\ \neg instanceof(Part, collective), \\ \neg functional\_complex(Part) \end{aligned}$$

—

Definição formal 8.9 (*quality_particular_multi_meta*)**.** Esta regra acrescenta outra restrição à capacidade de múltipla instanciação de Ontoprolog, mas agora no que se refere a Qualities, de modo a atender a restrição 6 do Quadro 13 (Quality), p. 178: particulares T de instâncias de entidades `quality` da UFO não podem instanciar direta e simultaneamente outra entidade. Essa restrição é verificada pela seguinte regra:

$$\begin{aligned} ngrule(quality\_particular\_multi\_meta) \leftarrow \\ entity(T), \\ setof(M, dio(T, M), MM), \\ util\_filter(instance\_quality, MM, UMM), \\ (lenght(UMM, L), L > 1) \end{aligned}$$

Em que $util\_filter/3$ é apresentada na Definição formal 8.10 e $instance\_quality/1$ sucede para todas as entidades instâncias de `quality`, via $instanceof/2$. —

Além dessas regras, acrescentam-se, entre outras, as seguintes, que referem-se a regras gerais de ontologias especificadas com base na UFO.

Definição formal 8.10 ($dio\_more\_than\_one\_ufo\_entity$)**.** Acrescenta restrição à capacidade de múltipla instanciação de Ontoprolog: instâncias T de entidades da UFO não podem instanciar simultaneamente outra entidade. Trata-se de um tipo de disjunção generalizada das entidades da teoria da UFO. Essa restrição é verifica pela seguinte regra:

$$\begin{aligned} ngrule(dio\_more\_than\_one\_ufo\_entity) \leftarrow \\ & entity(T), \\ & setof(M, dio(T, M), MM), \\ & util\_filter(ufo\_type, MM, UMM), \\ & (lenght(UMM, L), L > 1) \end{aligned}$$

Em que $util\_filter/3$ é um predicado de alta ordem que sucede quando UMM é a lista de indivíduos de MM no qual o predicado $ufo\_type/1$ sucede. $ufo\_type/1$ sucede para todas as entidades que estão na teoria de entidades da UFO (Apêndice B). —

Definição formal 8.11 ($deo\_and\_dio\_ufo\_type$)**.** Outra restrição aos modelos bases de Ontoprolog refere-se à restrição adicional de que uma entidade T não pode instanciar e estender, simultaneamente, uma entidade da UFO. Isso é verificado por esta regra:

$$\begin{aligned} ngrule(deo\_and\_dio\_ufo\_type) \leftarrow \\ & extensionof(T, S), \\ & ufo\_type(E), \\ & instanceof(T, M), \\ & ufo\_type(M) \end{aligned}$$

Em que $ufo\_type/1$ é apresentado na Definição formal 8.10. —

8.4 Discussões e exemplos

Esta seção contém discussões e exemplos de uso de Ontoprolog com UFO.

8.4.1 Extensões com regras mais específicas

Pelo modo com Ontoprolog é projetado, além da extensão realizada para a UFO, outras podem ser construídas. Isso se deve ao fato de que as extensões complementam a sistema base, e não o revogam. No entanto, é importante salientar que as extensões semânticas devem sempre restringir o número válido de modelos, no sentido de que devem acrescentar mais regras, e não flexibilizá-las. Do ponto de vista sintático, entretanto, o que se deseja é justamente o oposto: estender a linguagem de modo que mais sentenças sejam válidas. Em geral, nas extensões valem a regra geral para subsunção: uma regra mais específica deve sempre restringir uma regra mais geral, e não o contrário.

8.4.2 Especificações apenas com universais

Na maneira proposta, a extensão UFO de Ontoprolog não contém regras que imponham existência de particulares de entidades universais. Ou seja, não existe qualquer regra do tipo $ngrule/n$ que verifique se instâncias de `universal` possuem instâncias. Isso significa que em Ontoprolog é possível especificar apenas "classes" de entidades, assim como se faz em OntoUML tradicionalmente. Dessa forma, o uso de particulares se torna complementar, e não obrigatório.

O contrário também é o caso. Um programador pode declarar entidades como sendo instâncias diretas de `particular`, algo como `casa :: particular`, `carro :: particular`, etc. No entanto, esse tipo de abordagem não parece muito salutar, especialmente porque se perde a capacidade de se categorizar as entidades da ontologia. De toda forma, trata-se de um uso válido.

8.4.3 Especificação de ontologia UFO e diagramações OntoUML

O Código 11 contém uma versão especificada com base na UFO da ontologia "Sinatra" apresentada no Código 8, originalmente descrita diretamente a partir da Metateoria de *Hypertypes*. Essencialmente, pretende-se descrever uma ontologia sobre o onto *criatura*. Deseja-se descrever os conceitos gerais de *vivo* e de *morto*, e de *pessoa*, cujas instâncias podem estar em uma dessas situações em determinado mundo possível. Além disso, deseja-se descrever ser inerente ao conceito de *pessoa* um Momento Intrínseco (Intrinsic Moment) que represente o nome da pessoa. Inerente ao conceito de *morto*, deve constar um Momento Intrínseco que denote a data da morte, referente a um espaço de qualidade de datas. Para exemplificar um mundo possível dessa ontologia, apresenta-se como exemplo dois particulares de *pessoa*: `lauro` e `sinatra`.

Código 11 – Especificação "Sinatra" com base na UFO

```
criatura :: kind.

disjoint [vivo, morto] :: phase extends criatura.
data_morte :: quality => date characterizes morto.

pessoa :: subkind extends criatura.
nome :: quality => text characterizes pessoa.

[ pessoa_viva, pessoa_morta ] :: phase cover pessoa.
pessoa_viva extends vivo.
pessoa_morta extends morto.

lauro :: pessoa_viva.
lauro_nome :: nome <= quale 'Lauro Cesar' inheresin lauro.

sinatra :: [pessoa_morta, vivo].
[ sinatra_nome       :: nome      <= quale 'Frank Sinatra',
  sinatra_data_morte :: data_morte <= quale '1998-05-14' ] inherein sinatra.

% Theory compilation and semantic checking
:- otp_compile,
   check_semantics.
```

Na linha 1 do Código 11 descreve-se `criatura` como um Kind. Na linha 3, define-se que os conceitos de `vivo` e `morto` são disjuntos e são instâncias de `phase` (Phase). Essa linha é instância da gramática do Código 6, que combina a declaração simultânea de duas relações semânticas: a instanciação e a disjunção de conceitos. Já a linha 4 é instância da gramática estendida descrita no Código 10. Ela denota a existência um conceito universal `data_morte`, instância da entidade `quality` (Quality), que está associado (`dassoc`) (Association Relation) a um Quality Structure chamado `date`. Além disso, uma instância da relação `characterization` (Characterization) também é criada, de modo a relacionar `data_morte` com `morto`. Se nada mais for dito a respeito de `date`, o sistema lógico assumirá que se trata de uma instância de `qualityDimension` (Quality Dimension). Esse é um dos casos de aplicação de raciocínio não monotônico por meio de *defeasible rules* (subseção 6.3.3). Na linha 13 consta um exemplo da instância `lauro_nome` da *quality* `nome`, cujo valor do *quale* associado a ela é *'Lauro Cesar'*, e a vinculação do particular `lauro_nome` com o particular `lauro`, declarado na linha anterior como instância de `pessoa_viva`. Nas linhas 16 e 17 constam exemplo de açúcar sintático em que declara-se a instanciação e a vinculação de duas *quality* particulares com o particular `sinatra`, declarado na linha 15 como instância simultânea de duas entidades, `pessoa_morta` e `vivo`. Evidentemente, essa conceituação não respeita o compromisso ontológico com senso comum corrente, que nos informa que *Frank Sinatra* está morto desde 1998.

O Código 12 contém a saída da avaliação semântica do Código 11. A exemplo do que é apresentado no Código 9 a respeito do Código 8, o mecanismo de avaliação da Teoria denotada pela Especificação verifica que há uma inconsistência a respeito do conceito `sinatra`, capturado pela regra `dio_disjoint_types`, como era esperado. No caso, `sinatra` não pode instanciar simultaneamente os tipos `pessoa_morta` e `vivo` por eles serem disjuntos. Esse resultado demonstra o mecanismo que define a noção de consistência no âmbito das teorias de Ontoprolog. De fato isso exercita a regra de disjunção exclusiva, que ontologicamente pode ser interpretado como "não é o caso que em um mesmo mundo seja possível que `sinatra` esteja simultaneamente vivo e morto".

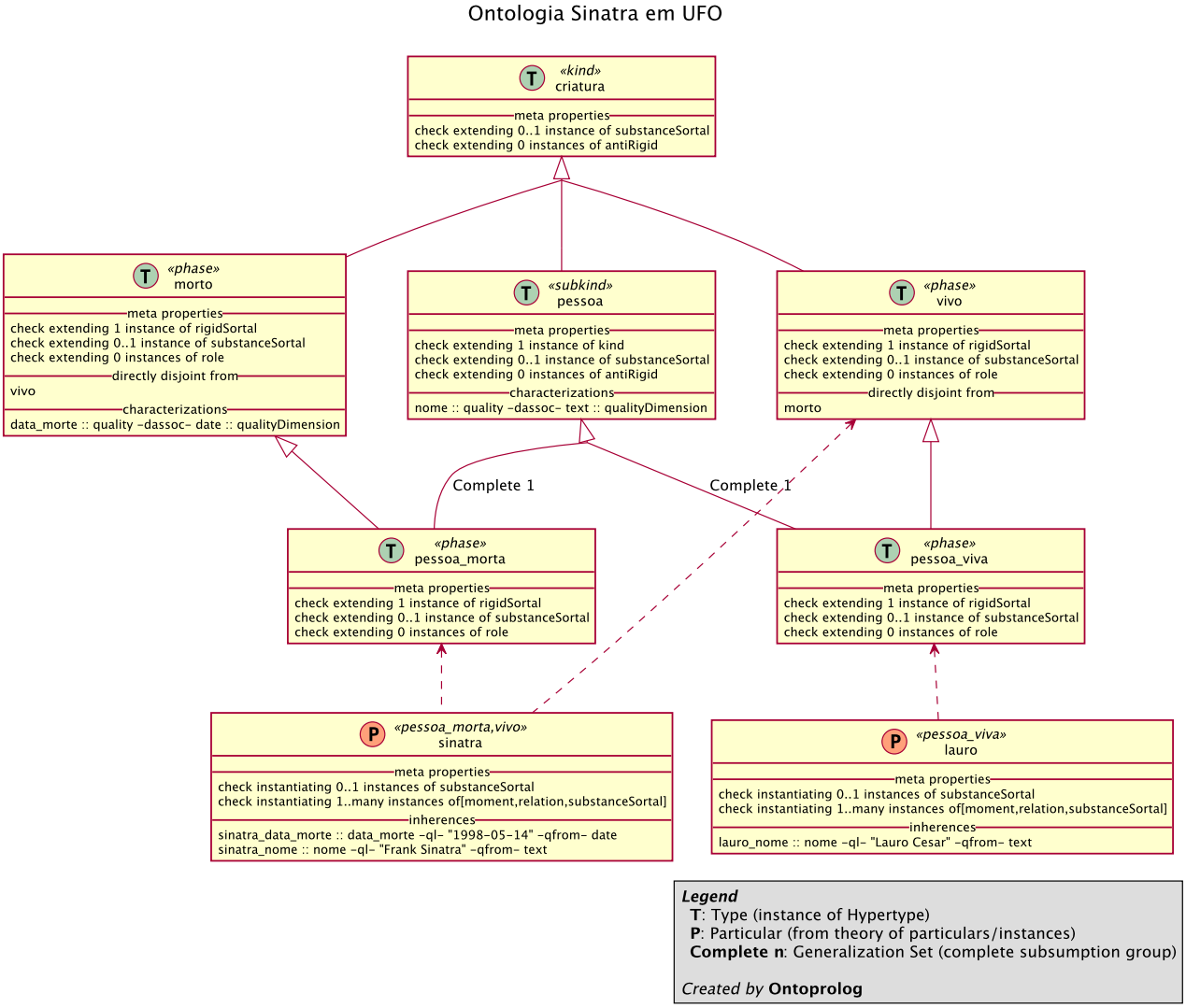
Código 12 – Fragmento da saída da avaliação semântica do Código 11

```
Compilation of the Ontoprolog specification was successful.

Checking semantics using negative rules of the current Ontoprolog theories...
--------------------------------------------------
Checking ngrule/3...
 sinatra:
   -> "sinatra" instantiates disjoint types "[pessoa_morta,vivo]".
    (dio_disjoint_types)
--------------------------------------------------
```

A exemplo da Figura 34, a Figura 38 contém uma representação em linguagem visual baseada em diagramas OntoUML da especificação contida no Código 11. Os diagramas são criados conforme tradução descrita na seção 10.3.

Na figura, as Qualidades são representadas de modo compacto, em um comparti-

Figura 38 – Diagramação da teoria especificada no Código 11 com decoração de Momentos Intrínsecos

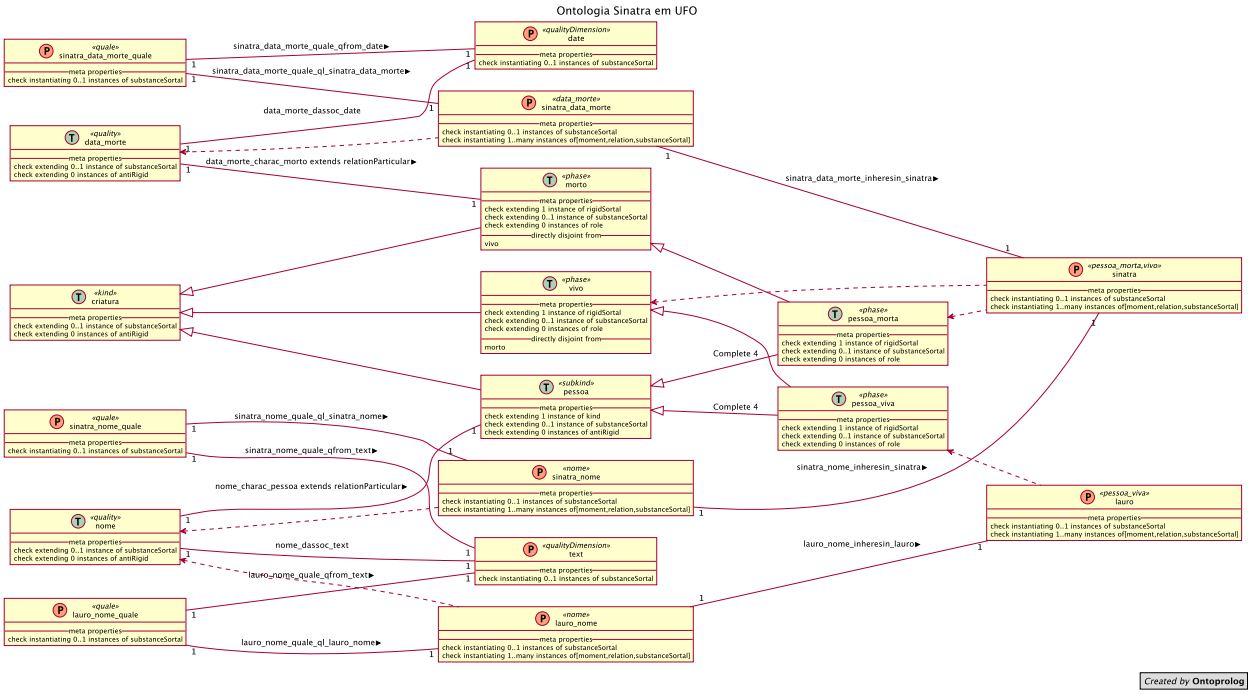


Fonte: Os autores

mento próprio, chamado *"characterizations"*, dentro do Classifier respectivo. Da mesma forma, as qualidades particulares são representadas na seção *"inherences"* (Inherence). A série de relações referentes aos Momentos Intrínsecos são sumarizadas em uma linha nos Classifier devido à uma personalização no mecanismo de tradução de teorias Ontoprolog para diagramas OntoUML. Sem o uso dessa personalização, a representação da especificação do Figura 38 teria a forma da Figura 39, em que as entidades referentes aos Momentos Intrínsecos, discutidas na seção 8.1, aparecem de forma explícita.

A possibilidade de representar visualmente diferentes níveis de abstração da ontologia exercita um resultado do trabalho anterior dos pesquisadores desta tese (subseção 2.2.1.1). No caso, trata-se da aplicação da *operação zoom* na composição de configurações representadas pelas entidades da ontologia. Especialmente, o critério adotado para

Figura 39 – Diagramação da teoria especificada no Código 11 sem decoração de Momentos Intrínsecos



Fonte: Os autores

navegar nas composições dos objetos da ontologias podem ser variados. Na seção 5.5, por exemplo esses critérios são baseados a relação de instanciação, que separa um nível teórico mais geral, de outro mais específico. No caso do que é ilustrado pela distinção da Figura 38 e da Figura 39, o critério adotado são os padrões de modelagem referentes ao Momentos Intrínsecos, que permitem que agregações sejam realizadas com base nas relações entre objetos que envolvem qualidades.

8.4.4 Definição de restrições sobre Espaços Conceituais

Conforme definido na subseção 8.2.2.3, são intrínsecas a um determinado Espaço conceitual (Definição 8.1) restrições sobre as dimensões e domínios da Estrutura de Qualidade (Quality Structure) que ele denota. Essas restrições são usadas como base para que sejam definidas álgebras entre diferentes instâncias desses Espaços Conceituais (Conceptual Space). Em Ontoprolog há um mecanismo padrão de definição de restrições, que é o descrito na subseção 6.4.4 (Restrições a partir de meta-propriedades). Com base nesse mecanismo, é possível atribuir meta-propriedades a instâncias de alguma entidade e então verificar se os modelos que possuem entidades em que aquela meta-propriedade está presente atendem a determina regra do tipo `nrulemeta` (Código 5).

Dessa forma, o Código 13 apresenta um exemplo de como esse mecanismo pode

ser usado para verificar Espaços Conceituais. Esse exemplo visa atender a restrição 2 do Quadro 16 (Quale), p. 180 e a restrição 3 do Quadro 15 (QualityDomain como StructuredDataType), p. 180.

Código 13 – Exemplo de verificação de Espaços Conceituais

```
% definindo uma estrutura de qualidade para numero_positivo
numero_positivo :: qualityDimension.

meta check is_positivo
  on numero_positivo.

% definindo o conceito de numero_positivo
nrulemeta check is_positivo
      for Quality_Dimension
  message ['"', Value, '" is not a positive number for "', Quale, '".'] :-
      dpv(codomain at QFrom, Quality_Dimension),
      dpv(domain at QFrom, Quale),
      dpv(qualityRegion at Quale, Value),
      \+ (number(Value),
          Value >= 0).

% definindo a teoria de universais
idade :: quality => numero_positivo characterizes pessoa :: kind.

% definindo uma teoria de particulares
[cesar, jones] :: pessoa.

% definindo uma teoria de particulares
idade_cesar :: idade <= quale -2 inheresin cesar.
idade_jones :: idade <= quale 66 inheresin jones.

:- otp_compile,
```

Essencialmente, a regra `nrulemeta check is_positivo` consulta as relações semânticas produzidas pela especificação contida no Código 13 e as entidades específicas da UFO contidas na metateoria descrita no Apêndice C de modo a obter o valor `Value` de determinado Quale `Quale`, associado via instância `QL` de uma relação `ql` (Quale of a quality) a uma instância `Quality_Instance` de um Quality `Quality`, que por sua vez está associado via uma instância `DASSOC` da relação `dassoc` (Association Relation) ao Quality Dimension `numero_positivo`. No caso, a regra sucede quando `Value` denotar um valor menor do que zero, o que é oposto a uma ideia de número positivo.

O resultado da avaliação semântica dessa especificação é descrita no Código 14.

Código 14 – Fragmento da saída da avaliação semântica do Código 13

```
Checking semantics using negative rules of the current Ontoprolog theories...
-------------------------------------------------
Checking ngrule/3...
 numero_positivo:
   -> Meta "check is_positivo" on "numero_positivo" have not passed: "-2" is not a
    positive number for "idade_cesar". (ncheck_for_meta_not_passing)
```

Como era de se esperar, a regra `check is_positivo` não é válida para o caso em que se aponta para a região -2 (Quality Region) do Espaço Conceitual denotado por `numero_positivo`.

8.4.5 Subsunção de Qualidades

O Código 15 exercita a definição de hierarquias de Qualidades (Quality) e a definição de Quality Domain.

Código 15 – Exemplo de especificação UFO com subsunção de Qualidades

```
% RGB conceptual space
rgb :: qualityDomain => space [r, g, b].

cor :: quality => rgb.
meta abstract on cor.
[cor_olhos, cor_pele] :: quality extends cor.

% person universal theory
pessoa :: kind.
[cor_olhos, cor_pele] characterize pessoa.

% person particular theory
marco_polo :: pessoa.
cor_olhos_marco_polo :: cor_olhos <= quale [r<= 10, g<= 20, b<= 30] from rgb.
cor_pele_marco_polo  :: cor_pele <= quale [r<= 20, g<= 40, b<= 50].

[ cor_olhos_marco_polo, cor_pele_marco_polo ] inherein marco_polo.

%[ cor_olhos_marco_polo :: cor_olhos <= quale [r<= 10, g<= 20, b<= 30],
%  cor_pele_marco_polo :: cor_pele <= quale [r<= 20, g<= 40, b<= 50]
% ] inhere in marco_polo.

lauro :: pessoa.
[ cor_olhos_marco_polo, cor_pele_marco_polo ] inherein lauro.

:- otp_compile,
   check_semantics.
```

Na linha 2 o Quality Domain `rgb` refere-se a um Conceptual Space que contempla os Dimensões de Qualidade `r`, `g` e `b`. A Qualidade `cor` é uma entidade abstrada associada (Association Relation) a esse espaço. As Qualidades `cor_olhos` e `cor_pele` estendem `cor` e caracterizam o conceito de `pessoa`. A partir da linha 13 apresenta-se diferentes formas de se definir particulares dessa teoria, com as respectivas observações de Quale. Observa-se, nesse caso, a diferença entre os operadores `=>` e `<=`. O primeiro é sempre utilizado o âmbito de universais, para declarar relações de associação entre elementos dos Momentos Intrínsecos (seção 8.1). Já o operador `=>` refere-se sempre a observações de valores de Quale referentes em teorias de particulares. Restrições aos Conceptual Space podem ser definidos conforme apresentados pela subseção 8.4.4.

Do ponto de vista ontológico, as Qualidades são existencialmente dependentes de seus Portadores (Bearer). Por isso, embora a especificação do Código 16 seja sintaticamente válida, pelas regras atribuídas às instâncias de Momentos Intrínsecos (Intrinsic Moment), os conceitos `cor_olhos_marco_polo` e `cor_pele_marco_polo` não podem ser inerentes a dois particulares diferentes: `marco_polo`, na linha 17, e `lauro`, na linha 24. Essa inconsistência é devidamente apresentada pela avaliação semântica da teoria denotada pela especificação, conforme o Código 16.

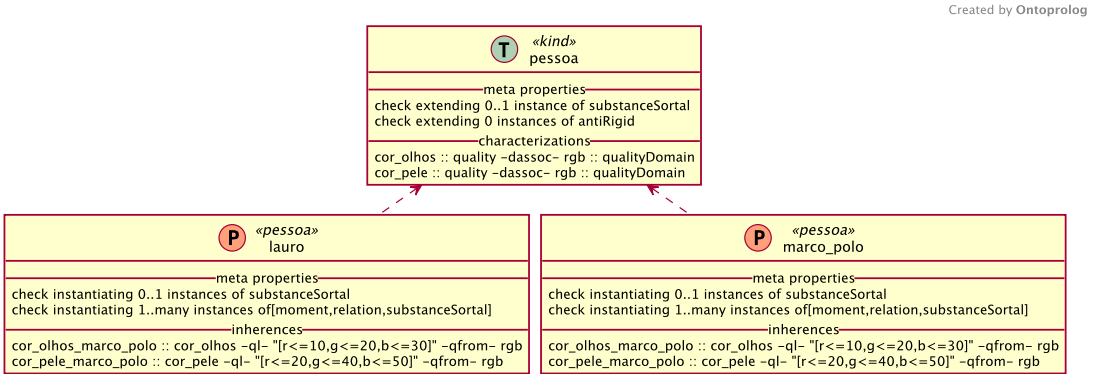
Código 16 – Fragmento da saída da avaliação semântica do Código 15

```
Compilation of the Ontoprolog specification was successful.

Checking semantics using negative rules of the current Ontoprolog theories...
```

```
-----------------------------------------------------
Checking ngrule/3...
 [cor_olhos_marco_polo]:
   -> "cor_olhos_marco_polo" should function "1" time(s) as domain in relations
    instanciating "cor_olhos_charac_pessoa", but it occurs "2" time(s) in
    "[lauro,marco_polo]". (dio_relation_wrong_codomain_cc)

 [cor_pele_marco_polo]:
   -> "cor_pele_marco_polo" should function "1" time(s) as domain in relations
    instanciating "cor_pele_charac_pessoa", but it occurs "2" time(s) in
    "[lauro,marco_polo]". (dio_relation_wrong_codomain_cc)
```

A Figura 40 ilustra a diagramação da teoria denotada pela especificação contida no Código 15, que pode ser produzida mesmo a teoria não sendo adequada às restrições lógicas da ontologia de fundamentação.

Figura 40 – Diagramação da teoria especificada no Código 15



Fonte: Os autores

8.4.6 Conversão de banco de dados

Um uso prático de Ontoprolog, que exemplifica o que é abordado sobre particulares na subseção 8.4.2, trata-se da conversão de bases de dados existentes em modelos baseados em ontologias, no caso, ontologias UFO. Por exemplo, imaginemos que em um banco de dados qualquer exista uma tabela chamada `cliente`, que possui quatro campos: `id`, `nome`, `genero` e `idade`. Nesse caso, pode-se definir uma conversão das tuplas dessa tabela em particulares de uma teoria UFO. Para isso, considere o Código 17.

Código 17 – Exemplo de tradução de banco de dados

```
% cliente(id, nome, genero, idade)
cliente(id1, 'Lauro Cesar',     masculino, 30).
cliente(id2, 'Laura Vicuna',    feminino,  12).
cliente(id3, 'Merceditas Pino', feminino,  50).

% UFO ontology

[ nome   :: quality => string,
  idade  :: quality => numero_positivo ]
  characterizes
  pessoa :: kind.

disjoint [homem, mulher] :: subkind extends pessoa.

% mapeamentos
```

```
map_genero(masculino, homem).
map_genero(feminino, mulher).

% UFO particulars

[ Pessoa_Indiv_Nome  :: nome  <= quale Nome,
  Pessoa_Indiv_Idade :: idade <= quale Idade]
  inherein
  Pessoa_Indiv :: SubKind :-
        cliente(Pessoa_ID, Nome, Genero, Idade),
        map_genero(Genero, SubKind),
        atom_concat('p_', Pessoa_ID, Pessoa_Indiv),
        atom_concat(Pessoa_Indiv, '_nome', Pessoa_Indiv_Nome),
        atom_concat(Pessoa_Indiv, '_idade', Pessoa_Indiv_Idade).

:- otp_compile,
   check_semantics.
```

No exemplo, `cliente/4` representa uma tabela existente em um banco de dados. A seção `UFO ontology` é uma teoria Ontoprolog simples, que consiste na afirmação de cinco universais principais (`pessoa`, `idade`, `nome`, `homem` e `mulher`). Com base nessas entidades, outras são derivadas, conforme se confirma pela execução da Consulta definida seguinte, baseada em *instanceof*/2 (Definição formal 6.5):

```
| ?- instanceof(X, universal).
X = particular ? ;
X = pessoa ? ;
X = idade ? ;
X = idade_charac_pessoa ? ;
X = nome ? ;
X = nome_charac_pessoa ? ;
X = homem ? ;
X = mulher ? ;
no
```

Na teoria de universais construída, os campos `nome` e `idade` de `cliente/4` devem ser transcritos para Qualities, enquanto o campo `genero`, que denota um tipo rígido (Rigid), deve ser traduzido em um Subkind.

A seção `mapeamentos` contém a definição do mapeamento entre as entidades de domínio, `masculino` e `feminino`, e as entidades da ontologia, no caso, os SubKinds `homem` e `mulher`. Esse mapeamento é apenas um exemplo que exercita os mapeamentos que devem ser feito em aplicações reais.

Por fim, na seção `UFO particulars`, exercita-se a tradução em si das tuplas da tabela `cliente/4` em uma teoria de particulares da UFO. A coluna `id` é o identificador da tupla. Esse identificador é usado como identificador das entidades da teoria, acrescido de prefixos e de sufixos, por meio do predicado `atom_concat/3`.

Uma vez executada essa especificação (observar os predicados `otp_compile/0` e `check_semantics/0`), é possível realizar a consulta definida abaixo, que demonstra os indivíduos criados com base na especificação do Código 17.

```
| ?- instanceof(X, particular).
X = idade_dassoc_numero_positivo ? ;
X = nome_dassoc_string ? ;
X = p_id1 ? ;
X = p_id1_idade ? ;
X = p_id1_idade_quale_ql_p_id1_idade ? ;
X = p_id1_idade_quale ? ;
X = p_id1_idade_inheresin_p_id1 ? ;
```

```
X = p_id1_nome ? ;
X = p_id1_nome_quale_ql_p_id1_nome ? ;
X = p_id1_nome_quale ? ;
X = p_id1_nome_inheresin_p_id1 ? ;
X = p_id2 ? ;
X = p_id2_idade ? ;
X = p_id2_idade_quale_ql_p_id2_idade ? ;
X = p_id2_idade_quale ? ;
X = p_id2_idade_inheresin_p_id2 ? ;
X = p_id2_nome ? ;
X = p_id2_nome_quale_ql_p_id2_nome ? ;
X = p_id2_nome_quale ? ;
X = p_id2_nome_inheresin_p_id2 ? ;
X = p_id3 ? ;
X = p_id3_idade ? ;
X = p_id3_idade_quale_ql_p_id3_idade ? ;
X = p_id3_idade_quale ? ;
X = p_id3_idade_inheresin_p_id3 ? ;
X = p_id3_nome ? ;
X = p_id3_nome_quale_ql_p_id3_nome ? ;
X = p_id3_nome_quale ? ;
X = p_id3_nome_inheresin_p_id3 ? ;
X = numero_positivo ? ;
X = string ? ;
X = p_id1_idade_quale_qfrom_numero_positivo ? ;
X = p_id1_nome_quale_qfrom_string ? ;
X = p_id2_idade_quale_qfrom_numero_positivo ? ;
X = p_id2_nome_quale_qfrom_string ? ;
X = p_id3_idade_quale_qfrom_numero_positivo ? ;
X = p_id3_nome_quale_qfrom_string ? ;
no
```

Observa-se que como toda sentença Ontoprolog é também uma sentença Prolog, a definição da sentença `inherein` (por volta da linha 24 do Código 17) é realizada como uma regra Prolog simples, uma vez que é seguida de `:-`. Dessa forma, a instanciação dos particulares ocorre naturalmente pelo algoritmo de *Resolução* da Programação em Lógica, que tenta provar universalmente a conjunção que segue `:-`.

8.4.7 Discussão e exemplo de uso de Relator

Os açúcares sintáticos introduzidos na subseção 8.2.2.10 para declaração de Relators são traduzidos num conjunto de diversas relações semânticas. Essas relações visam traduzir a definição do conceito conforme apresentado no Glossário da UFO, do qual extrai-se esta nota:

"*Nota:* É importante destacar que, com base em Guizzardi (2005) e em Guizzardi (2006), no âmbito do estudo dos Relators existem 2 Whole envolvidos:

a) o Relator em si é um Whole composto dos Qua Individuals;

b) o portador do Relator é um Whole composto pelos indivíduos mediados pelo relator (ou seja, os indivíduos aos quais os Qua Individuals se referem). No caso desse Whole portador, o autor não menciona como ele deve ser identificado ou mesmo representado em modelos conceituais."

Nas obras citadas não se encontrou qual exatamente é a natureza das relações que ligam os Wholes com as partes. No caso do Item b), como o Relator é um Momento (Moment), é necessário que haja uma entidade que seja seu portador. No nível de universais,

Characterization é a relação que liga um Moment com seu portador. Isso justifica a criação de um Collective arbitrário pela tradução descrita na subseção 8.2.2.10, bem como justifica a escolha da relação Component of para ligar as entidades mediadas pelo Relator.

Porém, avaliação semelhante não pode ser feita no que tange ao Item a). Isso se deve ao fato de o Relator e os Qua Individuals, por serem Moments, não se enquadrarem em nenhum dos domínios e contra-domínios das relações Component of, Member of, Subquantity of ou Subcollection of. As duas primeiras exigem que ao menos um dos lados da relação sejam Functional Complexes, e as duas últimas relacionam ou Quantities, ou Collectives, o que não é o caso das entidades em tela. Com isso, resta apenas o gênero dessas relações, no caso, a própria relação Parts of, que foi exatamente o tipo de relação da UFO escolhida por nós para relacionar as entidades descritas no Item a).

Diferentemente do que ocorre com as demais entidades envolvidas nos Relators, as relações de agregação (no caso, instâncias de Part of e Component of) são marcadas com as meta-propriedades `abstract` e `final`. Isso é devido ao axioma Weak Supplementation[4], inerente a todas as entidades Part of, que exige que os Wholes sejam formados por ao menos duas partes. Ocorre que embora no âmbito de universais as relações parte-todo em tela relacionem ao menos duas partes, não se poderia exigir que instâncias dessas relações ligassem ao menos duas partes, uma vez que se tratam de relações diferentes. Dessa forma, o Relator particular, descrito na subseção 8.2.2.11, não instancia essas relações de parte-todo, mas cria instâncias diretamente de `partOfParticular`, de modo a vincular as instâncias particulares das entidades relacionadas via Part of e Component of. Dessarte, o axioma Weak Supplementation é atendido também no âmbito de particulares.

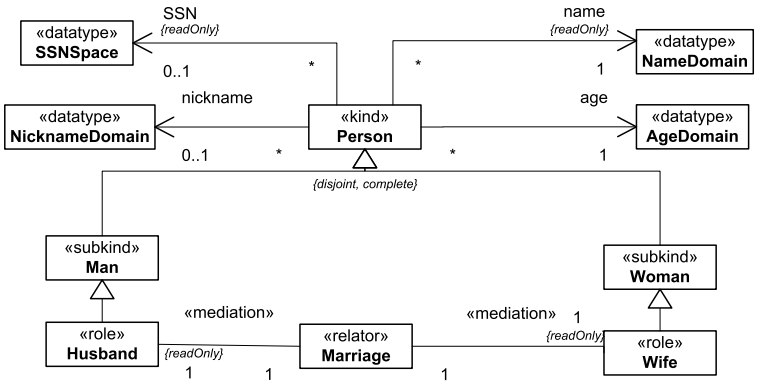
Com isso posto, o Código 18 apresenta uma especificação[5] em Ontoprolog de parte de uma ontologia escrita em OntoUML extraída de Guizzardi e Zamborlini (2014, p. 13). O diagrama em foco é o presente na Figura 41. Além das entidades originais, o exemplo é ampliado demonstrando a especificação de particulares. Observa-se no exemplo o uso do construtor próprio para Qua Individuals, que passam a funcionar como entidades na teoria, conforme definições da subseção 8.2.2.1. Por isso, a consulta `instanceof(john qua husband in marriage_j_m, X)` na teoria do Código 18 sucede da seguinte forma:

```
| ?- instanceof(john qua husband in marriage_j_m, X).
X = man qua husband in marriage ? ;
X = man qua husband ? ;
X = quaIndividualParticular ? ;
X = momentParticular ? ;
X = monadicParticular ? ;
X = concreteParticular ? ;
X = particular ? ;
X = being ? ;
no
```

[4] Ver a especificação em Ontoprolog no Apêndice C das regras que regem as meta-propriedades mencionadas, bem como o axioma Weak Supplementation.

[5] Omite-se propositalmente as relações semânticas derivadas dessa especificação, que podem ser consultadas diretamente nos casos de teste da implementação de Ontoprolog (subseção 10.2.2).

Figura 41 – Modelo conceitual de domínio em OntoUML



Fonte: Guizzardi e Zamborlini (2014, p. 13)

Código 18 – Exemplo de especificação de Relators

```
% UNIVERSALS
% Exemplo obtido de Guizzardi e Zamborlini (2014, p. 13)

person :: kind.
disjoint [man, woman] :: subkind cover person.
husband :: role extends man.
wife :: role extends woman.

marriage :: relator mediates
     husband
     and wife
     deriving married.

%% this is the same of the prior sentence
%marriage :: relator mediates
%     1 instance of husband bearing 1 of this
%     and 1 instance of wife bearing 1 of this
%     deriving married.

% PARTICULARS

john :: man.
mary :: woman.

marriage_j_m :: marriage mediates
      john qua husband
      and mary qua wife
      deriving married_with_j_m.
```

8.4.7.1 Ausência de regras do tipo $ngrule/n$ para Relators

Como os constructos sintáticos descritos na subseção 8.2.2.10 são meros açúcares sintáticos, é perfeitamente possível que se especifique individualmente cada uma das entidades e relações usadas para definir um Relator, de modo a se obter o mesmo resultado semântico. Porém, nenhuma regra do tipo $ngrule/n$ foi definida para as entidades que representam o conceito de Relator. Ou seja, embora os atalhos sintáticos possam de certa forma induzir a construção de modelos consistentes, apenas com definições $ngrule/n$ se pode garantir a consistência dos modelos das teorias Ontoprolog. Porém, regras desse tipo a respeito de especificamente de Relators não estão presentes neste texto, e são anotadas na seção 11.3 como trabalhos futuros.

8.4.8 Exemplo estendido

Para demonstrar o resultado do redesenho da UML baseado na UFO, em Guizzardi (2005, p. 360) encontra-se um diagrama integrado escrito em OntoUML que exercita diversos aspectos da ontologia de fundamentação desenvolvida pelo autor. A Figura 42 foi extraída da obra mencionada e ilustra aspectos da UFO que envolvem Relators, Relações Meronímicas (Meronymic relation), Estruturas de Qualidade (Quality Structure), Modos (Mode), Relações Formais (Formal Relation), entre outros. O exemplo é importante porque sumariza os principais resultados obtidos com a linguagem diagramática de formalização de ontologias criada, e ilustra como os fundamentos ontológicos podem ser usados para uso prático.

Aproveitando esse mesmo modelo, para exemplificar como uma ontologia baseada em UFO pode ser definida em Ontoprolog, o Código 19 contém a especificação da Figura 42 escrita com base na sintaxe textual definida neste trabalho. No código, entradas "`Nota X`" são apontamentos para as seguintes observações:

a) `Nota A`: Na Figura 42, em `Intention` há uma seta que aponta para `PriorityLevel` nomeada de `priority`. No caso, `priority` é uma Função de Atribuição (Attribute Function), uma vez que indica um Espaço Conceitual (Conceptual Space) representado por um DataType. Conforme a subseção 4.4.4, em Ontoprolog esse tipo de relação é representado diretamente por Qualidades (Quality). Por isso, no Código 19 `priority` é representado como instância de um Quality;

b) `Nota B`: Na Figura 42 o termo `location` é utilizado duas vezes para representar Funções de Atribuição (Attribute Function) que ligam `SpecialThing` e `Physical Object`, respectivamente, à `LocationCoordinates`. Porém, para destacar o fato de `location` representar um universal de um Momento Intrínseco (Intrinsic Moment), no Código 19 opta-se por representar essa relação por meio duas instâncias diferentes de Quality: `thingLocation` e `objectLocation`, caracterizam, respectivamente, `spatialThing` e `physicalObject`. Além de tornar mais mais explícita a relação intrínseca que ocorre nas instâncias do modelo universal retratado, a distinção permite tratamento individual a cada uma das relações;

c) `Nota C`: Considerando a definição de Mediation do Quadro 19, os Relators relacionam-se com Endurantes Universais (Endurant Universal) via essas relações de mediação. Além disso, conforme consta na definição de Relator Universal apresentada na entrada do Glossário da UFO, o portador dos Relators são somas mereológicas dos Qua Individuals (Qua Individual) mediados por ele. Porém, os portadores dos Relator não são representados nos modelos visuais, mas estão presentes ontologicamente. Por isso, conforme apresentado na subseção 8.2.2.10 e na subseção 8.2.2.11, Ontoprolog infere o Todo (Whole) que representa essa

Soma Mereológica (Mereological Sum), de modo que não seja necessário defini-lo explicitamente na especificação, mas que ele esteja presente nas relações semânticas. Para isso, a estratégia utilizada em Ontoprolog foi identificar o Sortal relacionado a cada um dos Papéis (Role) mediados pelo Relator, a partir do contexto sintático no momento da tradução da sintaxe nas relações semânticas (seção 7.3). No entanto, para obter o Sortal, é necessário que sejam relacionados via mediação apenas entidades anti-rígidas (Anti Rigid) do tipo `role` e `roleMixin`. Por isso, conforme o Código 33 (Apêndice C), a relação Mediation em Ontoprolog media apenas essas entidades, representada por uma entidade mais geral do que aquelas duas, chamada de `abstractRole`, conforme trecho extraído do Código 33:

```
mediation :: directedRelationship relates
        1..many instance of relator to 2..many instance of abstractRole.
```

Desse modo, porém, o Relator `Authorship` da Figura 42, que media a entidade `Performer  Artistic`, do tipo `mixin`, e `Album`, do tipo `kind`, não pode ser diretamente representado, uma vez que não media entidades universais do do tipo `abstractRole`. A solução, então, é tornar explícitos os papeis que cada entidade exerce nas relações de mediação de `Authorship`, conforme é apresentado no Código 19 por meio dos papeis `bandArtist` e `creation`, e dos Relators `individualAuthorship` e `bandAuthorship`;

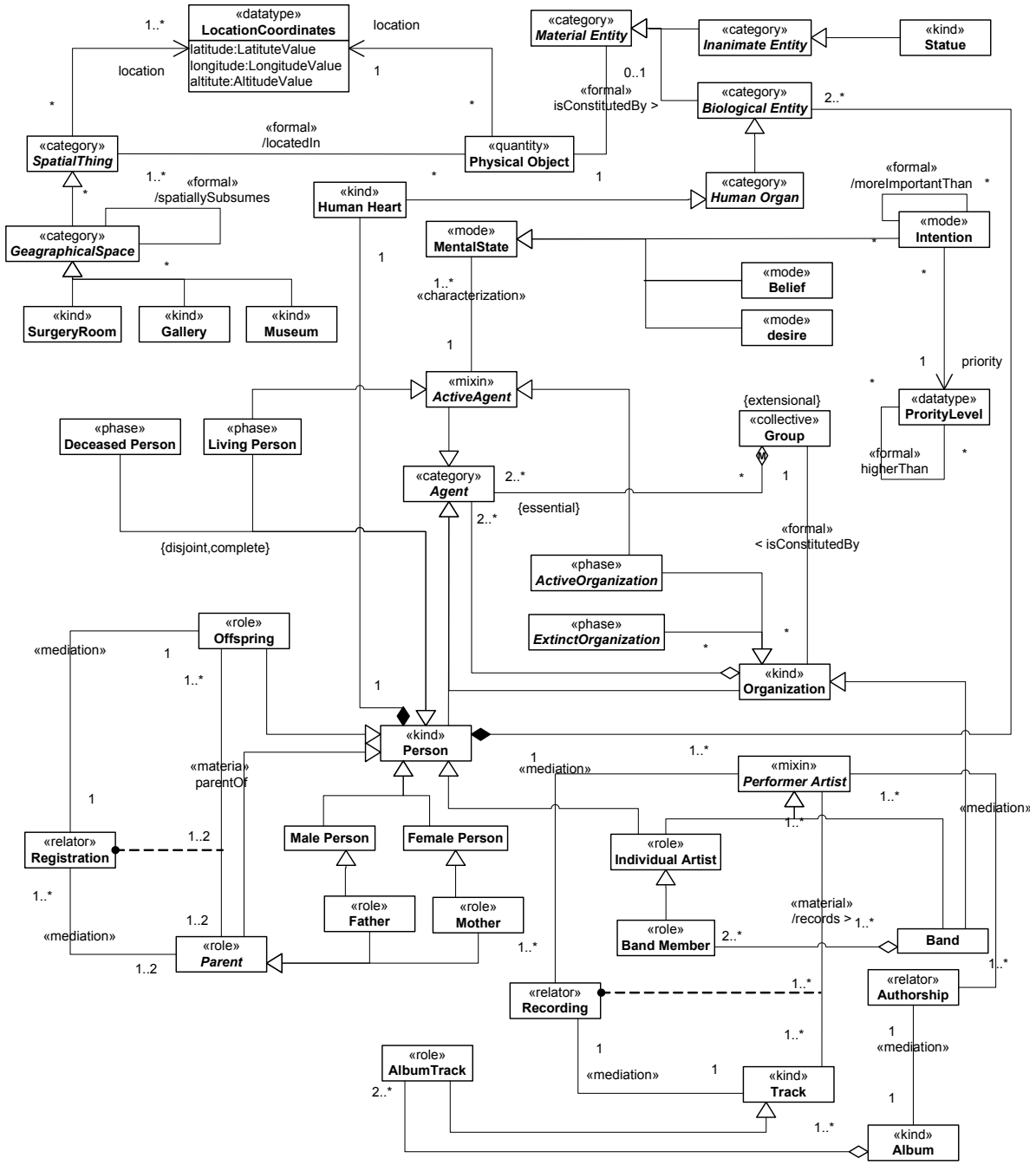
d) `Nota D`: As mesmas observações da *Nota C* se aplicam a esta, porém, direcionadas ao Relator `recording`, que é especificado por meio de `individualRecording` e `bandRecording`;

Uma observação geral a ser realizada é o uso de descritores em letras minúsculas no Código, e não letras maiúsculas, conforme consta na Figura. Isso refere-se ao fato de que em Prolog construtores iniciados em letras maiúsculas representarem variáveis.

Código 19 – Exemplo completo de especificação

```
% Exemplo obtido de Guizzardi (2005, p. 360)

% agent

agent :: category.
activeAgent :: mixin extends agent.

1..many of mentalState :: mode
      characterizes activeAgent.

[intention, belief, desire] :: mode extends mentalState.
moreImportantThan :: formal relates
       many instances of intention and
       many instances of intention.
meta derived on moreImportantThan.

priority :: quality => priorityLevel :: qualityDimension.
higherThan :: formal relates
        priorityLevel and
        priorityLevel.

priority characterizes intention. %% Nota A
```

Figura 42 – Diagrama OntoUML de uma ontologia baseada em UFO



Fonte: Guizzardi (2005, p. 360)

```
% organization and group

organization :: kind extends agent.                                          26
[activeOrganization, extinctOrganization] :: phase extend organization.

agentPartOfOrg :: componentOf relates
        2..many instances of agent                                           30
        partof 1 instance of organization.

group :: collective.
meta extensional on group.                                                   34

isConstituedBy :: formal relates
```

```
        1 instances of group
        to many instances of organization.

agentMemberOfGroup :: memberOf relates
        2..many instances of agent
        partof many instances of group.
meta essential on codomain at agentMemberOfGroup.

% material entity

materialEntity :: category.
[inanimateEntity, biologicalEntity] :: category extend materialEntity.
status :: kind extends inanimateEntity.

physicalObject :: quantity.
isConstitutedBy :: formal relates
        0..1 instances of materialEntity
        and 1 instance of physicalObject.

spatialThing :: category.

locatedIn :: formal relates
        spatialThing
        and many of physicalObject.

geographicalSpace :: category extends spatialThing. % GeagraphicalSpace
spatiallySubsumes :: formal relates
        many of geographicalSpace
        and many of geographicalSpace.
meta derived on spatiallySubsumes.
[surgeryRoom, gallery, museum] :: kind extend geographicalSpace.

%% Nota B
thingLocation :: quality => locationCoordinates
         characterizes spatialThing.
objectLocation :: quality => locationCoordinates
         characterizes physicalObject.

locationCoordinates :: qualityDomain => space [ latitude, longitude, altitute ].

% biological Entity

biologicalEntityCompOfPerson :: nonshared componentOf relates
         2..many instances of biologicalEntity
         partof person.

humanOrgan :: category extends biologicalEntity.
humanHeart :: kind extends humanOrgan.

humanHeartCompOfPerson :: nonshared componentOf relates
        1 of humanHeart
        partof 1 of person.

% person
person :: kind extends agent.

% living and deceased
[deceasedPerson, livingPerson] :: phase extend person.
livingPerson extends activeAgent.

% registration
disjoint [malePerson, femalePerson] :: subkind extend person.

[parent, offspring] :: role extends person.
meta abstract on parent.
father :: role extends [malePerson, parent].
mother :: role extends [femalePerson, parent].

registration :: relator mediates
     1..2 of parent bearing 1..many of this
     and 1 of offspring bearing 1 of this
     deriving parentOf.

% artists
performerArtist :: mixin.

individualArtist :: role extends person.
bandMember :: role extends [individualArtist, performerArtist].

band :: subkind extends [performerArtist, organization].
memberOfBand :: componentOf relates
```

```
     2..many of bandMember
     partof 1..many of band.

[album, track] :: kind.
albumTrack :: role extends track.
trackOf :: componentOf relates
     2..many of albumTrack
     and 1..many of album.

% authorship - Nota C

%authorship :: relator mediates
%     1..many of performerArtist bearing 1..many of this
%     and 1 of creation bearing 1 of this. % ???

authorship :: relator.
meta abstract on authorship.
bandArtist :: role extends band.
creation :: role extends album. % ??? Resolve o problema de album :: kind no relator

individualAuthorship :: relator mediates
     1..many of individualArtist bearing 1..many of this
     and 1 of creation bearing 1 of this. % ???
bandAuthorship :: relator mediates
     1..many of bandArtist bearing 1..many of this
     and 1 of creation bearing 1 of this. % ???

[individualAuthorship, bandAuthorship] extend authorship.

% recording  - Nota D

%recording :: relator mediates
%     1 of albumTrack bearing 1 of this
%     and 1..many of performerArtist bearing 1..many of this
%     deriving records.

recording :: relator.
meta abstract on recording.

individualRecording :: relator mediates
     1 of albumTrack bearing 1 of this
     and 1..many of individualArtist bearing 1..many of this
     deriving individualRecord.
bandRecording :: relator mediates
     1 of albumTrack bearing 1 of this
     and 1..many of bandArtist bearing 1..many of this
     deriving bandRecord.

[individualRecording, bandRecording] extend recording.

:- otp_compile,
   check_semantics.
```

Como ganhos diretos obtidos pela especificação da ontologia em Ontoprolog, pode-se destacar:

a) *Validação do modelo realizada automaticamente pelas definições lógicas da meta-teoria*: a validação do modelo concreto do Código 19 é realizada automaticamente pelas regras lógicas criadas na definição da UFO em Ontoprolog, conforme constam no Apêndice C. Ou seja, da própria especificação formal da UFO pode-se verificar a consistência de instâncias daquela teoria. Isso é possível devido à estratégia de metamodelagem utilizada em Ontoprolog: teorias mais gerais são usadas como um conjunto de regras (ou restrições) aplicáveis às teorias mais específicas;

b) *Deduções e inferência de entidades*: conforme abordado pela `Nota C`, a definição da sintaxe específica da UFO em Ontoprolog contempla a inferência

de entidades ontológicas a partir de regras lógicas. Por exemplo, o predicado *dcharacterization*/2 (Definição formal 8.2) pode ser usado para obter a entidade Whole que é o portador do Relator `bandAuthorship`, do seguinte modo:

```
| ?- dcharacterization(bandAuthorship, A).
A = bandAuthorship_collective ? ;
```

No caso, a entidade `bandAuthorship_collective` é a instância de Coletivo (Collective) que é caracterizado por `bandAuthorship`. Da mesma forma, o predicado *drel*/5 (Definição formal 6.12) pode ser usado para listar as instâncias de Relações Meronímicas (Meronymic relation) R e as entidades que participam como partes P de `bandAuthorship_collective`:

```
| ?- drel(R, P, _, bandAuthorship_collective, _).
R = bandArtist_partof_bandAuthorship_collective,
P = bandArtist ? ;
R = creation_partof_bandAuthorship_collective,
P = creation ? ;
R = bandAuthorship_charac_bandAuthorship_collective,
P = bandAuthorship ? ;
no
```

Dezenas de outras consultas podem ser definidas de modo similar àquelas apresentadas na subseção 8.3.1 e exemplificadas nesta seção. Essas definições representam predicados a respeito dos modelos produzidos pelas relações semânticas que denotam o significado das especificações das ontologias descritas em Ontoprolog;

c) *Declaração de instâncias*: complementando o exemplo apresentado na subseção 8.4.6, que demonstra que é possível incorporar qualquer predicado de Programação em Lógica em programas que contém especificações Ontoprolog, o Código 20 contém breve especificação de instâncias de `gallery` e do DataType `locationCoordinates` presentes na teoria descrita no Código 19. Esse tipo de especificação reforça a característica abordada no Item a) referente à capacidade de Ontoprolog de metamodelar teorias com base em ontologias gerais. Além disso, o excerto exemplifica o uso da sintaxe específica de Quales (Quale).

Código 20 – Exemplo de instâncias de Quales

```
:- include('guizzardi2005p360.pl').

gallery1 :: gallery.
loc1 :: thingLocation <= quale [latitude <= 1, longitude <=2, altitute
    <=3] from locationCoordinates
        inheresin gallery1.
```

8.5 Fechamento

As extensões apresentadas neste capítulo aperfeiçoam a linguagem proposta para Ontoprolog de modo a atingir concisão nas sentenças que referem-se à especificação de discursos sobre ontologias com base na UFO-A, conforme apresentados pelos exemplos

da seção 8.4. Aprimoramentos futuros podem ser adicionados à linguagem, especialmente quando outras ontologias de fundamentação forem incorporadas, como a UFO-B, por exemplo. É importante ressaltar que, conforme discutido na seção 8.4, apenas regras mais específicas devem ser definidas para a validação dos modelos de entidade, ao passo que do ponto de vista sintático, construções sintáticas mais gerais podem ser incorporadas com intuito de ampliar as frases sintáticas válidas.

Idealmente, a sintaxe poderia ser expandida até o limite de se tornar a própria linguagem natural. Nesse sentido, pesquisas sobre "ontologias automáticas", como as baseadas em linguística computacional propostas por Gottschalg-Duque (2005), poderiam ser combinadas com a característica de fundamentação da UFO, de modo que análises automáticas de textos possam extrair ontologias bem fundamentadas. No entanto, não há elementos suficientes para que se creia que esse limite teórico possa ser atingido, uma vez que a proposta de Guizzardi (2005), em que a ontologia é cognitivamente bem fundamentada, vai na contra-mão da ideia geral que se possa extrair ontologias coerentes de textos em linguagem natural. De todo modo, entende-se que esse tipo de integração é alvo natural a ser atingido no âmbito de Inteligência Artificial, mesmo que não seja possível, de fato, alcançá-lo de forma absoluta. De todo modo, pesquisas como as descritas neste trabalho podem evoluir para a busca desse tipo de resultado, principalmente devido a própria "vocação" de Prolog referente à inteligência artificial.

Quanto aos resultados obtidos neste capítulo, descam-se a capacidade de validar modelos de ontologias de domínio com base na própria formalização da UFO em Ontoprolog; a possibilidade de se realizar deduções e inferência de entidades ontológicas a partir de especificações base; e a declaração de instâncias de uma teoria de modo sintaticamente otimizado para a UFO, conforme apresentado e discutido na subseção 8.4.8.

9 Modalidades de MProlog com Ontoprolog

A linguagem de Ontoprolog apresentada nos capítulos anteriores denota teorias clássicas de descrição de ontologias. Entretanto, a semântica atribuída a essa linguagem é sustentada por um fragmento da Lógica de Primeira Ordem que pode ser estendido para contemplar expressões de contextos modais. Nessa linha, este capítulo apresenta resultados obtidos a partir da integração do arcabouço criado por Ontoprolog nos capítulos anteriores com o paradigma de Programação em Lógica Modal oferecido por MProlog (seção 3.2). Os resultados consistem em três estratégias diferentes para extensão da expressividade originalmente construída para Ontoprolog. Cada proposta é uma das combinações possíveis de incorporação de expressões modais sobre componentes da linguagem: a) modalidades apenas no âmbito das sentenças; b) modalidades somente nas teorias; e c) modalidades nas sentenças e nas teorias de Ontoprolog. Os resultados se baseiam no fato de terem sido propostas traduções de sentenças (especificações) em relações semânticas (teorias) em que ambos os componentes podem ser regidos por sistemas lógicos diferentes.

As seções seguintes descrevem como as construções dos capítulos anteriores podem ser alteradas para atingir os objetivos apresentados. Na seção 9.1 discute-se as características e ganhos obtidos em termos de poder expressivo com a substituição da lógica subjacente às especificações sintáticas da linguagem[1]. Nessa seção apresenta-se como as sentenças (sintáticas) da linguagem podem ser escritas no âmbito de contextos modais, e utilizadas como insumos para produção de teorias clássicas comuns de Ontoprolog. A seção 9.2, por sua vez, avalia uma estratégia que visa alterar a sintaxe da linguagem de modo que seja possível anotar a modalidade a ser embutida como contexto modal nas teorias denotadas pelas especificações. Porém, nesse método, as sentenças são regidas por um sistema não modal. Já a seção 9.3 mostra como as estratégias anteriores podem ser combinadas, no âmbito da Engenharia Lógica, para compor sistemas que incorporem poder expressivo tanto nas especificações quanto nas teorias. Como encerramento do capítulo, a seção 9.4 consiste em comentários a respeito das abordagens adotadas para incorporar expressões modais à linguagem de Ontoprolog.

Embora a implementação da semântica alética originalmente proposta para a UFO seja alvo de interesse de pesquisa, ressalta-se que neste capítulo apresenta-se resultados de investigações com foco em outros tipos de semântica, especialmente aquelas com maiores propensões a favorecer o alcance dos objetivos traçados na seção 1.2. Dessa forma, o foco de estudo deste capítulo não se restringe à aplicação daquela ontologia de fundamentação.

[1] A proposta de substituição da lógica subjacente de Ontoprolog é apresentada originalmente na seção 5.2.

9.1 Sentenças modais

É possível distinguir ao menos dois componentes do arcabouço proposto à Ontoprolog: a *especificação* (Definição 5.1) e a *teoria* (Definição 5.2). A especificação refere-se ao discurso em si[2], e não exatamente ao que o discurso significa ou representa. É claro que espera-se que o discurso seja significativo do ponto de vista ontológico, tanto é que são propostas traduções entre sentenças das especificações e relações semânticas que denotam as *teorias*. Entretanto, considerando uma visão semiótica, o discurso é uma codificação de uma ideia sobre um referente (seção 5.3). Em consequência, o discurso não se confunde com a ideia, nem com o referente. Haja vista disso, parece lícito desenvolver a ideia de *modalidades sobre o discurso*, no sentido de modalidades *a respeito do discurso em si*, e não modalidades do que o discurso se refere ou do que ele trata. Portanto, a primeira estratégia de incorporação de modalidades em Ontoprolog se sustenta sobre essa ideia, de modo que se apresenta uma forma de incorporar modalidades sobre as sentenças dos discursos sem que essas modalidades invadam o espaço semântico a respeito do que esses discursos se referem. Ou seja, explora-se o caso em que há dois sistemas lógicos diferentes: um sistema modal que rege as sentenças, e um sistema clássico padrão, que rege a teoria produzida por essas sentenças. Do ponto de vista de aplicação, um empreendimento de Engenharia Lógica (seção 2.4) poderia utilizar essa estratégia para construir sistemas que integrem e gerenciem discursos de vários agentes, ou ainda, que avaliem modos (aléticos, epistêmicos, temporais...) a respeito de discursos sobre ontologias e de fatos provenientes de bases de dados. As possibilidade de aplicação são diversas, principalmente considerando a pluralidade de sistemas modais oferecidos por MProlog (seção 3.2), mas de fato não é objetivo desta seção exauri-las, embora se explore aspectos de alguns deles.

Por Ontoprolog ser projetada com base em operadores, toda especificação composta por sentenças nas formas sintáticas descritas no Capítulo 7 e na seção 8.2 representa um programa Prolog. Por isso, a integração de Ontoprolog com outros arcabouços da Programação em Lógica pode ser realizada de modo prosaico, ao menos do ponto de vista sintático. Notadamente, a integração sintática de especificações Ontoprolog com modalidades do MProlog pode ser dada nos moldes que seguem. Considerando as formas sintáticas de programas modais apresentadas na subseção 3.2.2:

```
Context : (Head :- Body).
```

```
Head :- Body.
```

em que `Context` é uma lista que representa modalidades, `Head` ou possui a forma E ou $\nabla : E$, sendo que E é um átomo clássico Prolog, e ∇ é uma lista contendo uma ou mais modalidades de algum dos sistemas modais de MProlog, e considerando φ um esquema para uma sentença clássica Ontoprolog, então E pode conter qualquer φ, uma vez que

[2] No caso, os discursos sobre ontologias.

toda sentença clássica Ontoprolog é um átomo clássico de Prolog.

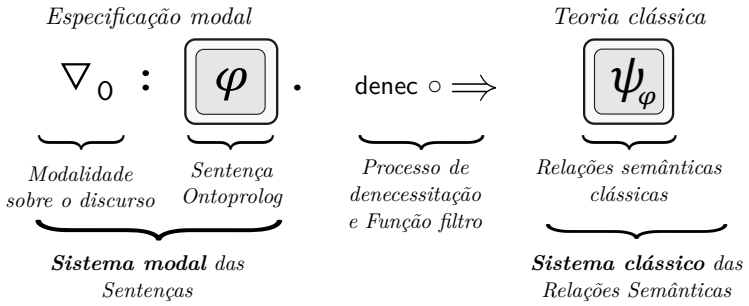
Pelo fato de as cláusulas de MProlog serem escritas com contextos modais, não é possível unificar uma variável com todo Fato ou cláusula unitária (Definição 3.15) indexado por uma modalidade. Ou seja, uma consulta como `mcall([M] : X)`, ou mesmo `mcall([pos(_)] : X)`, não faz com que o provador instancie em `X` os contra-exemplos de cláusulas presentes na base de fatos e de regras. Esses tipos de consulta são avaliadas com variáveis quantificadas existencialmente, o que faz com que o programa apenas responda *Sim*, no caso de existir *ao menos um fato* no contexto modal indicado, e *Não* no caso oposto. Por isso, para listar as afirmações em um determinado contexto modal, é necessário definir essas asserções como instâncias de um predicado específico. Dessa forma, define-se `otp/1` como o predicado que contém as instâncias de sentenças Ontoprolog (φ) contextualizadas com operadores modais no âmbito de um sistema modal.

Doravante nesta e nas seções subsequentes deste capítulo, o predicado `otp/1` será utilizado para indicar um caso φ de uma Sentença Ontoprolog, em que φ ocorre como `otp(`φ`)`. Por sua vez, ψ_φ é utilizada para indicar uma ou mais relações semânticas de $\mathcal{H}$ da estrutura $\mathcal{OT}$ (Capítulo 6) produzida como resultado de uma Função filtro (Definição 7.1), indicada por $\Longrightarrow$, que realiza tradução conservativa da instância φ da linguagem de Ontoprolog. Esses símbolos são utilizados na Figura 43, e em figuras posteriores deste capítulo, para descrever as diferentes formas de composição dos sistemas lógicos de Prolog e MProlog com Ontoprolog. Os retângulos nas figuras não são constructos das linguagens formais envolvidas, mas tratam-se de meros destaques utilizados no auxílio didático da apresentação dos argumentos discutidos no texto ilustrados nas figuras.

A Figura 43 descreve o caso discutido nesta seção em que há no programa dois sistemas lógicos diferentes, um que rege a especificação e outro a teoria produzida por essa especificação. No caso, a especificação é realizada com expressividade modal, em que ∇_0 representa alguma lista de operadores modais de algum sistema modal específico de MProlog. Apenas um único sistema modal pode ser utilizado para as sentenças. Porém, casa sistema modal pode conter diferentes tipos de modalidades, com semânticas específicas, conforme o caso. A lista ∇_0 refere-se à *modalidade sobre o discurso*, por isso o escopo dos operadores modais são as sentenças φ. O símbolo `:` é o operador padrão de MProlog que indica que à direita está a cláusula de Horn regida pela lista de modalidades circunscritas à esquerda do operador. Como na estratégia de integração modal discutida nesta seção as modalidades dizem respeito à sentença como um todo, e não ao conteúdo semântico referente a sentença, a função filtro $\Longrightarrow$ é exatamente aquela criada para traduzir sentenças em relações semânticas. Porém, esse filtro, conforme definido no Capítulo 7, tem como entrada uma sentença clássica, sem indicação de modalidade. Por isso, um programa modal, digamos `denec`, deve ser definido como um aprimoramento do algoritmo Σ (subseção 7.3.3) para selecionar as sentenças que *interessam ser traduzidas*, de modo a remover o contexto

modal delas e, dessa forma, aplicá-las sobre a função $\Longrightarrow$. Esse processo é equivalente à produção de uma função (denec) que seleciona e denecessita as sentenças, e à composição ($\circ$) dessa com a função $\Longrightarrow$ que, por sua vez, produzirá uma teoria clássica de Ontoprolog. Esse procedimento resultará exatamente no mesmo sistema produzido nos capítulos anteriores. Entretanto, sofisticou-se o modo de lidar com as sentenças ao encapsulá-las em um sistema modal, mas não se aumentou a expressividade sobre o *referente* dos discursos. As sentenças *que interessam*, selecionadas por denec, representam uma abstração da aplicação de lógica modal para interpretar discursos.

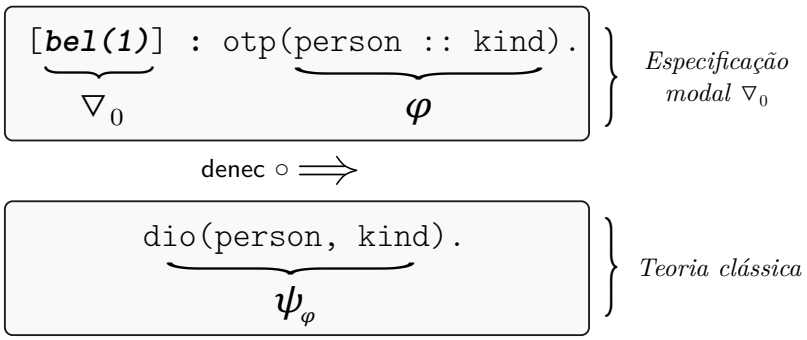
Figura 43 – Sentença modal com semântica clássica



Fonte: Os autores

Para exemplificar a abordagem desenvolvida, a Figura 44 apresenta uma sentença simples encapsulada em um contexto modal contido em uma lista de modalidades ∇_0 de um sistema modal de MProlog especificamente escolhido[3] para reger a *Especificação* ∇_0. No caso, trata-se de um sistema em que a modalidade `bel(1)` é válida. Na ilustração o predicado `otp/1` é usado para identificar a sentença Ontoprolog φ. No âmbito do quadro *Teoria clássica*, apresenta-se um possível resultado da tradução da especificação, indicado por ψ_φ. A definição concreta de denec não é apresentada.

Figura 44 – Exemplo de instanciação de sentença modal com semântica clássica



Fonte: Os autores

[3] Apresentar o sistema escolhido é irrelevante aos propósitos da exposição em curso, por isso opta-se por omiti-lo por enquanto.

Com a estratégia apresentada, as especificações de discursos sobre ontologias podem ser descritas com contextos modais sem a necessidade de qualquer ajuste adicional no sistema desenvolvido com Ontoprolog. Para discutir ainda mais e exemplificar essa abordagem, apresenta-se no Código 21 e no Código 22 uma especificação Ontoprolog cujas sentenças são formalizadas considerando contextos modais conforme desenvolvido nesta seção.

De acordo com a subseção 3.2.3, os programas MProlog consistem em ao menos duas partes: um programa clássico (Código 21) e um programa modal (Código 22). Na parte modal do exemplo, as três primeiras linhas consistem na especificação do sistema modal a ser utilizado pelo arcabouço dedutivo do MProlog. Isso é realizado por meio dos predicados `calculus/1` e `set_option/2`. A forma sintática a ser utilizada na especificação do contexto modal varia em relação ao sistema dedutivo escolhido. No caso, opta-se pelo sistema $KD45_m$, descrito na subseção 3.2.1 como um sistema adequado para interpretação epistêmica de multiagentes, em que os agentes não possuem acesso à base de conhecimento um dos outros. Nesse sistema, as modalidades possuem as formas sintáticas `bel(i)` γ e `pos(i)` γ, e são interpretadas como "o agente i acredita que γ é verdadeiro" e "γ é considerado possível pelo agente i", respectivamente[4], em que γ é uma sentença Prolog qualquer.

Seguindo o Código 22, entre as linhas 5 e 8 encontra-se um conjunto de afirmações que descrevem o discurso proferido pelo agente identificado como `1`. Conforme já apresentado, o predicado `otp/1` é utilizado para determinar as sentenças da forma φ de Ontoprolog, para $\varphi \subseteq \gamma$. Nas linhas 6 e 7 o agente `1` afirma a certeza que possui sobre as relações das entidades identificadas por `ser` e `pessoa`. Já na linha 8, as relações ontológicas que `vivo` e `morto` possuem são afirmadas de forma hesitante, ou apenas possíveis. O mesmo ocorre com o agente identificado por `2`. Entre as linhas 10 e 13, o discurso desse agente é formalizado e, novamente, uma afirmação hipotética possível é realizada sobre as relações ontológicas de `vivo` e de `morto`. No caso, é fácil perceber que os agentes divergem quanto a meta-classe dessas entidades, em que o primeiro acredita que `vivo` e `morto` são instâncias de `category` e o segundo, que são instâncias de `kind`. Por fim, na linha 16, exemplifica-se como definir relações entre os agentes modais. No caso, define-se a crença de um agente hipotético identificado por `10`, que crê seguramente em tudo que ao menos um agente crer com certeza. Ou seja, a crença do agente `10` denota o consenso entre os agentes. Trata-se de um tipo de axioma de interação entre as modalidades que denotam o conhecimento dos agentes `1` e `2`. É importante lembrar que, devido à presença do axioma (D) (Tabela 2), as afirmações γ no contexto de uma `[bel(i)] :` γ são consistentes com `[pos(i)] :` γ, ou seja, tudo aquilo que é conhecido certamente é também conhecido possivelmente. Ou, posto de outra forma, o conhecimento do agente é consistente, pois ele não crê simultaneamente

[4] Tratam-se da versão textual para as modalidades universal ($\Box$) e existencial ($\Diamond$) com interpretação doxástica. Ver também seção 3.2 a respeito de interpretações epistêmicas e doxásticas.

em uma afirmação e na negação dela[5].

Já o Código 21 funciona como um controlador do programa modal. A linha 2 incorpora no programa em tela o arcabouço do Ontoprolog-Modal, que consiste na inclusão da biblioteca do MProlog[6] e do Ontoprolog-Clássico. A linha 5 consulta a biblioteca de cálculos modais epistêmicos contida no arquivo `belief.cal`[7]. É nessa biblioteca que o sistema $KD45_m$ e outros estão definidos. A linha 8 efetivamente consulta o programa modal descrito pelo Código 22. Entre as linhas 10 e 22 define-se o predicado `otp_m_compile/0`, que consiste no seguinte:

a) produzir uma lista LMA com as sentenças Ontoprolog afirmadas pelo agente `1`;

b) produzir uma lista LMB com as sentenças Ontoprolog afirmadas pelo agente `2`;

c) produzir um conjunto $LMAB$ com a união das listas LMA e LMB. Por se tratar de um conjunto, $LMAB$ não possui afirmações repetidas;

d) produzir uma linha LC com todas as sentenças clássicas presentes no contexto do programa em tela. Isso é necessário para resgatar a base de fatos que contém a Definição 5.10 (Metateoria de *Hypertypes*) (Apêndice A) subjacente a toda especificação Ontoprolog;

e) produzir um conjunto $LCMAB$ com a união de $LMAB$ e de LC;

f) compilar o conjunto de sentenças $LCMAB$ como uma especificação Ontoprolog-Clássico.

Por fim, nas linhas 23 e 24 efetivamente executa-se o predicado `otp_m_compile/0` e o predicado `check_semantics/0` (subseção 10.2.1), utilizado para verificar a consistência do modelo produzido pela especificação. O Código 23 contém a saída da execução do Código 21.

Código 21 – Exemplo de Ontoprolog-Modal: arquivo Prolog

```
% Include ONTOPROLOG-MODAL libraries
:- include('../../../ontoprolog/ontoprolog-m').

% Consult calculi
:- consult_calculi('../../../ontoprolog/modal/mprolog2-custom/belief.cal').

% Consult modal program
:- mconsult('modal_theory_integracao_agentes.mpl').

% denec
otp_m_compile :-
   write('Getting candidate modal sentences...'), nl,
   findall(A, mcall([pos(1)] : otp(A)), LMA),
   findall(B, mcall([pos(2)] : otp(B)), LMB),
   util_append(LMA, LMB, LMAB),

   write('Getting candidate classical sentences...'), nl,
```

[5] O agente consistente é uma característica implicada pelo axioma (D) e pela interdefinição dos operadores modais $\Diamond$ (`pos`) e $\Box$ (`bel`), conforme é apresentado na subseção 3.2.1.

[6] O MProlog é adaptado para que possa ser utilizado com o arcabouço de Ontoprolog, conforme descrito na seção 10.5.

[7] Esta e outras bibliotecas são apresentadas na subseção 3.2.3.

```
   setof(S, otp_classic_sentence(S), LC),
   util_append(LMAB, LC, LCMAB),

   otp_compile(LCMAB).

:- otp_m_compile,   % denec
   check_semantics.
```

Código 22 – Exemplo de Ontoprolog-Modal: arquivo MProlog

```
% Modal Epistemic Logic
:- calculus cKD45m.
:- set_option(current_calculus, cKD45m).

% Agent 1's ontological discourse specification
[bel(1)] : otp(ser :: kind).
[bel(1)] : otp(pessoa :: subkind extends ser).
[pos(1)] : otp(disjoint [vivo, morto] :: category).

% Agent 2's ontological discourse specification
[bel(2)] : otp(ser :: kind).
[bel(2)] : otp(pessoa :: subkind extends ser).
[pos(2)] : otp(disjoint [vivo, morto] :: kind).

% Artificial agent integrating consensus among agents
[bel(10)] : opt(S) :- [bel(_)] : otp(S).
```

Código 23 – Saída da execução do Código 21

```
Getting candidate modal sentences...
Getting candidate classical sentences...

Starting Ontoprolog compilation...
Retracting current semantic terms...
Checking syntax of Ontoprolog candidate sentences...
Compiling sentences ...
  iteration 1...
All sentences are syntactically well-formed.

Asserting semantic terms produced by sentence decoding... 644
Asserting semantic expansions helpers...
   asserting semantic_expansion_dio_hierarchy/2...192
   asserting semantic_expansion_deo_hierarchy/2...401
Asserting semantic expansions...587
Asserting extensionof_irreflexive/2 relations...401
Asserting instanceof/2 relations...226
Asserting drefine/3 relations...8
Asserting propertyon/2 relations...74


Compilation of the Ontoprolog specification was successful.

Checking semantics using negative rules of the current Ontoprolog theories...
-------------------------------------------------
Checking ngrule/3...
 morto:
   -> "morto" instantiates more than one UFO meta-type: "[category,kind]"
    (dio_more_than_one_ufo_entity)

 vivo:
   -> "vivo" instantiates more than one UFO meta-type: "[category,kind]"
    (dio_more_than_one_ufo_entity)
```

A saída da execução do programa demonstra que a verificação de consistência baseada nas regras RNP da forma *ngrule/n* (subseção 6.3.1) identifica um problema de consistência com os conceitos de `vivo` e de `morto`, conforme era esperado. No caso, tratam-se de contra-exemplos à RNP descrita na Definição formal 8.10 (*dio_more_than_one_ufo_entity*). Esse tipo de resultado demonstra a capacidade de se realizar *verificação global de inconsistência* nas teorias denotadas pela conjunção das especificações individuais do discurso de cada agente, no caso, os agentes identificados como `1` e `2`. O mecanismo auxilia no processo

de obtenção de acordos a respeito da realidade, na medida que expõe a inconsistência nos discursos dos agentes.

É na possibilidade de se definir um predicado como `otp_m_compile/0` que reside uma das principais características dessa abordagem de integração entre os arcabouços lógicos, uma vez que essa definição demonstra que é possível construir predicados que identifiquem, produzam e controlem as sentenças da especificação a serem utilizadas para algum raciocínio especializado. Nesse caso, seria possível definir novos comportamentos para o predicado em tela, baseado no exemplo apresentado. De fato, `otp_m_compile/0` é a definição de `denec` da Figura 43 e da Figura 44.

Quanto ao arcabouço de deduções modais, argumenta-se que o sistema $KD45_m$ seja adequado ao propósito de formalizar ontologias com múltiplos agentes quando eles especificam concorrentemente discursos a respeito de uma única Ontologia de domínio (Definição 5.21). Os agentes observam o mundo e fazem afirmações sobre esse mundo com dois graus confiança: ou estão certos sobre o que dizem, caso em que as afirmações são realizadas com o contexto `bel`, ou estão conscientemente inseguros sobre suas afirmações, caso em que o contexto `pos` é utilizado. Interdefinições entre as modalidades e entre os agentes, como as apresentadas no Código 22, possibilitam relacionar, por meio de regras de inferências, os diferentes discursos. Evidentemente, devem ser realizados estudos a respeito das propriedades lógicas obtidas com cada tipo de axioma de interação entre os índices modais (no caso, entre os agentes), uma vez que definições como as apresentadas alteram as regras básicas do sistema modal em tela.

Em complemento às perspectivas da abordagem, consultas específicas à base de dados do MProlog podem ser realizadas de modo a identificar sentenças com base em critérios que considerem os contextos modais. Por exemplo, a consulta definida que sucede "`| ?-`" no próximo código pode ser utilizada para listar apenas as afirmações realizadas no contexto da modalidade `pos`. Essa consulta pode ser lida do seguinte modo: "prove que não existe um `X` em `otp(X)` que tenha sido afirmado como possível, mas não como necessário, por qualquer dos agentes `i`". A duas linhas seguidas por "`X = `" são os contra-exemplos encontrados pelo provador a respeito da solicitação de prova e que refletem exatamente as instâncias de fatos da base de afirmações do Código 22. Essas consultas podem ser utilizadas como base para a definição de predicados como o `otp_m_compile/0`.

```
| ?- mcall([pos(_)] : otp(X)), \+ mcall([bel(_)] : otp(X)).
X = disjoint[vivo,morto]::category ? ;
X = disjoint[vivo,morto]::kind ? ;
```

Com fulcro nessa discussão, evidencia-se que a integração sintática do arcabouço de Ontoprolog com MProlog é realizada de forma transparente, sem necessidade de qualquer adaptação no arcabouço da linguagem desenvolvida nesta pesquisa. Além disso, a abordagem permite obter um dos resultados esperados referente à integração de discursos de diferentes agentes. Esse resultado, no entando, é aprimorado nas seções seguintes.

9.2 Teorias modais

A integração de Ontoprolog com MProlog descrita na seção 9.1 refere-se a um tratamento direto de inclusão de operadores modais sobre as sentenças bases de Ontoprolog e de raciocínio modal sobre essas sentenças com algum dos sistemas de MProlog. Conforme apresentado, nessa abordagem nenhuma alteração no arcabouço semântico de Ontoprolog é necessário, uma vez que o raciocínio de MProlog é utilizado apenas para selecionar e produzir sentenças que serão, por sua vez, interpretadas pelo motor clássico de Ontoprolog[8].

Aquela estratégia de integração é suficiente aos objetivos desta tese, pois permite que sejam identificadas inconsistências globais em especificações de ontologias criadas por diferentes arquitetos, além de permitir tratamento modal às sentenças que refletem a especificação dos discursos sobre ontologias.

Porém, esse método não permite incorporar raciocínios e restrições ontológicas nas *relações semânticas* com base em modalidades lógicas, ou seja, não permite introduzir contextos modais a respeito do *referente* dos discursos. Isso ocorre porque a lógica subjacente de Ontoprolog é uma lógica clássica não modal, e essa base lógica que sustenta as teorias denotadas pelas especificações não é alterada naquele método. Ou seja, quando as sentenças de Ontoprolog são especificadas no âmbito de programas em Lógica sustentados por um sistema modal, o raciocínio modal pode ser utilizado de forma limitada somente a essas sentenças, o que não contempla automaticamente a semântica da linguagem. A semântica resultado da tradução sintática descrita na seção 7.3 continua sendo clássica e denota afirmações denecessitadas (seção 5.9). Baseando-se na Figura 23 (seção 5.2), a estratégia da seção 9.1 substitui o "Sistema lógico das Sentenças" por um sistema modal, mas mantém o mesmo sistema clássico de Programação em Lógica no "Sistema lógico das Relações Semânticas", conforme ilustrado pela Figura 43 e pela Figura 44.

No caso apresentado no Código 21 e no Código 22, embora seja exemplificado como realizar consultas definidas para conhecer sentenças proferidas por determinado agente, não é possível realizar uma consulta que retorne quais são os meta-tipos das entidades afirmadas apenas pelo agente identificado como 2, por exemplo, uma vez que uma consulta desse tipo é realizado no âmbito das relações semânticas, e não das especificações sintáticas.

Diante disso, uma segunda abordagem pode adaptar o arcabouço semântico de Ontoprolog para que ele contemple afirmações modais regidas por regras baseadas em modalidades. Não obstante, MProlog pode servir de sustentação lógica para todo o arcabouço de Ontoprolog, e não apenas para as sentenças da linguagem. Isso incorpora legitimamente modalidades no significado das especificações de ontologias, e permite que regras modais específicas da UFO sejam definidas, como aquela discutida na seção 5.9.

[8] No caso, o predicado `otp_m_compile/0` exemplifica a definição de `denec` com a seleção de sentenças modais e a execução da função $\Longrightarrow$ do arcabouço clássico de Ontoprolog.

A Figura 45 ilustra uma síntese de como o arcabouço semântico de Ontoprolog pode ser ampliado para suportar teorias com expressões modais. A estratégia consiste em adotar dois sistema lógicos diferentes: um sistema clássico para as *especificações* e um sistema modal para as *teorias* produto dessas especificações. Do ponto de vista sintático, as sentenças de Ontoprolog são enriquecidas com um operador que exerce a função de anotação do contexto modal a ser utilizado na tradução da sentença para as relações semânticas que ela denota. Se comparada com a abordagem da seção 9.1 (Figura 43), destacam-se o seguinte:

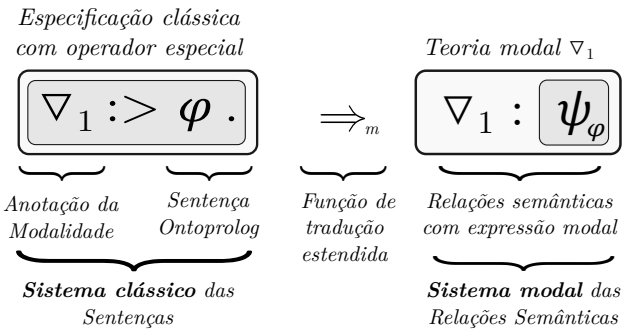
a) *sobre a semântica*: um arcabouço de definições e regras positivas e negativas deve ser definido levando em consideração cada sistema modal utilizado como lógica subjacente da *teoria*;

b) *sobre a sintaxe*: embora o sistema clássico padrão seja mantido no lado da especificação sintática, a sintaxe das sentenças e o mecanismo de tradução devem ser adaptados:

   – as sentenças φ de Ontoprolog são anotadas com contexto modal por meio do operador `:>`, em que o contexto modal ∇ está à esquerda e a sentença φ à direita. As sentenças não são modais, mas clássicas;

   – uma nova função filtro, $\Longrightarrow_m$, deve ser definida com intuito de replicar o comportamento conservativo padrão de $\Longrightarrow$, mas produzir relações semânticas anotadas com o contexto modal indicado no lado esquerdo de `:>`.

Figura 45 – Sentença clássica com semântica modal

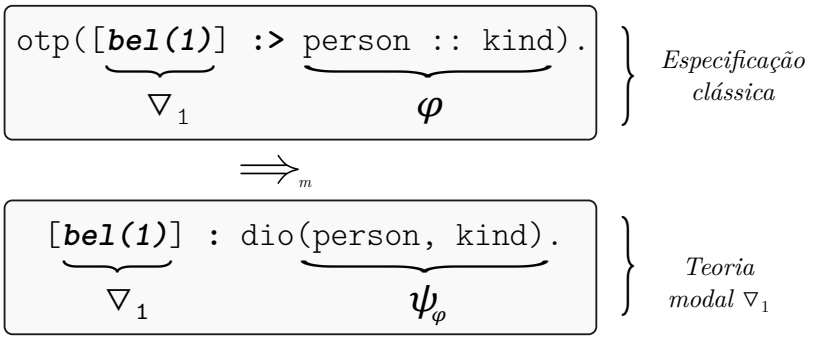


Fonte: Os autores

A Figura 46 exemplifica uma instância do modelo ilustrado na Figura 45. No caso, uma sentença clássica contida em `otp/1` é traduzida para uma relação semântica anotada com a modalidade `[bel(1)]`. A relação semântica está regida por um dos sistemas modais ∇ de MProlog. Embora no quadro superior `[bel(1)]` represente apenas uma anotação, no quadro inferior ele é de fato o contexto modal da cláusula de Horn `dio(person, kind)`.

Devido à forma como é apresentada a semântica das especificações da linguagem Ontoprolog-Clássico, a semântica de uma especificação com anotação de contexto modal

Figura 46 – Exemplo de instanciação de sentença clássica com semântica modal



Fonte: Os autores

pode ser dada no âmbito de MProlog acrescentando-se a cada relação semântica um operador modal. Assim como ocorre no paradigma clássico, essas relações – átomos na linguagem alvo (Definição 3.6) descritas por cláusulas de Horn – com contexto modal apoiam-se na *semantic embedding* (subseção 3.3.2) da lógica na qual são interpretadas, nesse caso, um dos sistemas modais providos pelo arcabouço de MProlog (listados na subseção 3.2.1). Porém, para cada sistema lógico específico, é necessário produzir um conjunto de definições e regras positivas e negativas que validem as *teorias* de modo adequado ao sistema lógico escolhido. Isso é necessário porque as regras de $\mathcal{H}$ da estrutura $\mathcal{OT}$ (seção 6.2) determinam a noção de consequência que definem o comportamento dedutivo das teorias sobre ontologias de Ontoprolog.

Desse modo, em uma teoria modal de Ontoprolog se pode identificar uma estrutura $\mathcal{OTM}_\nabla$ formada pelos elementos da estrutura $\mathcal{OT}$ em que as relações semânticas $h \in \mathcal{H}$ e as definições e regras positivas e negativas $r \in \mathcal{R}$ são precedidas de um ou mais operadores modais de ∇, na forma $\nabla : h$ e $\nabla : r$, em que ∇ é uma lista de operadores modais do sistema modal escolhido na lógica subjacente ao Programa:q

$$\mathcal{OTM}_\nabla = \langle \mathcal{C}, \mathcal{G}, \mathcal{R}_\nabla, \mathcal{H}_\nabla \rangle$$

Com base nessa construção base de estrutura semântica, pode-se construir regras RNP e definições positivas adequadas ao sistema modal escolhido e ao fim pretendido com a estrutura semântica das teorias a serem descritas. Ao alterar a lógica subjacente, de clássica para um dos sistemas modais de MProlog, necessariamente deve-se alterar os predicados que determinam as interpretações válidas.q

Por exemplo, quando o sistema $KD45_m$ é adotado como lógica subjacente às teorias, cada valor $i \in m$ que indexa os operadores modais `bel` e `pos` representam a posição de crença de agentes diferentes, como no caso apresentado pela seção 9.1. Já quando o sistema $KDI4_s$ (subseção 3.2.1) é utilizado, o índice $i \in s$ nos operadores modais denota o grau de confiabilidade da proposição contextualizada pela modalidade.

Por isso, dependendo do sistema modal adotado, as regras negativas que regulamentam as relações semânticas – propostos como casos de $ngrule/n$ (subseção 6.3.1) e descritos na subseção 6.4.3 – bem como as próprias definições positivas a respeito de teorias – apresentadas na subseção 6.3.2 e detalhadas na subseção 6.4.1 – devem ser reescritas para que possam verificar a consistência das relações semânticas levando em consideração não apenas uma interpretação modal específica, mas o conjunto de consequências lógicas que a combinação das regras daquele sistema incorporam às teorias.

Para exemplificar como as regras que regem a teorias de Ontoprolog podem ser adaptados para considerarem contextos modais, considerando o sistema modal $KD45_m$, a definição de irreflexibilidade e de assimetria simples da regra Definição formal 6.37 (Regras para $dio/2$) poderia ser reescrita conforme a Definição formal 9.1:

Definição formal 9.1 (Irreflexividade e assimetria simples de $dio/2$ em $KD45_m$)**.** As definições seguintes propagam os contextos modais `pos` e `bel` das relações bases $dio/2$ para as definições do tipo $ngrule(dio\_reflexive)$ e $ngrule(dio\_symmetric)$ no sistema $KD45_m$.

$$\begin{aligned}[pos(X)] : ngrule(dio\_reflexive) \leftarrow & [pos(X)] : dio(C, C)\\ [pos(X)] : ngrule(dio\_symmetric) \leftarrow & [pos(X)] : dio(A, B),\\ & [pos(X)] : dio(B, A)\end{aligned}$$

$$\begin{aligned}[bel(X)] : ngrule(dio\_reflexive) \leftarrow & [bel(X)] : dio(C, C)\\ [bel(X)] : ngrule(dio\_symmetric) \leftarrow & [bel(X)] : dio(A, B),\\ & [bel(X)] : dio(B, A)\end{aligned}$$

—

A transcrição sucessiva das definições e regras presentes na seção 6.4, nos moldes do que é apresentado na Definição formal 9.1, produzirá um conjunto de predicados e regras modais que poderão ser utilizadas para verificar a consistência individual de uma teoria, bem como para realizar consultas definidas sobre essa base de regras.

Do ponto de vista do tratamento da sintaxe das especificações, para que seja possível tratar modalmente as teorias de Ontoprolog ainda é necessário:

a) incluir o operador `:>` como anotação do contexto modal;

b) ajustar o algoritmo Σ de controle da tradução de sentenças sintáticas em sentenças semânticas (subseção 7.3.3) para que identifique as sentenças com anotação de contexto modal e para que o contexto Δ de relações semânticas intermediárias (subseção 7.3.2) seja produzido levando em consideração a modalidade do sistema modal vigente; e

c) definir uma Função filtro (Definição 7.1) $\Longrightarrow_m$ para tratamento de contextos modais.

A inclusão do novo operador tratada pelo Item a) é apresentada pela Definição 9.1:

Definição 9.1 (Sentença com contexto modal)**.** Seja φ uma sentença escrita na forma EBNF presente no Código 6 ou no Código 10. Então `otp(`∇ `:>` φ`)` é a uma sentença com contexto modal, em que ∇ é um contexto modal da lógica modal subjacente e `:>` é um Operador principal (Definição 7.4). —

Quanto ao Item b), conforme exposto na subseção 7.3.3, o primeiro passo do algoritmo controlador Σ é identificar as sentenças Prolog cujo predicado mais externo seja um Operador principal (Definição 7.4). Esse processo representa uma seleção na base de fatos e de regras do programa Prolog com intuito de identificar as cláusulas que possuem a forma de uma sentença. A própria definição de `:>` como um operador principal já seria suficiente para que o algoritmo identificasse as sentenças de Ontoprolog com contexto modal, de modo que o predicado `otp/1` não seja necessário. Porém, para manter a clareza na distinção de sentenças Ontoprolog padrões e de sentenças com contextos e anotações modais, opta-se por manter o predicado `otp/1`, mesmo ele sendo prescindível. Dessa forma, a única alteração em Σ consiste em, para cada sentença com contexto modal (Definição 9.1), identificar o operador `:>` e repassar à função filtro $\Longrightarrow_m$ não apenas a sentença φ à direita, mas também o contexto à esquerda do operador.

Com base nisso, no âmbito do que trata o Item c), uma função $\Longrightarrow_m$ é uma $\Longrightarrow$ que recebe sentenças clássicas Ontoprolog e que produz relações semânticas contextualizadas com uma determinada modalidade ∇. Por isso, além de receber um contexto Δ e uma sentença ou um fragmento de sentença α, a função $\Longrightarrow_m$ também recebe um contexto modal ∇. O contexto ∇ é usado como prefixo de toda relação semântica β completa produzida pela função. A disjunção dessas relações contextualizadas formam o conjunto $\mathcal{H}$ de $\mathcal{OTM}_{\nabla}$.

O contexto modal ∇ pode ser utilizado pelos predicados contidos no contexto de relações semânticas Δ (subseção 7.3.2). Dessa forma, os predicados contidos na Equação 7.1 e na Equação 7.2 poderiam ser reescritos conforme a Equação 9.1 e a Equação 7.2, respectivamente.

$$\begin{aligned}\forall T\ \forall S\ \forall X \in \Delta(\quad & \nabla : extensionof\_irreflexive(T,S) \leftarrow \nabla : deo(T,S) \\ & \nabla : extensionof\_irreflexive(T,S) \leftarrow \nabla : deo(T,X), \\ & \qquad \nabla : extensionof\_irreflexive(X,S) \quad)\end{aligned} \tag{9.1}$$

$$\forall T\ \forall S\ \forall M \in \Delta(\quad \nabla : instanceof(T, M) \leftarrow \nabla : dio(T, M)$$
$$\nabla : instanceof(T, S) \leftarrow \nabla : dio(T, M),$$
$$\nabla : extensionof\_irreflexive(M, S) \quad) \tag{9.2}$$

Uma vez realizadas essas definições, os casos de $\Longrightarrow_m$ dependentes do contexto Δ devem ser reescritos para que as consultas sejam realizadas no âmbito do arcabouço modal. Por exemplo as definições apresentadas para a tradução de Instância de caracterização (subseção 8.2.2.9), Relator universal (subseção 8.2.2.10) e discutida para a tradução de subseção 8.2.2.11, devem ser adaptadas. Já as demais definições não dependentes de consultas ao contexto Δ podem ser mantidas sem alterações. Evidentemente, uma implementação dessas traduções deve levar em consideração os predicados específicos de MProlog envolvidos nas deduções que lidam regras modais.

Na seção 9.1 apresenta-se uma forma de selecionar sentenças Ontoprolog por meio de consultas definidas modais. Trata-se da função `denec`, implementada com `otp_m_compile/0`. Porém, no âmbito da estratégia ilustrada pela Figura 45, `denec` deixa de ser necessária, uma vez que as sentenças não estão regidas por um sistema modal (e por isso não podem ser denecessitadas).

Com intuito de exemplificar as ideias apresentadas, apresenta-se uma sequência de três códigos em que se exercita um protótipo da versão modal das relações semânticas de Ontoprolog discutido nesta seção:

a) Código 24: contém o controlador Prolog que consulta os programas modais envolvidos. A primeira parte do código consiste na inclusão das regras presentes no Código 25. A parte identificada como *"Sentences in the form of extended Ontoprolog base language"* descreve uma especificação escrita com sentenças conforme a Definição 9.1. A especificação está marcada como "comentário" (cada linha está prefixada com `%`) porque a implementação concreta da tradução das sentenças com anotação de modalidade em relações semânticas modais não está de fato realizada, mas indicada. Essa implementação, que concretiza a função $\Longrightarrow_m$ em Programação em Lógica, será realizada no contexto do predicado `otp_mt_compile/0`. De toda forma, o resultado esperado dessa tradução está presente no Código 26;

b) Código 25: consiste em três categorias de regras modais criadas para o sistema $KD45_m$. As regras presentes nas parte dois e três são regras de $\mathcal{R}$ da estrutura $\mathcal{OTM}_\nabla$. Já as regras que tratam comportamento dos agentes, na primeira parte, são escopo dos axiomas base do sistema modal em tela, uma vez que governam todo o sistema modal:

- a primeira parte, identificada com *"Propagation rules of the beliefs of the agent 0"*, consiste em regras sobre os agentes de $KD45_m$. Trata-se de uma uma mudança no sistema original, em que se acrescenta uma regra de propagação aos agentes com número diferente de 0 das relações semânticas `entity/0`, `dio/2`, `deo/2` e `dd/1` quando no contexto `[bel(0)]`. Isso afirma o agente 0 como uma espécie de *agente perspicaz* em que os demais agentes acreditam (`bel`) naquilo que ele crê a respeito dos predicados indicados. Esse tipo de definição funciona como uma forma de evitar redundâncias de afirmações *grounds* (fatos) sobre aquelas relações semânticas. Porém, esse tipo de regra também é uma usurpação do sistema axiomático original, no caso $KD45_m$, de modo que é necessário investigar as consequências na lógica modal em tela com a introdução desse tipo regra. Porém, isso não é escopo deste estudo;
- a segunda parte, identificada por *"Closures"*, consiste na implementação da definição contida na Equação 9.1 e na Equação 9.2. Tratam-se dos fechos transitivos das relações `deo/2` e `dio/2`. Para se manter a clareza e abstração do exemplo, não foram empregadas as técnicas de *memoization* discutidas na Nota 6.1 (subseção 6.4.1);
- por fim, na terceira parte, identificada como *"Rigidity"*, consta a implementação de uma possível interpretação do conceito de Rigidez (Rigidity)[9,10] no âmbito do sistema $KD45_m$. Nesse sistema, uma entidade T pode ser considerada rígida se ela for instância de `kind` segundo o agente 0, e não for o caso de haver ao menos dois agentes que tenham afirmado que exista um conceito X que seja instância tanto de T quanto de U, para $T \neq U$, em que U também seja instância de `kind` conforme o agente 0. Além disso, T também é considerado rígido se ele for instância de `kind` segundo o agente 0 e não houver nenhum agente que afirme a existência de um X instância de T.

As regras contidas nesse arquivo são apenas um fragmento da versão de Ontoprolog-$KD45_m$. Uma implementação completa deve produzir uma versão modal de todas as regras descritas no Capítulo 6 e na seção 8.3;

c) Código 26: contém o programa MProlog que representa o resultado da aplicação de $\Longrightarrow_m$ sobre a especificação contida no Código 24. Essencialmente, a anotação do contexto modal das sentenças é transcrita como de fato uma modalidade no âmbito das relações semânticas de $\mathcal{H}$ que compõem a teoria denotada por aquela especificação.

Código 24 – Exemplo semântica modal: arquivo Prolog

[9] Além do Glossário da UFO, a ideia modal do conceito em foco também é tratada na seção 5.9.

[10] Não é foco deste exercício apresentar a definição original de rigidez da UFO, mas explorar uma interpretação possível considerando a semântica de múltiplos agentes.

```
% Include ONTOPROLOG-MODAL libraries
:- include('../../../ontoprolog/ontoprolog-m').

% Consult calculi belief.cal
:- consult_calculi('../../../ontoprolog/modal/mprolog2-custom/belief.cal').

% Consult modal definitions and rules (Código 25)
:- mconsult('modal_theory_rules.mpl').

%=========================================================================
% Sentences in the form of extended Ontoprolog base language (seção 9.1)
%=========================================================================

% opt([bel(0)] :> [alien, humano, nave, planeta] :: kind).
% opt([bel(0)] :> disjoint [adulto, menino] :: phase extend humano).
% opt([bel(0)] :> alien_visitante :: role extends alien).
% opt([bel(0)] :> et :: alien_visitante).
% opt([bel(0)] :> elliott :: menino).

% opt([pos(1)] :> nave_et :: nave).
% opt([pos(2)] :> nave_et :: alien).

%=========================================================================
% Compilation and semantic checking...
%=========================================================================

% Consult modal program:
otp_mt_compile:-
        % read opt/0 sentences and translate them to semantic relations...
        % it will produce (Código 26):
        mconsult('modal_theory_rigidity_belief.mpl').

check_semantics.  % check ngrules...

:- otp_mt_compile,
   check_semantics.

%=========================================================================
% Queries
%=========================================================================

%% Queries:
%% mcall([bel(0)] : rigid_type(alien)).
%% mcall([bel(0)] : rigid_type(humano)).
%% mcall([bel(0)] : rigid_type(nave)).
%% mcall([bel(0)] : rigid_type(planeta)).
```

Código 25 – Exemplo de semântica modal: arquivo MProlog com regras positivas

```
:- calculus cKD45m.
:- set_option(current_calculus, cKD45m).

%=========================================================================
% Propagation rules of the beliefs of the agent 0
%=========================================================================

[bel(N)] : entity(X) :-
  \=(N,0),
  [bel(0)] : entity(X).

[bel(N)] : dio(X, Y) :-
  \=(N,0),
  [bel(0)] : dio(X, Y).

[bel(N)] : deo(X, Y) :-
  \=(N,0),
  [bel(0)] : deo(X, Y).

[bel(N)] : dd(X) :-
  \=(N,0),
  [bel(0)] : dd(X).

%=========================================================================
% Closures
%
% extensionof_irreflexive/2
% instanceof/2
%=========================================================================

% extensionof/2
```

```
[bel(X)] : extensionof_irreflexive(A, B) :-
  [bel(X)] : deo(A,B).
[bel(X)] : extensionof_irreflexive(A, B) :-
  [bel(X)] : deo(A,Z),
  [bel(X)] : extensionof_irreflexive(Z,B).

[pos(X)] : extensionof_irreflexive(A, B) :-
  [pos(X)] : deo(A,B).
[pos(X)] : extensionof_irreflexive(A, B) :-
  [pos(X)] : deo(A,Z),
  [pos(X)] : extensionof_irreflexive(Z,B).

% instanceof/2
[bel(X)] : instanceof(A, B) :-
  [bel(X)] : dio(A,B).
[bel(X)] : instanceof(A, B) :-
  [bel(X)] : dio(A,Z),
  [bel(X)] : extensionof_irreflexive(Z,B).

[pos(X)] : instanceof(A, B) :-
  [pos(X)] : dio(A,B).
[pos(X)] : instanceof(A, B) :-
  [pos(X)] : dio(A,Z),
  [pos(X)] : extensionof_irreflexive(Z,B).

%=============================================================================
% Rigidity
%
% rigid_type/1
%=============================================================================

%% rigid_type(?T)
% rigid_type/1
% Succeeds whether T is an instance of "rigid" and has no
% particulars instantiating another O instance of "rigid"
% distinct of T.
[bel(0)] : rigid_type(T) :-          % universal modal rigidity
        [bel(0)] : instanceof(T, kind),
        % set Xs of the instances of the universal T
        findall(X,
                mcall([pos(_)] : instanceof(X, T)),
                Xs),
        findall(S,
                (member(X1, Xs),
                 mcall([pos(_)] : instanceof(X1, S)),
                 S \= T,
                 mcall([pos(_)] : instanceof(S, kind))),
                []).
[bel(0)] : rigid_type(R) :-          % universal modal rigidity
        [bel(0)] : dio(R, kind),
        \+ mcall([pos(_)] : instanceof(_, R)).
[pos(0)] : rigid_type(T) :-          % existential modal rigidity
        [pos(0)] : instanceof(T, kind).
```

Código 26 – Exemplo de semântica modal: arquivo MProlog relações semânticas

```
:- calculus cKD45m.
:- set_option(current_calculus, cKD45m).

%=============================================================================
% Semantic relations produced by translation filter (=>m)
%=============================================================================

% universals
[bel(0)] : entity(kind).
[bel(0)] : entity(phase).
[bel(0)] : entity(role).
[bel(0)] : entity(alien).
[bel(0)] : entity(humano).
[bel(0)] : entity(nave).
[bel(0)] : entity(planeta).
[bel(0)] : entity(adulto).
[bel(0)] : entity(menino).
[bel(0)] : entity(alien_visitante).
[bel(0)] : entity(et).
[bel(0)] : entity(elliott).
[bel(0)] : entity(nave_et).
[bel(0)] : dio(alien,kind).
[bel(0)] : dio(humano,kind).
[bel(0)] : dio(nave,kind).
```

```
[bel(0)] : dio(planeta,kind).
[bel(0)] : dio(adulto,phase).
[bel(0)] : dio(menino,phase).
[bel(0)] : dio(alien_visitante,role).
[bel(0)] : deo(adulto,humano).
[bel(0)] : deo(menino,humano).
[bel(0)] : deo(alien_visitante,alien).
[bel(0)] : dd([menino,adulto]).

% particulars
[bel(1)] : dio(et,alien_visitante).
[bel(1)] : dio(elliott,menino).
[pos(1)] : dio(nave_et,alien).
[pos(2)] : dio(nave_et,nave).
```

Como uma alegoria aos *objetos voadores não-identificados* – OVNI no Brasil e UFO nos Estados Unidos, sendo que este remete à sigla da ontologia de fundamentação investigada neste trabalho – a especificação contida no Código 24 consiste em descrever uma ontologia que alude ao filme *E.T. O Extraterrestre* (E.T.…, 1982). Especial interesse, no entanto, está sobre os conceitos `planeta` e `alien`. Segundo a especificação mencionada e sua contraparte semântica na Código 26, o conceito de `alien` não é um conceito rígido conforme definido nesta seção. No caso, o conceito de `nave_et`, que é instância de `alien`, é também instância de `nave`, sendo que esses dois últimos são instância de `kind`. Por isso, `nave_et` é um contra-exemplo às consultas definidas que implementam o predicado `rigid_type/1` do Código 25. O Código 27 contém o resultado de três consultas definidas realizadas à teoria modal construída, e evidencia as conclusões apresentadas. A última consulta, aliás, apresenta na variável `P` o contra-exemplo que evidencia a existência de ao menos outra entidade que instancia `kind` que não seja o próprio conceito original instanciado em `T = alien`.

Código 27 – Saída de consultas ao Código 24

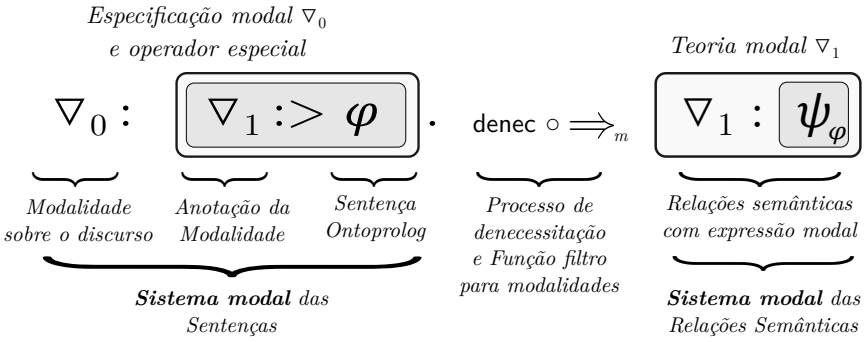
```
% Query 1: Is "planeta" a rigid type?
| ?- mcall([bel(0)] : rigid_type(planeta)).
yes
% Query 2: Is "alien" a rigid type?
| ?- mcall([bel(0)] : rigid_type(alien)).
no
% Query 3: Why "alien" is not a rigid type? Put the counterexamples on P.
| ?- T = alien,
     mcall(([bel(0)] : instanceof(T, kind),
          findall(X,
                  mcall([pos(_)] : instanceof(X, T)),
                  Xs),
          findall(S,
                  (member(X1, Xs),
                   mcall([pos(_)] : instanceof(X1, S)),
                   S \= T,
                   mcall([pos(_)] : instanceof(S, kind))),
                 P),
          P\=[]
         )).
T = alien,
Xs = [et,nave_et],
P = [nave] ? ;
no
```

9.3 Sentenças e teorias modais

A terceira estratégia de integração do arcabouço de Ontoprolog com o arcabouço modal de MProlog é uma combinação das duas outras descritas nas seções anteriores. Trata-se da combinação de sentenças modais regidas por alguma das lógicas modais de MProlog, com sentenças anotadas por contexto modal na forma da Definição 9.1 (Sentença com contexto modal). Naturalmente, por se tratar de uma combinação das estratégias anteriores, as *teorias* produzidas a partir da tradução de sentenças com anotação de modalidade será uma teoria modal, possivelmente sustentada por um sistema modal diferente do sistema modal que regia as *especificações*.

A Figura 47 apresenta a arquitetura geral da abordagem em tela. Nela destacam-se dois tipos de sistemas modais, o identificado por ∇_0 e o outro por ∇_1. O primeiro é o sistema modal que rege as sentenças, exatamente conforme é apresentado na seção 9.1, ou seja, as especificações estão sustentadas por um programa em Lógica Modal que é independente de Ontoprolog, e é usado para gerenciar e integrar as sentenças dos discursos em si. Por essa razão que a função denec está novamente representada na figura. Essa função tem o mesmo objetivo: identificar, selecionar e denecessitar sentenças que "interessam" serem traduzidas para teorias Ontoprolog, conforme objetivos definidos na engenharia lógica. Uma vez a função denec aplciada, obtém-se um conjunto de sentenças na forma da Definição 9.1. Diante disso, a função filtro $\Longrightarrow_m$ desenvolvida na seção 9.2 é utilizada para produzir teorias modais no sistema ∇_1.

Figura 47 – Sintaxe integrada de sistemas modais diferentes

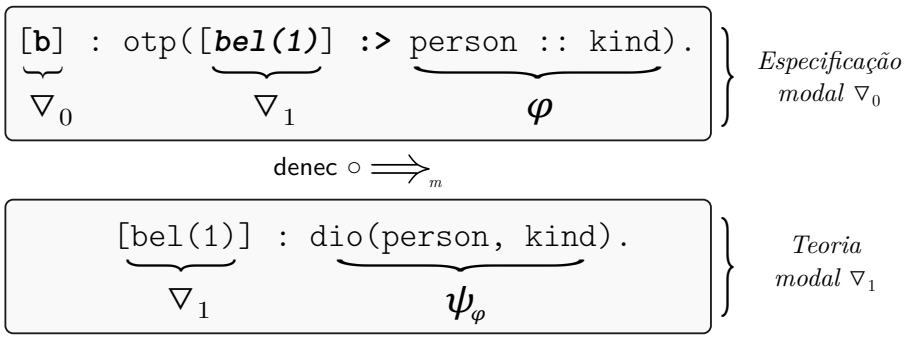


Fonte: Os autores

Com base na ideia apresentada, a Figura 48 ilustra um exemplo concreto, em que uma sentença φ é traduzida para uma relação semântica ψ devidamente contextualizada pelo operador modal anotado na sentença. Nessa ilustração evidencia-se que o sistema modal ∇_0, usado no lado sintático, não aparece no lado semântico. É importante ressaltar que, embora pareça que as sentenças da *especificação* possuam anotação de duas modalidades, há apenas um único sistema modal vigente, e isso é o caso porque apenas o operador : é usado para definir, à esquerda, o contexto modal do sistema modal vigente, enquanto o

operador :> é usado apenas para anotar o contexto modal a ser utilizado após a tradução.

Figura 48 – Exemplo de instanciação da sintaxe integrada de sistemas modais



Fonte: Os autores

Espera-se que o contexto modal aplicado às relações semânticas que denotam as sentenças (∇_1) tenha alguma relação com o significado original das modalidades que regiam as especificações sintáticas (∇_0). Porém, isso não é necessário, caso em que diferentes sistemas lógicos são utilizados como controladores dos programas em que as sentenças e as teorias são definidas.

O Código 28 contém uma especificação criada nos moldes da estrutura descrita nesta seção. As duas primeiras linhas definem o sistema modal que rege essa estrutura, no caso, o sistema monomodal T. A interpretação alética desse sistema provê dois operadores modais, b e d, caracteres que remetem aos símbolos $\Box$ e $\Diamond$, que são definidos de forma tradicional, conforme descrito em Nguyen (2010, p. 15). Portanto, as sentenças do exemplo devem ser interpretadas como "necessárias" ou "possíveis". Uma implementação concreta de denec pode utilizar essa informação modal para selecionar sentenças que atendam a determinado critério e, dessa forma, repassá-las à função filtro $\Longrightarrow_m$ e produzir a teoria modal.

Código 28 – Exemplo do sentença modal e anotação de modalidade

```
:- calculus cT.
:- set_option(current_calculus, cT).

[b] : opt([bel(0)] :> [alien, humano, nave, planeta] :: kind).
[b] : opt([bel(0)] :> disjoint [adulto, menino] :: phase extend humano).
[b] : opt([bel(0)] :> alien_visitante :: role extends alien).
[b] : opt([bel(0)] :> et :: alien_visitante).
[b] : opt([bel(0)] :> elliott :: menino).

[d] : opt([pos(1)] :> nave_et :: nave).
[d] : opt([pos(2)] :> nave_et :: alien).
```

9.4 Fechamento

As estratégias de incorporação de expressões modais à linguagem de Ontoprolog apresentadas nesta seção podem ser consideradas suficientes para o alcance do objetivo

específico "*estender o arcabouço clássico da linguagem textual para inclusão de operadores modais referentes a discursos de diferentes agentes*" (subseção 1.2.2). Com a primeira proposta, descrita na seção 9.1, que engloba apenas a avaliação modal nas sentenças da linguagem, além das consultas com contextos modais às sentenças, é possível realizar a verificação global de consistência entre os discursos especificados em Ontoprolog. Esse é um instrumento importante no processo de obtenção de acordos nos discursos a respeito da realidade porque evidencia as diferenças entre diferentes ponto de vista de cada agente envolvido na modelagem conceitual. Embora isso seja aceitável do ponto de vista de obtenção dos resultados esperados, essa não é uma solução que exercita ao máximo o potencial de uso de modalidades em formalizações de ontologias. Por isso, na seção 9.2 apresenta-se como o arcabouço de Ontoprolog-Clássico produzido nos capítulos anteriores pode ser alterado para que seja possível a incorporação legítima de modalidades no arcabouço dedutivo das teorias de Ontoprolog, de modo que a linguagem comporte expressões modais não apenas no âmbito das especificações de ontologias, mas também na esfera das teorias que essas especificações denotam. Como consequência, essa estratégia torna mais sofisticado o processo de construção de acordos, uma vez que permite que consultas específicas a respeito da ontologia construída por cada agente sejam realizadas. Dessa forma, é possível não apenas transcrever as regras já apresentadas na seção 6.4 e na seção 8.3, como também é factível definir novos axiomas e regras, como o de Rigidez, apresentado na entrada Rigidity do Glossário da UFO. Isso é possível, principalmente, porque a semântica dos contextos modais não está necessariamente restrita à interpretação epistêmica, como é apresentado no capítulo. Outras interpretações podem ser dadas às modalidades, sem ser necessário alterar o modo como combinar Ontoprolog com MProlog. Por fim, a combinação apresentada na seção 9.3 das duas estratégias base permite construir sistema ainda mais expressivos, que servem de instrumentos para casos concretos de aplicação de Engenharia Lógica.

Uma limitação das abordagens apresentadas no que tange à modelagem conceitual refere-se ao fato de que nas especificações as entidades da realidade são representadas indiretamente por seus mediadores, no caso, os *termos*. Esses, por sua vez, remetem aos conceitos. Por isso, eventualmente um mesmo termo pode ser usado para representar conceitos diferentes para agentes distintos. Garantir a *hipótese do nome único*, conforme é discutido na subseção 5.3.1, é uma abordagem potencialmente mais difícil em relação ao número de agentes. Porém, esse é um desafio comum na integração de ontologias, que só pode ser vencido com a combinação de diferentes técnicas, como o uso de ontologias de fundamentação; a negociação e discussão constante e irrestrita entre os ontologistas; o teste da representação de conceitual em relação aos objetos da realidade modelada; a delimitação do escopo da modelagem a uma mesma parcela da realidade, compartilhada entre os agentes; entre outras. Dessa forma, as estratégias apresentadas neste capítulo são complementares e auxiliares na tarefa de integração de ontologias e construção de acordos,

e produzem maior efeito se utilizadas em conjunto com outras técnicas.

Das três abordagens apresentadas para integração do arcabouço de Ontoprolog com o poder expressivo modal disponibilizado pelo arcabouço lógico de MProlog, apenas a integração sintática das sentenças, apresentada na seção 9.1, está incorporada ao software produzido nesta pesquisa (Capítulo 10). Já a implementação das demais estratégias é anotada como trabalho futuro na seção 11.3, uma vez que, seguindo-se os parâmetros apresentados na seção 9.2, espera-se que seja possível obter os resultados apresentados.

O código-fonte anexo à versão digital deste documento contém todos os exemplos transcritos no decorrer do texto, além de outros exemplos não apresentados neste capítulo. É o caso, por exemplo, da versão modal da ontologia apresentada no Código 19. A seção 10.2 descreve o conteúdo do código-fonte.

10 Implementação de referência do arcabouço de Ontoprolog

Este capítulo apresenta a implementação de referência de Ontoprolog em Prolog e em MProlog. Parte dessa implementação já é introduzida nos capítulos anteriores, e também é descrita por alguns apêndices. São descritas a arquitetura geral da ferramenta, bem como limitações conhecidas. Possibilidades de evoluções futuras também são indicadas.

Com esse objetivo, o restante deste capítulo é organizado do seguinte modo. A seção 10.1 estabelece a licença de uso e descreve como o software desenvolvido no escopo desta pesquisa pode ser obtido, utilizado e evoluído. A seção 10.2 apresenta um breve comentário sobre cada um dos arquivos codificados em Prolog da implementação de referência. A seção 10.3 contém resultados preliminares quanto a exportação de modelos de teorias Ontoprolog em diagramas UML e OntoUML. Na seção 10.4 lista-se algumas limitações da implementação atual em relação às regras da UFO. A seção 10.5 consiste na descrição de adaptações realizadas no código-fonte de MProlog para que a implementação de referências de Ontoprolog possa ser integrada ao arcabouço modal provido por ele. E, como finalização, breves comentários finais são tecidos na seção 10.6.

10.1 Código-fonte e licença de uso

A implementação concreta de Ontoprolog, baseada nas especificações desenvolvidas nesta pesquisa, adota a licença GPLv3 (<http://www.gnu.org/copyleft/gpl.html>), que é um tipo de licença *copyleft* forte. Isso significa que os trabalhos derivados dessa codificação devem obedecer a mesma licença.

O código-fonte de Ontoprolog é incorporado à versão eletrônica deste documento. Conforme descrito na subseção 1.4.4.3, a incorporação do código-fonte é realizada de acordo com a especificação do padrão PDF/A-3b da ISO 19005:2012 (ISO, 2012a)[1].

A ferramenta MProlog, versão 2.0, disponível para download em <http://www.mimuw.edu.pl/~nguyen/mprolog.html>[2], não apresenta informações sobre a licença de uso, o que induz considerá-lo de domínio público. Por isso, toma-se a liberdade de redistribuí-lo

[1] Eventualmente, pode ser necessário ajustar o nível de segurança do leitor de PDF para que o arquivo anexo com o código-fonte possa ser aberto. Como sugestão universal, o software livre *PDFSaM* (<http://www.pdfsam.org/>, acesso de 21 jan 2015), baseado em linha de comando, é capaz de extrair o conteúdo dos anexos do PDF. Como alternativa a esse, em ambiente Windows, o software *Foxit Reader* (<http://www.foxitsoftware.com/Secure_PDF_Reader/>, acesso de 21 jan 2015) pode ser utilizado para essa finalidade.

[2] Acesso de 29 out 2014.

no endereço mencionado no parágrafo anterior.

Os softwares MProlog e Ontoprolog são testados para execução nas seguintes implementações de Prolog:

a) SWI-Prolog(<http://www.swi-prolog.org/>), versão 6.6.6 de maio de 2014;

b) SICStus Prolog(<https://sicstus.sics.se/>), versões 4.2.3 de outubro de 2012 e 4.3.1 de dezembro de 2014;

A implementação de referência de Ontoprolog é baseada exclusivamente nas normas ISO/IEC 13211-1:1995 *Information technology – Programming languages – Prolog – Part 1 : General core* (ISO/IEC, 1995) – com as correções providas pela (ISO/IEC, 2007) e pela 2012a (ISO/IEC, 2012a) – e ISO/IEC 13211-2:2000 *Information technology – Programming languages – Prolog – Part 2 : Modules* (ISO/IEC, 2000). Por isso, é esperado que o software Ontoprolog funcione em qualquer implementação Prolog que atenda essas normas.

10.2 Descrição do código-fonte e componentes

O código de Ontoprolog é formado por componentes, contidos em arquivos individuais. Cada um dos arquivos principais que compõe a ferramenta está listado no Quadro 32, que apresenta o diretório e o nome do arquivo, bem como o papel do componente no sistema como um todo. Sempre que existentes, referências a seções deste documento também são apresentadas na descrição de cada arquivo.

Os arquivos de exemplos, contidos na pasta `demos`, contém uma série de exemplos, inclusive alguns adicionais, não apresentados no decorrer deste texto. Destaca-se, por exemplo, o arquivo `guizzardi2005p360.mpl`, que é uma versão modal da especificação apresentada pelo Código 19.

Quadro 32 – Arquivos do código-fonte da implementação de referência

demos/classical/tese/bd.pl:
Exemplo de especificação baseada na ontologia UFO contida no Código 17.

demos/classical/tese/constraints_espacos_conceituais.pl:
Exemplo de especificação baseada na ontologia UFO contida no Código 13.

demos/classical/tese/guizzardi2005p360.pl:
Exemplo de especificação de ontologia contida no Código 19 baseada no diagrama OntoUML de Guizzardi (2005, p. 360), apresentado na Figura 42.

demos/classical/tese/guizzardi2005p360_instances.pl:
Exemplos do Código 20 que contém especificações de instâncias da ontologia descrita no Código 19.

demos/classical/tese/guizzardi2014p13.pl:
Exemplo de especificação baseada na ontologia UFO contida no Código 18.

demos/classical/tese/inheres_marco_polo.pl:
Exemplo "Marco Polo" de especificação baseada na ontologia UFO contida no Código 15 que envolve subsunção de Qualidades e Domínios de Qualidades.

demos/classical/tese/ontoprolog_hypertypes.pl:
Exemplo simples de exportação da Metateoria de Hypertypes para PlantUML (<http://plantuml.com/>) contido no Código 29

demos/classical/tese/sinatra_base.pl:
Exemplo "Sinatra" de especificação Ontoprolog-base contida no Código 8 e ilustrada na Figura 34.

demos/classical/tese/sinatra_ufo.pl:
Exemplo "Sinatra" de especificação baseada na ontologia UFO contida no Código 11.

demos/modal/tese/guizzardi2005p360.mpl:
Versão modal da especificação da ontologia contida no Código 19.

demos/modal/tese/guizzardi2005p360.pl:
Arquivo Prolog que controla a versão modal em MProlog da especificação contida no Código 19.

demos/modal/tese/modal_sentence_integracao_agentes.mpl:
Exemplo de especificação modal contida no Código 22.

demos/modal/tese/modal_sentence_integracao_agentes.pl:
Arquivo Prolog que controla programa MProlog do exemplo contido no Código 21.

demos/modal/tese/modal_theory_rigidity_belief.mpl:
Especificação e semântica modal do exemplo contido no Código 26.

demos/modal/tese/modal_theory_rigidity_belief.pl:
Arquivo Prolog que controla a especificação modal em MProlog contido no Código 24.

demos/modal/tese/modal_theory_rules.mpl:
Arquivo MProlog com regras positivas consultadas pelo Código 24.

demos/modal/tese/rigidity_belief2.mpl:
Exemplo de especificação e semântica modal do exemplo contido no Código 26.

demos/modal/tese/rigidity_smnm.mpl:
Estudo sobre o conceito de rigidez denecessitada da UFO contido no Código 30.

demos/modal/tese/rigidity_smnm.pl:
Arquivo Prolog que controla a especificação modal em MProlog apresentada no Código 30.

demos/modal/tese/simple_example.mpl:
Exemplo simples de um programa MProlog, conforme abordado no Código 2. Baseado no exemplo smnm_test3.mpl, distribuído com o software MProlog.

demos/modal/tese/simple_example.pl:
Arquivo Prolog que controla programa MProlog do exemplo contido no Código 1.

export/plantuml/demos/plantuml_sinatra_base.pl:
Exemplos de exportação de teorias Ontoprolog instância da Metateoria de Hypertypes com PlantUML (<http://plantuml.com/>)

export/plantuml/plant_export_cases_printers.pl:
Casos comuns de consulta à teoria para exportação à PlantUML (<http://plantuml.com/>)

export/plantuml/plant_export_cases_theories.pl:
Conjunto de predicados para exportação de teorias baseadas na Metateoria de Hypertypes para visualização em PlantUML (<http://plantuml.com/>)

export/plantuml/plantuml.pl:
Biblioteca de exportaçãoo de teorias Ontoprolog para renderização em UML com PlantUML (<http://plantuml.com/>)

export/plantuml/plantuml_lib.pl:
Biblioteca de predicados utilitários para exportação de teorias Ontoprolog com PlantUML (<http://plantuml.com/>)

ontoprolog/classical/base/ontology.pl:
Metateoria de hypertypes. Trata-se do conteúdo do Apêndice A.

ontoprolog/classical/base/ontology_nrulemeta.pl:
Consiste em regras que verificam entidades lógicas atribuíveis como meta-propriedades (Apêndice A).

ontoprolog/classical/base/parser_dec_sentence_base.pl:
Definição das regras de tradução de sentenças sintáticas em relações semânticas. Trata-se de parte da implementação do Filtro de tradução da sintaxe (seção 7.3).

ontoprolog/classical/base/parser_dec_sentence_relation.pl:
Tradução do Açúcar sintático para relações binárias (seção 7.5).

ontoprolog/classical/base/parser_helper_base.pl:
Conjunto de definições auxiliares para o sistema de tradução da sintaxe base.

ontoprolog/classical/base/semantic_closures_base.pl:
Definições positivas construídas sobre as relações semânticas base. Contém implementações das definições e regras descritas na subseção 6.4.1, entre outras.

ontoprolog/classical/base/semantic_expansion_base.pl:
Contém o sistema de expansões das relações semânticas geradas pelo tradutor (Apêndice F).

ontoprolog/classical/base/semantic_ngrules.pl:
Conjunto de Regras negativas escritas de modo positivo (subseção 6.3.1), na forma $ngrule/n$, conforme apresentadas na subseção 6.4.3.

ontoprolog/classical/parser.pl:
Controlador global do sistema de tradução sintática. Contém parte da implementação dos algoritmos mencionados na subseção 7.3.3 (Algoritmo de controle da tradução).

ontoprolog/classical/parser_assert.pl:
Controlador das expansões geradas pelo tradutor. Contém parte da implementação dos algoritmos mencionados na subseção 7.3.3 (Algoritmo de controle da tradução).

ontoprolog/classical/parser_dec_sentence.pl:
Controlador das regras de tradução de sentenças sintáticas em relações semânticas. Contém parte da implementação dos algoritmos mencionados na subseção 7.3.3 (Algoritmo de controle da tradução).

ontoprolog/classical/parser_def.pl:
Definições globais do tradutor sentenças.

ontoprolog/classical/parser_infra.pl:
Infra-estrutura do tradutor de sentenças. Contém definições referentes o contexto Δ (subseção 7.3.2) e ao sistema de mensagens de verificações sintáticas.

ontoprolog/classical/parser_messages.pl:
Conjunto de definições de mensagens lançadas ao usuário durante o processo de tradução sintática.

ontoprolog/classical/semantic_ngrules.pl:
Sistema controlador das verificações semânticas na forma $ngrule/n$ (subseção 6.3.1)

ontoprolog/classical/semantic_theoretical_layers.pl:
Definições referentes aos níveis teóricos (seção 5.5)

ontoprolog/classical/ufo-a/ontology_ufo-a.pl:
Meta-teoria da UFO. Trata-se do conteúdo do Apêndice C e da figura do Apêndice B.

ontoprolog/classical/ufo-a/ontology_ufo-a_nrulemeta.pl:
Consiste em regras que verificam entidades lógicas atribuíveis como meta-propriedades específicas da UFO (Apêndice C).

ontoprolog/classical/ufo-a/parser_dec_sentence_general_ufo.pl:
Definição das regras de tradução de sentenças sintáticas específicas da UFO em relações semânticas. Trata-se de parte da implementação de parte da Extensão do filtro de tradução (subseção 8.2.2).

ontoprolog/classical/ufo-a/parser_dec_sentence_inherence_ufo.pl:
Definição das regras de tradução de sentenças sintáticas específicas da UFO que se referem à Instância de caracterização (subseção 8.2.2.9).

ontoprolog/classical/ufo-a/parser_dec_sentence_relator_particular_ufo.pl:
Definição das regras de tradução de sentenças sintáticas específicas da UFO que se referem à Relator particular (subseção 8.2.2.11).

ontoprolog/classical/ufo-a/parser_dec_sentence_relator_type_ufo.pl:
Definição das regras de tradução de sentenças sintáticas específicas da UFO que se referem à Relator universal (subseção 8.2.2.10).

ontoprolog/classical/ufo-a/parser_def_ufo.pl:
Definições complementares para o sistema de tradução da sintaxe específica da UFO.

ontoprolog/classical/ufo-a/parser_helper_context_ufo.pl:
Conjunto de definições específicas da UFO referentes ao contexto Δ (subseção 7.3.2).

ontoprolog/classical/ufo-a/parser_helper_general_ufo.pl:
Conjunto de definições auxiliares para o sistema de tradução da sintaxe UFO.

ontoprolog/classical/ufo-a/semantic_closures_ufo.pl:
Definições positivas específicas da UFO construídas sobre as relações semânticas base. Contém implementações das definições e regras descritas na subseção 8.3.1, entre outros.

ontoprolog/classical/ufo-a/semantic_expansion_ufo.pl:
Contém a extensão UFO do sistema de expansões das relações semânticas geradas pelo tradutor (Apêndice G).

ontoprolog/classical/ufo-a/semantic_ngrules_closures_ufo.pl:
Conjunto de Regras negativas escritas de modo positivo (subseção 6.3.1), na forma *ngrule*/*n*, conforme apresentados na subseção 8.3.2.

ontoprolog/classical/ufo-a/semantic_ngrules_domain_ufo.pl:
Conjunto das RNP forma *ngrule*/*n* apresentadas na subseção 8.3.2.

ontoprolog/common/cardinality_constraint.pl:
Definições da biblioteca de controle de restrições de cardinalidade (subseção 6.4.2).

ontoprolog/common/ontoprolog_operators.pl:
Operadores Prolog usados na definição da sintaxe de Ontoprolog (Apêndice E)

ontoprolog/common/options.pl:
Concentra o predicado *ontoprolog_option*/2, usado para alterar o comportamento do sistema.

ontoprolog/common/util.pl:
Biblioteca de utilitários gerais que abstrai eventuais necessidades de pacotes específicos. Todos os predicados deste componente são prefixados com *util_*. Os predicados deste componente também são independentes de qualquer outro componente do sistema.

ontoprolog/common/util_strings.pl:
Biblioteca de controle de *strings*.

ontoprolog/common/write_util.pl:
Biblioteca de utilitários para criação de arquivos. Utilizado para a integração e exportação de dados.

ontoprolog/modal/m_parser.plt:
Controlador global do sistema de tradução sintática modal.

ontoprolog/modal/m_parser_dec_sentence.plt:
Definição das regras de tradução de sentenças modais.

ontoprolog/modal/mprolog2-custom/README2.txt:
Descreve as adaptaçõeds ao código-fonte do MProlog, conforme seção 10.5.

ontoprolog/modal/ontoprolog-m_op.pl:
Operador de Prolog usado na definição de anotação de modalidade (Código 36)

ontoprolog/ontoprolog-m.pl:
Arquivo principal do fragmento modal. Como o fragmento modal é construído sobre o fragmento clássico, o arquivo `ontoprolog.pl` é importado automaticamente.

ontoprolog/ontoprolog.pl:
Arquivo principal do fragmento clássico da ferramenta, que concentra todas as inclusões dos demais arquivos necessários. É este arquivo que deve ser importado em projetos que visem utilizar Ontoprolog como pacote.

test/classical/tests_cases_base.pl:
Conjunto de casos de testes da sintaxe base.

test/classical/tests_cases_ufo.pl:
Conjunto de casos de testes da sintaxe complementar da UFO.

test/classical/tests_lib.pl:
Infra-estrutura para realização de testes do tradutor da sintaxe.

test/classical/tests_main.pl:
Controlador da biblioteca de testes.

10.2.1 Compilação de sentenças e verificação de teorias em Ontoprolog

Para compilar um conjunto de sentenças clássicas escritas na sintaxe de Ontoprolog, é suficiente que o seguinte arquétipo de programa seja atendido:

a) em um programa Prolog, carregar a biblioteca de Ontoprolog por meio do predicado `include/1`, em que o primeiro parâmetro é o caminho completo, relativo ao programa atual, para o arquivo principal de Ontoprolog. Um exemplo é este:

```
:- include('ontoprolog.pl').
```

b) escrever sentenças Ontoprolog, que podem ser combinadas com programas Prolog comuns;

c) para compilar as sentenças sintáticas em relações semânticas, utilizar o predicado `otp_compile/0`;

d) para verificar a consistência dos modelos produzidos pelas relações semânticas, utilizar o predicado `check_semantics/0`.

O Código 11 apresenta um exemplo completo de especificação, compilação e verificação da semântica. A seção 8.4, a seção 7.7 e o Capítulo 9 contém outros exemplos.

10.2.2 Testes do tradutor sintático

Um sistema de testes unitários foi escrito especialmente para verificação da correção do processo de tradução das sentenças sintáticas em relações semânticas. Com base nessa biblioteca de testes, estão disponíveis 161 casos de testes da tradução da sintaxe base (seção 7.4) e 140 referentes à tradução da sintaxe estendida (seção 8.2).

Os testes podem ser executados pelo predicado `run_tests/0` e por `run_tests/1`. O último predicado recebe como parâmetro `false` ou `true`. `false` indica que apenas casos de testes que não tenham sucesso sejam exibidos.

O sistema de testes e os casos de testes estão concentrados nos seguintes arquivos: `tests_lib.pl`, `tests_cases_base.pl` e `tests_cases_ufo.pl`.

Ao final do arquivo `tests_cases_ufo.pl`, apresenta-se uma série de casos de testes que exercitam as traduções de Relators, abordados pela subseção 8.4.7.

10.3 Representação de teorias em linguagem diagramática

Conforme comentam Costa, Grings e Santos (2008, p .437-438), linguagens com modelos computacionais que permitem inclusão automática de controle podem, ao menos em parte, ser representadas de modo diagramático, como é o caso de várias linguagens de programação por restrições e de alguns tipos de sistemas de prova automática de teoremas. Por isso, como um caso de sistema de programação em lógica que pode ser representado como problemas de satisfação de restrições – no caso, restrições sobre os modelos conceituais –, grande parte das teorias clássicas definidas em Ontoprolog podem ser representadas em diagramas UML. Dentre as características que não possuem representação em UML, destaca-se especialmente as restrições sobre meta-propriedades (subseção 6.4.4). No entanto, as demais entidades ontológicas e lógicas, e as relações formais básicas (seção 6.2), podem ser transcritas em diagramas UML. Da mesma forma, modelos conceituais especificados em Ontoprolog tendo como base a ontologia de fundamentação UFO podem igualmente ser representadas em OntoUML.

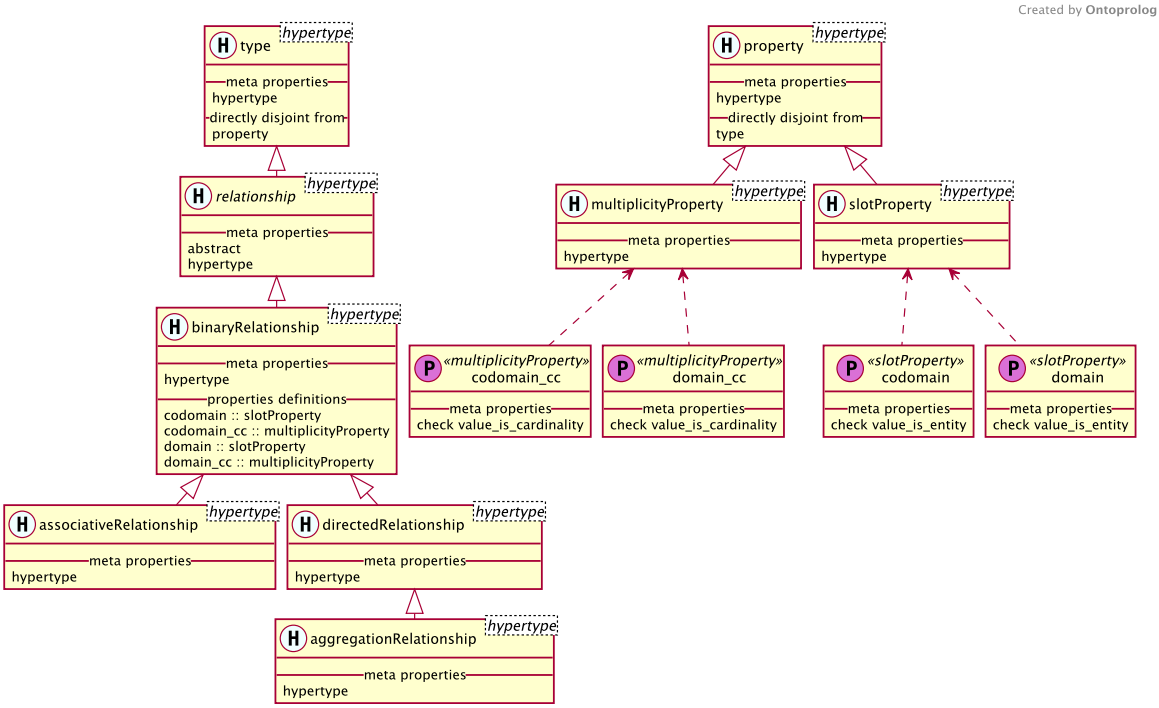
A diagramação das teorias de Ontoprolog em UML e OntoUML é resultado adicional deste trabalho, e está incluído no código-fonte listado na seção 10.2. A exportação das teorias em diagramas consiste em processo de tradução que transpõe as relações semânticas, que consistem em uma teoria Ontoprolog, em uma representação na linguagem alvo utilizada pelo mecanismo de plotagem.

Para exercício da representação de teorias Ontoprolog de forma diagramática, algumas bibliotecas e ferramentas de código aberto para representação UML são consideradas,

como UMLGraph[3], MetaUML[4], UMLLet[5] e PlantUML[6]. Utiliza-se neste trabalho o arcabouço oferecido por PlantUML porque a linguagem oferece constructos de representações visuais com grande poder expressivo, além de permitir uma série de personalizações na imagem final. No entanto, qualquer outro padrão serviria ao propósito de demonstrar a possibilidade de tradução dos modelos, inclusive RDF[7], UXF (SUZUKI; YAMAMOTO, 1999), XMI (OBJECT MANAGEMENT GROUP, 2014b) ou o padrão utilizado pela ferramenta OLED (SALES, 2014, p. 237).

Representações diagramáticas de teorias Ontoprolog são apresentadas ao longo deste texto, a exemplo da Figura 34, Figura 38, Figura 39 e da Código 15. A título de exemplo adicional, a Figura 49 ilustra um resultado do mecanismo de tradução para PlantUML referente à diagramação automática da Metateoria de *Hypertypes* (Definição 5.10), conforme especificação contida no Código 31. A figura congrega entidades ilustradas pela Figura 25 e pela Figura 26.

Figura 49 – Representação da Metateoria de *Hypertypes* com PlantUML



Fonte: Os autores

Na ilustração contida na Figura 49, destaca-se uso de recursos visuais como os recipientes inseridos no corpo e de símbolos no título dos Classifiers. Por exemplo, na entidade `type`, o recipiente `meta properties` contém as meta-propriedades daquela entidade. Da

[3] <http://www.umlgraph.org/index.html> e <https://github.com/dspinellis/UMLGraph>
[4] <http://metauml.sourceforge.net/old/class-diagram.html>
[5] <http://www.umlet.com/>
[6] <http://plantuml.sourceforge.net/>
[7] <http://www.w3.org/standards/techs/rdf#w3c_all>

mesma forma, o recipiente `properties definitions`, em `binaryRelationship`, ilustra as propriedades atribuídas a essa entidade. Os símbolos `H` e `P` funcionam como estereótipo adicional, com apelo visual, que representam o nível teórico da entidade (seção 5.5): `H` para *Hypertype* e `P` para *Particular*.

Embora a exportação de teorias de Ontoprolog para UML seja relativamente simples, a exportação para OntoUML deve observar algumas restrições, contidas nos quadros da subseção 4.5.1, conforme lista-se algumas a seguir:

a) observar o uso do símbolo "/" nas entidades derivadas, conforme a restrição 1 do Quadro 22 (Formal), p. 186;

b) a representação de *generalization sets* (subseção 4.4.3) deve observar a restrição 2 do Quadro 6 (Phase), p. 172, bem como disjunção e subsunção total. Por falta de suporte na linguagem PlantUML para representar adequadamente *generalization sets*, utiliza-se o artifício que consiste em nomear os grupos de generalização e representá-los pela palavra "Complete" sucedida de um número que indica o nome do grupo;

c) conforme Guizzardi (2005, p. 246-248, 325), geralmente os Quality Universals não são representados explicitamente em modelos OntoUML, mas sim via Attribute Functions. Dessa forma, um Attribute Functions é uma forma alternativa de expressar uma Quality. Quanto a isso, a tradução observa as regras do Quadro 13 (Quality) e permite que Momentos Intrínsecos sejam decorados de forma específica em teorias baseadas na UFO, conforme abordado pela subseção 8.4.3;

No código-fonte de Ontoprolog, o predicado `export_compile_plantuml/4` é o responsável por executar a tradução da teoria corrente em diagramas PlantUML. Uma série de opções de personalização do diagrama produto da tradução estão disponíveis e documentadas no arquivo `plantuml_lib.pl`. O Código 29 contém a definição utilizada para produzir a Figura 49 e ilustra o uso do predicado `export_compile_plantuml/4`. As linhas comentadas são algumas das possibilidades de personalização do processo de tradução. Especialmente, as opções do tipo `level/1` indicam o conjunto universo das entidades a serem consideradas na tradução. Essencialmente, tratam-se de consultas com base nos níveis teóricos (seção 5.5) à base de fatos das relações semânticas que obtém o universo de entidades. Evidentemente, outros universos podem ser definidos. De fato, a flexibilidade na definição das consultas às entidades ontológicas é exatamente uma das características desejadas com a linguagem, e que são demonstradas nessa tradução.

Código 29 – Exportação da Metateoria de *Hypertypes* e opções da tradução para PlantUML

```
:- include('../../../ontoprolog/ontoprolog').
:- include('../../../export/plantuml/plantuml').

% the ontoprolg specification goes here...
```

```
:- otp_compile.

:- write('Generating Hypertype PlantUML diagram...'), nl,
   export_compile_plantuml('../../../export/plantuml/java/',
                           svg,
                           'ontoprolog_hypertypes.plantuml',
                           [% level(domain_theories)
                            % level(theories)
                              level(hypertype_ontology)
                            % level(being_theory)
                            %, hide_entity([particular])
                            %, hide_particulars
                            %, hide_meta
                            %, hide_disjoint
                            , hide_disjoint_closure
                            %, hide_property_value
                            %, hide_property_definition
                            %, hide_property_definition_instance
                            %, hide_property_definition_subsetting
                            %, hide_property_definition_redefining
                            %, hide_property_definition_meta
                            %, hide_powertype
                            %, legend
                            %, ontoprolog
                            %, decorate_meta_note
                            , decorate_meta_abstract
                            %, decorate_relationship
                            %, decorate_ufo_intrinsic_moment
                            %, hide_ufo_characterization
                            %, hide_ufo_inherence
                            %, title('Ontologia de Hypertypes')
                            %, note(1, 'General note')
                           , plant_config(['hide empty members',
                                           'skinparam nodesep 10',
                                           'skinparam ranksep 25',
                                           'skinparam shadowing false',
                                           'header',
                                           'Created by <b>Ontoprolog</b>',
                                           'endheader'
                                          ])
                           ]),
   write('  Generating PDF...'),
   util_shell('java -jar
    /Users/laurocesar/git/rr-svg-to-pdf/target/SVG2PDF-jar-with-dependencies.jar
    ontoprolog_hypertypes.svg ontoprolog_hypertypes.pdf'),
   write('  Done!'), nl.
```

O código contido a partir da linha 44 indica exemplo de como os modelos produzidos com PlantUML no formato *Scalable Vector Graphics* (SVG) podem ser convertidos em documentos PDF. Utiliza-se, no exemplo, um conversor de SVG para PDF escrito em Java disponível como código-livre no endereço: <https://github.com/laurocesar/rr-svg-to-pdf>.

10.4 Regras não implementadas ou implementadas parcialmente

Apresenta-se, nesta seção, algumas pendências conhecidas de formalização de Ontoprolog em relação às regras descritas pelos quadros da subseção 4.5.1.

a) *Regras sobre relações binárias*: a restrição 19 do Item 1 (Mediation), p. 183, a restrição 21 do Item 1 (Derivation Relation), p. 185 e a restrição 20 do Item 1 (Characterization), p. 184 são regras que restringem a somente relações binárias determinados tipos de relações. Ocorre que no corrente estado de Ontoprolog, apenas relações binárias estão definidas, conforme descrito na seção 5.6. Dessa forma, essas regras são inócuas e não possuem nenhum tipo de implementação;

b) *Meta-propriedade* `immutable`: essa meta-propriedade, abordada pela restrição 19 do Item 4 (Mediation), p. 183, restrição 20 do Item 5 (Characterization), p. 184, e pela restrição 21 do Item 4 (Derivation Relation), p. 185, é incorporada na especificação de UFO em Ontoprolog(Apêndice C), mas nenhuma regra do tipo $ngrule/n$ é adicionada;

c) *Meta-propriedades* `derived`, `extensional` *e* `mandatory`: igualmente, essas meta-propriedades estão presentes na especificação de UFO em Ontoprolog, mas nenhuma regra é definida no universo de teorias denotadas pelas especificações;

d) *Alteração na regra original*: a restrição 19 do Item 3 (Mediation), p. 183 é implementada de forma diferente da estabelecida originalmente. Conforme abordado na seção 5.8, a regra de limitação da cardinalidade do lado do `role` foi alterada para considerar a nova entidade `abstractRole`;

e) *Alteração da regra do Quadro 20*: a restrição 2 do Quadro 20 (Characterization), p. 184 também é implementada de forma diferente da regra original. Na implementação, a regra é estabelecia para domínios os casos em que domínios são Moments, e não apenas para os casos específicos de Modes, como era a regra original;

f) *Inferência da meta-propriedade* `mandatory`: o Apêndice G contém regra referente à restrição 5 do Quadro 24 (Meronymic), p. 187. Essa regra visa inferir automaticamente a meta-propriedade `mandatory`. Ela foi inserida como uma expansão pelo fato de se tratar de uma regra geral positiva. Isso poderia ter sido realizado diretamente como uma regra do filtro da sintaxe, mas opta-se pela expansão por ser uma implementação mais simples e natural.

10.5 Adaptações ao MProlog original

O código-fonte original de MProlog precisa ser adaptado para que funcione corretamente com a implementação de referências de Ontoprolog. As alterações referem-se ao uso do operador `::`, que é usado na implementação original de MProlog para denotar regras de inferências no arcabouço de especificação de sistemas de resolução modal, conforme descrito em Nguyen (2010, p. 92-93).

Porém, por `::` ser um operador importante no âmbito de modelagem conceitual e de Ontoprolog, pois usualmente denota a relação de instância, opta-se por alterá-lo no MProlog. Embora fosse possível apenas ajustar a prioridade do operador no contexto de MProlog, a opção por redefini-lo garante que não ocorram eventuais problemas semânticos na interpretação dos operadores por alguma implementação específica de Prolog. Dessa forma, no âmbito de MProlog, todo o código-fonte que continha o operador `::` foi substituído

por outro com o operador `:::`. Essa alteração é registrada no código-fonte de MProlog distribuído com Ontoprolog (seção 10.1). Nenhuma alteração adicional é necessária.

10.6 Fechamento

A implementação de referência de Ontoprolog produzida nesta pesquisa serve como *benchmark* para comparação de desempenho e de design para futuras versões e implementações da ferramenta.

Embora a implementação apresentada seja funcional, o foco da pesquisa não é desenvolver uma solução final de software, que possa ser utilizada em aplicações práticas em tempo imediato. É necessário que projetos no âmbito industrial de engenharia de software incorporem abordagem mais pragmática dos resultados obtidos com as investigações acadêmicas realizadas neste trabalho a fim de que esses resultados possam refletir em soluções para o ambiente corporativo, por exemplo.

11 Considerações finais e conclusão

A experiência do pesquisador com os objetos científicos é uma ocorrência autêntica do fenômeno do conhecimento. Tanto que essa experiência é cerne do método fenomenológico de investigação científica. No entanto, embora seja desejável certa separação e isolamento do pesquisador com o objeto de estudo – para que as análises sejam as mais imparciais possíveis – é inalcançável uma absoluta *epoché*. Além disso, a experiência com o fenômeno do conhecimento é tão arraigada e visceral, que é impossível ignorá-la, mesmo quando tecnicamente outros métodos de investigação são adotados. Por isso mesmo, talvez o maior resultado deste trabalho de pesquisa social aplicada seja justamente o próprio pesquisador, que torna-se usuário e replicador da manifestação das experiências que sofre. Cabe ao investigante, portanto, expor esses resultados, para que outros, como ele, possam contribuir com o conhecimento científico. Este capítulo final tem esse objetivo.

As pesquisas teóricas deste trabalho percorreram várias tentativas humanas de explicar a experiência íntima com o mundo de modo racional – embora apenas uma pequena parte desses achados esteja documentada no texto. Ora[1], o que são as ontologias senão meios de usurpar o papel original da poesia em descrever o mundo? Ou qual o papel das lógicas formais senão o de sintetizar a complexidade do Ser em jogos previsíveis e calculáveis? Ou ainda, o que é a Arquitetura da Informação senão uma tentativa atrevida de não apenas entender, mas de avocar o papel de arquiteta da experiência dos sujeitos com a informação? Como era esperado ao permitir que o pesquisador experimente o mundo, no decorrer dos estudos esses achados influenciaram o modo de entender os objetivos originalmente propostos para o trabalho. O que inicialmente era concebido como o desenvolvimento genuíno de um *instrumento* sustentado por teorias consistentes da Lógica e da Ontologia, ao final das investigações passou a ser entendido apenas como mais algumas linhas num poema que está sendo escrito há milhares de anos a respeito de como exprimir as próprias inquietações.

Por outro lado, embora ainda que seja impossível descrever fielmente a experiência fenomenológica, esta pesquisa apresenta resultados que corroboram o projeto de "minguar a intrínseca subjugação subjetiva inerente à própria experiência" e "descrever a experiência em pura forma lógica" (Introdução, página 39). Ao menos em parte, esses resultados possibilitam que discursos sobre a experiência com o mundo possam ser descritos no âmbito da produção de modelos conceituais. Porém, mesmo com essa ressalva, entende-se que os objetivos do trabalho foram totalmente alcançados e extrapolaram aqueles originalmente propostos na seção 1.2, conforme é detalhado na seção 11.1 e na seção 11.2.

[1] Este parágrafo requer a mesma licença poética implícita na Introdução.

O arcabouço lógico produzido nesta pesquisa é extensível e permite que diferentes tipos de lógicas e de ontologias de fundamentação sejam combinadas. Isso é demonstrado no Capítulo 8 e no Capítulo 9 que, respetivamente, consistem na extensão do arcabouço de Ontoprolog com a UFO e na combinação da lógica subjacente original de Ontoprolog com lógicas modais providas por MProlog. A incorporação da UFO no arcabouço produzido tem caráter duplo: ela denota um tipo de formalização da Ontologia, uma vez que esta é descrita com base em fundamentos filosóficos, e representa também um meio de industrializar a produção de modelos conceituais, pois a modelagem da UFO é utilizada como base para a construção de novos modelos. Essa característica dupla representa um meio de sistematizar os conhecimentos adquiridos no decorrer desta pesquisa a respeito da UFO, de modo que eles possam ser compreendidos e aperfeiçoados teoricamente, além de utilizados em aplicações reais. As diferentes estratégias de incorporação de expressões e semântica modal ao arcabouço de Ontoprolog, discutidas no Capítulo 9, engendram técnicas para construção compartilhada de ontologias, pois oferecem instrumentos que auxiliam o processo obtenção de acordos a respeito da realidade, especialmente quando evidenciam as distinções nos discursos proferidos por diferentes agentes. Além disso, a exploração das formas de incorporação de raciocínios modais suportados por Programação em Lógica aclaram mecanismos, ferramentas e métodos de tratamento da informação úteis a projetos de Engenharia Lógica.

No que tange ao projeto de produção de um marco teórico para a Arquitetura da Informação por meio da sistematização de conceitos, conforme é discutido na seção 4.6 e no Capítulo 2, entende-se que a sistematização da UFO e sua descrição no arcabouço produzido por esta pesquisa exerce papel de referência teórica útil ao projeto da AI. Além disso, essa descrição funciona como instrumento para prototipagem de especificações de conceitos de AI que podem ser construídos com base naqueles providos pela UFO, ou mesmo por outras ontologias de fundamentação. Para avançar com primeiros experimentos a respeito da produção desse marco teórico de AI com base na UFO, na seção 11.3 apresenta-se o rascunho de uma ontologia referente à teoria do conhecimento utilizada na Arquitetura da Informação proposta por Lima-Marques (2011).

Por fim, a seção 11.4 descreve resultado referente ao software colaborativo abnTEX2, obtido no processo de investigação realizado, mas considerado extra-tese, pois não está listado nos objetivos originais do trabalho.

11.1 Alcance dos objetivos

Os objetivos específicos desta pesquisa, estabelecidos na subseção 1.2.2, são contribuições componentes do Objetivo Geral (subseção 1.2.1) *"propor uma linguagem formal textual para construção de discursos sobre entidades ontológicas"*. Na sequência, cada

objetivo específico é reescrito acrescido das evidências de como é alcançado nesta pesquisa:

a) *"definir uma especificação formal de uma linguagem textual de propósito específico para formalização de ontologias e implementar um interpretador dessa linguagem em Programação em Lógica Clássica"*: esse objetivo específico contempla dois objetivos diferentes, mas que são combinados propositalmente. O primeiro, de definir uma especificação formal de uma linguagem textual, é atingido no Capítulo 7, em que uma sintaxe formal, definida na notação EBNF, é proposta para Ontoprolog. A sintaxe é construída tendo em vista a otimização da leitura oral[2] das sentenças, para que ela possa ser facilmente verbalizada e utilizada diretamente na atividade de formalização de discursos sobre ontologia. Já a segunda parte do objetivo específico, a de implementar um interpretador dessa linguagem, é descrita no Capítulo 10. A implementação de referência presente naquele capítulo possui a característica de permitir que as sentenças da linguagem sejam incorporadas diretamente em programas Prolog que atendam à ISO/IEC 13211:1995 (ISO/IEC, 1995). A observação dessa norma visa garantir que o produto resultado dessa implementação seja funcional em implementações de Prolog de diferentes fornecedores, e que permita a utilização em programas existentes. O código-fonte produzido nesta pesquisa é disponibilizado como software livre, conforme seção 10.1. Do ponto de vista pragmático, a semântica operacional da linguagem permite que a especificação de modelos conceituais seja realizada de forma explorativa, por "tentativa e erro", uma vez que é possível obter respostas a respeito da adequação de determinada especificação em relação às regras da ontologia subjacente;

b) *"construir um sistema de regras que valide ontologias de universais e de particulares especificadas com base no fragmento endurante da Unified Foundational Ontology (UFO)"*: as regras de validação de ontologias de universais e de particulares da UFO são uma combinação das regras semânticas descritas no Capítulo 6 e no Capítulo 8, e que são baseadas no marco teórico descrito no Capítulo 4 e no Glossário da UFO. Este objetivo é totalmente alcançado devido à característica de metamodelagem incorporada em Ontoprolog, em que universais e particulares são passíveis de serem declarados como entidades de primeira classe no arcabouço lógico desenvolvido. Um aperfeiçoamento do sistema taxonômico da ontologia original da UFO (Apêndice D) é realizada com intuito de salientar as diferentes ontologias passíveis de serem declaradas e integradas na linguagem proposta. Essa personalização é abordada pelo Capítulo 5, e detalhada no Apêndice B e no Apêndice C. Embora este objetivo específico refira-se especificamente à UFO, o arcabouço de Ontoprolog é desenvolvido de

[2] Ver o Item a) da seção 11.3, que trata de trabalho futuro referente a esse aspecto da linguagem.

modo que outras ontologias de fundamentação possam ser incorporadas e até combinadas;

c) *"estender o arcabouço clássico da linguagem textual para inclusão de operadores modais referentes a discursos de diferentes agentes"*: três abordagens diferentes que visam ao alcance deste objetivo específico são apresentadas no Capítulo 9. A primeira demonstra como as sentenças sintáticas do arcabouço clássico de Ontoprolog, desenvolvidas no Capítulo 7 e no Capítulo 8, podem ser estendidas para contextos modais e submetidas a diferentes mecanismos de inferências no âmbito de raciocínio sobre conhecimento de vários agentes. Além disso, ainda na primeira abordagem, evidencia-se que o tratamento modal às sentenças é suficiente para verificar consistências globais na conjunção dos discursos de diferentes agentes a respeito de uma ontologia de domínio comum. Já a segunda abordagem reflete uma estratégia mais sofisticada de incorporação legítima de contextos modais não apenas nas especificações, mas também nos modelos que essas especificações denotam. Com base nessa estratégia, ainda no Capítulo 9 apresentam-se adaptações de regras de Ontoprolog baseadas no sistema $KD45_m$. A terceira abordagem descrita reflete a combinação lógica das duas anteriores, em que é possível tratar tanto sentenças quanto teorias com base em diferentes sistemas modais. A forma apresentada de combinação de Ontoprolog com o arcabouço de Programação em Lógica Modal MProlog possibilita que outras interpretações sejam dadas às modalidades, de modo não limitado às apresentadas. Porém, este trabalho não esgota o desenvolvimento de uma versão genuinamente modal das relações semânticas de Ontoprolog. Por isso, registra-se a implementação da solução apresentada no Capítulo 9 como indicação de trabalhos futuros.

11.2 Contribuições adicionais

Além dos objetivos inicialmente pretendidos, destaca-se que o desenvolvimento da pesquisa possibilitou que outras contribuições complementares fossem realizadas no âmbito da Ciência da Informação, Arquitetura da Informação, Programação em Lógica Modal e Modelagem Conceitual, conforme apresenta-se:

a) **Glossário da UFO**: a produção do glossário com termos e definição de conceitos encontrados na bibliografia da UFO, composto por 186 entradas, é uma sistematização conceitual que possibilita que se avance na racionalização e sistematização lógica da ontologia de fundamentação em tela. Além disso, o glossário traz ganhos bibliográficos e pedagógicos, porque organiza a bibliografia da área ao indicar a origem de cada definição obtida das obras originais. Do ponto

de vista do cientista da informação enquanto organizador do conhecimento científico, o desenvolvimento do glossário é uma contribuição direta à micro-biblioteconomia da área. Ainda no âmbito do glossário, ressalta-se a proposição de traduções para o português dos termos originais em inglês. Nesse aspecto, destaca-se a tradução do termo Endurante (Endurant), realizado com base em investigações linguísticas do uso do termo na língua portuguesa, conforme descrito em nota própria na entrada do glossário dedicada ao conceito;

b) **Sistematização de regras da UFO**: os quadros de regras para modelagem conceitual que utilizam a UFO como ontologia de fundamentação, apresentados na subseção 4.5.1, estendem e complementam aqueles apresentados em Guizzardi (2005, p. 317-320, 334-338, 348-352), Benevides (2010, p. 36-39, 45-50, 55-58) e trabalhos posteriores desses autores. Portanto, os quadros representam misto de marco teórico com resultados que complementam a sistematização compreendida no Item a);

c) **Abordagem de Programação em Lógica para tratamento de ontologias**: não se encontrou na bibliografia analisada abordagem semelhante à adotada nesta pesquisa para formalização de ontologias de modo metamodelado, construída com base em uma linguagem interna, criada de forma incorporada e acomodada na própria sintaxe base de sentenças Prolog. Por ser integrada à plataforma de prova assistida por computador da Programação em Lógica, a arquitetura construída permite o arranjo de diversas abordagens lógicas, e não apenas as adotadas nesta pesquisa. Diante disso, entende-se que esta pesquisa contribui diretamente com a área de Modelagem Conceitual ao demonstrar a possibilidade de combinação de técnicas e paradigmas lógicos diferentes e, dessa forma, apresentar novos e instrumentos de especificação de ontologias. Além disso, esse resultado não poderia ter sido atingido sem o desenvolvimento de uma linguagem nova, baseada em operadores, criada com fim específico de ser incorporada em programas Prolog;

d) **Uso de MProlog para formalização de ontologias com base em semânticas de multiagentes**: o uso de MProlog como instrumento para formalização de ontologias, especialmente ontologias baseadas em UFO, ainda não havia sido explorado. Dessa forma, trata-se de uma contribuição adicional desta pesquisa a demonstração da aplicabilidade no uso de linguagens com expressividade modal no paradigma de Programação em Lógica para tratamento de ontologias. Especialmente, destaca-se a proposta de semânticas de multi-agentes para tratamento de integração de ontologias e buscas de acordos e consensos entre vários ontologistas;

e) **Versão textual da UFO**: embora não tenha sido encontrado na literatura da

área clamor por uma versão textual da UFO, a sintaxe estendida de Ontoprolog, apresentada no Capítulo 8, pode ser entendida como uma versão textual de OntoUML, ou mais propriamente, como uma versão textual da linguagem inerente à ontologia UFO no que tange à construção de modelos conceituais. Dessa forma, a linguagem pode funcionar como uma representação textual de modelos conceituais visuais, uma vez que é independente do paradigma de Programação em Lógica e criada especificamente tendo a UFO como referência;

f) **Representação de teorias em linguagem diagramática**: além da versão textual da UFO, este trabalho também demonstra a possibilidade de tradução das teorias de Ontoprolog para outros formalismos, como os que envolvem linguagens de representação visual, a exemplo da OntoUML. A seção 10.3 descreve como diagramações visuais podem ser produzidas a partir de teorias Ontoprolog. Ao longo deste texto, muitas das representações visuais são geradas a partir de especificações de teorias Ontoprolog baseadas em modelos UML. Além disso, este trabalho apresenta aprimoramentos às representações de modelos conceituais baseado na linguagem visual OntoUML. Esses aprimoramentos incluem formas concisas de representar as relações entre Momentos Intrínsecos (seção 8.1), compartimentos nos Classifiers que detalham restrições lógicas, entre outros. Uma das vantagens de se utilizar representação diagramático em complemento à especificação textual é facilitar a comunicação dos modelos descritos em Ontoprolog a indivíduos não conhecedores da linguagem técnica;

g) **Novos instrumentos à Engenharia Lógica**: o arcabouço de Programação em Lógica de Ontoprolog e as integrações com o arcabouço modal de MProlog são resultados de Arquitetura da Informação obtidos por meio da Engenharia Lógica (seção 2.4). Porém, mais do que frutos, eles são também novos instrumentos de Engenharia Lógica disponíveis ao arquiteto da informação, uma vez que podem ser combinados com outras lógicas e ferramentas para obtenção de outros resultados.

11.3 Possibilidades de pesquisas futuras

Visando contribuir com trabalhos futuros, apresenta-se uma série de propostas que intercalam aprimoramentos possíveis à pesquisa atual, avanço em deficiências identificadas e ideias para propostas alternativas, complementares e aperfeiçoadas das empregadas neste trabalho. Primeiros esboços de trabalhos já em andamento também são indicados.

a) *validação da capacidade de oralizar a sintaxe de Ontoprolog*: a sintaxe de Ontoprolog apresentada no Capítulo 7 e estendida na seção 8.2 é proposta com a intenção de que sentenças concretas construídas com base nessa sintaxe possam

ser oralizadas, ou seja, lidas em voz alta, de modo que sirvam como instrumento operacional de construção de ontologias em modo colaborativo, e funcionem especialmente como uma espécie de "língua franca" entre especialistas. Porém, a eficácia e a adequação dos constructos sintáticos a esses objetivos não foram validados no âmbito desta pesquisa. Trabalhos futuros podem realizar testes empíricos com intuito de obter essas validações. Além disso, considerando o modo como a sintaxe e o protótipo de Ontoprolog são construídos, é perfeitamente possível, e relativamente simples, alterar as construções sintáticas e as funções filtro (subseção 7.3.1) para que reflitam construções mais adequadas aos objetivos apresentados. Essa simplicidade é justificada devido ao modo como os operadores de Prolog (base da sintaxe de Ontoprolog) podem ser combinados para formar árvores sintáticas;

b) *novas semânticas modais para as teorias de Ontoprolog*: na seção 9.2 apresenta-se os caminhos para o desenvolvimento de uma semântica modal de vários agentes com base nas relações que denotam o significado das especificações de Ontoprolog. Entretanto, embora sejam indicados os resultados e construída a estratégia de definição da semântica de vários agentes para as teorias de Ontoprolog, não se apresenta nesta pesquisa uma implementação total desses resultados. Por isso, trabalhos futuros poderão utilizar os resultados apresentados no Capítulo 9 para implementar a versão genuinamente modal de Ontoprolog com base nos sistemas modais estudados de MProlog. Cabe ressaltar, no entanto, que a estratégia de enriquecer as sentenças de Ontoprolog com expressões modais, conforme é discutido na seção 9.1, estão implementadas e disponíveis no código-fonte descrito pelo Capítulo 10. Adicionalmente, sugere-se o desenvolvimento de semânticas modais temporais, que podem ser atribuídas a conceitos ontológicos não rígidos, especialmente ontologias com perdurantes;

c) *implementação da semântica modal da UFO em MProlog*: a natureza alética da semântica original da UFO, apresentada em Guizzardi (2005) sobre um sistema $S5$, pode ser implementada utilizando um dos sistemas monomodais de MProlog. Porém, a atual implementação do sistema $S5$ de MProlog não se mostrou adequada, pois alguns resultados não esperados foram obtidos quando determinadas consultas que envolviam fatos clássicos e deduções modais eram combinadas. De toda forma, o Código 30 é uma adaptação da combinação do Código 25 e do Código 26 que expõe um estudo preliminar que usa o sistema monomodal serial T de MProlog para implementar uma noção de Rigidez (Rigidity) com interpretação alética. Trabalhos futuros podem investigar a implementação do sistema $S5$ de MProlog e transcrever as regras clássicas de Ontoprolog apresentadas no Capítulo 6 e na seção 8.3, conforme orientações contida no Capítulo 9, com objetivo de obter um sistema modal com a interpretação original proposta

para a UFO implementado em Programação em Lógica;

Código 30 – Estudo de semântica modal da UFO: arquivo MProlog

```
:- calculus cT.  % □φ → φ
:- set_option(current_calculus, cT).
% No sistema monomodal $T$ ($K + T$), \texttt{[b]} representa a
    modalidade universal.

%============================================================================
% Semantic relations produced by translation filter (=>m)
%============================================================================

[b] : entity(kind).
[b] : entity(phase).
[b] : entity(role).
[b] : entity(alien).
[b] : entity(humano).
[b] : entity(nave).
[b] : entity(planeta).
[b] : entity(adulto).
[b] : entity(menino).
[b] : entity(alien_visitante).
[b] : entity(et).
[b] : entity(elliott).
[b] : entity(nave_et).
[b] : dio(alien,kind).
[b] : dio(humano,kind).
[b] : dio(nave,kind).
[b] : dio(planeta,kind).
[b] : dio(adulto,phase).
[b] : dio(menino,phase).
[b] : dio(alien_visitante,role).
[b] : dio(et,alien_visitante).
[b] : dio(elliott,menino).
[d] : dio(nave_et,alien).
[d] : dio(nave_et,nave).
[b] : deo(adulto,humano).
[b] : deo(menino,humano).
[b] : deo(alien_visitante,alien).
[b] : dd([menino,adulto]).

%============================================================================
% Closures represented as ground facts in order to keep this simpler
%============================================================================

[b] : dio(elliott, humano).
[b] : dio(et, alien).

%============================================================================
% rigid_type/1
%============================================================================

%% rigid_type(?T)
[b] : rigid_type(T) :-          % universal modal rigidity
        [b] : dio(T, kind),
        % set Xs of the instances of the universal T
        findall(X,
                mcall([d] : dio(X, T)),
                Xs),
        findall(S,
                (member(X1, Xs),
                 mcall([d] : dio(X1, S)),
                 S \= T,
                 mcall([d] : dio(S, kind))),
                []).
[b] : rigid_type(R) :-          % universal modal rigidity
        [b] : dio(R, kind),
        \+ mcall([d] : dio(_, R)).
[d] : rigid_type(T) :-          % existential modal rigidity
        [d] : dio(T, kind).
```

d) *expansão e substituição das meta-teorias*: conforme justificativa apresentada na subseção 1.3.4, a restrição do escopo da pesquisa ao fragmento Endurante (Endurant) da UFO (UFO-A) é uma escolha metodológica, e não reflete a abrangência necessária da ontologia de fundamentação para aplicação em contextos reais.

Idealmente, todos os aspectos ontológicos de uma ontologia de fundamentação devem ser considerados em situações factuais, especialmente na construção de arquiteturas da informação, que envolvem aspectos transdisciplinares, conforme abordado na seção 2.1. Embora o corte epistemológico realizado no escopo do trabalho limite a abrangência da pesquisa, entende-se que os resultados obtidos podem ser estendidos para alcançar outros aspectos de ontologias de fundamentação, como os conceitos que envolvem perdurantes (UFO-B), objetos sociais (UFO-C) e serviços (UFO-S), conforme apresentadas na seção 4.3. Por isso, trabalhos futuros podem especificar essas ontologias com base no arcabouço lógico desenvolvido nesta pesquisa, de modo que funcionem como metateorias de Ontoprolog adequadas para especificação de ontologias de domínio. Nessa linha, o conceito de Disposição (Disposition), embora mencionado no Glossário da UFO, não foi tratado nesta pesquisa, mas sugere-se ser alvo de trabalhos futuros que envolvam a formalização da UFO-B em Ontoprolog;

e) *aperfeiçoamento da semântica de Qualities*: o mecanismo de formalização de Qualidades (Quality), e a ontologia UFO descrita no Apêndice B, podem ser aperfeiçoadas para que as categorias propostas por Probst (2007, p. 47-75) e arranjadas por Albuquerque e Guizzardi (2013) para o modelo fundacional da UFO. Exemplos são as taxonomias de conceitos como Região de Qualidade (Quality Region) e as subcategorias de Universais de Qualidade (Quality Universal) e Estruturas de Qualidade (Quality Structure), que podem figurar como vocabulário na linguagem de Ontoprolog, além de funcionarem como elementos semânticos às regras de verificações de modelos. Além desses, há também a possibilidade de inclusão do conceito de Espaço Métrico (Metric Space), entre outros;

f) *aprimoramento dos Relators*: a natureza ontológica dos Relators (Relator) é discutida em obra recente de Guarino e Guizzardi (2015), de modo que essas relações genuínas dependentes externamente de outros indivíduos são vistos com características substanciais, e não meros Momentos (Moment). Essa visão altera a estrutura base da ontologia UFO descrita nesta tese. Além disso, decisões técnicas específicas aos Relators foram tomadas em Ontoprolog de modo mais restrito do que aquelas encontradas na bibliografia original. Essas decisões são discutidas na subseção 8.4.8 tendo um exemplo estendido como pano de fundo. Outras limitações referentes a ausência de regras do tipo $ngrule/n$ referentes às entidades que denotam Relators também são apresentadas na subseção 8.4.7.1 Por isso, trabalhos futuros devem tanto aprimorar a descrição da UFO em Ontoprolog com base em resultados recentes de pesquisa, como afastar as limitações conhecidas;

g) *anti-padrões da UFO*: *padrões* e *anti-padrões* podem ser tanto sugeridos como

identificados com base em teorias de Ontoprolog que atendam determinadas formas lógicas. Nesse caso, resultados a respeito da identificação de padrões e anti-padrões na UFO podem ser utilizados como base para o desenvolvimento desse tipo de verificação na linguagem. Nessa linha, pode-se mencionar os trabalhos Guizzardi, Graças e Guizzardi (2011), Guizzardi (2014) e Sales e Guizzardi (2015), além de Sales, Barcelos e Guizzardi (2012), que descrevem estudo sobre anti-padrões baseado em simulação. Este trabalho apresenta resultados interessantes, como o fato de cerca de 30% dos estudos analisados possuírem modelagens que os autores chamaram de *pseudo-anti-rigid anti-pattern*. Tratam-se de modelos cujas restrições impedem de determinado tipo T nunca deixe de possuir uma característica anti-rígida, que pela própria definição de anti-rigidez deveria haver ao menos um mundo possível em que T não esteja associado a essa característica. Em Ontoprolog, esse tipo de verificação pode ser realizada definindo-se predicados que denotem regras negativas, a exemplo dos RNP (subseção 6.3.1);

h) *aperfeiçoamento da sintaxe específica da UFO*: a sintaxe de `relates` pode ser aprimorada para que possa englobar a declaração simultânea das relações *dso*/2 subseção 7.4.9 e *dro*/2 subseção 7.4.10. Isso visa aumentar a concisão da linguagem proposta para Ontoprolog;

i) *aprimoramento do aspecto meta-teórico de Ontoprolog*: em Carvalho et al. (2015) e Carvalho e Almeida (2015) propõe-se uma teoria para modelagem conceitual baseado em meta-modelagem (MLT). Trabalhos futuros podem investigar a proposta da MLT com intuito de aprimorar as relações básicas de Ontoprolog, especialmente as relações de *instanciação* e de *subsunção*. No caso da relação de instanciação, a distinção entre a instanciação de (meta)modelos e de particulares de uma teoria, discutida na seção 5.5, poderia enriquecer o arcabouço teórico de Ontoprolog com a ideia de modelagem multi-nível;

j) *description logic*: de certa forma, o que é desenvolvido neste trabalho é um tipo de *sistema de descrição*, no sentido de uma linguagem que permite descrever conceitos. Porém, esse tipo de classificação seria irrelevante aos objetivos, à característica e à justificativa da pesquisa, embora fosse útil para apresentar a correlação entre modelos lógicos desenvolvidos e outros sistemas de descrição. *Description Logic* (DL) é um importante conjunto de formalismos de representação de "conhecimento", propositalmente construídos de forma que se mantenha a propriedade da decibilidade. Conforme Baader, Horrocks e Sattler (2008, p. 135):

> *The name* description logics *is motivated by the fact that, on the one hand, the important notions of the domain are described by concept descriptions, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL; on the other hand, DLs*

> *differ from their predecessors, such as semantic networks and frames, in that they are equipped with a formal,* logic-*based semantics.*

O que é proposto neste trabalho é um sistema de descrição de conceitos cuja sintaxe e a semântica são formalmente definidas. Entretanto, trata-se de uma interpretação específica de linguagens formais baseada em ontologias de fundamentação e Programação em Lógica. Por isso, apresenta-se implementação em Prolog de uma linguagem de propósito específico com o fim estrito de formalizar discursos sobre ontologias de domínio sustentados por ontologias de fundamentação. DL, por outro lado, são lógicas de propósito geral, embora especializadas na atividade genérica de descrição de conceitos. Por conseguinte, trabalhos futuros interessados em propriedades lógicas específicas, como garantia de decidibilidade, podem definir o sistema construído nesta pesquisa em DL. Nesse sentido, este seria o caso em que uma DL específica serviria de lógica subjacente e substituiria (ou complementaria) as regras semânticas definidas no Capítulo 6. No entanto, isso não é realizado aqui porque o foco de interesse é a obtenção de resultados que exercitam características específicas de Prolog, e de seus módulos de programação modal. Nesses termos, o paradigma de DL não contribuiriam diretamente a esse propósito. De toda forma, sugere-se que evoluções futuras desta pesquisa nessa linha de investigação considerem o trabalho de Józefowska e Łukaszewski (2010), de Lukácsy, Szeredi e Kádár (2008) – referente a uma implementação do *DLog system*, um motor de raciocínio para a parte ABox da lógica $\mathcal{SHIQ}$ – e o trabalho de Herchenröder (2006) – que consistem em uma implementação de DL em Prolog pelo método semântico de *tableaux*;

k) *interação homem-máquina*: os resultados desta pesquisa que tangem ao protótipo de linguagem para descrição de ontologias poderiam ser utilizados como instrumento de fundação em investigações de aspectos pragmáticos de construção de ontologias com base em interação homem-máquina. Nessa linha, sugere-se a análise do método desenvolvido por Graças (2010), Graças e Guizzardi (2010), que visa construir ontologias com base na UFO por meio de perguntas em interfaces com usuários;

l) *integração com OLED* e *tradução para outros formalismos*: conforme Sales (2014, p. 237), o software *OntoUML Lightweight Editor* (OLED) é um aplicativo livre construído com base em resultados de pesquisas com a UFO. O sistema provê interface gráfica para descrever, manipular, validar e simular modelos OntoUML, além de permitir que os modelos produzidos sejam exportados em diferentes formatos. Nesse contexto, pesquisas futuras podem desenvolver mecanismos de transformação de modelos OLED em representações na sintaxe de Ontoprolog, e também traduções de teorias de Ontoprolog em modelos

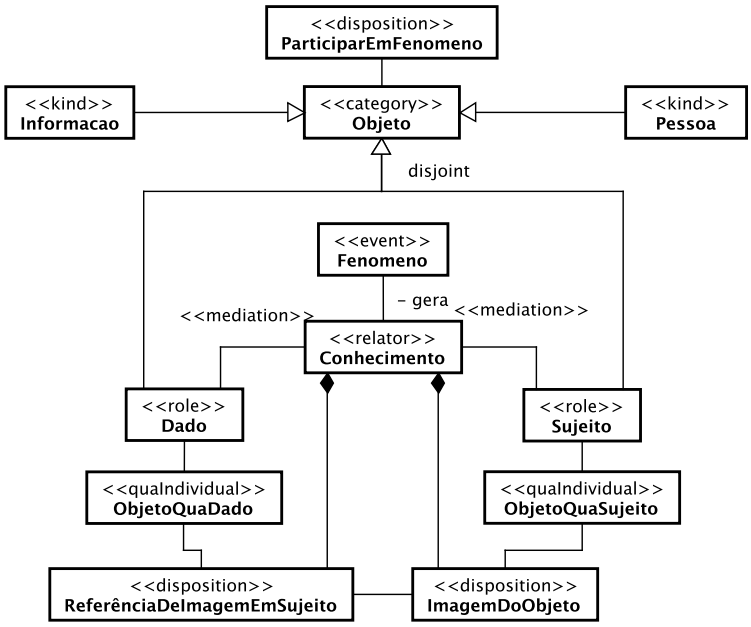
representáveis visualmente pelo OLED. Desse modo, seria possível combinar as características positivas de cada abordagem para se obter resultados ainda mais amplos. Especialmente, essa estratégia poderia explorar possibilidades de representação de conceitos no contexto de diferentes arcabouços meta-lógicos de Ontoprolog, como no âmbito de sistemas modais, além de permitir que diferentes motores de regras baseados em Programação em Lógica, que vão além do paradigma de OCL, sejam utilizados na verificação de padrões e anti-padrões de modelos de ontologias. Paralelamente, conforme abordado na seção 10.3, podem ser propostas traduções das teorias Ontoprolog para outros formalismos, de modo a aumentar interoperabilidade dos modelos produzidos com a linguagem, como RDF[3], UXF (SUZUKI; YAMAMOTO, 1999), XMI (OBJECT MANAGEMENT GROUP, 2014b), entre outros. Adicionalmente, trabalhos futuros podem propor representações visuais para as teorias modais, especialmente as que envolvem agentes, referentes aos sistemas lógicos abordados na seção 3.2 e no Capítulo 9;

m) *ontologia de fundamentação para Arquitetura da Informação*: o projeto de sistematização de conceitos da Arquitetura da Informação pode utilizar os marcos teóricos e as teorias desenvolvidas nesta pesquisa como base de referência sobre a UFO para que uma ontologia própria para AI seja definida. Do ponto de vista metodológico, a implementação de referência de Ontoprolog pode servir de instrumento auxiliar à prototipagem de modelos conceituais de AI baseados na UFO, ou em outra ontologia de fundamentação que seja considerada mais adequada. Para exercitar essa tese, a Figura 50 ilustra uma proposta de ontologia de fundamentação para a AI. Na figura está retratada, por meio de conceitos da UFO-A e da UFO-B, o aspecto fenomenológico básico da correlação entre *Sujeito* e *Objeto* entendida como *Conhecimento* (seção 2.2). No caso, *Objeto* é proposta como uma categoria endurante rígida, da qual *Pessoa* e *Informação* são espécies com critérios de identidade próprios e rígidos em todos os mundos possíveis (Kind). Essa classificação estabelece que *Informação* não se confunde com Trope, no sentido comumente encontrado na literatura de informação como *propriedade de algo*. Pelo contrário, informação é compreendida como uma entidade rígida e própria da natureza. Essa posição está alinhada com Hofkirchner (1998), Hofkirchner (2011) e Lima-Marques (2011). Ainda na proposta da ontologia, *Dado* é um papel exercido por um *Objeto* – que na figura pode ser exercido tanto por uma *Pessoa*, quanto por uma *Informação* – em um Relator que correlaciona outro *Objeto* no papel de *Sujeito*. Essa correlação é fruto do Evento (Event) conhecido como *fenômeno da experiência* entre *objetos* nos papeis de *Sujeito* e *Dado*. Esses objetos exercem seus papeis no como Qua Individuals que possuem

[3] <http://www.w3.org/standards/techs/rdf#w3c_all>

Dispositions específicos no evento da experiência. No caso do *ObjetoQuaDado*, a disposição é de ser referência a uma imagem em um *sujeito*. No caso do *ObjetoQuaSujeito*, a disposição é a própria captura da imagem do dado a partir do fenômeno do *Conhecimento*. Esta proposta de ontologia de fundamentação da Arquitetura da Informação ainda carece de aperfeiçoamento e, por não ser foco específico desta tese, é indicada como trabalho futuro;

Figura 50 – Proposta inicial de uma ontologia de fundamentação para AI



Fonte: Os autores

n) *adoção de paradigmas paraconsistentes*: os sistemas lógicos construídos neste trabalho possuem a capacidade de identificar e de realizar tratamento não trivial na presença de incompatibilidade entre teorias e metateorias. A "incompatibilidade", entendida como um tipo de inconsistência entre modelos, acontece quando uma teoria mais específica não é *adequadamente* uma instância da teoria mais geral. Isso pode acontecer devido à inobservância na teoria específica de restrições da teoria geral, por exemplo. O sistema Ontoprolog é capaz de reportar essas inconsistências e mesmo assim não assumir qualquer conclusão de suas consultas, ou seja, de não se tornar trivialmente satisfatível. Essa característica pode ser uma centelha de raciocínio paraconsistente. Pela ausência de uma negação explícita, Ontoprolog é em alguma medida um sistema anódico, embora a negação por falha esteja presente na meta-teoria da programação em Lógica. Nesse enfoque, trabalhos futuros podem avançar pelo caminho de raciocínio paraconsistente atribuído à noção de prova dos modelos conceituais baseados em ontologias. Sugere-se como referência inicial o trabalho de Leite

(2003) e, especialmente, de Bueno-Soler (2009), devido a resultados com sistemas modais paraconsistentes, que podem servir de base para aperfeiçoamento dos sistemas modais de MProlog. Especificamente no âmbito de Programação em Lógica, Pais (2004) e Amo e Pais (2007) realizaram contribuições no âmbito de sistemas e de banco de dados dedutivos paraconsistentes. A pesquisa básica de Rodrigues (2010) apresenta resultados referentes modelos de Herbrand paraconsistentes, e são a base para a criação de um sistema de Programação em Lógica genuinamente paraconsistente;

o) *aprimoramento do Glossário*: o Glossário da UFO foi produzido com o objetivo de servir de referência secundária aos trabalhos da UFO, e tem o intuito de ser uma base inicial para a construção de um tesauro do tema, nos moldes apresentados pela ISO 25964:2011 (*Information and documentation - Thesauri and interoperability with other vocabularies - Part 1: Thesauri for information retrieval* (ISO, 2011). Porém, na atual versão, diversos conceitos da UFO não estão presentes. O glossário contém os conceitos utilizados no âmbito desta pesquisa e poucos outros, usados para complementá-los. Além disso, vários conceitos foram intencionalmente mantidos fora do escopo, como *Factual Universal* (GUIZZARDI, 2002; GUIZZARDI; WAGNER, 2008), e *unifying condition* (GUIZZARDI, 2011), apenas para citar alguns. Por isso, trabalhos futuros podem ser empreendidos para a organização de um tesauro mais amplo a respeito de termos e conceitos da UFO.

11.4 Resultado extra-tese: abnTEX2

Além dos resultados diretos obtidos com a pesquisa em Arquitetura da Informação, no processo de doutoramento também foi oportuno desenvolver um projeto colaborativo de software livre, chamado abnTEX2 (<http://www.abntex.net.br>). Trata-se da evolução do abnTEX (*ABsurd Norms for TeX*), uma suíte para LATEX que atende os requisitos das normas da Associação Brasileira de Normas Técnicas (ABNT) para elaboração de documentos técnicos e científicos brasileiros, como artigos científicos, relatórios técnicos, trabalhos acadêmicos como teses, dissertações, projetos de pesquisa e outros documentos do gênero.

O trabalho com abnTEX2 já resultou em dezenas de customizações específicas para diversas universidades brasileiras, e já recebeu colaboração direta de dezenas de pessoas.

Uma apresentação sobre o trabalho foi realizada na *XI Conferência Latino-americana de SOftware Livre* (Latinoware 2014), realizada entre os dias 15 e 17 de outubro de 2014 em Foz do Iguaçu, PR.

Referências

ABBAGNANO, N. *Dicionário de Filosofia*. Edição revista e ampliada. São Paulo: Martins Fontes, 2007. Citado na página 129.

ALBUQUERQUE, A.; GUIZZARDI, G. An ontological foundation for conceptual modeling datatypes based on semantic reference spaces. In: *Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference on*. [s.n.], 2013. p. 1–12. ISSN 2151-1349. Disponível em: <http://www.inf.ufes.br/~gguizzardi/PID2733627.pdf>. Acesso em: 2 jan 2014. Citado 14 vezes nas páginas 59, 60, 143, 144, 307, 399, 438, 439, 472, 480, 481, 482, 483 e 484.

ALBUQUERQUE, A. R. R. de. *Discurso sobre fundamentos de Arquitetura da Informação*. Tese (Doutorado) — Universidade de Brasília, Brasília, 2010. Citado 3 vezes nas páginas 62, 76 e 77.

ALBUQUERQUE, A. R. R. de; LIMA-MARQUES, M. Sobre os fundamentos da arquitetura da informação. *Perspectivas em Ciência da Informação*, v. 1, n. Especial (2011), p. 60–72, 2011. Disponível em: <http://periodicos.ufpb.br/ojs2/index.php/pgc/article/view/10827/6075>. Acesso em: 2 nov 2011. Citado na página 62.

ALMEIDA, J.; GUIZZARDI, G. On the foundation for roles in RM-ODP: Contributions from conceptual modelling. In: *EDOC Conference Workshop, 2007. EDOC '07. Eleventh International IEEE*. [S.l.: s.n.], 2007. p. 205–215. Citado na página 437.

ALMEIDA, M. B. Uma abordagem integrada sobre ontologias: Ciência da Informação, Ciência da Computação e Filosofia. *Perspectivas em Ciência da Informação*, v. 19, n. 3, p. 242–258, 2014. Disponível em: <http://portaldeperiodicos.eci.ufmg.br/index.php/pci/article/view/1736/1448>. Acesso em: 30 jun 2015. Citado 3 vezes nas páginas 51, 130 e 131.

AMERICAN ASSOCIATION OF WORKERS FOR THE BLIND. *The Nemeth Braille code for mathematics and science notation: 1972 revision*. Louisville, Kentucky: American Printing House for the blind, 1987. Disponível em: <http://www.brailleauthority.org/mathscience/nemeth1972.pdf>. Acesso em: 25 abr 2014. Citado na página 115.

AMO, S. de; PAIS, M. S. A paraconsistent logic programming approach for querying inconsistent databases. *International Journal of Approximate Reasoning*, v. 46, n. 2, p. 366 – 386, 2007. ISSN 0888-613X. Special Track on Uncertain Reasoning of the 18th International Florida Artificial Intelligence Research Symposium (FLAIRS 2005). Disponível em: <http://www.sciencedirect.com/science/article/pii/S0888613X06001307>. Acesso em: 2 nov 2013. Citado 2 vezes nas páginas 122 e 404.

ANTONIOU, G. *Nonmonotonic Reasoning*. [S.l.]: The MIT Press, 1997. (Artificial Intelligence). With contributions by Mary-Anne Wiliams. Citado na página 65.

ARARIBÓIA, G. *Inteligência Artificial: Um Curso Prático*. Rio de Janeiro: Livros Técnicos e Científicos Editora Ltda., 1988. (Aplicações de Computadores). Citado na página 93.

ARAUJO, L. C. *Configuração: uma perspectiva de Arquitetura da Informação da Escola de Brasília*. Dissertação (Mestrado) — Universidade de Brasília, Brasília, Março 2012. Citado 8 vezes nas páginas 59, 62, 82, 83, 84, 85, 86 e 125.

ARAUJO, L. C.; LIMA-MARQUES, M. Configuração da informação? *Ciência da Informação*, 2014. Artigo enviado para publicação em 12 jun 2014. Citado 3 vezes nas páginas 75, 82 e 83.

ARAÚJO, S. L.; ACIÓLY, B. M. *Introdução ao Haskell*. Vitória da Conquista, BA: Edições Uesb, 2008. Citado na página 70.

ARMS, C. et al. *The benefits and risks of the PDF/A-3 file format for archival institutions*: an NDSA report. [S.l.], 2014. National Digital Stewardship Alliance. Disponível em: <http://hdl.loc.gov/loc.gdc/lcpub.2013655115.1>. Acesso em: 23 set 2014. Citado na página 70.

ARMSTRONG, J. L.; VIRDING, S. R.; WILLIAMS, M. C. Use of Prolog for developing a new programming language. *The Practical Application of Prolog*, Institute of Electrical Engineers, London, v. 1, n. 3, April 1992. Disponível em: <http://www.erlang.se/publications/prac_appl_prolog.pdf>. Acesso em: 2 dez 2013. Citado 3 vezes nas páginas 115, 125 e 260.

ARNDT, R. et al. Comm: Designing a well-founded multimedia ontology for the web. In: ABERER, K. et al. (Ed.). *The Semantic Web*. Springer Berlin Heidelberg, 2007, (Lecture Notes in Computer Science, v. 4825). p. 30–43. ISBN 978-3-540-76297-3. Disponível em: <http://iswc2007.semanticweb.org/papers/029.pdf>. Acesso em: 26 ago 2014. Citado na página 60.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *ABNT NBR 14724:2011*: Informação e documentação — trabalhos acadêmicos — apresentação. Rio de Janeiro, 2011. 15 p. Citado na página 42.

ATKINSON, C.; KÜHNE, T. Meta-level independent modelling. In: *Proceedings of the International Workshop Model Engineering (in conjunction with ECOOP'2000)*. Cannes, France: [s.n.], 2000. Disponível em: <http://homepages.mcs.vuw.ac.nz/~tk/publications/papers/level-indep.pdf>. Acesso em: 23 jul 2014. Citado na página 161.

AUSTIN, J. L. *How to do things with words*. Great Britain: Oxford University Press, 1962. Citado na página 87.

AZEVEDO, R. F. de B. C. *Um modelo ontológico do sistema eleitoral brasileiro*. Dissertação (Mestrado) — Universidade de Brasília, Faculdade de Ciência da Informação, Brasília, agosto 2014. Citado na página 60.

BAADER, F.; HORROCKS, I.; SATTLER, U. Description logics. In: HARMELEN, F. van; LIFSCHITZ, V.; PORTER, B. (Ed.). *Handbook of Knowledge Representation*. [S.l.]: Elsevier B.V., 2008. p. 135–179. Citado na página 400.

BACKUS, J. W. et al. Report on the algorithmic language ALGOL 60. *Commun. ACM*, ACM, New York, NY, USA, v. 3, n. 5, p. 299–314, maio 1960. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/367236.367262>. Acesso em: 5 dez 2013. Citado na página 262.

_____. Revised report on the algorithm language ALGOL 60. *Commun. ACM*, ACM, New York, NY, USA, v. 6, n. 1, p. 1–17, jan. 1963. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/366193.366201>. Acesso em: 5 dez 2013. Citado na página 262.

BALBIANI, P.; HERZIG, A.; LIMA-MARQUES, M. TIM: The Toulouse inference machine for non-classical logic programming. In: . [S.l.]: Springer-Verlag, 1991. p. 365–382. Citado na página 104.

_____. Implementing Prolog extensions: A parallel inference machine. In: . [S.l.: s.n.], 1992. p. 833–842. Citado na página 104.

BARCELLOS, M. P.; FALBO, R. A.; DAL MORO, R. A well-founded software measurement ontology. In: *Proceedings of the 2010 Conference on Formal Ontology in Information Systems: Proceedings of the Sixth International Conference (FOIS 2010)*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2010. p. 213–226. Disponível em: <http://www.nemo.inf.ufes.br/files/a_well_founded_software_measurement_ontology_2010.pdf>. Acesso em: 26 ago 2014. Citado na página 61.

BARCELOS, P. P. F.; GUIZZARDI, R. S. S.; GARCIA, A. S. An ontology reference model for normative acts. In: BAX, M. P.; ALMEIDA, M. B.; WASSERMANN, R. (Ed.). *Proceedings of the 6th Seminar on Ontology Research in Brazil*. Belo Horizonte, Brazil: [s.n.], 2013. p. 35–36. Disponível em: <http://ceur-ws.org/Vol-1041/ontobras-2013_paper14.pdf>. Acesso em: 29 jul 2014. Citado na página 61.

BATISTA, R. B. Reposicionando o endurantismo na actual metafísica da persistência, segundo a analogia do ser. *Revista Portuguesa de Filosofia*, tomo 69, fasc. 2, p. 215–240, 2013. Disponível em: <http://www.jstor.org/discover/10.2307/23631106>. Acesso em: 27 out 2014. Citado 2 vezes nas páginas 449 e 477.

BENEVIDES, A. B. *A Model-based Graphical Editor for Supporting the Creation, Verification and Validation of OntoUML Conceptual Models*. Dissertação (Mestrado) — Universidade do Espírito Santo. Departamento de Informática, Vitória, February 2010. Citado 12 vezes nas páginas 141, 144, 147, 164, 167, 395, 439, 445, 457, 463, 479 e 495.

BENEVIDES, A. B. et al. Assessing modal aspects of OntoUML conceptual models in Alloy. In: HEUSER, C. A.; PERNUL, G. (Ed.). *ER 2009 Workshops, LNCS 5833*. Berlin Heidelberg: Springer-Verlag, 2009. p. 55–64. Disponível em: <http://www.inf.ufes.br/~gguizzardi/benevides-et-al-2009.pdf>. Acesso em: 20 ago 2013. Citado na página 164.

_____. Validating modal aspects of ontouml conceptual models using automatically generated visual world structures. *Journal of Universal Computer Science*, v. 16, n. 20, p. 2904–2933, 2010. Disponível em: <http://www.jucs.org/jucs_16_20/validating_modal_aspects_of/jucs_16_20_2904_2933_benevides.pdf>. Acesso em: 29 jul 2014. Citado na página 58.

BORGO, S.; MASOLO, C. Foundational choices in DOLCE. In: STAAB, S.; STUDER, R. (Ed.). *Handbook on Ontologies*. Second edition. EUA: Springer, 2008. p. 361–381. Citado na página 133.

BOWEN, J.; JIFENG, H.; PANDYA, P. An approach to verifiable compiling specification and prototyping. In: DERANSART, P.; MALUSZYŃSKI, J. (Ed.). *Programming Language Implementation and Logic Programming*. Springer Berlin Heidelberg, 1990, (Lecture Notes in Computer Science, v. 456). p. 45–59. ISBN 978-3-540-53010-7. Disponível em: <http://dx.doi.org/10.1007/BFb0024175>. Acesso em: 12 dez 2013. Citado na página 116.

BRAGA, B. F. B. et al. Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal method. *Innovations Syst Softw Eng*, v. 6, p. 55–63, 2010. Disponível em: <http://www.inf.ufes.br/~gguizzardi/braga-et-al-ISSE10.pdf>. Acesso em: 20 ago 2013. Citado na página 62.

BRANDÃO, H. H. N. *Introdução à análise do discurso*. 2. ed. Campinas, SP: Editora da Unicamp, 2004. Citado na página 89.

_____. Analisando o discurso. *Estação da Luz, Museu da Língua Portuguesa*, may 2009. Disponível em: <http://www.estacaodaluz.org.br/colunas_interna.php?id_coluna=1>. Acesso em: 31 jan 2013. Citado na página 87.

BRINGHURST, R. *Elementos do estilo tipográfico*. Versão 3.0. Canadá: Cosac Naify, 2005. Citado na página 81.

BRINGUENTE, A. C. de O.; FALBO, R. de A.; GUIZZARDI, G. Using a foundational ontology for reengineering a software process ontology. *Journal of Information and Data Management*, v. 2, n. 3, p. 511–526, 2011. Disponível em: <http://seer.lcc.ufmg.br/index.php/jidm/article/view/164>. Acesso em: 13 set 2013. Citado 3 vezes nas páginas 60, 61 e 136.

BRYANT, B.; PAN, A. Rapid prototyping of programming language semantics using Prolog. In: *Computer Software and Applications Conference, 1989. COMPSAC 89., Proceedings of the 13th Annual International*. [S.l.: s.n.], 1989. p. 439–446. Citado na página 116.

BU, Y. et al. Scaling Datalog for machine learning on big data. *CoRR*, abs/1203.0160, 2012. Citado na página 103.

BUENO-SOLER, J. *Multimodalidades anódicas e catódicas: a negação controlada em lógicas multimodais e seu poder expressivo*. Tese (Doutorado) — Instituto de Filosofia e Ciências Humanas da Universidade Estadual de Campinas, Campinas, Julho 2009. Citado na página 404.

BURGIN, M. *Theory of Information*: Fundamentality, diversity and unification. Los Angeles, USA: World Scientific, 2010. v. 1. (Information Studies, v. 1). ISSN 1793-7876. Citado na página 77.

_____. Information: Concept clarification and theoretical representation. *tripleC - Cognition, Communication, Co-operation, Vienna University of Technology*, v. 9, n. 2, p. 347–357, 2011. Disponível em: <http://www.triple-c.at/index.php/tripleC/article/view/284>. Acesso em: 4 jul 2013. Citado na página 77.

CANN, R. Sense relations. In: MAIENBORN, C.; HEUSINGER, K. von; PORTNER, P. (Ed.). *Semantics: An International Handbook of Natural Language Meaning*. Berlin/Boston:

De Gruyter Mouton, 2011, (Handbooks of Linguistics and Communication Science, v. 1). p. 456–478. Citado na página 470.

CARBONERA, J. L. *Raciocínio sobre conhecimento visual*: um estudo em estratigrafia sedimentar. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação, Porto Alegre, RS, Abril 2012. Disponível em: <http://www.lume.ufrgs.br/handle/10183/54864>. Acesso em: 12 out 2013. Citado 3 vezes nas páginas 449, 477 e 485.

CARLSSON, M. *SICStus Prolog User's Manual*: Release 4.2.3. Kista, Sweden, 2012. SICS Swedish ICT AB. Disponível em: <https://sicstus.sics.se/sicstus/docs/4.2.3/pdf/sicstus.pdf>. Acesso em: 2 jul 2013. Citado 2 vezes nas páginas 228 e 229.

CARNIELLI, W.; PIZZI, C. *Modalities and Multimodalities*. With the assistance and collaboration of Juliana Bueno-Soler. USA: Springer, 2008. v. 12. (Logic, Epistemology, and the Unity of Science, v. 12). ISBN 978-1-4020-8589-5. Citado 4 vezes nas páginas 108, 141, 223 e 459.

CAROLO, F.; BURLAMAQUI, L. *Improving web content management with Semantic Technologies*. San Francisco, CA, USA: [s.n.], 2011. Estudo de caso apresentado na 2011 Semantic Technology Conference. Disponível em: <http://www.estacazero.com/publicacoes/websemantica_globo.pdf>. Acesso em: 8 ago 2014. Citado na página 60.

CARVALHO, V. A. et al. Extending the foundations of ontology-based conceptual modeling with a multi-level theory. In: *34th International Conference on Conceptual Modeling (ER 2015)*. [s.n.], 2015. Disponível em: <http://www.inf.ufes.br/~gguizzardi/ER2015-MLT-UFO.pdf>. Acesso em: 12 jul 2015. Citado na página 400.

CARVALHO, V. A. de; ALMEIDA, J. P. A. Towards a well-founded theory for multi-level conceptual modelling. *Unpublished*, 2015. Disponível em: <http://nemo.inf.ufes.br/wp-content/uploads/2015/04/mlt-carvalho-almeida-2015submitted1.pdf>. Acesso em: 12 jul 2015. Citado na página 400.

CASTELNOU, A. M. N. *Arquitetura como arte: tradução livre dos alunos do 4o. período (1980) do Curso de Graduação em Arquitetura e Urbanismo da UFPR do livro intitulado Teoría de la Arquitectura, da autoria de Enrico Tedeschi*. Curitiba: [s.n.], 2011. Universidade Federal do Paraná. Departamento do Livro e Publicações. Disponível em: <http://eusouarquiteto.weebly.com/uploads/3/0/2/0/3020261/arquitetura_como_arte.pdf>. Acesso em: 21 jan 2012. Citado na página 125.

CERBONE, D. R. *Fenomenologia*. Tradução de Caesar Souza. Petrópolis, RJ: Vozes, 2012. (Pensamento Moderno). Citado na página 82.

CHIMIAK-OPOKA, J. et al. Querying UML models using OCL and Prolog: A performance study. In: *Software Testing Verification and Validation Workshop, 2008. ICSTW '08. IEEE International Conference on*. [S.l.: s.n.], 2008. p. 81–88. Citado na página 123.

CHRISTIANSEN, H. Using Prolog as metalanguage for teaching programming language concepts. In: J., M. K. K.; ZADROZNY, S. (Ed.). *Issues in Information Technology*. Warszawa: EXIT, 2002. p. 59–82. Disponível em: <http://akira.ruc.dk/~henning/publications/UnknownPolish2001.pdf>. Acesso em: 2 dez 2013. Citado na página 54.

CLARK, T.; SAMMUT, P.; WILLANS, J. *Applied Metamodelling*: A foundation for language driven development. Second edition. Ceteva, 2008. Disponível em: <http://eprints.mdx.ac.uk/6060/>. Acesso em: 22 out 2014. Citado na página 51.

COAN, M. et al. As categorias verbais tempo, aspecto, modalidade e referência: pressupostos teóricos para uma análise semântico-discursiva. *Revista Estudos Linguísticos*, XXXV, p. 1463–1472, 2006. Disponível em: <http://gel.org.br/estudoslinguisticos/edicoesanteriores/4publica-estudos-2006/sistema06/523.pdf>. Acesso em: 7 jan 2015. Citado na página 449.

COHEN, J.; HICKEY, T. J. Parsing and compiling using Prolog. *ACM Transactions on Programming Languages and Systems*, v. 9, n. 2, April 1987. Citado na página 116.

COLOMB, R. M. *Deductive Databases and Their Applications*. London, United Kingdom: Taylor and Francis Ltd, 1998. Citado 4 vezes nas páginas 64, 95, 204 e 236.

COSTA, A. A. *Direito e Método: diálogos entre a hermenêutica filosófica e a hermenêutica jurídica*. Tese (Doutorado) — Universidade de Brasília, Faculdade de Direito - Curso de Doutorado, Brasília, Março 2008. Disponível em: <http://dominiopublico.mec.gov.br/download/teste/arqs/cp149009.pdf>. Acesso em: 23 out 2014. Citado na página 57.

COSTA, E.; GRINGS, A.; SANTOS, M. V. Documentation methods for visual languages. In: FERRI, F. (Ed.). *Visual Languages for Interactive Computing: Definitions and Formalizations*. [S.l.]: Information Science Reference, 2008. p. 436–454. Citado 2 vezes nas páginas 53 e 384.

COSTA, I. de M. *Um Método para Arquitetura da Informação: Fenomenologia como base para o desenvolvimento de arquiteturas da informação aplicadas*. Dissertação (Dissertação de Mestrado) — Universidade de Brasília, 2010. Citado 2 vezes nas páginas 82 e 83.

COSTA, L. *Arquitetura*. 2a. ed. Rio de Janeiro: José Olympio, 2003. Citado 2 vezes nas páginas 81 e 125.

COSTA-LEITE, A. Book review: Walter Carnielli & Claudio Pizzi, Modalities and Multimodalities, Spring, 2008. *Manuscrito - Rev. Int. Fil.*, v. 36, n. 1, p. 191–195, jan.-jun. 2013. Citado na página 10.

COSTAL, D.; GÓMEZ, C.; GUIZZARDI, G. Formal semantics and ontological analysis for understanding subsetting, specialization and redefinition of associations in UML. In: *Proceedings of the 30th International Conference on Conceptual Modeling*. Berlin, Heidelberg: Springer-Verlag, 2011. (ER'11), p. 189–203. ISBN 978-3-642-24605-0. Disponível em: <http://www.inf.ufes.br/~gguizzardi/paper184.pdf>. Acesso em: 2 jan 2014. Citado 11 vezes nas páginas 147, 148, 149, 192, 193, 194, 455, 485, 486, 492 e 495.

_____. Relatório técnico, *On the meanings of subsetting, specialization and redefinition in UML*. 2011. Universitat Politécnica de Catalunya. Disponível em: <http://upcommons.upc.edu/e-prints/handle/2117/12827>. Acesso em: 2 jan 2014. Citado na página 148.

COVINGTON, M. A. Some coding guidelines for Prolog. Draft document of Artificial Intelligence Center, The University of Georgia. 2002. Disponível em:

<http://www.ai.uga.edu/mc/plcoding2002.pdf>. Acesso em: 9 nov 2013. Citado na página 124.

COVINGTON, M. A. et al. Coding guidelines for Prolog. *Theory Pract. Log. Program.*, Cambridge University Press, New York, NY, USA, v. 12, n. 6, p. 889–927, nov. 2012. ISSN 1471-0684. Disponível em: <http://arxiv.org/pdf/0911.2899v3.pdf>. Acesso em: 23 ago 2013. Citado 4 vezes nas páginas 55, 56, 124 e 125.

CRNKOVIC, G. D.; HOFKIRCHNER, W. Floridi's "open problems in philosophy of information", ten years later. *Information*, v. 2, n. 2, p. 327–359, 2011. Disponível em: <http://www.mdpi.com/2078-2489/2/2/327/>. Acesso em: 2 nov 2011. Citado 2 vezes nas páginas 53 e 77.

CROFT, W.; CRUSE, D. A. *Cognitive Linguistics*. New York: Cambridge University Press, 2004. Citado na página 470.

DECKER, S. On domain-specific declarative knowledge representation and database languages. In: *Proceedings of the 5th KRDB Workshop*. Seattle: [s.n.], 1998. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.633&rep=rep1&type=pdf>. Acesso em: 3 dez 2013. Citado 6 vezes nas páginas 113, 114, 115, 119, 126 e 269.

DEGEN, W. et al. GOL: Towards an Axiomatized Upper-Level Ontology. In: SMITH, B.; GUARINO, N. (Ed.). *Proceedings of FOIS'01*. Ogunquit, Maine, USA: ACM Press, 2001. Citado 4 vezes nas páginas 58, 59, 136 e 498.

_____. GOL: A general ontological language. In: . [s.n.], 2011. Disponível em: <http://ontology.buffalo.edu/smith/articles/gol_fois2001.pdf>. Acesso em: 8 jun 2014. Citado 2 vezes nas páginas 59 e 136.

DÍAZ NAFRÍA, J. M. What is information? a multidimensional concern. *tripleC - Cognition, Communication, Co-operation, Vienna University of Technology*, v. 8, n. 1, p. 77–108, 2010. Disponível em: <http://www.triple-c.at/index.php/tripleC/article/view/76/462>. Acesso em: 4 jul 2013. Citado na página 79.

DÍAZ NAFRÍA, J. M.; ALEMANY, F. S. Towards a transdisciplinary frame: Bridging domains, a multidimensional approach to information. *tripleC - Cognition, Communication, Co-operation, Vienna University of Technology*, v. 9, n. 2, p. 286–294, 2011. Citado 2 vezes nas páginas 78 e 79.

DIETZ, J. L. G. Lecture, *Is it $\phi\tau\psi$ or bullshit?* Netherlands: Delft University of Technology, 2009. Disponível em: <http://repository.tudelft.nl/assets/uuid:358ae2b8-faa5-4034-81e0-4d668e2bd716/20100326084714_001.pdf>. Acesso em: 16 jun 2013. Citado na página 76.

DING, W.; LIN, X. *Information Architecture: The Design and Integration of Information Spaces*. EUA: Morgan & Claypool Publishers, 2010. (Synthesis Lectures on Information Concepts, Retrieval, and Services). Citado 3 vezes nas páginas 53, 76 e 80.

DODEBEI, V. L. D. *Tesauro*: linguagem de representação da memória documentária. Niteroi: Intertexto, 2002. Citado na página 43.

DUARTE, J. C.; LIMA-MARQUES, M. Modeling and simulation competency center for mature enterprises. In: *Proceedings of the International Workshop on Enterprises & Organizational Modeling and Simulation*. New York, NY, USA: ACM, 2009. (EOMAS '09), p. 11:1–11:12. Disponível em: <http://doi.acm.org/10.1145/1750405.1750418>. Acesso em: 17 nov 2011. Citado na página 76.

DUROZOI, G.; ROUSSEL, A. *Dicionário de Filosofia*. 5a. ed. Campinas, SP: Papirus. Tradução de Marina Appenzeller, 1993. Citado na página 128.

ECHEVERRÍA, R. *Ontología del Lenguaje*. Chile: Lom Ediciones S.A, 2005. Disponível em: <http://www.uchile.cl/documentos/ontologia-del-lenguaje-echeverria-pdf_90752_0_5938.pdf>. Acesso em: 4 set 2014. Citado na página 83.

E.T. - O Extraterrestre. Direção: Steven Spielberg. Roteiro: Melissa Mathison. Elenco: Henry Thomans, Drew Barrymore, Peter Coyote, Dee Wallace, e outros. EUA: Universal Pictures do Brasil, 1982. 1 filme (115 min), son., color. Citado na página 372.

ERWIG, M.; WALKINGSHAW, E. Semantics first! - rethinking the language design process. In: *Software Language Engineering - 4th International Conference, SLE 2011*. Braga, Portugal: [s.n.], 2011. p. 243–262. Disponível em: <http://web.engr.oregonstate.edu/~erwig/papers/SemanticsFirst_SLE11.pdf>. Acesso em: 3 dez 2013. Citado na página 227.

_____. Semantics-driven DSL design. In: _____. *Formal and Practical Aspects of Domain-Specific Languages: Recent Developments*. Maribor, Slovenia: University of Maribor, 2012. p. 56–60. Disponível em: <http://web.engr.oregonstate.edu/~erwig/papers/SemanticDSLDesign-12.pdf>. Acesso em: 3 dez 2013. Citado 2 vezes nas páginas 119 e 227.

EUZENAT, J.; SHVAIKO, P. *Ontology Matching*. EUA: Springer Berlin Heidelberg, 2007. ISBN 978-3-540-49611-3. Disponível em: <http://dx.doi.org/10.1007/978-3-540-49612-0_4>. Acesso em: 13 set 2013. Citado na página 133.

EVERNDEN, R.; EVERNDEN, E. *Information First*: Integrating knowledge and information architecture for business advantage. Wheeler Road, Burlington MA: Elsevier Butterworth-Heinemann, 2003. Citado na página 78.

FALBO, R. d. A. et al. Towards an enterprise ontology pattern language. In: *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2014. (SAC '14), p. 323–330. ISBN 978-1-4503-2469-4. Disponível em: <http://doi.acm.org/10.1145/2554850.2554983>. Acesso em: 13 set 2015. Citado na página 61.

FARIÑAS DEL CERRO, L. Molog: A system that extends Prolog with modal logic. *New Generation Computing*, Springer-Verlag, v. 4, n. 1, p. 35–50, 1986. ISSN 0288-3635. Disponível em: <http://dx.doi.org/10.1007/BF03037381>. Acesso em: 4 ago 2013. Citado 2 vezes nas páginas 56 e 104.

FARINELLI, F.; MELO, S.; ALMEIDA, M. B. O papel das ontologias na interoperabilidade de sistemas de informação: reflexões na esfera governamental. In: *XIV Encontro Nacional de Pesquisa em Ciência da Informação - XIV ENANCIB 2013 - GT 8: Informação e Tecnologia*. [s.n.], 2013. Disponível em:

<http://mba.eci.ufmg.br/downloads/Interoperab_Enancib_2013_camera-ready.pdf>. Acesso em: 14 jun 2014. Citado na página 60.

FITTING, M.; MENDELSOHN, R. L. *First-Order Modal Logic*. [S.l.]: Springer Science+Business Media B.V., 1998. (Studies in Epistemology, Logic, Methodology, and Philosophy of Science). Citado na página 57.

FLUSSER, V. *O mundo codificado*: por uma filosofia da comunicação. São Paulo: Cosac Naif, 2007. Citado na página 76.

FOWLER, M. *Analysis Patterns*: Reusable object models. [S.l.]: Addison-Wesley Professional, 1996. Citado na página 158.

FOWLER, M.; PARSONS, R. *Domain Specific Languages*. United States: Addison-Wesley, 2010. Citado na página 117.

FREITAS, A. T. de. *Nova apostila à censura do Senhor Alberto de Moraes Carvalho sobre o Projecto do Código Civil Portuguez*. Rua dos Inválidos, 61B, Rio de Janeiro: Typographia Universal de Laemmert, 1859. Citado na página 57.

_____. *Consolidação das leis civis*. Ed. fac-sim. Brasília: Senado Federal, Conselho Editorial, 2003. (Coleção história do direito brasileiro. Direito civil). Prefácio de Ruy Rosado de Aguiar. Citado na página 57.

FRÜHWIRTH, T. Theory and practice of constraint handling rules. *The Journal of Logic Programming*, v. 37, n. 1–3, p. 95 – 138, 1998. ISSN 0743-1066. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0743106698100055>. Acesso em: 5 ago 2013. Citado 2 vezes nas páginas 55 e 95.

GANGEMI, A. et al. Understanding top-level ontological distinctions. In: *Proc. of IJCAI 2001 workshop on Ontologies and Information Sharing*. [s.n.], 2001. p. 26–33. Disponível em: <http://www.loa.istc.cnr.it/old/Papers/IJCAI2001ws.pdf>. Acesso em: 20 ago 2014. Citado na página 135.

GÄRDENFORS, P. *Conceptual Spaces*: The geometry of thought. Cambridge, Massachusetts: The MIT Press, 2000. Citado 7 vezes nas páginas 143, 444, 445, 471, 480, 481 e 483.

_____. Conceptual spaces as a framework for knowledge representation. *Mind and Matter*, v. 2, n. 2, p. 9–27, 2004. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.126.5207>. Acesso em: 20 set 2013. Citado na página 143.

GERSTL, P.; PRIBBENOW, S. Midwinters, end games, and body parts: a classification of part-whole relations. *Int. J. Hum.-Comput. Stud.*, Academic Press, Inc., Duluth, MN, USA, v. 43, n. 5-6, p. 865–889, dez. 1995. ISSN 1071-5819. Disponível em: <http://www.ontology.buffalo.edu/smith/courses03/tb/PribbenowMidwinters.pdf>. Acesso em: 12 out 2013. Citado 2 vezes nas páginas 456 e 457.

GODINEZ, M. et al. *The Art of Enterprise Information Architecture*: A systems-based approach for unlocking business insight. [S.l.]: IBM Press, 2010. Citado na página 76.

GONÇALVES, B.; GUIZZARDI, G.; FILHO, J. G. P. Using an ECG reference ontology for semantic interoperability of ECG data. *Journal of Biomedical Informatics*, v. 44, n. 1, p. 126 – 136, 2011. ISSN 1532-0464. Ontologies for Clinical and Translational Research. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1532046410001188>. Acesso em: 21 ago 2014. Citado na página 61.

GONZALEZ-PEREZ, C.; HENDERSON-SELLERS, B. *Metamodelling for Software Engineering*. [S.l.]: John Wiley & Sons, Ltd, 2008. Citado na página 158.

GOTTSCHALG-DUQUE, C. *SiRILiCO*: Uma proposta para um sistema de recuperação de informação baseado em teorias da lingüística computacional e ontologia. Tese (Doutorado) — Escola de Ciência da Informação - UFMG, Belo Horizonte, 2005. Citado na página 354.

GRAÇAS, A. P. das. *Suporte Automatizado para Construção de Modelos Conceituais bem fundamentados*. Dissertação (Mestrado em Informática) — Universidade Federal do Espírito Santo. Departamento de Informática., Vitória, Agosto 2010. Disponível em: <http://www.dominiopublico.gov.br/download/texto/cp153427.pdf>. Acesso em: 28 dez 2013. Citado 2 vezes nas páginas 60 e 401.

GRAÇAS, A. P. das; GUIZZARDI, G. Padrões de modelagem e regras de construção de modelos para a criação de ontologias de domínio bem-fundamentadas em OntoUML. In: *Seminário de Pesquisa em Ontologia no Brasil 3º Ontobras*. Florianópolis, SC: [s.n.], 2010. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/ontobras/2010/0030.pdf>. Acesso em: 28 dez 2013. Citado 2 vezes nas páginas 60 e 401.

GRAY, J. et al. Dsls: The good, the bad, and the ugly. In: *Companion to the 23rd ACM SIGPLAN Conference on Object-oriented Programming Systems Languages and Applications*. New York, NY, USA: ACM, 2008. (OOPSLA Companion '08), p. 791–794. ISBN 978-1-60558-220-7. Disponível em: <http://doi.acm.org/10.1145/1449814.1449863>. Acesso em: 3 dez 2013. Citado na página 122.

GRIFFO, C.; ALMEIDA, J. P. A.; GUIZZARDI, G. Towards a Legal Core Ontology based on Alexy's Theory of Fundamental Rights. In: *ICAIL Multi-Lingual Workshop on AI and Law (MWAIL 2015), 15th International Conference on Artificial Intelligence and Law (ICAIL 2015)*. San Diego, CA, USA: [s.n.], 2015. Disponível em: <http://www.inf.ufes.br/~gguizzardi/UFO-L.pdf>. Acesso em: 13 set 2015. Citado na página 61.

GRUBER, T. Ontology. In: LIU, L.; ÖZSU, M. T. (Ed.). *Encyclopedia of Database Systems*. Springer-Verlag, 2009. Disponível em: <http://tomgruber.org/writing/ontology-definition-2007.htm>. Acesso em: 16 mar 2014. Citado na página 129.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing. *International Journal Human-Computer Studies*, v. 43, p. 907–928, August 1993. Disponível em: <http://tomgruber.org/writing/onto-design.pdf>. Acesso em: 16 mar 2014. Citado na página 48.

_____. A translation approach to portable ontology specifications. *Knowledge Acquisition*, v. 5, p. 199–220, 1993. Citado na página 58.

GUARINO, N. The ontological level. In: _____. *Philosophy and the Cognitive Science*. Vienna: Holder-Pivhler-Tempsky, 1995. p. 443–456. Disponível em: <http://wiki.loa-cnr.it/Papers/OntLev.pdf>. Acesso em: 2 jan 2012. Citado na página 195.

_____. Formal ontology and information systems. In: *Proceedings of FOIS'98*. Trento, Italy: IOS Press, 1998. p. 3–15. Amended version. Disponível em: <http://uosis.mif.vu.lt/~donatas/Vadovavimas/Temos/OntologiskaiTeisingasKoncepcinisModeliavimas/papildoma/Guarino98-Formal%20Ontology%20and%20Information%20Systems.pdf>. Acesso em: 30 jul 2014. Citado na página 63.

_____. The ontological level: Revisiting 30 years of knowledge representation. In: BORGIDA, A. et al. (Ed.). *Conceptual Modelling: Foundations and Applications. Essays in Honor of John Mylopoulos*. New York: Springer-Verlag, 2009. p. 52–67. Disponível em: <http://wiki.loa-cnr.it/Papers/2009_Guarino-3.pdf>. Acesso em: 2 jan 2012. Citado na página 195.

GUARINO, N.; GUIZZARDI, G. "We Need to Discuss the Relationship": Revisiting Relationships as Modeling Constructs. In: ZDRAVKOVIC, J.; KIRIKOVA, M.; JOHANNESSON, P. (Ed.). *Advanced Information Systems Engineering*. Springer International Publishing, 2015, (Lecture Notes in Computer Science, v. 9097). p. 279–294. ISBN 978-3-319-19068-6. Disponível em: <http://www.inf.ufes.br/~gguizzardi/CAISE2015-CR.pdf>. Acesso em: 4 jun 2015. Citado 2 vezes nas páginas 60 e 399.

GUARINO, N.; WELTY, C. Evaluating ontological decisions with OntoClean. *Commun. ACM*, ACM, New York, NY, USA, v. 45, n. 2, p. 61–65, fev. 2002. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/503124.503150>. Acesso em: 21 set 2013. Citado 3 vezes nas páginas 59, 136 e 498.

GUARINO, N.; WELTY, C. A. An overview of OntoClean. In: STAAB, S.; STUDER, R. (Ed.). *Handbook on Ontologies*. Second edition. EUA: Springer, 2008. p. 201–220. Citado 2 vezes nas páginas 59 e 136.

GUERSON, J.; ALMEIDA, J.; GUIZZARDI, G. Support for domain constraints in the validation of ontologically well-founded conceptual models. In: BIDER, I. et al. (Ed.). *Enterprise, Business-Process and Information Systems Modeling*. [S.l.]: Springer Berlin Heidelberg, 2014, (Lecture Notes in Business Information Processing, v. 175). p. 302–316. ISBN 978-3-662-43744-5. Citado 2 vezes nas páginas 62 e 164.

GUIZZARDI, G. On the general ontological foundations of conceptual modeling. *Proceedings of 21th International Conference on Conceptual Modeling (ER 2002)*, Springer-Verlag, Berlin, 2002. Disponível em: <http://www.informatik.tu-cottbus.de/~gwagner/papers/ER2002.pdf>. Acesso em: 16 fev 2014. Citado 2 vezes nas páginas 404 e 496.

GUIZZARDI, G. *Ontological Foundations for Structural Conceptual Models*. Tese (Doutorado) — Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands, 2005. Disponível em: <http://www.loa.istc.cnr.it/Guizzardi/SELMAS-CR.pdf>. Acesso em: 3 jul 2011. Citado 106 vezes nas páginas 40, 43, 48, 53, 61, 62, 67, 122, 127, 134, 135, 136, 137, 138, 139, 140, 143, 144, 145, 146, 148, 149, 150, 151, 163, 165, 166, 167, 169, 170, 171, 172, 173, 175, 178, 183, 185, 223, 345, 348, 349, 350,

354, 378, 386, 395, 397, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 487, 488, 489, 490, 491, 492, 493, 495, 496, 497, 498 e 499.

_____. Agent roles, qua individuals and the counting problem. In: GARCIA, A. et al. (Ed.). *Software Engineering for Multi-Agent Systems IV*. EUA: Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 3914). p. 143–160. Citado 19 vezes nas páginas 60, 122, 136, 137, 149, 345, 440, 441, 452, 454, 455, 473, 474, 476, 480, 487, 488, 496 e 497.

_____. Modal aspects of object types and part-whole relations and the de re/de dicto distinction. In: *Proceedings of the 19th international conference on Advanced information systems engineering*. Berlin, Heidelberg: Springer-Verlag, 2007. (CAiSE'07), p. 5–20. ISBN 978-3-540-72987-7. Disponível em: <http://dl.acm.org/citation.cfm?id=1768029.1768033>. Acesso em: 22 set 2013. Citado 13 vezes nas páginas 59, 60, 136, 157, 438, 450, 451, 457, 459, 462, 464, 487 e 489.

_____. On ontology, ontologies, conceptualizations, modeling languages, and (meta)models. In: *Proceedings of the 2007 Conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DB&IS'2006*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2007. p. 18–39. ISBN 978-1-58603-715-4. Disponível em: <http://dl.acm.org/citation.cfm?id=1565421.1565425>. Acesso em: 4 jun 2014. Citado 6 vezes nas páginas 58, 63, 127, 133, 134 e 139.

_____. The problem of transitivity of part-whole relations in conceptual modeling revisited. In: *Proceedings of the 21st International Conference on Advanced Information Systems Engineering*. Berlin, Heidelberg: Springer-Verlag, 2009. (CAiSE '09), p. 94–109. ISBN 978-3-642-02143-5. Disponível em: <http://dx.doi.org/10.1007/978-3-642-02144-2_12>. Acesso em: 20 set 2013. Citado 13 vezes nas páginas 60, 62, 136, 163, 195, 455, 456, 461, 467, 472, 487, 490 e 497.

_____. On the representation of quantities and their parts in conceptual modeling. In: GALTON, A.; MIZOGUCHI, R. (Ed.). *Formal Ontology in Information Systems, Proceedings of the Sixth International Conference, FOIS 2010*. Toronto, Canada: IOS Press, 2010. p. 103–116. Disponível em: <http://www.inf.ufes.br/~gguizzardi/FOIS2010.pdf>. Acesso em: 30 set 2013. Citado 8 vezes nas páginas 60, 136, 151, 163, 189, 465, 466 e 485.

_____. Representing collectives and their members in UML conceptual models: An ontological analysis. In: *Proceedings of the 2010 International Conference on Advances in Conceptual Modeling: Applications and Challenges*. Berlin, Heidelberg: Springer-Verlag, 2010. (ER'10), p. 265–274. ISBN 3-642-16384-X, 978-3-642-16384-5. Disponível em: <http://dl.acm.org/citation.cfm?id=1927973.1928024>. Acesso em: 16 fev 2014. Citado 5 vezes nas páginas 60, 136, 151, 467 e 493.

_____. Ontological foundations for conceptual part-whole relations: the case of collectives and their parts. In: *Proceedings of the 23rd international conference on Advanced information systems engineering*. Berlin, Heidelberg: Springer-Verlag, 2011. (CAiSE'11), p. 138–153. ISBN 978-3-642-21639-8. Disponível em: <http://www.inf.ufes.br/~gguizzardi/CAISE2011-CR.pdf>. Acesso em: 30 set 2013. Citado 20 vezes nas páginas 60, 136, 151, 157, 190, 404, 442, 452, 453, 456, 458, 465, 467, 468, 476, 478, 479, 485, 493 e 494.

_____. Ontology-based evaluation and design of visual conceptual modeling languages. In: REINHARTZ-BERGER, I. et al. (Ed.). *Domain Engineering*. Springer Berlin Heidelberg, 2013. p. 317–347. Disponível em: <http://dx.doi.org/10.1007/978-3-642-36654-3_13>. Acesso em: 2 jan 2014. Citado na página 63.

_____. Ontological patterns, anti-patterns and pattern languages for next-generation conceptual modeling. In: YU, E. et al. (Ed.). *Conceptual Modeling*. Springer International Publishing, 2014, (Lecture Notes in Computer Science, v. 8824). p. 13–27. Invited companion paper to the Keynote Speech. Disponível em: <http://www.inf.ufes.br/~gguizzardi/ER2014-keynote-CR.pdf>. Acesso em: 29 jul 2014. Citado 4 vezes nas páginas 62, 195, 400 e 455.

GUIZZARDI, G. et al. Ontologias de fundamentação, modelagem conceitual e interoperabilidade semântica. In: GUIZZARDI, G.; BAIÃO, F. A.; OLIVEIRA, J. P. M. de (Ed.). *Proceedings of the Iberoamerican Meeting of Ontological Research*. CEUR-WS.org, 2011. Disponível em: <http://ceur-ws.org/Vol-728/paper6.pdf>. Acesso em: 14 jun 2014. Citado 4 vezes nas páginas 61, 129, 136 e 163.

_____. Ontologias de fundamentação e modelagem conceitual. In: II SEMINÁRIO DE PESQUISA EM ONTOLOGIA NO BRASIL, 2009. *Anais eletrônicos...* Rio de Janeiro: IME, 2009. Disponível em: <http://ontobra.comp.ime.eb.br/artigos/Gr17.pdf>. Acesso em: 14 jun 2014. Citado na página 61.

GUIZZARDI, G.; FALBO, R. de A.; GUIZZARDI, R. S. S. Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE software process ontology. In: *Proceedings of the XI Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS)*. Recife: [s.n.], 2008. p. 244–251. Citado 4 vezes nas páginas 60, 61, 136 e 497.

GUIZZARDI, G.; GRAÇAS, A. P. das; GUIZZARDI, R. S. S. Design patterns and inductive modeling rules to support the construction of ontologically well-founded conceptual models in OntoUML. In: SALINESI, C.; PASTOR, O. (Ed.). *Advanced Information Systems Engineering Workshops*. Springer, 2011, (Lecture Notes in Business Information Processing, v. 83). p. 402–413. Disponível em: <http://www.nemo.inf.ufes.br/files/design_patterns_and_inductive_modeling_rules_to_support_the_construction_of_ontologically_well_founded_conceptual_models_in_ontouml_2011.pdf>. Acesso em: 16 fev 2014. Citado 3 vezes nas páginas 62, 195 e 400.

GUIZZARDI, G. et al. On the importance of truly ontological distinctions for ontology representation languages: An industrial case study in the domain of oil and gas. In: HALPIN, T. et al. (Ed.). *Enterprise, Business-Process and Information Systems Modeling*. Springer Berlin Heidelberg, 2009, (Lecture Notes in Business Information Processing, v. 29). p. 224–236. ISBN 978-3-642-01861-9. Disponível em: <http://dx.doi.org/10.1007/978-3-642-01862-6_19>. Acesso em: 29 jul 2014. Citado na página 60.

GUIZZARDI, G.; MASOLO, C.; BORGO, S. In defense of a trope-based ontology for conceptual modeling: An example with the foundations of attributes, weak entities and datatypes. In: *Proceedings of the 25th International Conference on Conceptual Modeling*. Berlin, Heidelberg: Springer-Verlag, 2006. (ER'06), p. 112–125. ISBN 3-540-47224-X, 978-3-540-47224-7. Disponível em: <http://dx.doi.org/10.1007/11901181_10>. Acesso em: 15 fev 2014. Citado 5 vezes nas páginas 58, 60, 136, 163 e 497.

GUIZZARDI, G.; PIRES, L.; SINDEREN, M. Ontology-based evaluation and design of domain-specific visual modeling languages. In: NILSSON, A. et al. (Ed.). *Advances in Information Systems Development*. Springer US, 2006. p. 217–228. ISBN 978-0-387-30834-0. Disponível em: <http://www.loa.istc.cnr.it/Guizzardi/ISD2005.pdf>. Acesso em: 3 dez 2013. Citado na página 122.

GUIZZARDI, G.; WAGNER, G. Some applications of a unified foundational ontology in business modeling. In: GREEN, P.; ROSEMANN, M. (Ed.). *Business Systems Analysis with Ontologies*. Hershey, PA: Idea Group Publishing, 2005. cap. Chapter XIII, p. 345–367. Disponível em: <http://www.loa.istc.cnr.it/Guizzardi/book.pdf>. Acesso em: 21 set 2013. Citado 2 vezes nas páginas 135 e 498.

_____. Towards ontological foundations for agent modeling concepts using the Unified Fundational Ontology (UFO). In: BRESCIANI, P. et al. (Ed.). *Agent-Oriented Information Systems II*. EUA: Springer Berlin Heidelberg, 2005, (Lecture Notes in Computer Science, v. 3508). p. 110–124. ISBN 978-3-540-25911-4. Citado 3 vezes nas páginas 135, 497 e 498.

_____. What's in a relationship: An ontological analysis. In: *Proceedings of the 27th International Conference on Conceptual Modeling*. Berlin, Heidelberg: Springer-Verlag, 2008. (ER '08), p. 83–97. ISBN 978-3-540-87876-6. Disponível em: <http://www.inf.ufes.br/~gguizzardi/ER2008-CR-GuizzardiWagner.pdf>. Acesso em: 2 jan 2014. Citado 15 vezes nas páginas 48, 60, 136, 148, 163, 404, 440, 448, 451, 455, 461, 466, 474, 487 e 497.

_____. Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages. In: POLI, R.; HEALY, M.; KAMEAS, A. (Ed.). *Theory and Applications of Ontology: Computer Applications*. Netherlands: Springer Netherlands, 2010. p. 175–196. ISBN 978-90-481-8846-8. Disponível em: <http://dx.doi.org/10.1007/978-90-481-8847-5_8>. Acesso em: 21 set 2013. Citado 34 vezes nas páginas 136, 137, 138, 139, 143, 221, 438, 440, 442, 445, 448, 452, 454, 455, 461, 463, 466, 471, 472, 473, 475, 476, 477, 479, 480, 481, 483, 484, 486, 487, 489, 492, 498 e 517.

_____. Conceptual simulation modeling with Onto-UML. In: *Proceedings of the Winter Simulation Conference*. Winter Simulation Conference, 2012. (WSC '12), p. 5:1–5:15. Disponível em: <http://dl.acm.org/citation.cfm?id=2429759.2429765>. Acesso em: 24 set 2013. Citado na página 489.

_____. Dispositions and causal laws as the ontological foundation of transition rules in simulation models. In: *Simulation Conference (WSC), 2013 Winter*. [S.l.: s.n.], 2013. p. 1335–1346. Citado 3 vezes nas páginas 447, 449 e 477.

GUIZZARDI, G. et al. Towards ontological foundations for the conceptual modeling of events. In: NG, W.; STOREY, V.; TRUJILLO, J. (Ed.). *Conceptual Modeling*. [S.l.]: Springer Berlin Heidelberg, 2013, (Lecture Notes in Computer Science, v. 8217). p. 327–341. ISBN 978-3-642-41923-2. Citado 2 vezes nas páginas 60 e 136.

_____. An ontologically well-founded profile for UML conceptual models. In: *In Proceedings of the 16 th International Conference on Advanced Information Systems Engineering (CAiSE*. [S.l.]: Springer, 2004. p. 112–126. Citado 4 vezes nas páginas 60, 136, 163 e 164.

GUIZZARDI, G.; WAGNER, G.; VAN SINDEREN, M. J. A formal theory of conceptual modeling universals. In: BÜCHEL, G.; KLEIN, B.; ROTH-BERGHOFER, T. (Ed.). *First International Workshop on Philosophy and Informatics, WSPI 2004*. Cologne, Germany: CEUR-WS.org, 2004. (CEUR Workshop Proceedings, v. 112), p. 10. ISSN 1613-0073. Citado na página 141.

GUIZZARDI, G.; ZAMBORLINI, V. Using a trope-based foundational ontology for bridging different areas of concern in ontology-driven conceptual modeling. *Science of Computer Programming*, v. 96, part 4, p. 417–443, December 2014. ISSN 0167-6423. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167642314000896>. Acesso em: 29 jul 2014. Citado 17 vezes nas páginas 60, 62, 137, 166, 220, 223, 346, 347, 437, 463, 474, 476, 484, 489, 491, 496 e 497.

GUIZZARDI, R. S. S. *Agent-Oriented Constructivist Knowledge Mangament*. Tese (Doutorado) — Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands, 2006. Disponível em: <http://doc.utwente.nl/56967/1/r_guizzardi.pdf>. Acesso em: 11 set 2013. Citado 3 vezes nas páginas 60, 136 e 497.

GUIZZARDI, R. S. S.; GUIZZARDI, G. Applying the UFO ontology to design an agent-oriented engineering language. In: *Proceedings of the 14th East European Conference on Advances in Databases and Information Systems*. Berlin, Heidelberg: Springer-Verlag, 2010. (ADBIS'10), p. 190–203. ISBN 3-642-15575-8, 978-3-642-15575-8. Disponível em: <http://dl.acm.org/citation.cfm?id=1885872.1885890>. Acesso em: 29 jul 2014. Citado 2 vezes nas páginas 60 e 136.

GUPTA, G. Horn logic denotations and their applications. In: *The Logic Programming Paradigm: a 25 year perspective (Proceedings of Workshop on Current trends and Future Directions in Logic Programming Research)*. [s.n.], 1998. p. 127–160. Disponível em: <http://www.cs.nmsu.edu/~gupta/logden/lnai.ps>. Acesso em: 3 dez 2013. Citado 6 vezes nas páginas 54, 114, 115, 125, 261 e 270.

GUPTA, G.; PONTELLI, E. *A Horn Logic Denotational Framework for Specification, Implementation, and Verification of Domain Specific Languages*. New Mexico, 1999. Laboratory for Logic, Databases and Advanced Programming. Disponível em: <https://www.cis.uab.edu/courses/cs593/spring2008/gupta99horn.pdf>. Acesso em: 3 dez 2013. Citado na página 114.

_____. Specification, implementation, and verification of domain specific languages: A logic programming-based approach. In: KAKAS, A. C.; SADRI, F. (Ed.). *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*. London, UK, UK: Springer-Verlag, 2002. (Lecture Notes in Artificial Intelligence), p. 211–239. ISBN 3-540-43959-5. Disponível em: <http://dl.acm.org/citation.cfm?id=646001.675779>. Acesso em: 3 dez 2013. Citado na página 114.

HALPIN, T.; MORGAN, T. *Information Modeling and Relational Databases*. Second edition. Burlington: Morgan Kaufmann, 2008. Citado 2 vezes nas páginas 63 e 205.

HAREL, D.; RUMPE, B. *Modeling Languages*: Syntax, semantics and all that stuff - part I: The basic stuff. [S.l.], 2000. Relatório técnico. Disponível em: <http://www4.in.tum.de/publ/papers/HR00.pdf>. Acesso em: 23 jul 2014. Citado na página 113.

HELLER, B.; HERRE, H. Ontological categories in GOL. *Axiomathes*, v. 14, n. 1, p. 57–76, 2004. Disponível em: <http://www.onto-med.de/publications/2004/heller-b-2004-57-b.pdf>. Acesso em: 16 fev 2014. Citado na página 487.

HENDERSON-SELLERS, B. Bridging metamodels and ontologies in software engineering. *Journal of Systems and Software*, v. 84, n. 2, p. 301–313, 2011. ISSN 0164-1212. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0164121210002906>. Acesso em: 29 jul 2014. Citado na página 60.

_____. *On the Mathematics of Modelling, Metamodelling, Ontologies and Modelling Languages*. Sydney, Australia: Springer, 2012. (Springer Briefs in Computer Science). Citado na página 60.

HENDERSON-SELLERS, B.; GONZALEZ-PEREZ, C. Connecting powertypes and stereotypes. *Journal of Object Technology*, v. 4, n. 7, October 2005. Disponível em: <http://www.jot.fm/issues/issue_2005_09/article3/article3.pdf>. Acesso em: 23 jul 2014. Citado 2 vezes nas páginas 158 e 159.

HERCHENRÖDER, T. *Lightweight Semantic Web Oriented Reasoning in Prolog*: Tableaux inference for description logics. Dissertação (Mestrado) — School of Informatics, University of Edinburgh, 2006. Disponível em: <http://www.inf.ed.ac.uk/publications/thesis/online/IM060364.pdf>. Acesso em: 21 ago 2014. Citado na página 401.

HERRE, H. et al. Onto-Med Report, *General Ontological Language*: A formal framework for building and representing ontologies (version 1.0). 2004. Technical Report, Nr. 7. Reseach Group Ontologies in Medicine, University of Leipzig. Disponível em: <https://www.academia.edu/6524083/General_Ontological_Language_GOL>. Acesso em: 8 jun 2014. Citado na página 480.

HESSEN, J. *Teoria do conhecimento*. São Paulo: Martins Fontes, 1998. Citado na página 82.

HOARE, C. A. R. An axiomatic basis for computer programming. *Commun. ACM*, ACM, New York, NY, USA, v. 12, n. 10, p. 576–580, out. 1969. Disponível em: <http://doi.acm.org/10.1145/363235.363259>. Acesso em: 5 dez 2013. Citado na página 112.

HOFKIRCHNER, W. Towards a Unified Theory of Information: The merging of second-order cybernetics and semiotics into a single and comprehensive information science. *15e Congrès International de Cybernétique*, n. Namur 1998, Namur 1999, p. 175–180, 1998. Disponível em: <http://igw.tuwien.ac.at/igw/Menschen/hofkirchner/papers/InfoScience/Unified_Infotheory/namur.html>. Acesso em: 22 out 2011. Citado 2 vezes nas páginas 77 e 402.

_____. Toward a new science of information. *Information*, v. 2, p. 372–382, 2011. Disponível em: <http://www.mdpi.com/2078-2489/2/2/372>. Acesso em: 29 jun 2013. Citado 3 vezes nas páginas 75, 77 e 402.

HUDAK, P.; WADLER, P. *Report on the Programming Language Haskell*: A non-strict, purely functional language, version 1.0. 1990. Disponível em: <https://www.haskell.org/definition/haskell-report-1.0.tar.gz>. Acesso em: 13 jan 2014. Citado na página 119.

HUSSERL, E. *Logical Investigations, Vols I and II*. New York: Routledge, 2001. Citado na página 82.

IBM. *Dave Ferrucci at Computer History Museum: How it all began and what's next*. 2011. Blog da IBM Research. Disponível em: <http://ibmresearchnews.blogspot.com.br/2011/12/dave-ferrucci-at-computer-history.html>. Acesso em: 22 out 2014. Citado na página 54.

ISO. *Information technology - Syntatic metalanguage - Extended BNF*. [S.l.], 1996. Disponível em: <http://standards.iso.org/ittf/PubliclyAvailableStandards/s026153_ISO_IEC_14977_1996(E).zip>. Acesso em: 5 dez 2013. Citado na página 262.

_____. *ISO 25964:2011 - Information and documentation - Thesauri and interoperability with other vocabularies - Part 1: Thesauri for information retrieval*. First edition. [S.l.], 2011. Citado 2 vezes nas páginas 43 e 404.

ISO. *ISO/IEC 19005-3:2012 PDF/A Part3*. EUA, 2012. Citado 2 vezes nas páginas 70 e 377.

_____. *ISO/IEC 19505-1:2012(E): Information technology - Object Management Group Unified Modeling Language (OMG UML), Infrastructure*: Version 2.4.1, formal/2012-05-06. EUA, 2012. Disponível em: <http://www.omg.org/spec/UML/ISO/19505-1/PDF/>. Acesso em: 26 set 2014. Citado na página 247.

ISO/IEC. *ISO/IEC 13211-1:1995 Information technology – Programming languages – Prolog – Part 1 : General core*. [S.l.], 1995. 199 p. Citado 6 vezes nas páginas 41, 56, 225, 234, 378 e 393.

_____. *ISO/IEC 13211-2:2000 Information technology – Programming languages – Prolog – Part 2 : Modules*. [S.l.], 2000. 23 p. Citado 2 vezes nas páginas 56 e 378.

_____. *ISO/IEC 13211-1:1995/Cor 1: 2007*. [S.l.], 2007. 5 p. Citado 2 vezes nas páginas 56 e 378.

_____. *ISO/IEC 13211-1:1995/Cor 2: 2012*. [S.l.], 2012. 28 p. Citado 2 vezes nas páginas 56 e 378.

_____. *ISO/IEC 19507:2012(E) - Information technology - Object Management Group Object Constraint Language (OCL)*. [S.l.], 2012. Disponível em: <http://www.omg.org/spec/OCL/ISO/19507/PDF/>. Acesso em: 7 jun 2014. Citado 3 vezes nas páginas 62, 121 e 164.

ISRAEL, D.; BRACHMAN, R. Some remarks on the semantics of representation languages. In: BRODIE, M.; MYLOPOULOS, J.; SCHMIDT, J. (Ed.). *On Conceptual Modelling*. [S.l.]: Springer New York, 1984, (Topics in Information Systems). p. 119–146. ISBN 978-1-4612-9732-1. Citado na página 52.

JACKSON, D. *Software Abstractions*: Logic, language, and analysis. Revised edition. Cambridge, Massachusetts, London, England: The MIT Press, 2012. Citado na página 164.

JAPIASSU, H.; MARCONDES, D. *Dicionário básico de filosofia*. 5a. ed. Rio de Janeiro: Jorge Zahar, 2008. Citado na página 89.

JAŚKOWSKI, S. On the rules of suppositions in formal logic. *Studia Logica*, Piotr Pyz i S-ka, Warszawa, Miodown, n. 1, p. 5–32, 1934. Disponível em: <http://www.logik.ch/daten/jaskowski.pdf>. Acesso em: 23 jun 2014. Citado na página 152.

JOHANSSON, I. On the transitivity of the parthood relations. In: HOCHBERG, H.; MULLIGAN, K. (Ed.). *Relations and Predicates*. Germany: Ontos Verlag, 2004, (Philosophische Analyse, v. 11). p. 161–181. Citado na página 465.

JONES, S. P. The haskell 98 language. *Journal of Functional Programming*, v. 13, n. 01, p. 7–255, Jan 2003. Special Issue. Citado na página 119.

JOUAULT, F.; BÉZIVIN, J. KM3: A DSL for Metamodel Specification. In: GORRIERI, R.; WEHRHEIM, H. (Ed.). *Formal Methods for Open Object-Based Distributed Systems*. Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 4037). p. 171–185. ISBN 978-3-540-34893-1. Disponível em: <http://dx.doi.org/10.1007/11768869_14>. Acesso em: 14 jan 2015. Citado na página 121.

JÓZEFOWSKA, A. Ł. J.; ŁUKASZEWSKI, T. The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *Theory and Practice of Logic Programming*, v. 10, n. Special Issue 03, p. 251–289, May 2010. Disponível em: <http://dx.doi.org/10.1017/S1471068410000098>. Acesso em: 24 ago 2013. Citado na página 401.

KERNIGHAN, B. W.; PIKE, R. *The Practice of Programming*. [S.l.]: Addison-Wesley, 2009. (Professional Computing). Citado na página 124.

KERNIGHAN, B. W.; PLAUGER, P. J. *The elements of programming style*. Second edition. New York, NY, USA: McGraw-Hill, 1978. Citado na página 124.

KHAI, Z.; NADEEM, A.; LEE, G. A Prolog based approach to consistency checking of UML class and sequence diagrams. In: KIM, T. et al. (Ed.). *Software Engineering, Business Continuity, and Education: FGIT-ASEA/DRBC/EL*. Germany: Springer, 2011. (Communications in Computer and Information Science, v. 257), p. 85–96. Citado na página 123.

KIFER, M.; LAUSEN, G. F-logic: A higher-order language for reasoning about objects, inheritance, and scheme. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 18, n. 2, p. 134–146, jun. 1989. ISSN 0163-5808. Disponível em: <http://doi.acm.org/10.1145/66926.66939>. Acesso em: 25 out 2014. Citado na página 114.

KLEENE, S. C. *Introduction to Metamathematics*. Amsterdam: North-Holland Publishing Company, 1952. (Bibliotheca Mathematica). Citado na página 437.

KNUTH, D. E. Semantics of context-free languages. *Mathematical systems theory*, Springer-Verlag, v. 2, n. 2, p. 127–145, 1968. ISSN 0025-5661. Disponível em: <http://dx.doi.org/10.1007/BF01692511>. Acesso em: 5 dez 2013. Citado na página 112.

KOONS, R. Defeasible reasoning. In: ZALTA, E. N. (Ed.). *The Stanford Encyclopedia of Philosophy*. Spring 2014. [s.n.], 2014. Disponível em: <http://plato.stanford.edu/archives/spr2014/entries/reasoning-defeasible/>. Acesso em: 23 out 2014. Citado na página 65.

KOSAR, T.; MERNIK, M. Embedded domain-specific languages in Prolog. *Acta Electrotechnica et Informatica*, v. 6, n. 3, 2006. Disponível em: <http://www.aei.tuke.sk/papers/2006/3/Kosar.pdf>. Acesso em: 14 abr 2014. Citado 2 vezes nas páginas 119 e 126.

KOWALSKI, R. Predicate logic as programming language. *Information Processing*, North-Holland Publishing Company, v. 74, p. 569–574, 1974. Disponível em: <http://www-public.it-sudparis.eu/~gibson/Teaching/CSC4504/ReadingMaterial/Kowalski74.pdf>. Acesso em: 6 mar 2014. Citado 6 vezes nas páginas 56, 57, 93, 94, 95 e 101.

_____. *Logic for Problem Solving*. United States of America: North Holland, 1979. (Artificial Intelligence Series 7). Disponível em: <http://www.doc.ic.ac.uk/~rak/papers/LogicForProblemSolving.pdf>. Acesso em: 13 abr 2014. Citado na página 95.

KOWALSKI, R. A. The early years of logic programming. *Communications of the ACM*, v. 31, n. 1, p. 38–43, January 1988. Disponível em: <http://www.doc.ic.ac.uk/~rak/papers/the%20early%20years.pdf>. Acesso em: 22 out 2014. Citado 2 vezes nas páginas 54 e 94.

KRIPKE, S. A. *Naming and Necessity*. Twelfth printing. United States of America: Harvard University Press, 2001. Citado 2 vezes nas páginas 222 e 450.

LACERDA, F. *Arquitetura da Informação: aspectos espistemológicos, científicos e práticos*. Dissertação (Dissertação de Mestrado) — Universidade de Brasília, 2005. Citado 2 vezes nas páginas 67 e 82.

LACERDA, F.; LIMA-MARQUES, M. Information architecture as a discipline - a methodological approach. In: RESMINI, A. (Ed.). *Reframing Information Architecture*. [S.l.]: Springer, 2014, (Human-Computer Interaction Series). Citado 2 vezes nas páginas 67 e 75.

LALLY, A.; FODOR, P. *Natural Language Processing With Prolog in the IBM Watson System*. 2011. Disponível online no sítio "Association for Logic Programming". Disponível em: <http://www.cs.nmsu.edu/ALP/2011/03/natural-language-processing-with-prolog-in-the-ibm-watson-system/>. Acesso em: 22 out 2014. Citado na página 54.

LAMPORT, L. How to write a 21^{st} century proof. *Journal of Fixed Point Theory and Applications*, v. 11, n. 1, p. 43–63, 2012. Disponível em: <http://dx.doi.org/10.1007/s11784-012-0071-6>. Acesso em: 27 set 2014. Citado na página 249.

LANDIN, P. J. The mechanical evaluation of expressions. *The Computer Journal*, v. 6, n. 4, p. 308–320, 1964. Disponível em: <http://comjnl.oxfordjournals.org/content/6/4/308.full.pdf+html>. Acesso em: 12 abr 2015. Citado na página 70.

LEITE, A. F. B. C. *Paraconsistência, Modalidade e Cognoscibilidade*. Dissertação (Mestrado) — Departamento de Filosofia do Instituto de Filosofia e Ciências Humanas da Universidade Estadual de Campinas, Campinas, Abril 2003. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=vtls000298918>. Acesso em: 8 jul 2013. Citado na página 404.

LI, F.-L. et al. Non-functional requirements as qualities, with a spice of ontology. In: *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*. [S.l.: s.n.], 2014. p. 293–302. Citado na página 60.

LIMA-MARQUES, M. *De la connaissance à la paraconsistance: un modèle d'application pour la résolution des conflits aériens*. Tese (Doutorado) — L'Universite Paul Sabatier de Toulouse, Toulouse Cedex – France, Décembre 1992. Citado 4 vezes nas páginas 57, 59, 90 e 104.

LIMA-MARQUES, M. Outline of a theoretical framework of architecture of information: a School of Brasilia proposal. In: BÉZIAU, J.-Y.; CONIGLIO, M. E. (Ed.). *Logic without Frontiers: Festschrift for Walter Alexandre Carnielli on the occasion of his 60th Birthday*. London: College Publications, 2011, (Tribute Series, v. 17). Citado 6 vezes nas páginas 59, 75, 77, 82, 392 e 402.

LLOYD, J. W. *Foundations of logic programming*. Second, extended edition. Germany: Springer-Verlag, 1993. (Symbolic Computation). Citado 6 vezes nas páginas 93, 94, 95, 100, 102 e 125.

LOEBE, F. *An Analysis of Roles*: Towards ontology-based modelling. Dissertação (Master's thesis) — Institute of Medical Informatics, Statistics and Epidemiology (IMISE), University of Leipzig, Germany, August 2003. Disponível em: <http://www.onto-med.de/publications/2003/loebe-f-2003-a.pdf>. Acesso em: 28 set 2013. Citado na página 479.

LOHMANN, D.; EBERT, J. A generalization of the hyperspace approach using meta-models. In: ARAÚJO, J. et al. (Ed.). *Early Aspects 2003: Aspect-Oriented Requirements Engineering and Architecture Design (AOSD 2003)*. [s.n.], 2003. p. 1–6. Disponível em: <http://www4.cs.fau.de/~lohmann/download/LohEbe_ExHyper.pdf>. Acesso em: 25 jul 2014. Citado na página 162.

LOWE, E. J. The particular–universal distinction: A reply to macbride. *Dialectica*, Blackwell Publishing Ltd, v. 58, n. 3, p. 335–340, 2004. ISSN 1746-8361. Disponível em: <http://dx.doi.org/10.1111/j.1746-8361.2004.tb00308.x>. Acesso em: 8 ago 2014. Citado na página 161.

_____. *The Four-Category Ontology*: A metaphysical foundation for natural science. Oxford: Clarendon Press, 2006. Citado na página 137.

LUGER, G. F. *Artificial Intelligence*: Structures and strategies for complex problem solving. Sixth edition. Boston, MA: Addison-Wesley, 2009. Citado na página 64.

LUKÁCSY, G.; SZEREDI, P.; KÁDÁR, B. Prolog based description logic reasoning. In: *Proceedings of the 24th International Conference on Logic Programming*. Berlin, Heidelberg: Springer-Verlag, 2008. (ICLP '08), p. 455–469. ISBN 978-3-540-89981-5. Citado na página 401.

MACBRIDE, F. The particular–universal distinction: A dogma of metaphysics? *Mind*, v. 114, n. 455, p. 565–614, 2005. Disponível em: <http://mind.oxfordjournals.org/content/114/455/565.abstract>. Acesso em: 8 ago 2014. Citado na página 161.

MAFFEI, F. E. El valor ontológico de la poesía. In: *Actas del Primer Congreso Nacional de Filosofía*. Mendoza, Argentina: [s.n.], 1949. tomo 3, p. 1506–1510. Disponível em: <http://www.filosofia.org/aut/003/m49a1506.pdf>. Acesso em: 28 out 2014. Citado na página 39.

MASOLO, C. et al. WonderWeb Deliverable D18 Ontology Library (final) WonderWeb Project. *Communities*, p. 343, 2001. Citado na página 63.

_____. *WonderWeb Deliverable D18*. Trento, Italy, 2003. Disponível em: <http://www.loa.istc.cnr.it/Papers/D18.pdf>. Acesso em: 2 out 2013. Citado 4 vezes nas páginas 52, 460, 480 e 483.

_____. Relational roles and qua-individuals. In: BOELLA, G. et al. (Ed.). *AAAI Fall Symposium on Roles, an Interdisciplinary Perspective*. Menlo Park, California: American Association for Artificial Intelligence (AAAI), 2005. Disponível em: <http://www.irit.fr/publis/LILAC/MGVBF-rolesAAAI05.pdf>. Acesso em: 14 jun 2014. Citado 2 vezes nas páginas 60 e 136.

_____. Social roles and their descriptions. In: DUBOIS, D.; WELTY, C.; WILLIAMS, M.-A. (Ed.). *Principles of Knowledge representation and reasoning: Proceedings of the ninth International Conference (KR2004)*. [s.n.], 2004. p. 267–277. Disponível em: <http://www.aaai.org/Papers/KR/2004/KR04-029.pdf>. Acesso em: 18 set 2013. Citado na página 479.

MCCARTHY, J. Towards a mathematical science of computation. In: *IFIP Congress*. [s.n.], 1962. p. 21–28. Versão disponível para download produzida em 1996. Disponível em: <http://www-formal.stanford.edu/jmc/towards.ps>. Acesso em: 24 out 2014. Citado na página 112.

MEDEIROS, J. da S. *Tesauros conceituais e ontologias de fundamentação*: Análise comparativa entre as bases teórico-metodológicas utilizadas em seus modelos de representação de domínios. Dissertação (Mestrado) — Universidade Federal Fluminense. Instituto de Arte e Comunicação Social. Programa de Pós-graduação em Ciência da Informação, Niterói, 2011. Disponível em: <http://eprints.rclis.org/16094/1/medeiros_js_me_2011.pdf>. Acesso em: 14 jun 2014. Citado na página 60.

MEDEIROS, M. L. C. e Linair Campos e J. A representação de domínios de conhecimento e uma teoria de representação: a ontologia de fundamentação; la representación de dominios de conocimientos y uma teoría de la representación la ontología de fundación. *Informação & Informação*, v. 16, n. 2, 2011. ISSN 1981-8920. Disponível em: <http://www.uel.br/revistas/uel/index.php/informacao/article/view/10389>. Acesso em: 23 out 2014. Citado na página 60.

MELO, A. M. C. de. *Um modelo de Arquitetura da Informação para processos de investigação científica*. Dissertação (Mestrado) — Universidade de Brasília, Brasília, Setembro 2010. Citado na página 66.

MERNIK, M.; HEERING, J.; SLOANE, A. M. When and how to develop domain-specific languages. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 37, n. 4, p. 316–344, dez. 2005. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/1118890.1118892>. Acesso em: 23 ago 2012. Citado na página 118.

MORA, J. F. *Dicionário de Filosofia*: Tomo iii (K-P). Revisão e atualização de Josep-Maria Terricabras. São Paulo, SP: Edições Loyola, 1994. v. 3. (Ariel referencia, v. 3). Citado na página 128.

MORTARI, C. A. *Introdução à Lógica*. São Paulo: Editora UNESP: Imprensa Oficial do Estado, 2001. Citado na página 224.

MORVILLE, P. *Ambient Findability*. Sebastopol, CA: O'Reilly Media, 2005. Citado na página 76.

MORVILLE, P.; ROSENFELD, L. *Information Architecture for the World Wide Web*. Third edition. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, 2006. Citado 3 vezes nas páginas 76, 77 e 78.

MOSS, C. D. How to define a language using PROLOG. In: *Proceedings of the 1982 ACM Symposium on LISP and Functional Programming*. New York, NY, USA: ACM, 1982. (LFP '82), p. 67–73. ISBN 0-89791-082-6. Disponível em: <http://doi.acm.org/10.1145/800068.802136>. Acesso em: 5 dez 2013. Citado 2 vezes nas páginas 112 e 117.

MOTA, E. et al. Veriagent: an approach to integrating UML and formal verification tools. In: *Sixth Brazilian Workshop on Formal Methods (WMF 2003)*. Universidade Federal de Campina Grande, Brazil: [s.n.], 2003. p. 111–129. Disponível em: <http://web.engr.oregonstate.edu/~alex/WMF03.pdf>. Acesso em: 24 ago 2013. Citado na página 123.

MULLIGAN, K.; SMITH, B. A relational theory of the act. *Topoi*, v. 5, n. 2, p. 115–145, 1986. Disponível em: <http://www.unige.ch/lettres/philo/enseignants/km/doc/RelationalThAct.pdf>. Acesso em: 16 fev 2014. Citado na página 487.

MURPHY, M. L. *Semantic Relations and the Lexion*: Antonymy, synonymy, and other paradigms. New York: Cambridge University Press, 2003. Citado na página 470.

MYLOPOULOS, J. Conceptual Modelling and Telos. In: LOUCOPOULOS, P.; ZICARI, R. (Ed.). *Conceptual Modeling, Databases, and Case: An Integrated View of Information Systems Development*. Wiley, 1992. p. 57–68. Disponível em: <http://www.cs.toronto.edu/~jm/2507S/Readings/CM+Telos.pdf>. Acesso em: 29 jul 2014. Citado na página 120.

MYLOPOULOS, J. et al. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, v. 8, n. 4, p. 325–362, October 1990. Disponível em: <http://www.cs.utoronto.ca/~jm/Pub/Telos.pdf>. Acesso em: 12 ago 2014. Citado na página 120.

NAISH, L. *Negation and Control in Prolog*. [S.l.]: Springer-Verlag, 1986. Citado 2 vezes nas páginas 101 e 102.

_____. *Higher-order logic programming in Prolog*. Australia: [s.n.], 1996. University of Melbourne, Department of Computer Science. Techinal Report. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.4505&rep=rep1&type=pdf>. Acesso em: 25 jan 2014. Citado 2 vezes nas páginas 56 e 234.

NARDI, J. C. et al. Towards a commitment-based reference ontology for services. In: *IEEE EDOC (Enterprise Computing Conference)*. Vancouver: [s.n.], 2013. Citado 2 vezes nas páginas 61 e 136.

NGUYEN, L. A. *Results on Modal Reasoning with Applications to Modal Deductive Databases*. Tese (Doutorado) — Institute of Informaticsm Warsaw University, Warsaw, October 1999. Citado 3 vezes nas páginas 103, 122 e 225.

NGUYEN, L. A. The modal query language MDatalog. *Fundamenta Informaticae*, IOS Press, v. 46, p. 315–342, 2001. Citado 2 vezes nas páginas 103 e 122.

_____. A fixpoint semantics and an SLD-Resolution calculus for modal logic programs. *Fundamenta Informaticae*, IOS Press, v. 55, n. 1, p. 63–100, 2003. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.3761&rep=rep1&type=pdf>. Acesso em: 20 dez 2013. Citado 2 vezes nas páginas 103 e 104.

_____. The modal logic programming system MProlog. In: ALFERES, J. J.; LEITE, J. (Ed.). *Logics in Artificial Intelligence. 9th European Conference, JELIA 2004*. Lisbon, Portugal: Springer, 2004. (Lecture Notes in Artificial Intelligence. Subseries of Lecture Notes in Computer Science), p. 266–278. Citado na página 103.

_____. The modal logic programming system MProlog. In: ALFERES, J.; LEITE, J. (Ed.). *JELIA 2004, LNAI 3229*. Berlin Heidelberg: Springer-Verlag, 2004. p. 266–278. Citado 3 vezes nas páginas 103, 107 e 109.

_____. An SLD-Resolution calculus for basic serial multimodal logics. In: HUNG, D.; WIRSING, M. (Ed.). *ICTAC 2005, LNCS 3722*. Berlin Heidelberg: Springer-Verlag, 2005. p. 151–165. Citado na página 103.

_____. Multimodal logic programming. *Theoretical Computer Science*, v. 360, n. 1–3, p. 247 – 288, 2006. ISSN 0304-3975. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0304397506002830>. Acesso em: 11 set 2014. Citado 3 vezes nas páginas 41, 56 e 105.

_____. Foundations of modal deductive databases. *Fundamenta Informaticae*, v. 79, n. 1-2, p. 85–135, November 2007. Revised version: May 2010. Citado 2 vezes nas páginas 103 e 104.

_____. Modal logic programming revisited. *Journal of Applied Non-Classical Logics*, v. 19, n. 2, p. 167–181, 2009. Citado 4 vezes nas páginas 56, 103, 105 e 122.

_____. *Foundations of Modal Logic Programming: The Direct Approach*. Warsaw, Poland, 2010. 147 p. Institute of Informatics, University of Warsaw. Release 2.2. Citado 8 vezes nas páginas 103, 104, 105, 106, 107, 109, 374 e 388.

NIELSON, H. R.; NIELSON, F. *Semantics with Applications*: A formal introduction. 3rd. ed. John Wiley & Sons (First edition), 1999. Disponível em: <http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.html>. Acesso em: 30 nov 2013. Citado na página 112.

_____. *Semantics with Applications*: An appetizer. London: Springer-Verlag, 2007. (Undergraduate topics in Computer Science). Citado 4 vezes nas páginas 111, 112, 269 e 270.

NIETO, P.; COSTAL, D.; GÓMEZ, C. Enhancing the semantics of UML association redefinition. *Data and Knowledge Engineering*, v. 70, n. 2, p. 182–207, 2011. ISSN 0169-023X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169023X10001278>. Acesso em: 2 jan 2014. Citado 2 vezes nas páginas 148 e 149.

NILSSON, N. J. *Problem-Solving methods in Artificial Intelligence*. EUA: MacGraw-Hill Book Company, 1971. (Computer Science Series). Citado na página 56.

NILSSON, U.; MALUSZYNSKI, J. *Logic, Programming and Prolog*. 2. ed. EUA: John Wiley & Sons Ltd., 2000. Disponível em: <http://www.ida.liu.se/~ulfni/lpp/bok/bok.pdf>. Acesso em: 23 ago 2013. Citado na página 93.

OBERLE, D. *Semantic Management of Middleware (Semantic Web and Beyond: Computing for Human Experience)*. Secaucus, NJ, USA: Springer-Verlag, 2006. ISBN 0387276300. Citado 3 vezes nas páginas 131, 132 e 133.

OBJECT MANAGEMENT GROUP. *UML 2.0 Superstructure Specification*: Final adopted specification, ptc/03-08-02. EUA, 2003. 624 p. Disponível em: <http://www.omg.org/cgi-bin/doc?ptc/03-08-02.pdf>. Acesso em: 14 set 2013. Citado 5 vezes nas páginas 140, 163, 441, 445 e 475.

_____. *Human-Usable Textual Notation (HUTN) Specification*: Version 1.0. August, 2004. Disponível em: <http://doc.omg.org/formal/2004-08-01.pdf>. Acesso em: 10 ago 2014. Citado na página 120.

_____. *OMG Unified Modeling Language^{TM} (OMG UML), Superstructure*: Version 2.2, formal/2009-02-02. EUA, 2009. 724 p. Disponível em: <http://www.omg.org/spec/UML/2.2/Superstructure/PDF/>. Acesso em: 22 set 2013. Citado na página 141.

_____. *Business Process Model and Notation (BPMN)*: Version 2.0, formal/2011-01-03. EUA, 2011. 624 p. Disponível em: <http://www.omg.org/spec/BPMN/2.0>. Acesso em: 24 out 2014. Citado na página 64.

_____. *OMG Unified Modeling Language^{TM} (OMG UML), Superstructure*: Version 2.4.1, formal/2011-08-06. EUA, 2011. 624 p. Disponível em: <http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/>. Acesso em: 14 set 2013. Citado 11 vezes nas páginas 61, 64, 140, 141, 142, 158, 159, 160, 441, 445 e 475.

_____. *OMG Meta Object Facility (MOF) Core Specification*: Version 2.4.2, formal/2014-04-03. EUA, 2014. 69 p. Disponível em: <http://www.omg.org/spec/MOF/2.4.2/PDF/>. Acesso em: 14 jan 2015. Citado na página 121.

_____. *XML Meta Interchange (XMI) Specification*: Version 2.4.2, formal-14-04-04. EUA, 2014. 100 p. Disponível em: <http://www.omg.org/spec/XMI/2.4.2>. Acesso em: 31 out 2014. Citado 2 vezes nas páginas 385 e 402.

ODELL, J. Power types. *Journal of Object-Oriented Programming*, v. 7, n. 2, p. 8–12, 1994. Citado na página 158.

OGDEN, C. K.; RICHARDS, I. A. *The meaning of meaning*: A stydy of the influence of language upon thought and of the science of symbolism. New York: Harcourt, 1923. Disponível em: <http://s-f-walker.org.uk/pubsebooks/pdfs/ogden-richards-meaning-all.pdf>. Acesso em: 20 set 2014. Citado na página 209.

OLIVÉ, A. *Conceptual Modeling of Information Systems*. [S.l.]: Springer-Verlag, 2007. Citado na página 148.

ORGUN, M. A. On temporal deductive databases. *Computational Intelligence*, Blackwell Publishing Ltd, v. 12, n. 2, p. 235–259, 1996. ISSN 1467-8640. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.1428>. Acesso em: 25 out 2014. Citado na página 114.

ORLANDI, E. P. *Análise de Discurso: Princípios e Procedimentos*. 8. ed. [S.l.]: Pontes, 2009. Citado na página 87.

OSSHER, H.; TARR, P. Multi-dimensional separation of concerns and the hyperspace approach. In: AKŞIT, M. (Ed.). *Software Architectures and Component Technology*. Springer US, 2002, (The Springer International Series in Engineering and Computer Science, v. 648). p. 293–323. ISBN 978-1-4613-5286-0. Disponível em: <http://dx.doi.org/10.1007/978-1-4615-0883-0_10>. Acesso em: 25 jul 2014. Citado 3 vezes nas páginas 161, 162 e 214.

PAIS, M. S. *P-Datalog*$^{\neg}$: uma linguagem dedutiva para consultas a banco de dados com inconsistências. Dissertação (Mestrado) — Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, MG, Agosto 2004. Disponível em: <http://www.ppgcc.ufu.br/posgrad2/posgrad/defesas/dissertacoes/Monica%20Sakuray.pdf>. Acesso em: 2 nov 2013. Citado 2 vezes nas páginas 122 e 404.

PAP, I. M. Z.; PATARICZA, A.; SZEGI, A. Completeness and consistency analysis of UML statechart specifications. In: *IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop (DDECS'2001)*. Győr, Hungary: [s.n.], 2001. p. 83–90. Disponível em: <http://home.mit.bme.hu/~majzik/publicat/ddecs2001.pdf>. Acesso em: 24 ago 2013. Citado na página 123.

PASSOS, R.; LIMA-MARQUES, M.; MEALHA, Ó. Uma delimitação do conceito de informação para o design da informação. In: *Anais do 5o Congresso Internacional de Design da Informação*. Florianópolis, SC: [s.n.], 2011. Citado na página 76.

PEPINO, E. M. L. S. F.; GAVIORNO, G. V. S. de C.; FILGUEIRAS, S. V. A importância da jurisprudência dos conceitos para a metodologia jurídica. *Revista Depoimentos*, v. 4, n. 7, p. 137–148, Jul./Dez. 2003. Disponível em: <http://www.fdv.br/publicacoes/periodicos/revistadepoimentos/n7/6.pdf>. Acesso em: 23 out 2014. Citado na página 57.

PEREIRA, F. C. N.; WARREN, D. H. D. Definite clause grammars for language analysis: A survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, v. 13, p. 231–278, 1980. Disponível em: <http://cgi.di.uoa.gr/~takis/pereira-warren.pdf>. Acesso em: 22 out 2014. Citado na página 55.

PHILIPPOW, I.; RIEBISCH, M.; BOELLERT, K. The Hyper/UML approach for feature based software design. In: *The 4th AOSD Modeling With UML Workshop*. [s.n.], 2003. Disponível em: <http://www.cs.iit.edu/~oaldawud/AOM/AOM2003/Mathias-PhilRieBoel.pdf>. Acesso em: 25 jul 2014. Citado na página 162.

PONTELLI, E. et al. Phylog: A domain specific language for solving phylogenetic problems in biology. In: *First IEEE Computer Society Bioinformatics conference*. [s.n.],

2002. Disponível em: <http://www.cs.utdallas.edu/~gupta/phylog.ps>. Acesso em: 3 dez 2013. Citado na página 114.

PROBST, F. *Semantic Reference Systems for Observations and Measurements*. Tese (Doutorado) — Fachbereich Geowissenschaften der Westfälischen Wilhelms - Universität Münster, 2007. Disponível em: <http://ifgi.uni-muenster.de/~probsfl/publications/PROBST-Thesis-SemanticReferenceSystemsForObservationsAndMeasurements.pdf>. Acesso em: 7 jun 2014. Citado 4 vezes nas páginas 143, 399, 444 e 483.

PROBST, F. Observations, measurements and semantic reference spaces. *Appl. Ontol.*, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 3, n. 1-2, p. 63–89, jan. 2008. ISSN 1570-5838. Disponível em: <http://dl.acm.org/citation.cfm?id=1412417.1412420>. Acesso em: 7 jun 2014. Citado 2 vezes nas páginas 143 e 480.

QUINTON, A. Properties and classes. *Proceedings of the Aristotelian Society*, Wiley on behalf of The Aristotelian Society, v. 58, p. pp. 33–58, 1957. ISSN 00667374. Disponível em: <http://www.jstor.org/stable/4544588>. Acesso em: 26 nov 2013. Citado na página 498.

QUIRINO, G. et al. Towards a service ontology pattern language. In: *35th International Conference on Conceptual Modeling (ER 2015)*. Stockholm: [s.n.], 2015. Disponível em: <http://www.inf.ufes.br/~gguizzardi/S-OPL.pdf>. Acesso em: 13 set 2015. Citado na página 61.

REIMER, U.; HAHN, U. A formal approach to the semantics of a frame data model. In: *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 1*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1983. (IJCAI'83), p. 337–339. Disponível em: <http://dl.acm.org/citation.cfm?id=1623373.1623450>. Acesso em: 30 nov 2013. Citado na página 111.

REITER, R. Deductive question-answering on relational data bases. In: GALLAIRE, H.; MINKER, J. (Ed.). *Logic and Data Bases*. [S.l.]: Springer US, 1978. p. 149–177. ISBN 978-1-4684-3386-9. Citado na página 102.

RESMINI, A. *Information Architecture Modeling for Historical and Juridical Manuscript Collections*. Tese (Dottorato di ricerca in Informatica Giuridica e Diritto Dell'Informatica) — Alma Mater Studiorum Università di Bologna, 2010. Disponível em: <http://amsdottorato.cib.unibo.it/2941/1/andrea_resmini_tesi.pdf>. Acesso em: 6 jun 2013. Citado 2 vezes nas páginas 77 e 78.

RESMINI, A.; ROSATI, L. Information architecture for ubiquitous ecologies. In: *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*. New York, NY, USA: ACM, 2009. (MEDES '09), p. 29:196–29:199. ISBN 978-1-60558-829-2. Disponível em: <http://doi.acm.org/10.1145/1643823.1643859>. Acesso em: 19 nov 2011. Citado 2 vezes nas páginas 61 e 76.

_____. The semantic environment: Heuristics for a cross-context human–information interaction model. In: DUBOIS, E.; GRAY, P.; NIGAY, L. (Ed.). *The Engineering of Mixed Reality Systems*. [S.l.]: Springer London, 2010, (Human-Computer Interaction Series). p. 79–99. ISBN 978-1-84882-732-5. Citado na página 77.

_____. *Pervasive Information Architecture*. USA: Morgan Kaufmann, 2011. Citado na página 76.

RIOS FILHO, P. A. da C. *Um Modelo de Arquitetura da Informação Orientado a Serviços*. Dissertação (Mestrado) — Universidade de Brasília. Faculdade de Ciência da Informação, Julho 2014. Citado na página 61.

ROBINSON, J. A. A machine-oriented logic based on the resolution principle. *J. ACM*, ACM, New York, NY, USA, v. 12, n. 1, p. 23–41, jan. 1965. ISSN 0004-5411. Disponível em: <http://doi.acm.org/10.1145/321250.321253>. Acesso em: 23 ago 2014. Citado 2 vezes nas páginas 94 e 101.

RODRIGUES, T. G. *Sobre os Fundamentos da Programação Lógica Paraconsistente*. Dissertação (Mestrado) — Departamento de Filosofia do Instituto de Filosofia e Ciências Humanas da Universidade Estadual de Campinas, Campinas, SP, Setembro 2010. Disponível em: <http://www.cle.unicamp.br/prof/coniglio/dissertacao_Tarcisio.pdf>. Acesso em: 30 out 2013. Citado 3 vezes nas páginas 56, 93 e 404.

ROSENAN, B. Designing language-oriented programming languages. In: *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion*. New York, NY, USA: ACM, 2010. (OOPSLA '10), p. 207–208. ISBN 978-1-4503-0240-1. Disponível em: <http://doi.acm.org/10.1145/1869542.1869576>. Acesso em: 22 out 2014. Citado na página 51.

RUPP, N.; BURKE, D. From catalogers to ontologists. *The Serials Librarian*, v. 46, n. 3-4, p. 221–226, 2004. Disponível em: <http://dx.doi.org/10.1300/J123v46n03_04>. Acesso em: 22 out 2014. Citado na página 57.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence*: A modern approach. Third edition. [S.l.]: Prentice Hall, 2010. (Prentice Hall Series in Artificial Intelligence). Citado 2 vezes nas páginas 70 e 102.

_____. *Inteligência Artificial*: Tradução da terceira edição. Tradução de Regina Célia Simille. 3. ed. Rio de Janeiro: Elsevier e Editora Campus, 2013. (Prentice Hall Series in Artificial Intelligence). Citado na página 70.

SADILEK, D. A. Prototyping domain-specific language semantics. In: *OOPSLA Doctoral Symposium*. [s.n.], 2008. Disponível em: <http://www2.informatik.hu-berlin.de/~sadilek/publications/2008/OOPSLA08docsym.pdf>. Acesso em: 2 dez 2013. Citado na página 112.

SADILEK, D. A.; WACHSMUTH, G. Using grammarware languages to define operational semantics of modelled languages. In: *TOOLS '09: 47th International Conference Objects, Models, Components, Patterns*. Springer, 2009. Disponível em: <http://www2.informatik.hu-berlin.de/~sadilek/publications/2009/TOOLS09grammarwareOpSem.pdf>. Acesso em: 2 dez 2013. Citado na página 112.

SAINT-DIZIER, P. *An Introduction to Programming in Prolog*. [S.l.]: Springer-Verlag, 1990. Translated by Sharon J. Hamilton. This work is a translation of the French volume Introduction to *Programming in Prolog* by Patrick Saint-Dizier, published by Eyerolles - Paris. Citado 2 vezes nas páginas 93 e 95.

SALES, T. P. *Ontology Validation for Managers*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo. Departamento de Informática., Vitória, Outubro 2014.

Disponível em: <http://www.researchgate.net/publication/268220197_Ontology_Validation_for_Managers>. Acesso em: 12 abr 2015. Citado 2 vezes nas páginas 385 e 401.

SALES, T. P.; BARCELOS, P. P. F.; GUIZZARDI, G. Identification of semantic anti-patterns in ontology-driven conceptual modeling via visual simulation. In: *Proceedings of the Workshops of the 7th International Conference on Formal Ontology in Information Systems (FOIS)*. [s.n.], 2012. Disponível em: <http://kr-med.org/icbofois2012/proceedings/ICBOFOIS2012Workshops/FOIS2012ODISE/FOIS-2012-ODISE-Sales-Barcelos-Guizzardi.pdf>. Acesso em: 29 jul 2014. Citado 3 vezes nas páginas 62, 195 e 400.

SALES, T. P.; GUIZZARDI, G. Ontological anti-patterns: empirically uncovered error-prone structures in ontology-driven conceptual models. *Data and Knowledge Engineering*, v. 99, p. 72–104, 2015. ISSN 0169-023X. Selected Papers from the 33rd International Conference on Conceptual Modeling (ER 2014). Citado na página 400.

SCHERP, A. et al. Designing core ontologies. *Fachbereich Informatik*, v. 5, February 2011. Disponível em: <http://www.uni-koblenz.de/~fb4reports/2011/2011_05_Arbeitsberichte.pdf>. Acesso em: 11 set 2013. Citado na página 133.

SCHMIDT, D. A. *Denotational Semantics*: A methodology for language development. [S.l.]: Allyn and Banco, Inc, 1986. Citado na página 112.

SCOTT, D.; STRACHEY, C. Toward a mathematical semantics for computer languages. In: *Proceedings of the Symposium on Computers and Automata*. [S.l.: s.n.], 1971. p. 19–46. Citado na página 112.

SEBESTA, R. W. *Concepts of Programming Languages*. Tenth edition. [S.l.]: Pearson, 2012. Citado 4 vezes nas páginas 94, 95, 101 e 113.

SILVA, H. C. e et al. Well-founded it architecture ontology: An approach from a service continuity perspective. In: BENLAMRI, R. (Ed.). *Networked Digital Technologies*. [S.l.]: Springer Berlin Heidelberg, 2012, (Communications in Computer and Information Science, v. 294). p. 136–150. ISBN 978-3-642-30566-5. Citado na página 61.

SIMÕES, M. G. *Da Abstracção à Complexidade Formal*: relações conceituais num tesauro. Coimbra: Almedina, 2008. Citado na página 43.

SIMONS, P. Stanisław Leśniewski. In: ZALTA, E. N. (Ed.). *The Stanford Encyclopedia of Philosophy*. [s.n.], 2011. Disponível em: <http://plato.stanford.edu/entries/lesniewski/>. Acesso em: 15 jun 2014. Citado na página 469.

SIMONS, P. M. *Parts. A Study in Ontology*. Oxford: Clarendon Press, 1987. (Logical and philosophical study of mereology). Citado 2 vezes nas páginas 450 e 494.

SIQUEIRA, A. H. de. *A lógica e a linguagem como fundamentos da Arquitetura da Informação*. Dissertação (Dissertação de Mestrado) — Universidade de Brasília, Brasília, 2008. Citado 2 vezes nas páginas 54 e 62.

SIQUEIRA, A. H. de. *Arquitetura da Informação: uma proposta para fundamentação e caracterização da disciplina científica*. Tese (Doutorado) — Universidade de Brasília. Faculdade de Ciência da Informação, Brasília, 2012. Citado 5 vezes nas páginas 62, 76, 77, 82 e 84.

SIQUEIRA, A. H. de. Sobre os fundamentos filosóficos da arquitetura da informação. *RICI: R. Ibero-amerciana Ci. Inf.*, Brasília, v. 5, n. 1, p. 1–22, jan./jul.2012 2012. Citado 3 vezes nas páginas 76, 77 e 84.

SLONNEGER, K.; KURTZ, B. L. *Formal Syntax and Semantics of Programming Languages*: A laboratory based approach. Addison-Wesley Publishing Company, 1995. Disponível em: <http://homepage.divms.uiowa.edu/~slonnegr/plf/Book>. Acesso em: 10 dez 2013. Citado na página 116.

SMITH, B. Mereotopology: A theory of parts and boundaries. *Data and Knowledge Engineering*, v. 20, p. 287–303, 1996. Disponível em: <http://ontology.buffalo.edu/smith/articles/Mereotopology1.pdf>. Acesso em: 24 set 2013. Citado na página 450.

_____. Ontology. In: FLORIDI, L. (Ed.). *Blackwell Guide to the Philosophy of Computing and Information,*. Oxford: Blackwell, 2003. p. 155–166. Disponível em: <http://ontology.buffalo.edu/smith/articles/ontology_pic.pdf>. Acesso em: 24 set 2013. Citado na página 450.

SMULLYAN, R. M. Logicians who reason about themselves. In: *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1986. (TARK '86), p. 341–352. ISBN 0-934613-04-9. Disponível em: <http://dl.acm.org/citation.cfm?id=1029786.1029818>. Acesso em: 23 jan 2015. Citado na página 108.

SNEYERS, J. et al. As time goes by: Constraint Handling Rules: A survey of CHR research from 1998 to 2007. *Theory and Practice of Logic Programming*, v. 10, p. 1–47, 1 2010. ISSN 1475-3081. Disponível em: <http://journals.cambridge.org/article_S1471068409990123>. Acesso em: 6 ago 2013. Citado 2 vezes nas páginas 55 e 95.

SOARES, H. de F. *Uma contribuição da Fenomenologia para a Arquitetura da Informação*. Dissertação (Monografia) — Universidade de Brasília. Departamento de Ciência da Informação e Documentação, Brasília, Dezembro 2004. Disponível em: <http://rabci.org/rabci/sites/default/files/fenomenologia_0.pdf>. Acesso em: 4 ago 2013. Citado na página 59.

STERLING, L.; SHAPIRO, E. *The Art of Prolog*: Advanced programming techniques. With a foreword by David H. D. Warren. Second edition. Cambridge, Massachusetts, London, England: The MIT Press, 1999. Third printing. Citado na página 236.

STÖRRLE, H. A formal approach to the cross-language version management of models. In: KUZNIARZ, L. et al. (orgs.). *Nordic Workshop on Model Driven Engineering*. Ronneby, Sweden: [s.n.], 2007. p. 83–97. ISBN 978-91-7295-985-9. Citado na página 123.

_____. A prolog-based approach to representing and querying software engineering models. In: COX, P.; FISH, A.; HOWSE, J. (Ed.). *Proceedings of the VLL 2007 workshop on Visual Languages and Logic*. [s.n.], 2007. p. 71–83. Disponível em: <http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-274/paper6.pdf>. Acesso em: 22 ago 2013. Citado 3 vezes nas páginas 122, 123 e 124.

SUZUKI, J.; YAMAMOTO, Y. Making UML models interoperable with UXF. In: BÉZIVIN, J.; MULLER, P.-A. (Ed.). *The Unified Modeling Language. «UML»'98: Beyond the Notation*. Springer Berlin Heidelberg, 1999, (Lecture Notes

in Computer Science, v. 1618). p. 78–91. ISBN 978-3-540-66252-5. Disponível em: <http://www.cs.umb.edu/~jxs/pub/lncs_uml.ps>. Acesso em: 31 out 2014. Citado 2 vezes nas páginas 385 e 402.

SWIFT, T.; WARREN, D. S. XSB: Extending Prolog with Tabled Logic Programming. *Theory and Practice of Logic Programming*, v. 12, p. 157–187, jan 2012. Disponível em: <http://arxiv.org/abs/1012.5123>. Acesso em: 24 ago 2013. Citado na página 56.

SYRIANI, E. *A Multi-Paradigm Foundation for Model Transformation Language Engineering*. Tese (Doutorado) — School of Computer Science, McGill University, Montreal, Quebec, Canada, February 2011. Disponível em: <http://syriani.cs.ua.edu/publications/dissertation.pdf>. Acesso em: 23 jul 2014. Citado na página 113.

TARR, P.; OSSHER, H. *Hyper/J*^{TM}: User and installation manual. [S.l.], 2000. IBM Research. Disponível em: <http://www.cse.msu.edu/SENS/Software/hyperj/doc/hyperj-user-manual.pdf>. Acesso em: 25 jul 2014. Citado na página 162.

TURBAK, F.; GIFFORD, D. *Design Concepts in Programming Languages*. London, England: The MIT Press, 2008. Citado na página 112.

van BENTHEM, J. *Modal Logic for Open Minds*. Stanford, CA: CSLI Publications, 2010. Citado na página 106.

van EMDEN, M. H.; KOWALSKI, R. A. The semantics of predicate logic as a programming language. *Journal of the Association for Computing Machinery*, v. 23, n. 4, p. 733–742, October 1976. Disponível em: <http://athena.nitc.ac.in/~kmurali/Courses/LogicAug2009/Papers/Kowalski.pdf>. Acesso em: 13 abr 2014. Citado 2 vezes nas páginas 95 e 117.

van GIGCH, J. P.; PIPINO, L. L. In search for a paradigm for the discipline of information systems. *Future Computing Systems*, v. 1, n. 1, p. 71–97, 1986. Citado 2 vezes nas páginas 67 e 68.

VIEU, L.; AURNAGUE, M. Part-of relations, functionality and dependence. In: AURNAGUE, M.; HICKMANN, M.; VIEU, L. (Ed.). *The Categorization of Spatial Entities in Language and Cognition*. Amsterdam: John Benjamins Publishing Company, 2007. v. 20, p. 307–336. Disponível em: <http://clic.cimec.unitn.it/massimo/Teach/AI/Readings/vieu_arnague_07.pdf>. Acesso em: 24 jun 2014. Citado na página 493.

W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. [S.l.], 2008. Disponível em: <http://www.w3.org/TR/REC-xml/#sec-notation>. Acesso em: 5 dez 2013. Citado na página 262.

_____. *OWL 2 Web Ontology Language*: Document overview. W3C recomendation 11 december 2012. Second edition. [S.l.], 2012. Disponível em: <http://www.w3.org/2012/pdf/REC-owl2-overview-20121211.pdf>. Acesso em: 14 jan 2015. Citado na página 121.

WALLACE, M. et al. On benchmarking constraint logic programming platforms. response to fernandez and hill's "a comparative study of eight constraint programming languages over the boolean and finite domains". *Constraints*, v. 9, p. 5–34, 2004. Citado 2 vezes nas páginas 55 e 95.

WALTER, T. Combining domain-specific languages and ontology technologies. In: *Proceedings of the Doctoral Symposium at MODELS 2009*. [s.n.], 2009. Technical Report of School of Computing, Queen's University. Disponível em: <http://walter.semanticsoftware.eu/publications/Walter2009CDL.pdf>. Acesso em: 21 ago 2014. Citado na página 121.

WALTER, T.; PARREIRAS, F. S.; STAAB, S. en. Ontodsl: An ontology-based framework for domain-specific languages. In: SCHÜRR, A.; SELIC, B. (Ed.). *Model Driven Engineering Languages and Systems*. Springer Berlin Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5795). p. 408–422. Disponível em: <http://walter.semanticsoftware.eu/publications/WalterOAO2009.pdf>. Acesso em: 3 dez 2013. Citado 3 vezes nas páginas 118, 121 e 122.

WANG, Q.; GUPTA, G. Rapidly prototyping implementation infrastructure of domain specific languages: A semantics-based approach. In: *ACM Symposium on Applied Computing*. New Mexico: [s.n.], 2005. Disponível em: <http://www.di.unipi.it/ricerca/proceedings/AppliedComputing05/PDFs/papers/T25P07.pdf>. Acesso em: 3 dez 2013. Citado na página 117.

WANG, Q.; GUPTA, G.; LEUSCHEL, M. Towards provably correct code generation via horn logical continuation semantics. In: HERMENEGILDO, M.; CABEZA, D. (Ed.). *Practical Aspects of Declarative Languages*. [S.l.]: Springer Berlin Heidelberg, 2005, (Lecture Notes in Computer Science, v. 3350). p. 98–112. Citado na página 117.

WARD, M. P. Language-oriented programming. *Software - Concepts and Tools*, v. 15, n. 4, p. 147–161, 1994. Disponível em: <http://www.cse.dmu.ac.uk/~mward/martin/papers/middle-out-t.pdf>. Acesso em: 22 out 2014. Citado na página 51.

WARREN, D. H. D. Logic programming and compiler writing. *Software: Practice and Experience*, John Wiley & Sons, Ltd., v. 10, n. 2, p. 97–125, 1980. ISSN 1097-024X. Disponível em: <http://dx.doi.org/10.1002/spe.4380100203>. Acesso em: 24 out 2014. Citado 2 vezes nas páginas 93 e 116.

WATT, D. *Programming Language Design Concepts*. England: John Wiley & Sons Ltd., 2004. Citado na página 112.

WELTY, C.; GUARINO, N. Supporting Ontological Analysis of Taxonomic Relationships. *Data and Knowledge Engineering*, v. 39, p. 51–74, 2001. Citado 3 vezes nas páginas 59, 136 e 498.

WHITE, M. Information architecture. *Electronic Library, The*, v. 22, n. 3, p. 218–219, 2004. Disponível em: <http://dx.doi.org/10.1108/02640470410541606>. Acesso em: 7 jun 2013. Citado na página 77.

WIERINGA, R.; JONGE, W. de; SPRUIT, P. Using dynamic classes and role classes to model object migration. *Theory and Practice of Object Systems*, John Wiley & Sons, New York, NY, USA, v. 1, n. 1, p. 61–83, 1995. Disponível em: <http://doc.utwente.nl/67609/>. Acesso em: 16 out 2013. Citado na página 490.

WILLIAMS, D. C. On the elements of being: I. *The Review of Metaphysics*, Philosophy Education Society Inc., v. 7, n. 1, p. pp. 3–18, 1953. ISSN 00346632. Disponível em: <http://www.jstor.org/stable/20123348>. Acesso em: 27 nov 2013. Citado na página 497.

WILLIS, A.-M. Ontological designing. *Paper presented at Design Cultures, conference of the European Academy of Design*, May 1999. Citado 2 vezes nas páginas 83 e 91.

WINSTON, M. E.; CHAFFIN, R.; HERRMAN, D. A taxonomy of part-whole relations. *Cognitive Science*, n. 11, p. 417–444, 1987. Disponível em: <http://csjarchive.cogsci.rpi.edu/1987v11/i04/p0417p0444/MAIN.PDF>. Acesso em: 17 jun 2014. Citado na página 470.

WINTERS, E. *Aesthetics & Architecture*. London: Continuum International Publishing Group, 2007. Citado na página 125.

WIRTH, N. *The Programming Language Pascal (Revised Report)*. Zürich. [S.l.], 1973. Disponível em: <http://e-collection.library.ethz.ch/eserv/eth:3059/eth-3059-01.pdf>. Acesso em: 5 dez 2013. Citado na página 262.

_____. What can we do about the unnecessary diversity of notation for syntactic definitions? *Commun. ACM*, ACM, New York, NY, USA, v. 20, n. 11, p. 822–823, nov. 1977. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/359863.359883>. Acesso em: 5 dez 2013. Citado na página 262.

WODTKE, C.; GOVELLA, A. *Information Architecture*: Blueprints for the web. Second edition. Berkeley, CA: New Riders, 2009. Citado na página 76.

WURMAN, R. S. *Information Architects*. Lakewood: Watson-Guptill Pubns, 1997. v. 240 p. Citado na página 77.

WURMAN, R. S.; KATZ, J. Beyond graphics: The architecture of information. *AIA Journal*, p. 40;56, October 1975. Citado na página 77.

ZAMBORLINI, V. C. *Estudo de Alternativas de Mapeamento de Ontologias da Linguagem OntoUML para OWL*: Abordagens para representação da informação temporal. Dissertação (Mestrado) — Universidade do Espírito Santo. Departamento de Informática, Vitória, 2011. Citado 4 vezes nas páginas 127, 128, 138 e 139.

ZARETSKA, I. et al. Checking inconsistencies in UML design. In: ERMOLAYEV, V. et al. (Ed.). *Proceedings of the 8th International Conference on ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer*. Kherson, Ukraine: [s.n.], 2012. v. 848, p. 33–43. Disponível em: <http://ceur-ws.org/Vol-848/>. Acesso em: 24 ago 2013. Citado na página 123.

ZHANG; YUEXIAO. Definitions and sciences of information. *Information Processing and Management*, v. 24, n. 4, p. 479 – 491, 1988. ISSN 0306-4573. Disponível em: <http://www.sciencedirect.com/science/article/pii/0306457388900507>. Acesso em: 8 out 2011. Citado na página 53.

ZIMMERMAN, M. *The Watson Research Team Answers Your Questions*. 2011. Disponível online no sítio "Building a Smarter Planet". Disponível em: <http://asmarterplanet.com/blog/2011/02/the-watson-research-team-answers-your-questions.html>. Acesso em: 22 out 2014. Citado na página 54.

Glossário da *Unified Foundational Ontology*

Este glossário sumariza os termos e conceitos do fragmento da UFO abordados nesta tese. As definições são baseadas em publicações do autor da ontologia de fundamentação, realizadas de forma individual ou em colaboração.

Os termos são apresentados em inglês, língua original em que aparecem nas obras consultadas. Como muitos deles são estrangeiros, opta-se por não escrevê-los em itálico. Sugestões nossas de traduções para português dos referidos termos são apresentadas entre parênteses no título da entrada. Porém, as traduções são livres e não são baseadas em análise filológica rigorosa. De toda forma, sempre que considerado necessário, notas de tradução são incluídas na parte final do corpo da definição.

As formulações lógicas são baseadas em uma lógica modal S5 quantificada, com identidade e semânticas de mundos de Kripke definida em Guizzardi (2005, p. 121-134). Expressões como *"in (the) modal sense"*, *"necessarily"* e *"as a matter of necessity"* – e respectivas traduções vernáculas – referem-se à interpretação de modalidades aléticas. As fórmulas são adaptadas para seguir a Lista de símbolos deste trabalho. Conforme Guizzardi e Zamborlini (2014, p. 5), o quantificador $\exists!$ (quantificação de unicidade) deve ser lido como "existe apenas um". Esse quantificador é definido por Kleene (1952, p. 199) como (na notação original): $\exists! x A(x)$ é uma abreviação para $\exists! x[A(x) \& Ay(A(y) \supset x = y)]$ *"(read 'there exists a unique x such that A(x)')"*.

O glossário consiste em 186 entradas.

A | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **K** | **M** | **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **W**

A

Anti Rigid (Anti-rígido)

Ver: Anti Rigidity

Anti Rigid Mixin (*Mixin* anti-rígido)

Mixins anti-rígidos representam abstrações de propriedades comuns de dois ou mais Sortais anti-rígidos (Anti Rigid Sortal) que especializam ao menos duas Espécies (Kind) distintos. Conforme Almeida e Guizzardi (2007, p.209), um *Mixin* anti-rígido:

> *captures commonalities in various role universals. This universal is used in a conceptual modelling design pattern for "roles with multiple disjoint allowed types [...]. Intuitively, a role mixin universal allows us to add flexibility to a role universal, without tying its definition to a specific sortal universal.*

Na OntoUML recebem o estereótipo de Role Mixin (GUIZZARDI, 2005, p. 112).

Anti Rigid Sortal (Sortal anti-rígido)

Os Sortais anti-rígidos tratam-se de Sortais (Sortal) com a característica de Anti-rigidez (Anti Rigidity). Subdividem-se em Fase (Phase) e Papel (Role), conforme o critério de aplicação.

Ver também: Phased Sortal, Role

Anti Rigid Universal (Universal anti-rígido)

Ver: Anti Rigidity

Anti Rigidity (Anti-rigidez)

Conforme Guizzardi (2007a), Guizzardi e Wagner (2010), o conceito de Anti-rigidez AR é definido como:

> *A types* [sic.] T *is anti-rigid if for every instance* x *of* T*,* x *is possibly (in the modal sense) not an instance of* T*. In other words, if* x *instantiates* T *in a given world* w*, then there is a possible world* w' *in which* x *does not instantiate* T*:* $AR(T) \coloneqq \Box(\forall x T(x) \rightarrow \Diamond(\neg T(x)))$

Ver também: Rigidity, Kind

Association Relation (Relação de Associação)

Uma Relação de Associação, abreviada como *assoc*, é uma Relação Formal (Formal Relation) entre uma Estrutura de Qualidade (Quality Structure) e um Momento Intrínseco Universal (Intrinsic Moment Universal). Nessa relação, o papel do Momento Intrínseco Universal é exercido por um Universal de Qualidade (Quality Universal) que é relacionado a uma única Estrutura de Qualidade, e vice e versa. Conforme presente em Guizzardi (2005, p. 224), sendo $QS(x)$ uma Estrutura de Qualidade e $\exists!$ interpretado como "existe apenas um":

$$qualityUniversal(U) \coloneqq intrinsicMomentUniversal(U) \wedge \exists! x QS(x) \wedge assoc(x, U)$$

$$QS(x) \coloneqq \exists! U \ qualityUniversal(U) \wedge assoc(x, U)$$

Em Albuquerque e Guizzardi (2013), porém, Guizzardi revisa suas definições sobre os Momentos Intrínsecos (Intrinsic Moment) e a Relação de Associação é chamada de *"Structuration Relation"*. Além disso, os autores apresentam que a relação liga *ao menos uma Universal de Qualidade (Quality Universal) a ao menos um Quality Structure*. A possibilidade de uma *Quality Universal* estar associado a vários *Quality Structures* é necessária para que diferentes conceituações a respeito de um mesmo conceito universal de qualidade seja estruturado de diferentes formas lógicas.

> *For example, one could conceptualize the color quality structure under the HSB theory. However, one could alternatively consider the RGB theory or a monochromatic scale. In any case, all these theories offer different ways to approximate the same qualia".* (ALBUQUERQUE; GUIZZARDI, 2013)

Ver também: Attribute Function

Attribute Function (Função de Atribuição)

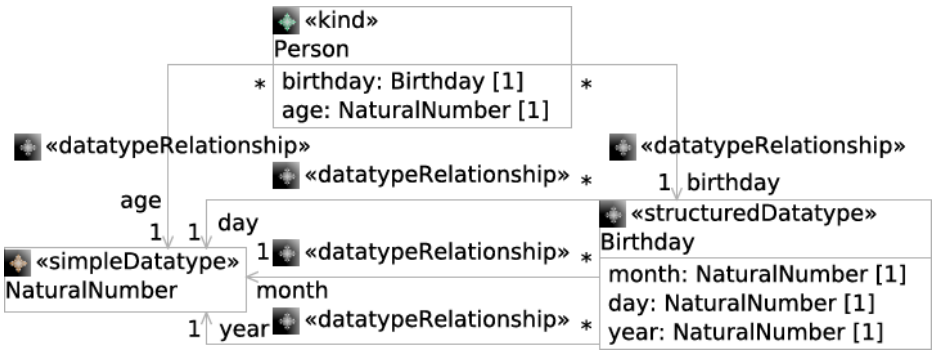
As Funções de Atribuição mapeiam instâncias de Universais de Qualidade (Quality Universal) em instâncias de Estruturas de Qualidade (Quality Structure), ou mais especificamente, mapeiam instâncias de Substanciais (Substantial) em seus respectivos Quales inerentes a uma Qualidade (Quality). Conforme Guizzardi (2005, p. 325), *Attribute Functions* são a interpretação ontológica das propriedades dos elementos da UML:

> *attribute functions are therefore the ontological interpretation of UML attributes, i.e., UML Properties which are owned by a given classifier. That is, an attribute of a class C representing a universal U is interpreted as an attribute function derived from the elementary specification of the universal U.*

Attribute Functions são representadas por DataType Relationships. A Figura 51 apresenta um exemplo que ilustra a relação entre `Pessoa`, sua data de nascimento e sua idade:

> *an Attribute Function age(Year), which maps each instance of Person (and in particular x) onto a point in the Quality Dimension ageValue (in OntoUML, we represented the Attribute Function age(Year) as a DataType Relationship with "age" as its navigable end name, from the Kind Person to the Simple DataType NaturalNumber)* (BENEVIDES, 2010, p. 41-42).

Figura 51 – Exemplo de Estruturas de Qualidade



Fonte: Benevides (2010, p. 42)

Ver também: Quale of a quality, Classifier, DataType, OntoUML, Association Relation

B

Basic Formal Relation (Relação Formal Básica)

Em complemento às Relações Formais Comparativas (Comparative Formal Relation), as Relações Formais Básicas (Basic Formal Relation) compõem a super-estrutura matemática do arcabouço da UFO. Tratam-se de Relações Formais (Formal Relation) que, conforme Guizzardi (2005, p. 236) e Guizzardi e Wagner (2010):

> *includes those relations that form the mathematical superstructure of our framework including existential dependence (ed), inherence (i), part-of (<), subset-of, instantiation, characterization, exemplification, among many others [...]*

Bearer (Portador)

Ver: Bearer of a Moment

Bearer of a Moment (Portador de um momento)

O indivíduo único que porta ou detém um Momento (Moment), o qual este depende existencialmente daquele. (GUIZZARDI, 2005, p. 214).

Ver também: Inherence, Existential Dependence, Bearer of a Trope

Bearer of a Trope (Portador de um *trope*)

Termo usado em Guizzardi e Wagner (2008) para se referir a Bearer of a Moment.

Ver também: Trope, Bearer of a Moment

C

Category (Categoria)

Ver: Rigid Mixin

Characterization (Caracterização)

A relação de Caracterização é uma Relação Formal (Formal Relation) entre um Momento Universal (Moment Universal) e um Objeto Universal (Object Universal) que ocorre se toda instância do Objeto Universal em questão *exemplificar* entidades do Momento Universal (GUIZZARDI; WAGNER, 2010). Em geral, especificações conceituais – como as apresentadas pelas classes em UML e diagramas de Entidade-Relationamento – representam conceitos no nível de tipos. Dessa forma, a ideia de *Characterization* – e também de Mediação (Mediation) – são contrapartes do nível de tipos para a relação *"inheres in"* (Inherence) – e *"mediates"* (Mediation), respectivamente, no nível dos particulares (GUIZZARDI, 2006; GUIZZARDI; WAGNER, 2008). Conforme Guizzardi (2005, p. 249):

> *the* characterization *relation between a quality universal and a substantial universal is mapped in the instance level to an inherence relation between the corresponding quality and substantial individuals*

Em Guizzardi (2005, p. 221) encontra-se que:

$$characterization(U, M) \coloneqq Universal(U) \wedge MomentUniversal(M) \wedge \forall x(x :: U \rightarrow \exists y\ y :: M \wedge i(y, x)) \tag{11.1}$$

Nota: Em Guizzardi (2005, p. 324), uma figura (Figure 8-9) apresenta que a relação *Characterization* é estabelecida entre um Universal e um Momento Intrínseco Universal (Intrinsic Moment Universal). Porém, conforme apresentado nesta entrada, a própria obra citada apresenta que a relação em tela é estabelecida entre um Universal e um Momento Universal (Moment Universal). De fato, a representação da figura indicada não é inconsistente, uma vez que se um tipo superior possui uma relação, o tipo inferior também a possui.

Ver também: Exemplification, Mediation, Quality Universal, Bearer of a Moment

Characterizing Universal (Universal caracterizador)

Tratam-se de Universais (Universal) que contribuem apenas com atributos que caracterizam indivíduos que já são instanciados como Sortais (Sortal). Conforme Guizzardi (2006):

> *Substantial Universals such as Person, Car, Dog, Student that carry a principle of identity, individuation and counting for its instances are named* Sortal Universals*. In contrast, universals such as Thing, Red, Tall, Heavy are named* Characterizing Universals*, since they only attribute properties to (characterize) individuals which have already being individuated by sortal-supplied principles. The distinction between sortal and characterizing universals is reflected in natural language in the distinction between common nouns and other general terms (e.g., adjectives, verbs), respectively*

Ver também: Moment Universal, Quality Universal, Sortal

Classifier

Entidade do metamodelo da UML. *Classifier* são Tipos (Type) que possuem propriedades e podem ser especializados em *Classes*, *Associações*, *Interfaces* e DataTypes (OBJECT MANAGEMENT GROUP, 2003, p. 61,) (OBJECT MANAGEMENT GROUP, 2011b, p. 51-56).

Collective (Coletivo)

Coletivos representam Sortais de Substância (Substance Sortal) cujas instâncias são coleções de entidades conectadas por uma relação de unificação, ou seja, que

possuem uma estrutura uniforme (GUIZZARDI, 2005, p. 181). Exemplo: matilha, manada, revoada, floresta, condomínio, os *Beatles*.

> *Finally, we emphasize that, differently from quantities, collectives do not necessarily obey an extensional principle of identity. Some collectives can be considered extensional by certain conceptualizations. In this case, the addition or subtraction of a member renders a different collective* (GUIZZARDI, 2005, p. 183).

Conforme Guizzardi (2011):

> *we are interested in aggregations of individuals that have a purpose for some cognitive task. So, we require all collectives in our system to form closure systems unified under a proper characterizing relation. For example, a group of visitors of interest can be composed by all those people that are attending a certain museum exhibition at a certain time.*

Ainda conforme a obra citada, *Collectives* não são Homeomeros e tratam-se de entidades maximais.

Nota: A entrada Functional Complex apresenta comparação dos Complexos (Complex) com as Quantidades (Quantity) e com Coletivos (Collective).

Ver também: Functional Complex, Quantity, Kind

Comparative Formal Relation (Relação Formal Comparativa)

Em complemento às Relações Formais Básicas (Basic Formal Relation), as Relações Formais Comparativas são um tipo de Relação Formal (Formal Relation) que, como o nome sugere, referem-se à relações de *comparação* entre as entidades, mediante a comparação de suas propriedades intrínsecas, o que implica que são relacionamentos entre Momentos Intrínsecos (Intrinsic Moment), e não entre Objetos (Object) (GUIZZARDI; WAGNER, 2010):

> *For instance, the relation heavier-than between two atoms is a formal relation that holds directly as soon as the relata (atoms) are given. The truth-value of a predicate representing this relation depends solely on the atomic number (a quality) of each atom and the material content of heavier-than is as it were distributed between the two relata. Moreover, to quote Mulligan and Smith, "once the distribution has been effected, the two relata are seen to fall apart, in such a way that they no longer have anything specifically to do with each other but can serve equally as terms in a potentially infinite number of comparisons"* (GUIZZARDI; WAGNER, 2010).

Ver também: Domain Formal Relation

Complex (Complexo)

Ver: Functional Complex

Complex Quality (Qualidade Complexa)

Em Guizzardi (2005, p. 232), o conceito de Qualidade Complexa é definido apenas como o complementar do conceito Qualidade Simples (Simple Quality):

$$complexQuality(x) \coloneqq quality(x) \wedge \neg simpleQuality(x)$$

Segundo o autor, a Qualidade (Quality) c da Figura 56 é um exemplo de *Complex Quality*. Na UFO não há Dimensões de Qualidade (Quality Dimension) multidimensionais, por isso, *Complex Qualities* só podem carregar Qualidades Simples:

$$\forall x\ complexQuality(x) \rightarrow (\forall y\ i(y,x) \rightarrow simpleQuality(y))$$

Ver também: Indirect Quality, Simple Quality

Complex Quality Universal (Qualidade Complexa Universal)

Trata-se do Universal instanciado por uma Qualidade Complexa (Complex Quality). Qualidades Complexas Universais são sempre associados a Domínios de Qualidades (Quality Domain) (GUIZZARDI, 2005, p. 233).

Ver também: Quality Universal

Component of (Componente de)

Estereótipo usado na OntoUML para representar a Relação componente-complexo funcional (Component-Functional Complex Relation).

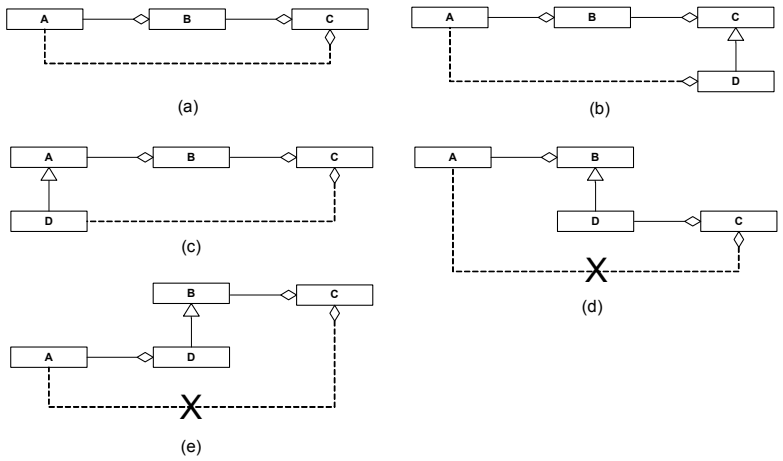
Ver também: Mereology, Subquantity of, Subcollection of, Member of

Component-Functional Complex Relation (Relação componente-complexo funcional)

Conforme Guizzardi (2005, p. 187-189, 342-344), a relação *component of*, ou *Component-Functional Complex Relation*, trata-se da mais comum, e por isso, da mais importante Relação Meronímica (Meronymic relation) que especializa a relação Parte de (Part of). Ela é uma relação estabelecida entre dois Complexos (Complex) que, diferente dos Coletivos (Collective), exercem diversos papeis no contexto do Whole. A relação possui as meta propriedades de *irreflexividade*, *anti-simetria*, *não-transitividade* e o axioma Suplementação Fraca (Weak Supplementation). A *não-transitividade* é devido ao fato de que por padrão a relação não ser *transitiva*, exceto nos casos em que a relação é marcada com o atributo `essential` (Essential Part) e nos casos apresentados na Figura 52, em que (a), (b) e (c) são casos que a transitividade é o caso e (d) e (e), os casos em que não. Pelo falo de tratar-se da relação mais

comum a ser representada em modelos conceituais, para representá-la utiliza-se o símbolo padrão de agregação da UML nos diagramas OntoUML. As partes da relação *component of* podem ser compartilháveis (Shareable) ou não compartilháveis (Exclusive part, Non-Shareable), imutáveis (Immutable Part), inseparáveis (Inseparable Part) e mandatórias (Mandatory part). Em diagramas OntoUML, assume-se que as partes mandatórias representadas pela cardinalidade mínima de 1 do lado da parte representa uma Dependência Genérica (Generic Dependence) do Whole para a parte (Mandatory part); a cardinalidade mínima de 1 do lado do Whole representa uma Dependência Genérica da parte para o todo (Mandatory Whole). Ainda sobre diagramas OntoUML, sempre que uma relação possuir a propriedade `inseparable`, a cardinalidade mínima da associação ligada ao Whole deve ser 1. De forma similar, sempre que uma relação possuir as propriedades `essential` ou `immutable`, a cardinalidade mínima da associação conectada à parte deve igualmente ser 1 (GUIZZARDI, 2005, p. 344).

Figura 52 – Casos de inferência de transitividade de relações



Fonte: Guizzardi (2005, p. 308)

Ver também: Functional Complex, Component of, Part of, Subcollection-Collection Relation, Mass-Quantity Relation, Member-Collection Relation

Conceptual Space (Espaço Conceitual)

Um Espaço Conceitual é um conceito apresentado por Gärdenfors (2000, p. 26) e definido na UFO como uma coleção de um ou mais Domínios de Qualidades (Quality Domain). A noção de Espaço Conceitual pode ser entendida literalmente, ou seja, as composições de *Quality Domains* são dotadas de certas estruturas geométricas (topológicas ou estruturas de ordem) que restringem as relações entre as dimensões constituintes dos domínios. De acordo com Probst (2007, p. 47), Espaço Conceitual

"is a geometrical or topological structure based on a set of quality dimensions." Para Gärdenfors (2000), os "quality dimension" podem ser integrais ou separáveis um dos outros. Na UFO, esses conceitos de dimensões "integrais" e "separáveis" são interpretados respectivamente como Dimensões de Qualidade (Quality Dimension) e Domínios de Qualidades (Quality Domain). A ideia de Espaço Conceitual é concepção usada para definir o conceito de Quality Structure na UFO (GUIZZARDI, 2005, p. 223), (GUIZZARDI; WAGNER, 2010).

Ver também: Quality Space

D

DataType

Entidade da OntoUML que representa um valor (OBJECT MANAGEMENT GROUP, 2003, p. 95-96), (OBJECT MANAGEMENT GROUP, 2011b, p. 59-61). Trata-se de uma especialização de Classifier. Os *DataTypes* são usados em OntoUML para representar Estruturas de Qualidade (Quality Structure).

DataType Relationship (Relacionamento *DataType*)

Entidade da OntoUML que representa uma Função de Atribuição (Attribute Function). Trata-se de uma relação entre algum Classifier, que possui um atributo, e um DataType. O nome da relação é o nome do atributo. Conforme Benevides (2010, p. 41), embora os *DataType Relationship* não estivessem presentes na proposta original da OntoUML de Guizzardi (2005), *"they are present at later revisions of the language as specializations of the metaclass Directed Binary Relationship."*

Derivation Relation (Relação de Derivação)

O relacionamento de derivação é uma Relação Formal (Formal Relation) entre um Universal de *Relator* (Relator Universal) e uma Relação Material (Material Relation), que expressa o fato de que a existência de uma Relação Material entre dois objetos pode ser inferida a partir da existência de um Relator que os liga através de Relações de Mediação (Mediation Relation) (BENEVIDES, 2010, p. 44) e (GUIZZARDI, 2005, p. 241). Considerando a Instantiation Relation ::, Guizzardi e Wagner (2010) apresenta a seguinte definição para *Derivation Relation*:

> *We define the formal relation of derivation* $derivation(R, U_R)$ *holding between a relator universal* U_R *and a material relation* R *such that a n-tuple* $\langle x_1 \dots x_n \rangle$ *instantiates* R *iff there is a relator* $r :: UR$ *such that* r *mediates every* x_i.

De acordo com Guizzardi (2005, p. 330), *derivation* é uma relação funcional total e também um tipo de Dependência Existencial (Existential Dependence). Por essa razão, "cada n-tupla x instância de uma relação material F é derivada de exatamente um Relator r" da qual é existencialmente dependente.

Ver também: Relation, Material Relation, Formal Relation, Inherence, Existential Dependence, Instantiation Relation

Derived (Derivado)

Derived é usado em modelos OntoUML como uma metapropriedade de Relações (Relation) para indicar que a relação é derivada, ou inferida, a partir de outros elementos do modelo. Veja um exemplo na descrição de Relação de Derivação (Derivation Relation). (GUIZZARDI, 2005, p. 331, 338).

Disjointness (Disjunção)

Dois universais são considerados *disjuntos* se não for possível existir uma instância comum a ambos. No contexto de Mereologias (Mereology), dois indivíduos são *disjuntos* se eles não se sobrepõem (Overlapping). Em outras palavras, x e y são disjuntos se, e somente se, eles não possuem parte em comum. Dessa forma, a disjunção, simbolizada por $\int$, é definida em Guizzardi (2005, p. 145) como:

$$(x \int y) \coloneqq \neg(x \bullet y)$$

em que $\bullet$ é definido em Overlapping.
A relação de disjunção é simétrica, assim como Overlapping:

$$\forall x, y \ (x \int y) \rightarrow (y \int x)$$

Ver também: Proper part, Weak Supplementation, Minimal Mereology, Ground Mereology

Dispersive Sortal (Sortal Dispersivo)

Ver: Dispersive Universal

Dispersive Universal (Universal Dispersivo)

Universais dispersivos são aqueles que abrangem conceitos com diferentes Princípios de identidade (Principle of identity). Guizzardi (2005, p. 105) apresenta o seguinte exemplo:

> *For instance, in the extension of Thing we might encounter an individual x that is a cow and an individual y that is a watch. Since the principles of identity for Cows and Watches are not the same, we conclude that Thing cannot supply a principle of identity for its instances. Otherwise, x and y would obey incompatible principles of identity and, thus, would not be determinate individuals.*

Dispersive Universal é usado como sinônimo de Mixin Universal.

Ver também: Mixin

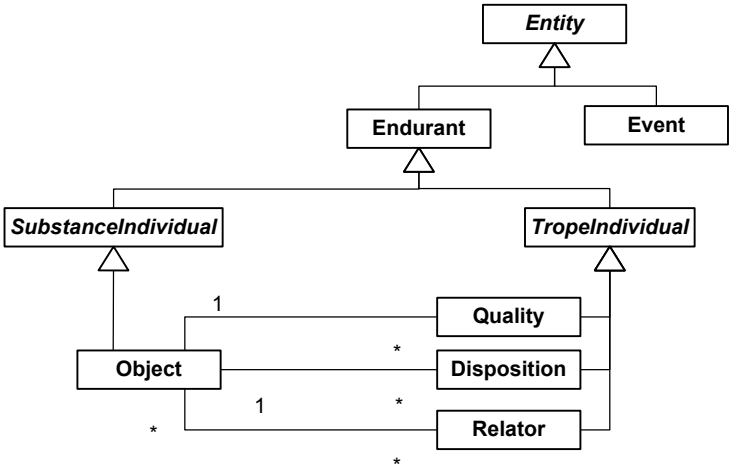
Disposition (Disposição)

Disposições na UFO-B são propriedades que se manifestam apenas em situações particulares, a partir da ocorrência de determinados eventos, e que também podem falhar em se manifestar. Quando manifestadas, as disposições são decorrência de eventos e de mudança de estados. Conforme Guizzardi e Wagner (2013, p. 1340):

> *As pointed out by Choi and Fara (2012), many terms have been used in philosophy to describe what is meant by dispositions: 'power', 'ability', 'potency', 'capability', 'tendency', 'potentiality' and 'capacity', among others. [. . .] Take for example the disposition of a magnet* m *to attract metallic objects. The magnet* m *has this disposition even if it is never manifested, for example, because* m *is never close to any magnetic object. Nonetheless, m can certainly be said to possess this intrinsic property, which it shares with other magnets. Now, consider a particular metallic object o that has the disposition of being attracted by magnets. Given a situation in which o is sufficiently close to* m *(on a surface with a sufficiently low friction, etc.), the disposition of these two objects can be manifested through the occurrence of a complex event, namely, the movement of o towards* m.

As Disposições são catalogadas como um tipo de Trope na UFO-A, conforme ilustrados na Figura 53.

Figura 53 – Fragmento da UFO-A que descreve Disposition



Fonte: Guizzardi e Wagner (2013, p. 1337)

Domain Abstraction (Abstração de Domínio)

Domain Abstraction são instâncias de Conceituações de domínio (Domain Conceptualization) (GUIZZARDI, 2005, p. 35-36), uma vez que são representações abstratas

de aspectos de entidades que existem em um domínio. Em suma, são abstrações de um certo *"state of affairs"*, expressos em termos de um conjunto de Conceitos de Domínio (Domain Concept) (GUIZZARDI, 2005, p. 26). No contexto de modelagem conceitual de domínios, um *Domain Abstraction* é sinônimo de "modelo" ou "modelo conceitual", que é uma diagramação das relações entre conceitos. Guizzardi e Wagner (2010) apresenta o seguinte exemplo:

> *Take as an example the domain of genealogical relations in reality. A certain conceptualization of this domain can be constructed by considering concepts such as* Person, Man, Woman, Father, Mother, Offspring, being the father of, being the mother of, *among others. By using these concepts, we can articulate a domain abstraction (i.e., a mental model) of certain facts in reality such as, for instance, that a man named John is the father of another man named Paul.*

Ver também: Domain Concept, Domain Ontology

Domain Concept (Conceito de Domínio)

É um conceito que existe no domínio a partir do qual está se construindo modelos conceituais por meio de Abstrações de Domínios (Domain Abstraction). Os Conceitos de Domínio fazem parte da Conceituação de domínio (Domain Conceptualization).

Ver também: Domain Ontology

Domain Conceptualization (Conceituação de domínio)

Tratam-se de um conjunto de Conceitos de Domínio (Domain Concept) (GUIZZARDI, 2005, p. 26). São expressos por meio de Ontologias de Domínio (Domain Ontology) (GUIZZARDI, 2005, p. 17).

Ver também: Domain Abstraction

Domain Formal Relation (Relação Formal de Domínio)

Expressão usada por Guizzardi e Wagner (2008) para se referir às Relações Formais Comparativas (Comparative Formal Relation).

Domain Ontology (Ontologia de Domínio)

No contexto da UFO, tratam-se de especificações conceituais compartilhadas de Conceituações de domínio (Domain Conceptualization) (GUIZZARDI, 2005, p. 17).

Ver também: Reference Ontology

E

Elementary specification (Especificação elementar)

Conforme Guizzardi (2005, p. 220), uma Especificação elementar de um universal U consiste em um conjunto de outros universais $U_1, \ldots, U_n$ e respectivas relações formais $R_1, \ldots, R_n$ que anexam instâncias de U_i a instâncias de U, expresso da seguinte forma:

$$\forall a(a :: U \leftarrow \exists e_1, \ldots, e_n \bigwedge_{i \leq n} (e_i :: U_i \wedge R_i(a, e_i)))$$

Os universais $U_1, \ldots, U_n$ são chamados Características (Feature) da especificação elementar. Um caso especial de especificação elementar é a que envolve a especificação de Momento Intrínseco (Intrinsic Moment).
Caracterização (Characterization) é uma relação entre um Universal e as *Features* de sua especificação elementar.

Ver também: Universal

Endurant (Endurante)

Endurantes são ditos como *"wholly present whenever they are present"*, ou seja, eles *são no tempo*, no sentido de que em uma determinada circunstância c_1 um endurante e tenha uma propriedade p_1, em outra circustância c_2, diferente de c_1, o mesmo endurante e pode ter uma propriedade p_2 possivelmente incompatível com p_1. É o caso, por exemplo, de alguém que pese 50Kg hoje, e pese 60Kg no próximo ano. Outros exemplos de endurantes são: uma casa, uma pessoa, a Lua, um monte de terra (GUIZZARDI, 2005, p. 210). "Por exemplo, é possível dizer que João pesa 80kg em c_1, mas 68kg em $c2$. Em todos os casos se está referindo à mesma pessoa" (GUIZZARDI; WAGNER, 2013, p. 1337).
Nota de tradução: Em português *endurant* poderia ser traduzido como "permansivo", no sentido linguístico de estado de coisa que não muda no decorrer do tempo (COAN et al., 2006, p. 1.466). Ou ainda, poderia ser simbolizado por "duradouro" ou "continuante", conforme Carbonera (2012, p. 42). Porém, opta-se pelo neologismo "endurante", pois este já é usado como termo técnico na filosofia, como evidência presente em Batista (2013).

Ver também: Perdurant

Endurant Universal (Endurante Universal)

Endurante Universal refere-se ao estereótipo usado na hierarquia da UFO para representar os Universais (Universal) que são Endurantes (Endurant). Trata-se da classe mais geral e que caracteriza os universais da UFO-A.

Essential Part (Parte Essencial)

Um indivíduo x é uma parte essencial de um indivíduo y se e somente se: y for, necessariamente, uma parte de x e x for existencialmente depente de y (GUIZZARDI, 2007a), (GUIZZARDI, 2005, p. 165). Formalmente:

> *An individual x is an essential part of another individual y iff, y is existentially dependent on x and x is, necessarily, a part of y: $EP(x, y) \coloneqq ed(y, x) \wedge \Box(x \leq y)$. This is equivalent to stating that $EP(x, y) \coloneqq \Box(\mathcal{E}(y) \rightarrow \mathcal{E}(x)) \wedge \Box(x \leq y)$, which is, in turn, equivalent to $EP(x, y) \coloneqq \Box(\mathcal{E}(y) \rightarrow \mathcal{E}(x) \wedge (x \leq y))$. We adopt here the* mereological continuism *defended by Simons (1987)[4], which states that the part-whole relation should only be considered to hold among existents, i.e., $\forall x, y(x \leq y) \rightarrow \mathcal{E}(x) \wedge \mathcal{E}(y)$. As a consequence, we can have this definition in its final simplification:*

$$EP(x, y) \coloneqq \Box(\mathcal{E}(y) \rightarrow (x \leq y))$$

em que x é a parte e y o todo, e $\leq$ é definido em Ground Mereology.

O uso da modalidade lógica universal, simbolizada por $\Box$, visa impor a ideia de que em qualquer situação em que exista um determinado y, y existe como parte de um x específico. Numa interpretação alética, utilizada largamente em textos sobre ontologia, e considerando uma interpretação de modalidade como mundos possíveis de Kripke (2001), afirmaria-se que "é *necessário* que sempre que um y exista, y exista como parte de um x, ou seja, em qualquer interpretação ou 'mundo possível e acessível', um x é parte essencial de um y se sempre que y existir ele é parte do x".

Conforme Guizzardi (2005, p. 188), a relação de parte essencial é sempre *transitiva*.

De acordo com Guizzardi (2005, p. 344), apenas Universais Rígidos (Rigid Universal) podem participar como Wholes em relações de parte essencial. No caso de Universais anti-rígidos (Anti Rigid Universal), a modalidade universal (representada por $\Box$) das partes só pode ser representada como *de dicto* e, nesse caso, as partes são chamadas de Partes Imutáveis (Immutable Part). Além disso, em modelos OntoUML, sempre que uma relação possuir a propriedade `essential`, a cardinalidade mínima da associação ligada à parte deve ser 1.

Toda Parte Essencial é também uma Parte Imutável (Immutable Part) (GUIZZARDI, 2007a).

Nota: Ver observações sobre o predicado $\mathcal{E}$ em Existential Dependence.

Ver também: Inseparable Part, Mandatory part, Exclusive part, Immutable Part, Inseparable Whole, Mandatory Whole, Generic Dependence, Mereology

[4] Guizzardi (2007a) cita um livro chamado *"Parts. An Essay in Ontology"* de Peter M. Simons – assim como também o faz Smith (1996) e Smith (2003). Porém, do mesmo autor, editora e ano de publicação, encontramos apenas uma obra entitulada *"Parts. A Study in Ontology"*, que mencionamos a referência na citação. De fato, em mensagem eletrônica trocada com Giancarlo Guizzardi em 25.9.2013, o autor confirmou que a obra indicada nesta bibliografia está correta.

Event (Evento)

Ver: Perdurant

Exclusive part (Parte exclusiva)

No contexto de Mereologias (Mereology) e de Relações Meronímicas (Meronymic relation), adaptado de Guizzardi (2005, p. 162), um indivíduo x de um tipo X é uma parte (própria) exclusiva de outro indivíduo y, do tipo Y se y é o único Y que possui x como parte, em que :: é definido em Instantiation Relation e $<$ em Proper part:

$$(x :: X) \wedge (y :: Y) \wedge (x < y) \wedge (\forall z\ ((z :: Y) \wedge (x < z) \rightarrow (y = z)))$$

Ainda conforme a obra citada, esse tipo de relação parte-todo exclusiva pode ser definida entre universais A e B, como uma forma de parte-todo exclusiva geral. Nesse sentido, um universal A é relacionado a um universal B pela relação de *parte exclusiva geral*, simbolizado por $A <_G XB$, sse toda instância x de A tem uma parte exclusiva de B:

$$A <_G XB \coloneqq \forall x\ (x :: A \leftarrow \exists! y(y :: B) \wedge (x < y))$$

Ver também: Non-Shareable, Shareable, Essential Part, Inseparable Part, Mandatory part, Immutable Part, Inseparable Whole, Mandatory Whole

Exemplification (Exemplificação)

Conforme Guizzardi (2005, p. 220), um Individual x exemplifica um Momento Universal (Moment Universal) U se, e somente se, há um Momento individual (Moment individual) y que é instância de U e que é inerente (Inherence) a x. Formalmente:

$$exemplification(x, U) \coloneqq MomentUniversal(U) \wedge \exists y(i(y, x) \wedge y :: U)$$

Ver também: Characterization

Existential Dependence (Dependência Existencial)

A Dependência Existencial refere-se à ideia de *dependência ontológica*. Uma forma da definição do conceito de dependência existencial *ed* aparece em Guizzardi (2005, p. 164), em Guizzardi (2007a), e em Guizzardi e Wagner (2008):

> *Let the predicate $\mathcal{E}$ denote existence. We have that an individual x is* existentially dependent *on another individual y (symbolized as $ed(x, y)$) iff, as a matter of necessity, y must exist whenever x exists, or formally:*

$$ed(x, y) \coloneqq \Box(\mathcal{E}(x) \rightarrow \mathcal{E}(y))$$

Em Guizzardi e Wagner (2010), o conceito é sumarizado como:

> We have that a particular x is existentially dependent (ed) on another particular y iff, as a matter of necessity, y must exist whenever x exists. Existential dependence is a modally constant relation, i.e., if x is dependent on y, this relation holds between these two specific particulars in all possible worlds in which x exists.

Guizzardi (2011) estabelece que a relação de Dependência Existencial é *transitiva*.
Conforme Guizzardi (2006), o predicado $ed(x, y)$ é naturalmente satisfeito no caso de x ser um Momento (Moment) e y um Substancial (Substantial), devido ao fato de que o primeiro representa, por definição, entidades existencialmente dependentes. Porém, o predicado $ed(x, y)$ pode ser satisfeito mesmo para os casos em que ambos x e y são Substanciais, como é o caso de Partes Essenciais (Essential Part) e Partes Inseparáveis (Inseparable Part), por exemplo.
Ainda conforme a obra citada, a dependência existencial trata-se de uma *relação transitiva*.
Nota: Nas obras citadas os autores não esclarecem precisamente o significado do predicado $\mathcal{E}$, sendo que as citações apresentadas são o que há de mais preciso a respeito desse predicado. Em Guizzardi (2005, p. 164), por exemplo, a frase *"Let the predicate $\mathcal{E}$ denote existence"* é interpretada como "Seja $\mathcal{E}$ um predicado monádico que denota que x, em $\mathcal{E}(x)$, é uma instância de um tipo T, na forma $\exists T\ x :: T$, conforme Instantiation Relation", de modo que:

$$\mathcal{E} \coloneqq \exists T\ x :: T$$

Ver também: Essential Part, Inherence, Moment Universal, Substance Sortal, Bearer of a Moment

Extension of a universal (Extensão de um universal)

A extensão de um universal é o conjunto de indivíduos que o instanciam em pelo menos um mundo possível. (GUIZZARDI, 2005, p. 219).

Extensional Individual (Indivíduo Extensional)

No contexto de Mereologia Extensional (Extensional Mereology), um indivíduo y é chamado extencional se, e somente se, para todo x tal que x é parte de y, x é uma Parte Essencial (Essential Part) EP de y (GUIZZARDI, 2005, p. 168):

$$E(y) \coloneqq \Box(\forall x\ (x < y) \rightarrow EP(x, y))$$

Dito de outra forma, um indivíduo é extencional sse todas as suas partes são essenciais (GUIZZARDI, 2005, p. 344).
Quantidades (Quantity) são conceitos necessariamente extensionais.

Ver também: Essential Part, Extensional Mereology, Quantity, Extensionality Principle

Extensional Mereology (Mereologia Extensional)

Conforme Guizzardi (2005, p. 146-147), *Extensional Mereology* (EM) é a teoria composta por Mereologia Base (Ground Mereology) e Suplementação Forte (Strong Supplementation) (SSP) que estende a Mereologia Mínima (Minimal Mereology). As seguintes formulações são teoremas da Mereologia Extensional, em que $\leq$ e $<$ são definidas nas entradas Ground Mereology e Proper part, respectivamente:

$$\exists z\ (z < x) \rightarrow (\forall z\ ((z < x) \rightarrow (z < y)) \rightarrow (z \leq y))$$

Desse teorema, segue este:

$$\exists z\ (z < x) \vee (z < y) \rightarrow ((x = y) \leftrightarrow \forall z\ ((z < x) \leftrightarrow (z < y)))$$

Este último teorema estabelece que dois objetos são idênticos se eles possuem as mesmas partes próprias, o que é a contraparte mereológica do princípio de extensionalidade da teoria de conjuntos, o qual estabelece que dois conjuntos são idênticos se, e somente se, possuem os mesmos membros. Conforme Guizzardi (2011):

> *A known consequence of the introduction of SSP is that in EM we have that two objects are identical iff they have the same parts, i.e., SSP entails a mereological counterpart of the extensionality principle (of identity) in set theory. As a consequence, if an entity is identical to the (mereological) sum of its parts, thus, changing any of its parts changes the identity of that entity. Ergo, an entity cannot exist without each of its parts, which is the same as saying that all its parts are essential parts.*

Ver também: Minimal Mereology, Essential Part, Extensional Individual, Extensionality Principle

Extensionality Principle (Princípio de Extensionalidade)

Conforme Guizzardi (2005, p. 205), o Princípio de Extensionalidade estabelece que dois indivíduos são idênticos sse possuem os mesmos membros.

Ver também: Extensional Individual, Essential Part, Extensional Mereology, Quantity, Category

External Dependence (Dependência externa)

Ver: Externally Dependent Mode

Externally Dependent Mode (Modo Externamente Dependente)

Um Modo (Mode) x é externamente dependente sse é existencialmente dependente (Existential Dependence) de um indivíduo que é independente de seu portador e de todas as suas partes (GUIZZARDI, 2006). Trata-se de um caso de Momento Externamente Dependente (Externally Dependent Moment). Formalmente, Guizzardi (2005, p. 238) apresenta o conceito $ExtDepMode$ de dependência externa baseado na relação assimétrica *indep* (Independence) e de dependência existencial *ed* (Existential Dependence):

$$ExtDepMode(x) \coloneqq Mode(x) \wedge \exists y\ indep(y, \beta) \wedge ed(x, y))$$

Ver também: Qua Individual

Externally Dependent Moment (Momento Externamente Dependente)

Momento Externamente Dependentes são Momentos Intrínsecos (Intrinsic Moment) inerentes a um único Particular mas que são existencialmente dependentes (Existential Dependence) de (possivelmente vários) outros Particulares: um Momento (Moment) x é externamente dependente sse for existencialmente dependente de um indivíduo que é independente de seu portador (GUIZZARDI; WAGNER, 2010). Usando a ideia de Fundação (Foundation), Guizzardi (2005, p. 239) coloca que: *"In the case of a material externally dependent moment x there is an individual external to its bearer (i.e., which is not one of its parts or intrinsic moments), which is the foundation of x".*
Termo alternativo: Externally Dependent Trope

Ver também: Externally Dependent Mode, Inherence, Foundation

Externally Dependent Trope (*Trope* Externamente Dependente)

Ver: Externally Dependent Moment
Ver também: Trope

Extrinsic Moment (Momento extrínseco)

Momentos extrínsecos são opostos a Momentos Intrínsecos (Intrinsic Moment). Tratam-se de Momentos dependentes de mais do que um Objeto. Eles são chamados

em Guizzardi (2005, p. 104) de *"extrinsic (relational) properties"*. Na UFO esse conceito é representado pela entidade Relator.

Ver também: Intrinsic Moment, Relator

F

Feature (Característica)

São chamados *features* os universais $U_1, \ldots, U_n$ participantes da definição de um universal em uma Especificação elementar (Elementary specification).

Formal Relation (Relação Formal)

As relações podem ser de dois tipos: formais ou materiais (Material Relation). As relações são ditas formais quando se referem a duas ou mais entidades diretamente, sem nenhuma intervensão de alguma outra entidade. Tratam-se de relações derivadas de propriedades intrínsecas das entidades relacionadas; elas são *"founded in qualities which are intrinsic to the their relata and, hence, can be reduced to relations between these qualities"* (GUIZZARDI, 2005, p. 242). Conforme Guizzardi (2014): *"A formal relation is a relation that holds directly between its relata and that is reducible to intrinsic properties of these relata"*. Guizzardi (2005, p. 236) apresenta os seguintes exemplos:

> *Examples of formal relations are: 5 is greater than 3, this day is part of this month, and N is subset of Q, but also the relations of instantiation (::), inherence (i), quale of a quality (ql), association (assoc), existential dependence (ed), among others*[5].

Segundo Guizzardi e Wagner (2010), as relações formais podem ser divididas em Relação Formal Básica (Basic Formal Relation) e Relação Formal Comparativa (Comparative Formal Relation).

Ver também: Relator, Relation, Derivation Relation, Inherence, Existential Dependence, Instantiation Relation, Association Relation, Quale, Domain Formal Relation

Foundation (Fundação)

No contexto de caracterização de Relators, a noção de Fundação é um tipo de Dependência Histórica (Historical Dependence) que caracteriza, ou que justifica, a existência de um Relator (GUIZZARDI, 2006; GUIZZARDI; WAGNER, 2008; GUIZZARDI, 2009). O seguinte exemplo está presente em Guizzardi (2009), Guizzardi e Wagner (2010) e também em Costal, Gómez e Guizzardi (2011a, p. 197):

> *Foundation can be seen as a type of historical dependence, in the way that, for example, an instance of* being kissed *is founded on an individual* kiss*, or an instance of* being punched *by is founded on*

[5] Como a dependência parte-todo formal, simbolizada por $\leq$ (GUIZZARDI, 2009).

> *an individual* punch *[...]. Suppose that John* is married to *Mary. In this case, we can assume that there is a particular relator (relational moment)* m_1 *of type* marriage *that mediates John and Mary. The foundation of this relator can be, for instance, a wedding event or the signing of a social contract between the involved parties. In other words, for instance, a certain event* e_1 *in which John and Mary participate can create a particular marriage* m_1 *which existentially depends on John and Mary and which mediates them. The event* e_1 *in this case is the foundation of relator* m_1 *and,* m_1 *is the so-called truthmaker of the propositions "John is married to Mary".*

A noção de Fundação está geralmente ligada a Perdurantes (Perdurant).

Functional Complex (Complexo Funcional)

Diferente dos Coletivos (Collective), os complexos são compostos por partes que exercem diferentes papeis no contexto de um Whole. Enquanto uma coleção possui uma estrutura *uniforme*, Complexos Funcionais possuem estrutura *heterogênea* (GUIZZARDI, 2005, p. 183). As partes de um complexo têm em comum que todas possuem uma ligação funcional com o complexo. Como exemplos, encontra-se em Guizzardi (2009): *"Persons, Cars, Computers, Cells, Organs, Organizations, Organizational Units"*. A Figura 54, extraída de Gerstl e Pribbenow (1995, p. 879), ilustra a relação entre massa, ou *quantidades*, que na UFO é tratada como Quantities, *coleções* ou *coletivos* (Collective) e os *complexos*. Baseado na obra de Gerstl e Pribbenow, Guizzardi (2005, p. 187)[6] apresenta que:

> *In other words, they all contribute to the functionality (or the behavior) of the complex. Therefore, if it is generally the case that essential parthood entails dependence*[7] *[...], in this type of parthood relation, an essential part represents a case of functional dependence. To put it more precisely, for all complexes, if* x *is an essential part of* y *then* y *is functionally dependent on* x*.*

Guizzardi (2011) comenta a diferença entre Coletivos e Complexos Funcionais:

> *In a collective,* all member parts play the same role type w.r.t. the whole*. For example, all trees in a forest can be said to play the role of a forest member. In complexes, conversely, a variety of roles can be played by different components. For example, if all ships of a fleet are conceptualized as playing solely the role of "member of a fleet" then it can be said to be a collection of ships. Contrariwise, if this role is further specialized in "leading ship", "defense ship", "storage ship" and so forth, the fleet must be conceived as a functional complex.*

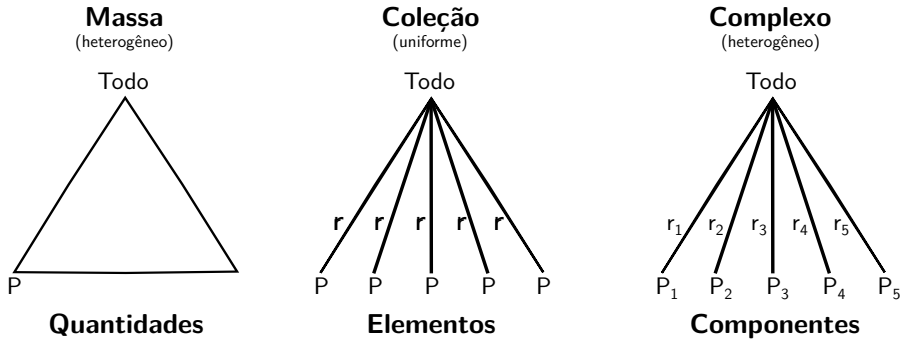
Conforme Guizzardi (2005, p. 350), no âmbito da OntoUML, um *functional complex* X é definido como:

1. Se X é um Universal de Sortal (Sortal Universal), então ele deve ser um Kind ou ser um subtipo de uma classe desse tipo;

[6] Ver também Guizzardi (2005, p. 183).
[7] Ver Essential Part.

2. Caso contrário, se X é um Mixin Universal, para todas as classes Y, tal que Y seja subtipo de X, Y não pode ser uma Quantidade (Quantity) nem um Coletivo (Collective), nem ser um subtipo de uma classe com um desses estereótipos.

Figura 54 – Tipos de Wholes induzidos pelas estruturas composicionais



Fonte: Gerstl e Pribbenow (1995, p. 879)

Ver também: Component of, Component-Functional Complex Relation

G

Generic Dependence (Dependência Genérica)

O conceito de Dependência Genérica *GD* determina a dependência em relação a outros indivíduos de determinado tipo, independente de qual seja a instância específica do outro elemento. Um exemplo é um coração e uma pessoa, ou um carro e um motor: uma pessoa (carro) depende de alguma instância de um objeto do tipo "coração" (ou "motor"), e não de uma instância específica de coração (ou de motor). Isso representa, por exemplo, a possibilidade de substituir um motor do carro sem que o carro perca sua identidade, ou de transplantar um coração de uma pessoa sem que a pessoa deixe de ser quem ela era. Trata-se da dependência "entre um Individual e um Universal" (BENEVIDES, 2010, p. 51). O conceito de Dependência Genérica é apresentado formalmente por Guizzardi (2007a)[8] como:

> *An individual y is generic dependent of a type T iff, whenever y exists it is necessary that an instance of T exists. This can be formally characterized by the following formula schema: $GD(y,T) \coloneqq \Box(\mathcal{E}(y) \rightarrow \exists T, x\mathcal{E}(x))$*

Nota: A exemplo do que ocorre com a entrada Mandatory part, e considerando a interpretação de $\mathcal{E}$ apresentada na entrada Existential Dependence, entende-se que a intenção dos autores citados é melhor representada por esta forma:

$$GD(y,T) \coloneqq \Box(\mathcal{E}(y) \rightarrow \exists x\ x :: T)$$

[8] E também por Guizzardi (2005, p. 167).

Em que a relação :: é definida na entrada Instantiation Relation.

Ver também: Mandatory part, Essential Part, Inseparable Part, Exclusive part, Existential Dependence

Ground Mereology (Mereologia Base)

Segundo Guizzardi (2005, p. 143), em teorias sobre partes, a relação mereológica mais básica, chamada Mereologia Base, é representada pela relação de ordem parcial, ou seja, é reflexiva, antissimétrica e transitiva. Uma relação precisa atender a esses requisitos para ser considerada uma relação de parte. Considerando $\leq$ um símbolo para a relação parte-todo, os seguintes axiomas refletem a *Ground Mereology*:

1. $\forall x \ (x \leq x)$
2. $\forall x, y \ (x \leq y) \wedge (y \leq x) \rightarrow (x = y)$
3. $\forall x, y, z \ (x \leq y) \wedge (y \leq z) \rightarrow (x \leq z)$

Embora necessários, esses requisitos não são suficientes para estabelecer que toda relação que atenda a essas restrições seja considerada uma relação parte-todo. Nesse caso, Mereologia Mínima (Minimal Mereology) e Mereologia Extensional (Extensional Mereology) são exemplos de extensões à *Ground Mereology*.
Ver nota em: Proper part

Ver também: Weak Supplementation, Strong Supplementation, Overlapping

H

Historical Dependence (Dependência Histórica)

Ver: Foundation

Homeomeros

No contexto de Mereologias (Mereology) e de Relações Meronímicas (Meronymic relation), *homeorosity* (*homeomeros* ou *homoeomerous*) significa que uma entidade é composta somente por partes do mesmo tipo (*homo* = mesmo, *mereos* = parte) (GUIZZARDI, 2011).

Ver também: Mereology, Meronymic relation, Quantity

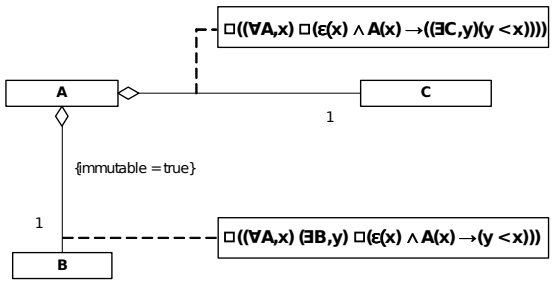
I

Immutable Part (Parte Imutável)

De acordo com Guizzardi (2005, p. 344), apenas Universais Rígidos (Rigid Universal) podem participar como Wholes em relações de Parte Essencial (Essential Part). No caso de Universais anti-rígidos (Anti Rigid Universal), a modalidade universal (representada por $\Box$) das partes só pode ser representada como *de dicto* e, nesse caso, as partes são chamadas de Partes Imutáveis: *"every whole bears a parthood relation to a specific part in all circumstances that it instantiates that specific whole universal"*. Toda Parte Essencial (Essential Part) é também uma *Immutable Part*.

> *In order to achieve a uniform axiomatization, we therefore propose the following formula schemas depicted in Figura 55*[9]*, which must hold irrespective of the type representing the whole being rigid or anti-rigid sortals. If the type A is rigid then $A(x)$ is necessarily true (if true) and the antecedent $(\mathcal{E}(x) \wedge A(x))$ can be expressed only by $(\mathcal{E}(x))$. In this case, the B's are truly essential parts of A's. We refrain from using the term essential part for the cases in which a mere de dicto modality is expressed. Therefore, for the case of specific dependence from instances of anti-rigid types to theirs part we adopt the term immutable part instead. Of course, every essential part is also immutable* (GUIZZARDI, 2007a).

Figura 55 – Representação geral de partes imutáveis e partes mandatórias



Fonte: Guizzardi (2007a)

Sobre a representação de modalidades nas formas *de dicto* e de *de re*, apresenta-se excerto da obra de Carnielli e Pizzi (2008, p. 3):

> *Aristotle was the first to emphasize the distinction between what Medieval logicians called de dicto and de re modalities, i.e., between saying that a proposition is necessary or possible (as, for instance, in $\Box \forall x P(x)$ or $\Diamond \exists x P(x)$) and saying that everything or something has a necessary or a possible property (as, for instance, in $\forall x \Box P(x)$ or $\exists x \Diamond P(x)$). In fact, there is a remarkable difference between saying:*
> *(i) There is someone who has the possibility to become President of the Republic, and*
> *(ii) It is possible that someone becomes President of the Republic.*

Em modelos OntoUML, sempre que uma relação possuir a propriedade `immutable`, a cardinalidade mínima da associação ligada ao Whole deve ser 1 (GUIZZARDI, 2005, p. 344).

[9] Originalmente: "figure 9".

Ver também: Mandatory Whole, Essential Part, Inseparable Part, Mandatory part, Exclusive part, Mereology

Independence (Independência)

O conceito de Independência é usado para formalizar a ideia de *disjunção*. Mais especificamente, para positivar que dois Substanciais (Substantial) são disjuntos, ou independentes um do outro. Isso é o caso, por exemplo, de uma pessoa que depende de seu cérebro, e um carro que depende de seu chassi. Porém, a pessoa, assim como o carro, são independentes de qualquer outro Substancial disjunto de si. Baseado no predicado de Dependência Existencial *ed* (Existential Dependence), Guizzardi (2005, p. 218) apresenta a seguinte definição formal para independência *indep*:

$$indep(x, y) \coloneqq \neg ed(x, y) \wedge \neg ed(y, x)$$

A fórmula seguinte também exclui o caso de dependência existencial mútua entre Substanciais que compartilham uma mesma Parte Essencial (Essential Part) (GUIZZARDI, 2005, p. 218):

$$\forall x, y Substantial(x) \wedge Substantial(y) \wedge (x f y) \rightarrow indep(x, y)$$

Conforme Guizzardi (2005, p.238), *indep* é uma relação simétrica.

Ver também: Substantial Universal

Indirect Quality (Qualidade Indireta)

Baseado em Masolo et al. (2003), Guizzardi (2005, p. 232) expõe que sempre que um Universal de Qualidade (Quality Universal) U está relacionado com um Domínio de Qualidade (Quality Domain) D – ou seja, para cada Qualidade (Quality) x tal que $x :: U$ – há qualidades indiretas (*"indirect qualities"*) inerentes a x para cada Dimensão de Qualidade (Quality Dimension) associado a D:

> *For instance, for every particular quality c instance of Color there are quality individuals h, s, b which are instances of quality universals Hue, Saturation and Brightness, respectively, and that inhere in c. The qualities h, s, b are named* indirect qualities *of c's bearer.*

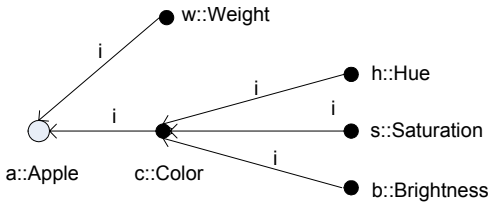
A Figura 56 ilustra a ideia exposta.

Ver também: Instantiation Relation, Simple Quality, Complex Quality, Inherence

Individual

Ver: Particular

Figura 56 – Exemplo de cadeia de inerências de Quality



Fonte: Guizzardi (2005, p. 232)

Inherence (Inerência)

A ideia de inerência trata-se de uma relação irreflexiva, assimétrica e intransitiva entre Momentos (Moment) e outro Endurante (Endurant) (GUIZZARDI, 2005, p. 213), (GUIZZARDI, 2009). Trata-se de um particular de uma instância de Dependência Existencial (Existential Dependence). As expressões seguintes descrevem o conceito de Inerência i de x em y ($i(x, y)$), em que $ed(x, y)$ refere-se à Dependência Existencial (GUIZZARDI, 2005, p. 213-214):

$$\forall x, y(i(x, y) \rightarrow Moment(x) \wedge Endurant(y))$$
$$\forall x, y(i(x, y) \rightarrow ed(x, y))$$
$$\forall x \neg i(x, x)$$
$$\forall x, y(i(x, y) \rightarrow \neg i(y, x))$$
$$\forall x, y, z(i(x, y) \wedge i(y, z) \rightarrow \neg i(x, z))$$

Por exemplo:

> *if x inheres in y then x cannot exist in a given situation without that very specific y existing in that same situation (existential dependence) and there is no z different from y such that x inheres in z* (GUIZZARDI, 2005, p. 215).

Conforme Guizzardi e Wagner (2008), os autores adotam o princípio de Não-Migração (Non Migration), que estabelece que não é possível que um Momento (Moment) m (também chamado de Trope) seja inerente a dois indivíduos a e b:

$$\forall x, y, z(Trope(x) \wedge i(x, y) \wedge i(x, z) \rightarrow y = z)$$

Conforme Guizzardi e Wagner (2010), a relação *Inherence* pode ser estabelecida entre dois Momentos:

> *Here, we admit that tropes can inhere in other tropes. Examples include the individualized time extension, or the graveness of a particular symptom. The infinite regress in the inherence chain is prevented by the fact that there are individuals that cannot inhere in other individuals, namely, Objects*[10].

[10] No caso, *"Objects"* é usado pelo autor como sinônimo de Substancial (Substantial).

Ver também: Bearer of a Moment

Inseparable Part (Parte Inseparável)

A ideia de Parte Inseparável, em contraste com Parte Essencial (Essential Part), é quando a existência do todo é indispensável à parte para existir. Ou seja, x é uma *Parte Inseparável* de um y se a parte x não puder existir sem ser parte do todo y. Trata-se de outra forma de analisar o exemplo entre o cérebro e a pessoa apresentado em Parte Mandatória (Mandatory part). Conforme exemplo de Guizzardi (2007a), o tempo de vida de uma pessoa e de seu cérebro devem, necessariamente, coincidir. Isso ocorre porque o cérebro é existencialmente dependente (Existential Dependence) de seu portador. Sempre que existe uma situação em que a parte é existencialmente dependente do todo o qual ela compõe, tem-se o conceito de Parte Inseparável. Uma Parte Inseparável IP é apresentado formalmente por Guizzardi (2007a) e por Guizzardi (2005, p. 169) como, em que x é a parte e y o todo:

> *An individual x is an inseparable part of another individual y iff, x is existentially dependent on y, and x is, necessarily, a part of y:*

$$IP(x, y) \coloneqq \Box(\mathcal{E}(x) \rightarrow (x \leq y))$$

Nota: Ver nota sobre o uso da modalidade universal em Essential Part.

Ver também: Inseparable Whole, Essential Part, Mandatory part, Exclusive part, Immutable Part, Mandatory Whole, Generic Dependence, Existential Dependence

Inseparable Whole (Todo Inseparável)

Todo Inseparável é uma noção derivada da ideia de Dependência Genérica (Generic Dependence). Um coração não é Parte Inseparável (Inseparable Part) de uma pessoa: um coração pode existir antes de uma pessoa, assim como pode continuar existindo depois desta pessoa. Entretanto, um coração necessita de uma pessoa para existir (e não de uma específica e determinada pessoa). Com isso em tela, Guizzardi (2007a) propõe a noção de Todos Inseparáveis:

> *An individual y is a mandatory whole for another individual x iff, x is generically dependent on a type T that y instantiates, and x is, necessarily, part of an individual instantiating T: $MW(T, x) \coloneqq \Box(\mathcal{E}(x) \rightarrow (\exists T, y)(x \leq y)))$.*

Ver também: Inseparable Part, Mandatory part, Exclusive part, Immutable Part, Inseparable Whole, Mandatory Whole, Generic Dependence, Mereology, Existential Dependence, Whole

Instance (Instância)

Ver: Instantiation Relation

Instantiation Relation (Relação de Instanciação)

A relação de instanciação é uma relação formal definida entre Individuais (Individual) e Universais (Universal). O símbolo :: denota a relação de instanciação, sendo que em $x :: U$, um indivíduo x é uma instância de U, ou x possui a propriedade de pertencer a U (GUIZZARDI, 2005, p. 218).

Internal Relation (Relação Interna)

Ver: Basic Formal Relation

Intrinsic Moment (Momento Intrínseco)

Ver: Intrinsic Moment Universal

Intrinsic Moment Universal (Momento Intrínseco Universal)

Momentos Intrínsecos Universais são Momentos (Moment) que dependem (Existential Dependence) de apenas um Particular, em oposição aos Relators, que dependem de mais de um deles (GUIZZARDI; WAGNER, 2010). Tratam-se das bases para atributos e relações formais comparativas (BENEVIDES, 2010, p. 41). São subdivididos em Universal de Qualidade (Quality Universal) e Modo Universal (Mode Universal). Os exemplos presentes em Guizzardi (2005, p. 213) são:

> *qualities such as a color, a weight, a electric charge, a circular shape; modes such as a thought, a skill, a belief, an intention, a headache, as well as dispositions such as the refrangibility property of light rays, or the disposition of a magnetic material to attract a metallic object;*

Ver também: Elementary specification, Extrinsic Moment

K

Kind (Espécie)

Um *Kind* é um Sortal Rígido (Rigid Sortal) que fornece um Princípio de identidade (Principle of identity) às suas instâncias (GUIZZARDI, 2005, p. 108). *"An important postulate of UFO is: Every object must instantiate exactly one kind"* (GUIZZARDI; WAGNER, 2010).

Em Guizzardi e Zamborlini (2014), *Kind* aparece como sinônimo de Tipo Natural (Natural type).

Nota de tradução: Opta-se pela palavra portuguesa "espécie" ao invés, por exemplo, de "tipo", para ressaltar a existência de distinção entre o conceito de *kind* proposto pelo autor em relação ao conceito de "tipo" de uma teoria de tipos.

Ver também: Subkind, Rigidity, Quantity, Collective

M

Mandatory part (Parte Mandatória)

O conceito de Parte Mandatória MP determina a dependência em relação a outros indivíduos *específicos* de determinado tipo, de forma dependente da instância. Trata-se de um tipo de Dependência Genérica (Generic Dependence) (GUIZZARDI, 2005, p. 167). Um exemplo é um cérebro e uma pessoa, ou um chassi e um motor: uma pessoa (carro) depende especificamente de um determinado objeto do tipo "cérebro" (ou "chassi"). Isso representa, por exemplo, a impossibilidade de substituir um chassi do carro sem que o carro perca sua identidade, ou de transplantar um cérebro de uma pessoa sem que a pessoa deixe de ser quem ela era. O conceito de parte mandatória é apresentado formalmente por Guizzardi (2007a)[11] como:

> *An individual x is a mandatory part of another individual y iff, y is generically dependent of an type T that x instantiates, and y has, necessarily, as a part an instance of T:* $MP(T, y) \coloneqq \Box(\mathcal{E}(y) \rightarrow (\exists T, x)(x < y))$.

Nota: Embora a transcrição da citação tenha sido feita exatamente como encontrada na obra citada, entende-se que a intenção do autor mencionado seria melhor representada nesta forma – em que a relação :: é definida conforme a entrada Instantiation Relation:

$$MP(T, y) \coloneqq \Box(\mathcal{E}(y) \rightarrow \exists x\ (x :: T \wedge x < y))$$

Ver também: Generic Dependence, Proper part, Essential Part, Inseparable Part, Exclusive part, Immutable Part, Mandatory Whole, Mereology

Mandatory Whole (Todo Mandatório)

Para capturar o caso em que um coração é dependente da existência de uma pessoa, mas não necessariamente da mesma pessoa (Generic Dependence), Guizzardi (2007a) apresenta a seguinte definição de *mandatory whole* (MW):

> An individual y is a mandatory whole for another individual x iff, x is generically dependent on a type T that y instantiates, and x is, necessarily, part of an individual instantiating T: $MW(T, x) \coloneqq \Box(\mathcal{E}(x) \rightarrow (\exists T, y)(x < y)))$

A seguir, a notação do autor é adaptada na seguinte formulação, em que :: é definido na entrada Instantiation Relation:

$$MW(T, x) \coloneqq \Box(\mathcal{E}(x) \rightarrow \exists T \exists y((y :: T) \wedge (x < y)))$$

[11] E também por Guizzardi (2005, p. 167), sendo que neste altera-se a variável T por U.

Ver também: Essential Part, Inseparable Part, Mandatory part, Exclusive part, Immutable Part, Mandatory part, Inseparable Whole, Whole, Generic Dependence

Mass-Quantity Relation (Relação Massa-Quantidade)

Conforme Guizzardi (2010a), *Sub Quantity of* ou *subQuantityOf*, conforme Guizzardi (2005, p. 184), *Mass-Quantity relation*, e conforme Guizzardi (2011), *subquantity-quantity*, é uma relação Parte de (Part of) entre duas Quantidades (Quantity). Trata-se de uma relação sempre *transitiva*, uma vez que as entidades envolvidas, Quantidades, são sempre quantidade de matéria maximalmente conectadas. Esse argumento é apresentado por Guizzardi (2010a) lançando mão do conceito de Topoid:

> *Another way to examine this situation is by inspecting A at an arbitrary time instant t. We can say that all parts of A are the quantities that are contained in a certain region of space R (i.e. a Topoid). Since A is an objectified matter, than the topoid R occupied by A must be self-connected. Therefore, if B is part of A then B must occupy a sub-region R′, which is part of R. Likewise, if C is part of B, it occupies a region R″, part of R′. Since spatial part-whole relations are always transitive (JOHANSSON, 2004), we have that R″ is part of R, and if C occupies R″, then it is contained in R. Ergo, by definition, C is a part of A.*

Da mesma forma, a relação *massa-quantidade* não pode ter como parte uma quantidade do mesmo tipo, todas as suas partes são não compartilháveis (Non-Shareable) e, no caso de modelagem com OntoUML, a restrição mínima de cardinalidade deve ser exatamente 1, uma vez que se tratam de partes maximalmente conectadas:

> *For example, take a case in which this relation holds between a quantity of alcohol x and a quantity of wine y. Since y is self-connected it occupies a self-connected portion of space. The same holds for x. In addition, the topoid occupied by x must be a (improper) part of the topoid occupied by y. Now suppose that there is a portion of wine z (different from y) such that x is a subQuantityOf z. A consequence of this is that z and x overlap*[12]*, and since they are both self-connected, we can define a portion of wine w which is itself self-connected. In this case, both z and x are part of w and therefore, they are not* maximally-self-connected-portions*. This contradicts the premises that x and z were quantities. Hence, we can conclude that the* subQuantityOf *relation is always non-sharable. Furthermore, since every part of a quantity is itself a quantity, subQuantityOf must also have a cardinality constraint of* one and exactly one *in the subquantity side. Take once more the alcohol-wine example above. Since alcohol is a quantity (and, hence, maximal), there is exactly one quantity of alcohol which is part of a specific quantity of wine* (GUIZZARDI, 2010a).

A alteração em qualquer das partes de um Whole – que naturalmente é uma Quantidade – causa a perda da identidade dessa entidade que funciona como o "todo". Por isso, todas as partes são essenciais (Essential Part). Este é o único caso de uma relação Part of que o axioma da Suplementação Fraca (Weak Supplementation) é

[12] Ver Overlapping.

substituído pelo axioma Suplementação Forte (Strong Supplementation). Por isso, *massa-quantidade* trata-se de uma relação que é um caso de Mereologia Extensional (Extensional Mereology) com partes não compartilháveis e essenciais, o que significa que duas quantidades são a mesma se, e somente se, possuem as mesmas partes e nenhum par de instâncias do mesmo tipo se sobrepõem (Overlapping).
Em Guizzardi (2005, p. 345) e em Guizzardi (2010a), as meta-propriedades da relação em tela são as propriedades da Parte Própria (Proper part), ou seja, *irreflexibilidade*, *anti-simetria* e *transitividade*. Já em Guizzardi (2005, p. 350), a propriedade *irreflexibilidade* é substituída por *não-reflexibidade*.
Nota: Considera-se meta-propriedades das relações as apresentadas por Guizzardi (2010a).

Ver também: Subquantity of, Part of, Subcollection-Collection Relation, Member-Collection Relation, Component-Functional Complex Relation, Extensional Individual

Material Relation (Relação Material)

As relações podem ser de dois tipos: formais (Formal Relation) ou materiais. Relações Materiais são dependentes de relações extrínsecas, ou seja, tratam-se de relações entre indivíduos mediados (Mediation) por Universais de *Relator* (Relator Universal) (GUIZZARDI, 2005, p. 236). *"Moreover, material relations are founded by material entities in reality, typically perdurants, which are external to their relata"* (GUIZZARDI, 2005, p. 242).
Cada associação do tipo material deve estar relacionada com exatamente uma Relação de Derivação (Derivation Relation) (GUIZZARDI; WAGNER, 2010). Como exemplos, temos que:

> *Material relations [...] have material structure of their own and include examples such as working at, being enrolled at, and being connected to* (GUIZZARDI; WAGNER, 2010).

Apenas as relações materiais são genuinamente ontológicas, enquanto as Relações Formais (Formal Relation) podem ser caracterizadas como meras "construções lógicas", que não significam propriedades das coisas em si (GUIZZARDI; WAGNER, 2008). De fato, as relações materiais apenas expõem os fatos derivados de Relators e suas entidades mediadas (GUIZZARDI, 2005, p. 331), por isso devem possuir a propriedade `isDerived=True` em diagramas OntoUML.

Ver também: Relator, Relation, Mediation

Mediation (Mediação)

De forma análoga à relação de Caracterização (Characterization), a relação de Mediação é uma Relação Formal (Formal Relation) que acontece entre Universais

de *Relator* (Relator Universal) e aqueles Universais (Universal) de suas entidades mediadas. *"The mediation relation holds between a universal U and a relator universal U_R iff every instance of U is mediated by an instance of U_R"* (GUIZZARDI, 2005, p. 240-241). A definição formal de *Mediation* é feita do seguinte modo: Seja x, y e z três distintos indivíduos, de modo que (a) x é um *relator*; (b) y é um Qua Individual e y é parte de x; (c) y é inerente (Inherence) a z. Nesse caso, diz-se que x media z, simbolizado por m. Formalmente[13]:

$$\forall x, y, m(x, y) \rightarrow relator(x) \wedge Endurant(y)$$

Com base nessa definição da relação m , o autor apresenta a seguinte definição formal para *Mediation* entre Universais:

$$mediation(U, U_R) :=$$
$$Universal(U) \wedge RelatorUniversal(U_R) \wedge \forall x(x :: U \rightarrow \exists r\ r :: U_R \wedge m(r, x))$$

Guizzardi (2009) acrescenta que:

> *The relation of mediation (symbolized m) between a relator r and the entities r connects is a sort of (non-exclusive) inherence and, hence, a special type of existential dependence relation.*

Ver também: Relator, Mode

Mediation Relation (Relação de Mediação)

Ver: Mediation

Member of (Membro de)

Estereótipo usado na OntoUML para representar a Relação Membro-Coleção (Member-Collection Relation).

Ver também: Subcollection of, Component of, Subquantity of

Member-Collection Relation (Relação Membro-Coleção)

Trata-se de uma relação do tipo Parte de (Part of) *intransitiva* estabelecida entre uma *entidade singular* e uma entidade *plural*, ou seja, um Coletivo (Collective). A Relação Membro-Coleção possui o axioma de Suplementação Fraca (Weak Supplementation) e quebra a transitivade natural que existe nas relações Parte de (Part of). Desse modo, baseado em Guizzardi (2010b), Guizzardi (2011) e em Guizzardi (2005, p. 185-186), se $M(X, Y)$ é uma Relação Membro-Coleção em que X é o membro e Y é a coleção, se $M(x, W)$ então x é um átomo de W, e se $M(y, x)$ é o caso, então se obtém $\neg M(y, W)$. Extraído de Guizzardi (2011):

[13] Esta definição está presente também na entrada Relator

> *The following example illustrates the intransitivity of the member-collection relation: "I am member of a club C (collective) and my club is a member of an International Association of clubs C' (collective). However, it does not follow that I am a member of this International Association of Clubs C' since this only has clubs as members, not individuals". However, an even more general statement about the intransitivity of this relation can be made. Since members of a collective are considered to be atomic w.r.t. the context in which the collective is defined, if an individual x is a part of (member of) a collection y, then for every z which is part of (member of, functional part of, sub-collection of) x, then z is not a part of (member of) y. In other words, the member-collective relation causes the part to necessarily be seen as atomic in the context of the whole, hence, "blocking" a possible transitive chain of part-whole relations. Thus, for instance, although an individual John can be part of (member of) a Club, none of John's parts (e.g., his heart) is part of that Club.*

Um ponto referente à modelagem conceitual ontologicamente bem-fundamentada que se obtém dessa relação é a noção de *extensão* e de Parte Essencial (Essential Part), de modo que *member-collection* não é necessariamente extensional (Extensional Individual), mas se for, exige-se que suas partes sejam essenciais:

> *unlike sets and mereological sums, collectives do not necessarily have an extensional criterion of identity. That is, whereas for some collectives the addition or subtraction of a member renders a different individual, it is not the case that this holds for all of them. However, when this is the case, all member-collective relations that the extensional collective participates as a whole are relations of essential parthood. This is because, since a collective (by definition) has a uniform structure, then all members of a collective must be indistinguishable w.r.t. the whole. As a consequence, it cannot be the case that some members of a collection are essential while others are not. In summary, member-collective relations are only relations of essential parthood if the collective in the association end connected to the whole is an extensional individual. In the converse reading, if a collective is extensional then all its parts (members) are essential (*GUIZZARDI, 2011, passim*).*

Conforme apresentado na entrada Relação Subcoleção-Coleção (Subcollection-Collection Relation), a Relação Membro-Coleção é sempre *transitiva* quando é combinada com uma Relação Subcoleção-Coleção (GUIZZARDI, 2011).
Em Guizzardi (2005, p. 352), a relação em tela é apresentada como *não-reflexiva* e *anti-simétrica*. Já em Guizzardi (2011), ela é apresentada como *irreflexiva* e *assimétrica*. Considerando que este trabalho é mais recente, as duas últimas propriedades são adotadas em detrimento das primeiras.

Ver também: Member of, Part of, Subcollection-Collection Relation, Mass-Quantity Relation, Component-Functional Complex Relation

Member-Collective relation (Relação Membro-Coletivo)

Ver: Member-Collection Relation

Mereological Sum (Soma Mereológica)

O conceito de Soma Mereológica é apresentado por Guizzardi (2005, p. 240) como: a fórmula $\sigma x \phi$ representa um indivíduo x composto por todos os indivíduos que satisfazem o predicado ϕ.

Ver também: Relator

Mereology (Mereologia)

Mereologia é um termo inventado pelo matemático, lógico e filósofo Stanisław Leśniewski (1886-1939), que significa *'teoria das partes'* (SIMONS, 2011). De fato, Leśniewski propôs uma mereologia especialmente focada nas "partes", que foi utilizada por diferentes trabalhos posteriores. A teoria de Leśniewski é formada originalmente por quadro axiomas e três definições. Abstém-se de apresentar as definições, que podem ser consultadas na obra citada. Já os axiomas opta-se por transcrevê-los, por serem utilizados na UFO. O primeiros dois axiomas referem-se às propriedades de simetria e transitividade da relação de parte. Os dois últimos referem-se ao conceito de *classe*, que permite falar de *a classe de m* sempre que houver ao menos um objeto m:

a) Axioma 1: Se o objeto A é parte do objeto B, então B não é parte do objeto A;

b) Axioma 2: Se o objeto A é parte do objeto B, e o objeto B é parte do objeto C, então A é parte de C;

c) Axioma 3: Se alguma parte é (um) m, então algum objeto é uma classe de objetos m;

d) Axioma 4: Se A é uma classe de objetos m, e B é uma classe de objetos m, então A é B.

Na UFO são utilizados quatro tipos de relações conceituais de parte-todo (Part of): Componente de (Component of), Membro de (Member of), Subcoleção de (Subcollection of) e Subquantidade de (Subquantity of) (GUIZZARDI, 2005, p. 183-189, 341).

Ver também: Meronymic relation, Ground Mereology, Minimal Mereology, Extensional Mereology, Proper part, Disjointness, Overlapping, Proper Overlapping, Weak Supplementation, Strong Supplementation, Mereological Sum, Generic Dependence, Essential Part, Inseparable Part, Mandatory part, Exclusive part, Immutable Part, Inseparable Whole, Mandatory Whole, Whole, Extensional Individual, Extensionality Principle, Member-Collection Relation, Subcollection-Collection Relation, Mass-Quantity Relation, Component-Functional Complex Relation, Topoid, Homeomeros

Meronymic relation (Relação Meronímica)

Meronímia – e *relações de meronímia* ou *relações meronímicas* – é um termo da linguística que se refere às relações do tipo "parte-todo" que ocorrem entre termos. Conforme Cann (2011, p. 464):

> Meronymy*: If X is part-of Y or Y has X then X is a* meronym *of Y and Y is a* holonym *of X.*

De forma similar, em Murphy (2003, p. 230) encontra-se esta definição:

> ***Meronymy** is the* IS-A-PART-OF *(or* HAS-A*) relation, and (like* hyponymy*) the term refers either to the directional relation from whole to part or collectively to that relation and its converse,* ***holonymy****. So, for example,* cockpit *is a meronym of* airplane *and* airplane *is a holonym of* cockpit*, and the relation between these two items is meronymy.*

Embora aparentemente simples, o tema das *meronímias* é um assunto complexo que possui diferentes abordagens na linguística, lógica e ciências cognitivas (WINSTON; CHAFFIN; HERRMAN, 1987), (CANN; MURPHY, 2011, 2003, loc. cit.), (CROFT; CRUSE, 2004, p. 159). Nesse sentido, as teorias que tratam as relações parte-todo são chamadas de Mereologias (Mereology).

No âmbito da UFO, Guizzardi (2005, p. 141-200) desenvolve quatro tipos de diferentes relações meronímicas: Componente de (Component of), Membro de (Member of), Subcoleção de (Subcollection of) e Subquantidade de (Subquantity of).

Ver também: Mereology

Metric Space (Espaço Métrico)

No contexto de Qualidades (Quality) e de Quales, os espaços métricos são definidos com base em *funções de distância*, que permitem atribuir conceitos de *similaridade* entre diferentes objetos em um domínio. Conforme Guizzardi (2005, p. 228):

> *The definition of a distance function is very important for many reasons, both in theoretical and in practical terms. Given a quality domain and its constituting dimensions, the quality function represents the degree of similarity between two individuals. One*

> *can for example give a precise account for why a certain shade of unique red is more similar to orange than to a shade of green, or why the taste of hazelnut is more similar to the taste of walnut than to the one of limes. The other reason is that, as shown by (GÄRDENFORS, 2000), given a number of* prototype *points in a metric quality domain, and a similarity function, a partition of the domain in different quality regions can be generated. Moreover, all generated regions are* convex regions.

Minimal Mereology (Mereologia Mínima)

Minimal Mereology é a teoria composta por Mereologia Base (Ground Mereology) e pot Suplementação Fraca (Weak Supplementation) (GUIZZARDI, 2005, p. 145).

Ver também: Extensional Mereology, Proper part

Mixin

Mixins representam entidades não-rígidas e não-sortais com propriedades que são essenciais para algumas entidades mas acidentais para outras. Exemplo presente em Guizzardi (2005, p. 113) é a entidade `Sentável`, que é essencial para uma `Cadeira`, mas é acidental para alguma `Pedra` ou alguma `Caixa`. Um *Mixin* não pode ser uma especialização de um Sortal, nem pode ter instâncias diretas (GUIZZARDI; WAGNER, 2010).
Na hierarquia de conceitos da UFO, o *Mixin* é o estereótipo que dos *Mixins* Semi-rígidos (Semi Rigid Mixin). Os *Mixins* são o complemento das disjunções entre *Mixin* de Papel (Role Mixin) e Categoria (Category).

Ver também: Mixin Universal, Dispersive Universal

Mixin Universal

Mixin Universals são abstrações de propriedades comuns para vários tipos disjuntos (GUIZZARDI, 2005, p. 112). Eles não podem ter instâncias diretas (GUIZZARDI; WAGNER, 2010) e se subdividem em *Mixins* Rígidos (Rigid Mixin) e *Mixins* Não-Rígidos (Non Rigid Mixin).

Ver também: Mixin, Role Mixin, Category, Dispersive Universal

Mode (Modo)

Ver: Mode Universal

Mode Universal (Modo Universal)

Um Modo Universal é um Momento Intrínseco Universal (Intrinsic Moment Universal) que "não é diretamente relacionado a Estruturas de Qualidade (Quality Structure)" (GUIZZARDI; WAGNER, 2010); *"A mode is an intrinsic moment individual which is not a quality."* (GUIZZARDI, 2005, p. 237). Exemplo: habilidades, crenças e pensamentos, que são existencialmente dependentes (Existential Dependence) de uma única `Pessoa`.

$$mode(x) \coloneqq intrinsicMoment(x) \wedge \neg quality(x)$$

Conforme Guizzardi e Wagner (2010):

> *Like objects, modes can bear other moments, and each of these moments can refer to separable quality dimensions. However, since they are moments, differently from objects, modes are necessarily existentially dependent of some particular*[14].

De forma concisa, Albuquerque e Guizzardi (2013) apresenta o conceito do seguinte modo:

> *In contrast to qualities, Modes are moments which cannot be directly evaluated in terms of single value space. Examples of modes include beliefs, intentions, goals and dispositions (e.g., the disposition of a magnet to attract electric material).*

Ver também: Externally Dependent Mode, Qua Individual, Inherence

Moment (Momento)

Ver: Moment Universal

Ver também: Moment individual

Moment individual (Momento individual)

Uma instância (Particular) de um Momento Universal (Moment Universal). Em Guizzardi (2009) o conceito é apresentado com o termo *Trope*.

Moment Universal (Momento Universal)

Diferente dos Universais Substanciais (Substantial Universal), os Momentos Universais tratam-se de categoria de elementos que dependem de ao menos outro, do qual são inerentes, para existir. Diz-se que são existencialmente dependentes (Existential Dependence), ou seja, apenas existem em outros indivíduos. Conforme Guizzardi (2005, p. 212)[15]:

[14] Em Guizzardi (2005, p. 238): *"Like substantials, modes can bear other moments, and each of these moments can be qualities referring to separable quality dimensions. However, since they are moments, differently from substantials, modes inhere necessarily in some bearer."*

[15] As referências a obras foram substituídas por "[. . .]".

> *The word Moment is derived from the german Momente in the writings of Husserl and it denotes, in general terms, what is sometimes named* trope [...], abstract particular [...], mode [...], particular quality*, individual accident, or* property instance*. In the scope of this work, the term bears no relation to the notion of time instant in ordinary parlance. The origin of the notion of moment lies in the theory of individual accidents developed by Aristotle in his Metaphysics and Categories. For him, an accident is an individualized property, event or process that is not a part of the essence of a thing. We here use the term "moment" in a more general sense and do not distinguish a priori between essential and inessential moments.*

Conforme Guizzardi e Wagner (2010), o termo *moment* não possui relação com a noção de 'instante no tempo', conforme é comum na linguagem coloquial. Ainda segundo o autor, exemplos típicos de *moments* são: uma cor, a carga elétrica (que só pode existir em um condutor), um acordo, ou compromisso social.
Considerando a noção de Inerência (Inherence) i, Guizzardi e Wagner (2010) apresenta que:

> *for a particular x to be a moment of another particular y, the relation $i(x, y)$ must hold between the two. For example, inherence glues your smile to your face, or the charge in a specific conductor to the conductor itself.*

Ainda em Guizzardi e Wagner (2010), mas também analisando Guizzardi (2006), temos que um *moment* pode ser inerente em outro *moment*, conforme segue:

> *Here, we admit that moments can inhere in other moments. Examples include the individualized time extension, or the graveness of a particular symptom. The infinite regress in the inherence chain is prevented by the fact that there are individuals that cannot inhere in other individuals, namely, objects.*

Ver também: Intrinsic Moment Universal, Relator, Characterizing Universal

Monadic (Monádico)

Ver: Monadic Universal

Monadic Universal (Universal Monádico)

Universais Monádicos são conceitos que se referem a uma única entidade, como Pessoa, Cão, Flor, Banco de Dados, Bruxa.
Os Universais Monádicos se subdividem em Universais Substanciais (Substantial Universal) e Momentos Universais (Moment Universal).

N

Natural type (Tipo Natural)

Ver: Kind

Non Migration (Não-Migração)

Não-Migração ou *non-transferability* é um princípio que estabelece que não não é possível que um Momento (Moment) m (também chamado de *Trope*) seja inerente a dois indivíduos a e b (GUIZZARDI, 2006; GUIZZARDI; WAGNER, 2008):

$$\forall x, y, z(Trope(x) \wedge i(x, y) \wedge i(x, z) \rightarrow y = z)$$

Ver também: Inherence, Bearer of a Moment

Non Rigid Mixin (*Mixin* Não-Rígido)

Mixins Não-Rígidos representam características que são essenciais para algumas entidades mas acidentais para outras. Eles se subdividem em *Mixins* Semi-rígidos (Semi Rigid Mixin) e *Mixins* anti-rígidos (Anti Rigid Mixin).

Non Sortal (Não-*Sortal*)

Ver: Mixin Universal

Non Sortal Universal (Não-*Sortal* Universal)

Ver: Mixin Universal

Non Transferability (Não-transferibilidade)

Ver: Non Migration

Non-Shareable (Não-Compartilhável)

Ver: Exclusive part

Ver também: Shareable, Component of, Member of, Subcollection of, Subquantity of, Mereology

O

Object (Objeto)

Usado em Guizzardi e Zamborlini (2014) para se referir a Universal Substancial (Substantial Universal):

> *As a synonym for the term* Object *as employed here, we have used the term* Substantial *elsewhere (GUIZZARDI, 2005). The variation, which is only terminological, is again motivated by the goal of harmonizing this theory with a jargon closer to the one already used by the conceptual modeling community.*

Object Universal (Objeto Universal)

Ver: Substantial Universal[16]

OntoUML

A OntoUML é um perfil (*profile*) da *Unified Modeling Language* (UML) (OBJECT MANAGEMENT GROUP, 2003; OBJECT MANAGEMENT GROUP, 2011b) construído como uma linguagem de diagramação para modelagem conceitual fundamentada na UFO. O perfil abrange: um conjunto de estereótipos que representam distinções ontológicas propostas pela teoria da UFO e restrições nas relações possíveis a serem estabelecidas por esses elementos que representam os postulados da teoria (GUIZZARDI; WAGNER, 2010).

Overlapping (Sobreposição)

No contexto de Mereologias (Mereology), dois indivíduos se sobrepõem (*overlap*) se eles possuem partes em comum. Isso inclui o caso no qual um indivíduo é parte de outro e também o caso de identidade. Trata-se de uma relação reflexiva, simétrica e não transitiva, que é equivalente à relação de intersecção da teoria de conjuntos. Considerando $\bullet$ um símbolo para *overlap*, e $\leq$ a definição contida na entrada Ground Mereology, tem-se a seguinte definição (GUIZZARDI, 2005, p. 144):

1. $(x \bullet y) \coloneqq \exists z\ (z \leq x) \wedge (z \leq y)$
2. $\forall x\ (x \bullet x)$
3. $\forall x, y\ (x \bullet y) \rightarrow (y \bullet x)$

Ver também: Ground Mereology, Proper Overlapping, Proper part, Disjointness

P

Part of (Parte de)

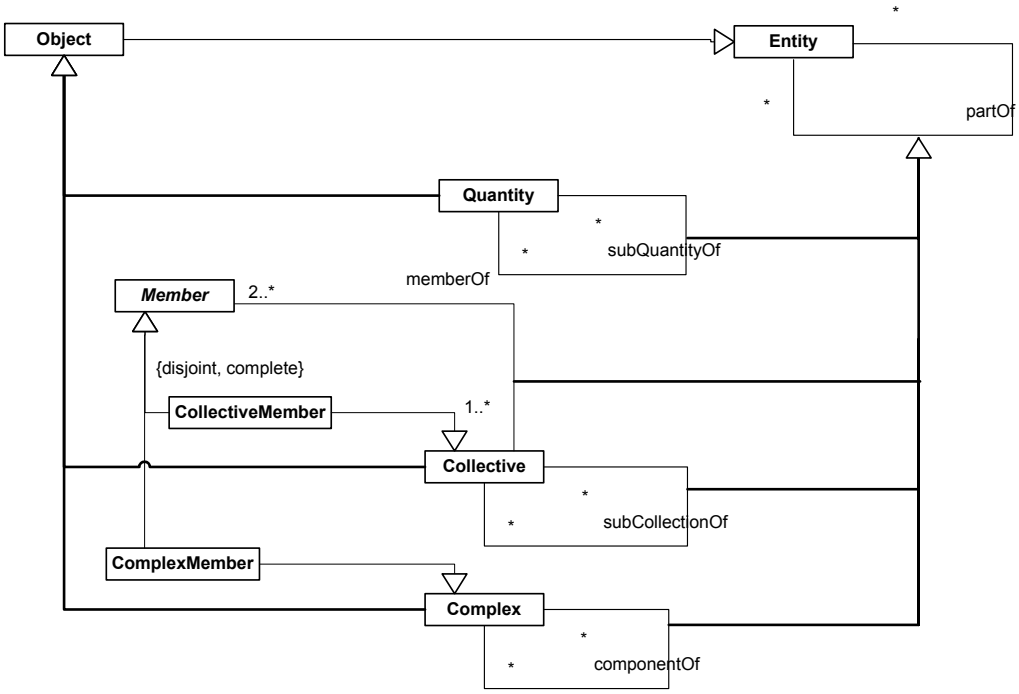
Trata-se da Relação Meronímica (Meronymic relation) mais geral, gênero em que Componente de (Component of), Membro de (Member of), Subcoleção de (Subcollection of) e Subquantidade de (Subquantity of) são espécies. *Part of* é uma relação irreflexiva, anti-simétrica e não-transitiva – uma vez que duas de suas subclasses (Subquantity of e Subcollection of) são transitivas, uma é intransitiva (Member of) e

[16] A expressão *object universal* aparece em Guizzardi e Wagner (2010), enquanto *substantial universal* é a expressão original presente em Guizzardi (2005).

outra é não-transitiva (Component of) – que obedece ao princípio Suplementação Fraca (Weak Supplementation) (GUIZZARDI, 2005, p. 341-342). A Figura 57 ilustra a relação de *Part of* com suas subclasses. Em Guizzardi (2011), encontra-se uma sumarização dessas relações:

> (a) *subquantity-quantity*: modeling parts of an amount of matter (e.g., alcohol-wine, gin-Martini, Chocolate-Toddy); (b) *member-collective*: modeling a collective entity in which all parts play an equal role w.r.t. the whole (e.g., tree–forest, card–deck, lion-pack); (c) *subcollective-collective*: modeling a relation between a collective and the subcollectives that provide further structure to the former (e.g., the north part of the black forest-black forest, the underage children of John-the children on John); (d) *component-functional complex*: modeling an entity in which all parts play a different role w.r.t. the whole, thus, contributing to the functionality of the latter (e.g., heart-circulatory system, engine-car).

Figura 57 – Diferentes tipos de relações de meronímia



Fonte: Guizzardi (2005, p. 341)

Ver também: Mereology, Meronymic relation, Functional Complex

Particular

Os particulares – indivíduos ou individuais, do inglês *Individual*, conforme aparece em trabalhos mais recentes, como em Guizzardi (2006) e em Guizzardi e Zamborlini (2014) – são entidades que existem na realidade e possuem uma identidade única (GUIZZARDI; WAGNER, 2010).

Ver também: Universal, Urelement

Perdurant (Perdurante)

Perdurants – que em português podem ser chamados "ocorrentes" (CARBONERA, 2012, p. 42) – são Individuais (Individual) compostos por partes temporais, eles *acontecem no tempo* no sentido de que se estendem pelo tempo e acumulam partes temporais. Exemplos são uma corrida, uma conversa, um jogo de futebol, a Segunda Gerra Mundial (GUIZZARDI, 2005, p. 211). *"Whenever an event is present, it is not the case that all its temporal parts are present"* (GUIZZARDI; WAGNER, 2013, p. 1337).

Nota: A palavra em português "perdurante" é usada como termo técnico na filosofia, como se encontra em evidência Batista (2013).

Ver também: Endurant

Phase (Fase)

Phases são Sortais anti-rígidos (Anti Rigid Sortal) cujas instâncias sempre são especializações de um Sortal Rígido (Rigid Sortal). Diferenciam-se de Papéis (Role) devido a condição de especialização, que no caso das Fases refere-se à uma condição intrínseca, ou seja, uma Fase é um tipo que um objeto instancia em um certo período de tempo devido a características próprias, intrínsecas (GUIZZARDI, 2005, p. 104). Por exemplo, uma `Pessoa` possui as fases `Criança` e `Adulto`, que podem ser estabelecidas somente a partir de propriedades da própria entidade `Pessoa` (como a idade, por exemplo). Em Guizzardi e Wagner (2010), encontra-se a seguinte explicação:

> *Phases constitute possible stages in the history of a particular. Examples include: (a) Alive and Deceased: as possible stages of a Person; (b) Catterpillar and Butterfly of a Lepidopteran; (c) Town and Metropolis of a City; (d) Boy, Male Teenager and Adult Male of a Male Person.*

Ver também: Phased Sortal

Phased Sortal (Sortal em fases)

Os *Phased Sortals* são Universais (Universal) do tipo anti-rígidos (Anti Rigid Sortal). Subdividem-se em Papel (Role) e Fase (Phase). *Phased Sortals* são opostos à categoria dos Sortais Rígidos (Rigid Sortal). Um exemplo que destaca essa diferença está presente em Guizzardi e Wagner (2010):

> *The prototypical example highlighting the modal distinction between these two categories is the difference between the Kind Person and the Phase-Sortals Student and Adolescent instantiated by the particular John in a given circumstance. Whilst John can cease to*

> *be a Student and Adolescent (and there were circumstances in which John was not one), he cannot cease to be a Person. In other words, while the instantiation of the phase-sortals Student and Adolescent has no impact on the identity of a particular, if a particular ceases to instantiate the universal Person, then she ceases to exist as the same particular.*

Principle of application (Princípio de aplicação)

O Princípio de aplicação é aquele em que termos gerais se aplicam a particulares, ou seja, trata-se do princípio que pode ser usado para julgar se algo é instância de um determinado tipo. Por exemplo, julgar se algo é uma Pessoa, um Cachorro, uma Cadeira ou um Estudante (GUIZZARDI, 2005, p. 98). Todos os Universais (Universal) carregam um Princípio de aplicação.

Principle of identity (Princípio de identidade)

O Princípio de identidade é aquele em que se julga que dois particulares são o mesmo, ou seja, em quais situações a relação de identidade é o caso (GUIZZARDI, 2005, p. 98). Apenas Sortais (Sortal) carregam o Princípio de identidade.

Proper Overlapping (Sobreposição Própria)

Simétrico aos conceitos de Mereologia Base (Ground Mereology) e de Parte Própria (Proper part), simbolizado por $\leq$, e considerando a definição de $\bullet$ presente na entrada Overlapping, tem-se o conceito de Sobreposição Própria, simbolizado por $\circ$, que é definido como (GUIZZARDI, 2005, p. 144):

1. $(x \circ y) \coloneqq (x \bullet y) \wedge \neg(x \leq y) \wedge \neg(y \leq x)$

Ver também: Proper part, Overlapping, Ground Mereology,Disjointness

Proper part (Parte Própria)

Conforme Guizzardi (2005, p. 143-144), no contexto de teorias mereológicas (Mereology), a relação de Parte Própria, simbolizada por $<$ é definida como, em que $\leq$ é a relação de parte definida na entrada Mereologia Base (Ground Mereology):

$$(x < y) \coloneqq (x \leq y) \wedge \neg(y \leq x)$$

A Parte Própria é uma relação em que não ocorre de um x ser parte de si mesmo. Trata-se de uma relação de ordem parcial estrita, ou seja, em que a relação é assimétrica e transitiva, da qual a irreflexibilidade decorre (GUIZZARDI, 2011):

1. $\forall x \neg(x < x)$
2. $\forall x, y\ (x < y) \rightarrow \neg(y < x)$

3. $\forall x, y, z\ (x < y) \wedge (y < z) \rightarrow (x < z)$

De acordo com Guizzardi (2005, p. 145):

> *In summary, if we interpret the concept of* being a part *as that of* being a proper part*, we have that for all individuals x, y, z:*
> *(i) x is not a part of itself;*
> *(ii) if x is part of y then y is not part of x;*
> *(iii) if x is part of y and y is part of z then x is part of z.*
> *Either the* proper part of *or* improper part of *relations can be taken as primitive, and the choice is more a matter of convenience than anything else. While the former is a more natural concept, the latter is algebraically more convenient.*

Nota: Em Guizzardi (2011) menciona-se que os axiomas apresentados nesta entrada são conhecidos na literatura pelo nome de *Ground Mereology*, embora em Guizzardi (2005, p. 143) a *Ground Mereology* refira-se aos axiomas de ordem parcial apresentados na entrada Mereologia Base (Ground Mereology), e não ao de ordem parcial estrita apresentados nesta entrada de glossário.

Ver também: Ground Mereology, Weak Supplementation, Minimal Mereology, Proper Overlapping, Overlapping

Q

Qua Individual

Um *Qua Individual* é um indivíduo que possui todas as instâncias de Modos Externamente Dependentes (Externally Dependent Mode) – Modos Universais (Mode Universal) externalmente dependentes, ou Momento Externamente Dependentes (Externally Dependent Moment) – de uma entidade que compartilha as mesmas dependências e a mesma base de fundamentos em certa relação (BENEVIDES, 2010, p. 43). Trata-se de um Particular que é portador de todos os Momento Externamente Dependentes que compartilham a mesma dependência externa e a mesma Fundação (Foundation) (GUIZZARDI; WAGNER, 2010). Como Momentos (Moment), os *Qua Individual* são inerentes a exatamente um outro Endurante (Endurant). Em Guizzardi (2005, p. 239-240) apresenta-se a ideia de *Qua Individual* como:

> *Qua individuals are, thus, treated here as a special type of complex* externally dependent modes*. Intuitively, a qua individual is "the way an object participates in a certain relation" (LOEBE, 2003), and the name comes from considering an individual only w.r.t. certain aspects (e.g., John qua student; Mary qua musician)* (MASOLO et al., 2004).

Por fim, Guizzardi (2005, p. 242) sumariza: *"Qua individuals are (potentially complex) externally dependent modes exemplifying all the properties that an individual has in the scope of a certain material relation".*

Ver também: Relator

Termo alternativo: Role Instance, Relator

Quale

Baseado em Gärdenfors (2000) e em Masolo et al. (2003), as obras Guizzardi (2005, p. 223) e Guizzardi e Wagner (2010) definem um *Quale* como a percepção ou concepção de um Momento Intrínseco Universal (Intrinsic Moment Universal) representado como um ponto em um Domínio de Qualidade (Quality Domain). O conceito é introduzido em Guizzardi (2005, p. 215):

> *In conformance with DOLCE (MASOLO et al., 2003), we distinguish between the color of a particular apple (its quality) and its 'value' (e.g., a particular shade of red). The latter is named quale, and describes the position of an individual quality within a certain quality dimension.*

De forma mais precisa, a seguinte definição é apresentada em Guizzardi (2005, p. 227):

> *A point in a n-dimensional quality domain can be represented as a vector* $v =< x_1 \dots x_n >$ *where each* x_i *represents each of the integral dimensions that constitute the domain. A multidimensional quale is therefore the vector representing the several quality dimensions that are mutually dependent in a quality domain.*

Conforme Guizzardi (2006):

> *we distinguish between the color of a particular apple (its quality) and its 'value' (e.g., a particular shade of red). The latter is named* quale*, and describes a projection of an individual quality into a certain conceptual space*

Em seu estudo dedicado às Qualidades (Quality), Albuquerque e Guizzardi (2013) apresentam a seguinte explicação sobre *Quale*:

> *The percept of a quality is referred in the literature as quale (GUIZZARDI, 2005; PROBST, 2008; MASOLO et al., 2003; HERRE et al., 2004). Originally, the term quale refers to the percept or mental state that is evoked in cognitive agents while observing some particular quality. Thus, qualia are by nature intrinsic to (the minds of) cognitive agents and therefore cannot be shared. In order to be communicated qualia needs to be approximated and then referred by symbols like "1m", "40oC" or "Crimson Red". As perception is only possible on the magnitudes of substantial qualities like color and height, we use the term quale also to denote the "conceived value" of abstract qualities like currency value and name.*

Nota: Em trabalhos recentes, a expressão *Quale* parece ser substituída por Valor de Qualidade (Quality Value).

Ver também: Quale of a quality, Conceptual Space, Quality Region, Quality

Quale of a quality (*Quale* de uma qualidade)

Trata-se do predicado $ql(x, y)$ que representa uma Relação Formal (Formal Relation) entre uma Qualidade (Quality) y e um Quale x (GUIZZARDI, 2005, p. 233):

$$\forall x, y \; ql(x, y) \rightarrow quale(x) \wedge quality(y)$$

Ver também: Attribute Function

Qualia

Ver: Quale

Quality (Qualidade)

Uma Qualidade é um Momento individual (Moment individual) que instancia um Universal de Qualidade (Quality Universal) e que possui um Quale como valor. Qualidades são, essencialmente, tipos de propriedades que um tipo e seus indivíduos possuem. "Trata-se de uma tentativa de modelar a relação entre Momentos Intrínsecos (Intrinsic Moment) e sua representação em estruturas cognitivas" (GUIZZARDI; WAGNER, 2010). Segundo Albuquerque e Guizzardi (2013):

> *Qualities are objectification of object properties which can be directly evaluated (projected) into a certain value space (depending on the type of quality, i.e., on the quality universal). Examples of qualities include mass, height, electric charge and color. An important aspect of qualities is that they can (as much as objects) endur maintaining their identity even through qualitative changes. For example, when state that the color of an apple is changing, we do not mean that red is changing. From a cognitive perspective, we countenance the existence of an entity which is existentially dependent on that apple and which can qualitatively change from greenish to reddish and then to brownish, while maintaining its identity.*

Ver também: Quality Dimension, Quality Domain, Conceptual Space, Quality Universal

Quality Dimension (Dimensão de Qualidade)

Baseado em Gärdenfors (2000), Guizzardi (2005) estabelece que Dimensões de Qualidade são compostas pelo conjunto de valores que uma Qualidade (Quality) pode possuir, além das relações formais entre esses valores. Tratam-se de partes de Domínios de Qualidades (Quality Domain). Conforme Albuquerque e Guizzardi (2013), Dimensões de Qualidade representam a categoria mais elementar, uni-dimensional, de Quality Structures. Em OntoUML, Dimensões de Qualidade são representados por *Datatype* simples (Simple DataType).

Quality Domain (Domínio de Qualidade)

Um Domínio de Qualidade é um conjunto de Dimensões de Qualidade (Quality Dimension) compostos de tal forma que o valor de uma dimensão não pode ser representada sem o valor das outras:

> *Quality domains are composed of integral dimensions. This means that the value of one dimension cannot be represented without representing the values of others. By representing the color quality domain in terms of a structured data type we can reinforce (via its constructor method) that its tuples will always have values for all the integral dimensions. Moreover, the representation of a quality domain should account not only for its quality dimensions but also for the constraints on the relation between them imposed by its structure. To mention another example, consider the Gregorian calendar as a quality domain (composed of the linear quality dimensions days, months and years) in which date qualities can be represented. The value of one dimension constrains the value of the others in a way that, for example, the points [31-April-2004] and [29-February-2003] do not belong to this quality structure. Once more, constraints represented on the constructor method of a data type can be used to restrict the possible tuples that can be instantiated* (GUIZZARDI, 2005, p. 250).

Quality Particular (Particular de Qualidade)

Ver: Quality

Quality Region (Região de Qualidade)

Trata-se de uma interpretação fenomenológica da ideia de Quale, uma vez que assume-se a impossibilidade de conhecer direta e absolutamente o valor de uma qualidade em espaço de valores possíveis, devido às limitações sensoriais inerentes a qualquer sujeito cognoscente. O que se tem, no entanto, são aproximações dos valores que são pressupostos como absolutos na natureza. Conforme Albuquerque e Guizzardi (2013):

> [...] *the qualia of particular qualities cannot be directly shared; they need to be approximated and referred by lexical symbols. The quality structures conceived as the value space for quality universals are composed by a number of inner regions, which the qualia of particular qualities are located at by the process of approximation. We introduce the category Quality Region in UFO to denote the regions that approximate qualia for the purpose of reference and communication. In this way, qualia are explicitly separated from what is being referred and communicated about the particular qualities (i.e., the quality regions which approximate the qualia). At the same time, the meaning of lexical elements like "1,86", "small" or "John" is grounded by the quality regions they denote.*

Nota: Conforme a subseção 4.4.4, o conceito de Região de Qualidade não é diretamente incorporado neste trabalho, e é alvo de sugestão de trabalho futuro registrado na seção 11.3.

Quality Space (Espaço de qualidade)

O conceito de Espaço de qualidade é usado na ontologia de fundamentação DOLCE (MASOLO et al., 2003) como analogia a ideia de Espaço Conceitual de Gärdenfors (2000). Porém, de acordo com Probst (2007, p. 47), a DOLCE define Espaço de qualidade como *soma mereológicas* de todas as Quality Region em que as qualidades de certo tipo estão localizadas. Já as Espaços Conceituais tratam-se de pontos representados por regiões, que são compostos de modo a formar espaços topológicos. A UFO conforme apresentada em Guizzardi (2005) utiliza o conceito de Gärdenfors (2000). Já conforme apresentada em Albuquerque e Guizzardi (2013), o conceito de Espaço de qualidade é desenvolvido de modo mais evidente.

Ver também: Conceptual Space

Quality Structure (Estrutura de Qualidade)

O termo *Quality Structure* é definido (GUIZZARDI, 2005, p. 223) com base no conceito de Espaço Conceitual (Conceptual Space), e é usado para se referir a Dimensões de Qualidade (Quality Dimension) e a Domínios de Qualidades (Quality Domain). Uma *"quality structure is the ontological interpretation of the UML DataType construct"*. Em OntoUML, Estruturas de Qualidade são representadas por *Datatype* estruturado (Structured DataType) (GUIZZARDI, 2005, p. 326). Na obra citada, uma Estrutura de Qualidade é sempre relacionada a um único Universal de Qualidade (Quality Universal) (GUIZZARDI, 2005, p. 224) por meio de uma Relação de Associação (Association Relation). Por exemplo, Guizzardi e Wagner (2010) colocam que *"a quality structure associated with the universal Weight cannot be associated with the universal Color."*
De forma mais didática, a seguinte explicação é encontrada em Albuquerque e Guizzardi (2013):

> *The definition of quality structures in UFO was motivated by the theory of Conceptual Spaces introduced by Gärdenfors (GÄRDENFORS, 2000), which offers a framework to represent knowledge about perceivable or conceivable qualities such as color, height, temperature and currency value. The cornerstone of Conceptual Space theory is the notion of quality dimension. According to Gärdenfors, conceptual spaces are sets of quality dimensions that can be integral or separable from each other. Quality dimensions are integral w.r.t. others when a particular quality cannot be associated to a region in one quality dimension without associating with a region in the other. Quality domains then can be defined as a "set of integral dimensions separable from all the other" (GÄRDENFORS, 2000), therefore constituting geometrical structures. Mass and temperature are quality universals that can be structured by separable quality dimensions, while color (HSB) and volume can be structured by three-dimensional quality domains. UFO introduces the category quality structure as a general category to the categories quality dimension and quality domain.*

Em Albuquerque e Guizzardi (2013), porém, Guizzardi revisa sua definição de *Quality Structure* e apresenta que ele deve estar associado a *ao menos uma* Universal de Qualidade (Quality Universal), assim com o Universal de Qualidade deve estar associado a *ao menos um Quality Structure* pela Relação de Associação (Association Relation).

Ver também: Indirect Quality

Quality Universal (Universal de Qualidade)

Uma Universal de Qualidade é um Momento Intrínseco Universal (Intrinsic Moment Universal) associada a Estruturas de Qualidade (Quality Structure) por meio de uma Relação de Associação (Association Relation) (GUIZZARDI, 2005, p. 222-224), (GUIZZARDI; WAGNER, 2010). "Para uma mesma Estrutura de Qualidade pode haver potencialmente muitos Espaços Conceituais (Conceptual Space) associados a ela, dos quais um é adotado para atender os objetivos de uma conceituação específica" (GUIZZARDI, 2005, p. 226). Qualidades (Quality) instanciam Universais de Qualidade.
Albuquerque e Guizzardi (2013) apresenta uma explanação mais ampla do conceito, que entende-se pertinente transcrever:

> *Quality universals are always associated with value spaces or quality structures that can be understood as the set of all possible regions that delimits the space of values that can be associated to a particular quality universal. Moreover, quality structures can provide ordering for these values, allowing the comparison of qualities associated with the same or equivalent quality structures. For example the height quality universal can be associated with a quality structure isomorphic to the positive halfline of real numbers, thus, defining the set of all possible values for particular heights. Making qualities comparable is a requirement for the establishment of formal relations that are based in the entities qualities like* older-than(John, Peter), heavier-than(John, Mary) *and determining equivalence between qualities like* same-as(color1, color2).

Conforme Guizzardi e Zamborlini (2014), os *Quality types* são entidades rígidas (Rigid).

Ver também: Quality, Complex Quality Universal, Complex Quality, Indirect Quality, Conceptual Space, Characterizing Universal

Quality Value (Valor de Qualidade)

Valor de Qualidade aparece em Guizzardi e Zamborlini (2014, p. 6) como sinônimo de Quale: *"the perception or conception of an intrinsic aspect can be represented as a point in a quality domain. This point is named here a quality value."*

Quantity (Quantidade)

Ver: Quantity Universal

Quantity Universal (Universal de Quantidade)

Quantidades são partes *maximalmente auto-conectadas genericamente* dependentes do todo da qual são constituintes. Tratam-se de Universais de Sortais (Sortal Universal) que representam porções de matéria que possuem identidade e unicidade determinadas. "É importante salientar que Quantidades referem-se a porções determinadas de matéria, de modo que podem ser compreendidos como objetos genuínos (porções objetivadas de matéria)" (CARBONERA, 2012, p. 46). Em Guizzardi (2005, p. 179) encontra-se a seguinte explicação:

> *[...] a quantity of K [...] is an instance of a genuine sortal universal, i.e., it has definite individuation, identity and counting principles. Moreover, it is not homeomerous, however it can still be composed of other quantities K' in the same sense of quantity [...]. All its parts are also essential, and it does not contain the infinite regress problems mentioned for the first case. Nonetheless, [...] the dependence relation between a quantity and its container is a generic not a specific one. For this reason we can state that for the same maximally self-connected quantity of wine, there can be several "container phases".*

Conforme Guizzardi (2010a), Quantidades (Quantity) são *"amounts of matter, masses"*, que no contexto de serem adequadamente representados em modelos conceituais, suas instâncias obedecem um Princípio de identidade (Principle of identity). De acordo com Guizzardi (2011), quantidades são Homeomeros.

Nota: A entrada Functional Complex apresenta comparação dos Complexos (Complex) com as Quantidades (Quantity) e com Coletivos (Collective).

Ver também: Functional Complex, Kind, Collective, Homeomeros

R

Redefinition relation (Relação de Redefinição)

A função da Relação de Redefinição é estabelecer uma nova definição para um ou mais terminais (Terminal) de outra associação de modo específico a determinado contexto. No caso, a nova definição trata-se de algo significativo no domínio da ontologia descrita. Um ou ambos os terminais de uma associação podem ser redefinidos e redefinições sempre envolvem duas associações.

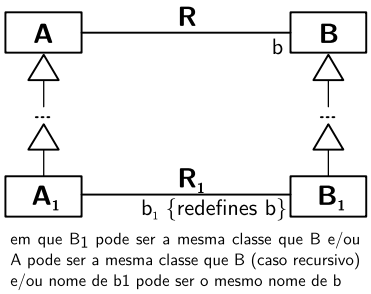
A Figura 58 apresenta o esquema geral da relação de redefinição. Na figura, B_1 pode ser a mesma classe que B e/ou A pode ser a mesma classe que B e/ou nome de b_1 pode ser o mesmo nome de b. Conforme Costal, Gómez e Guizzardi (2011a):

> *in the case of redefinition the type A1 in the opposite end of the redefining association end of association R1 must be a subclass*

> *of type A. This is due to the fact that the difference in ways that instances of A participate in association R is determined by differences in intrinsic properties of these instances. Thus, if we have at least of property p that some instances of A must have in order to participate in R in a specific way (e.g., R1) then we can define the type A1 such that A1(x) iff A(x) and (x possesses p). Conversely, notice that without a single property that differentiates particular subtypes of A then we cannot explain why they participate in association R in different manners and subject to different constraints. After all, in that case, all instance of A would have exactly the same properties including the specific relator type that binds then to instances of B.*

Essa análise gera o seguinte postulado pragmático: uma relação de redefinição deve ser estabelecida entre duas relações materiais R_2 e R_1 ($R_2 redefines R_1$) se: (i) ambas as relações são derivadas do mesmo Relator RR; (ii) existe um tipo A_i conectado a um dos terminais de R_2 tal que A_1 é uma especialização de A ($A \leftarrowtriangle A_1$) e A está conectado ao terminal de R_1 equivalente ao de A_1 (COSTAL; GÓMEZ; GUIZZARDI, 2011a, postulate 3).

Figura 58 – Estrutura da Relação de Redefinição



Fonte: Costal, Gómez e Guizzardi (2011a)

Ver também: Subsetting relation, Specialization relation (for relations)

Reference Ontology (Ontologia de Referência)

Trata-se de uma representação concreta de uma Conceituação de domínio (Domain Conceptualization) (GUIZZARDI; WAGNER, 2010).

Ver também: Domain Ontology

Relation (Relação)

Relações são Universais (Universal) poliádicos, ou seja, que se referem a mais de uma entidade, como `Casamento` (no sentido de que são necessários ao menos duas entidades para que uma relação como essa possa existir). As relações se dividem em Relação Formal (Formal Relation) e Relação Material (Material Relation).

Relational dependence (Dependência Relacional)

Trata-se da meta-propriedade dos Papéis (Role). Guizzardi (2007a) define formalmente um conceito de dependência relacional R como:

> A type T is relationally dependent on another type P via relation R iff for every instance x of T there is an instance y of P such that x and y are related via R: $R(T, P, R) \coloneqq (\forall x T(x) \rightarrow \exists y P(y) \wedge R(x, y))$

Relational Trope (*Trope* Relacional)

Expressão usada por Guizzardi e Wagner (2008) para se referir a Relator.

Ver também: Trope

Relator

Ver: Relator Universal

Relator Individual

Uma instância de um Universal de *Relator* (Relator Universal).

Ver também: Instance

Relator Universal (Universal de *Relator*)

Os *relators* têm a capacidade de conectar entidades, *"for example, founds a relator that connects airports, an enrollment is a relator that connects a student with an educational institution"* (GUIZZARDI, 2005, p. 236-237). Tratam-se de agregações de *mais de um* Qua Individual que compartilham as mesmas características (GUIZZARDI, 2005, p. 240) – que compartilham a mesma Fundação (Foundation) (GUIZZARDI, 2006; GUIZZARDI; WAGNER, 2010). De acordo com Guizzardi e Wagner (2008):

> *The notion of relator (Relational Tropes) is supported by several works in the philosophical literature (MULLIGAN; SMITH, 1986; HELLER; HERRE, 2004) and, the position advocated here is that they play an important role in answering questions of the sort: what does it mean to say that John is married to Mary? Why is it true to say that Bill works for Company X but not for Company Y?*

Conforme Guizzardi (2009), *"the entity which is the sum of all qua individuals that share the same foundation is a relator."* Os *relators* são casos especiais de Momento (Moment). Eles são opostos aos Momentos Intrínsecos Universais (Intrinsic Moment Universal), que se relacionam somente a um Particular (GUIZZARDI; WAGNER, 2010), ou seja, são casos de Momentos extrínsecos (Extrinsic Moment). O autor explica o conceito de *relator* do seguinte modo:

Seja x, y e z três distintos indivíduos, de modo que (a) x é um *relator*; (b) y é um

Qua Individual e y é parte de x; (c) y é inerente (Inherence) a z. Nesse caso, diz-se que x media z, simbolizado por m. Formalmente:

$$\forall x, y, m(x, y) \rightarrow relator(x) \wedge Endurant(y)$$

e

$$\forall x\ relator(x) \rightarrow \forall y(m(x, y) \leftrightarrow (\exists z\ quaIndividual(z) \wedge (z < x) \wedge i(z, y)))$$

Adicionalmente, requere-se que um relator medie ao menos dois Universais Substanciais (Substantial Universal) distintos:

$$\forall x\ relator(x) \rightarrow \exists y, z(y \neq z \wedge m(x, y) \wedge m(x, z))$$

Um *relator*, sendo um Momento, deve pertencer (Inherence) a um único indivíduo, ou seja, deve possuir um portador. Dessa forma, Guizzardi (2005, p. 240) define que o portador de um *relator* r é a Soma Mereológica (Mereological Sum) dos indivíduos mediados por r:

$$\forall x\ relator(x) \rightarrow (\beta(x) = \sigma ym(x, y))$$

No caso, o portador do *relator* x é o indivíduo y composto por todas as entidades mediadas por x. Conforme a obra citada:

> *Relators are genuine integral wholes according to the criteria posed in section 5.4*[17]*, since the relation of common foundation can be used as a suitable characterizing relation. A similar argument can be made in favour of the bearers of relators [...]*[18] *in terms of the relation of common mediation.*

Relator Universals são a base para Relações Materiais (Material Relation).

Nota: É importante destacar que, com base em Guizzardi (2005) e em Guizzardi (2006), no âmbito do estudo dos Relators existem 2 Whole envolvidos:

- o Relator em si é um Whole composto dos Qua Individuals;
- o portador do Relator é um Whole composto pelos indivíduos mediados pelo relator (ou seja, os indivíduos aos quais os Qua Individuals se referem). No caso desse Whole portador, o autor não menciona como ele deve ser identificado ou mesmo representado em modelos conceituais.

Ver também: Foundation, Endurant, Mediation, Qua Individual, Extrinsic Moment, Intrinsic Moment

Rigid (Rígido)

Ver: Rigidity

[17] Na seção indicada da obra em tela, o autor desenvolve os conceitos indicados na entrada Mereology.
[18] Na obra, no local onde colocamos reticências, o autor indica a fórmula presente imediatamente antes da citação.

Rigid Mixin (*Mixin* Rígido)

Trata-se de um universal Rígido (Rigid) que não é um Sortal. Representa uma abstração de propriedades rígidas comuns a dois ou mais sortais que especializam Espécies (Kind) distintos (GUIZZARDI, 2005, p. 112-113). O exemplo do autor é uma categoria chamada `Entidade Inteligente` que é uma abstração de `Pessoa` e `Agente artificial`.

Ver também: Mixin Universal, Mixin, Role Mixin, Category

Rigid Sortal (Sortal Rígido)

Sortais Rígidos são Sortais (Sortal) com a característica de Rigidez (Rigidity). Subdividem-se em Sortal de Substância (Substance Sortal) e em Subespécie (Subkind). *Rigid Sortals* são opostos à categoria dos Sortais em fases (Phased Sortal)[19]. Um Universal anti rígido não pode ter, como sua super classe, um Universal anti-rígido (Anti Rigid Universal) (GUIZZARDI; WAGNER, 2010).

Rigid Universal (Universal Rígido)

Ver: Rigidity

Rigidity (Rigidez)

Um tipo de objeto O é rígido se todas as instâncias de O são necessariamente instâncias de O em todas as modalidades consideradas. Um tipo que satisfaça essa condição também é chamado Universal Rígido (Rigid Universal). *"In other words, if x instantiates O in some possible world, then x must instantiate O in all some possible worlds, in which x exists"* (GUIZZARDI; WAGNER, 2012). Conforme Guizzardi (2007a), um conceito R de "rigidez" é definido formalmente como:

> *A type T is rigid if for every instance x of T, x is necessarily (in the modal sense) an instance of T. In other words, if x instantiates T in a given world w, then x must instantiate T in every possible world w':* $R(T) := \Box(\forall x T(x) \rightarrow \Box(T(x)))$

Definição semelhante é a presente em Guizzardi e Wagner (2010):

> *A universal U is rigid if for every instance x of U, x is necessarily (in the modal sense) an instance of U. In other words, if x instantiates U in a given world w, then x must instantiate U in every possible world w'.*

Em Guizzardi e Zamborlini (2014), a formulação de rigidez é mais precisa, uma vez que considera os casos em que determinado tipo T é rígido mesmo quando não

[19] Veja na definição de Phased Sortal uma explicação mais detalhada sobre a distinção entre esses e os Sortal Rígido (Rigid Sortal).

existem instâncias x de T:

$$Rigid(T) \coloneqq \forall x\ (x :: T \rightarrow \Box(\mathcal{E}(x) \rightarrow x :: T))$$

Nota: Entende-se que o conceito de Rigidez é melhor representado nesta formulação, que deixa explícito a modalidade existencial, e que poderia ser lido como 'para todo x, se x é um particular do tipo T *em algum* mundo possível acessível, então *em todo* mundo possível acessível em que x está presente, ele está presente como instância de T':

$$Rigid(T) \coloneqq \forall x\ (\Diamond(x :: T) \rightarrow \Box(\mathcal{E}(x) \rightarrow x :: T))$$

Nota: Ver observações sobre o predicado $\mathcal{E}$ em Existential Dependence.

Role (Papel)

Papéis são Sortais anti-rígidos (Anti Rigid Sortal) cujas instâncias são sempre especializações de um Sortal Rígido (Rigid Sortal). Diferenciam-se de Fases (Phase) devido a condição de especialização, que no caso do *Roles* refere-se à uma condição relacional (extrínseca) de dependência (Relational dependence), ou seja, relações em um certo contexto, mediadas por um Relator (GUIZZARDI, 2005, p. 294), ou participantes de um evento. Por exemplo uma `Pessoa` pode ter o papel de `Estudante` quando numa relação de `Matrícula` em uma `Universidade`.
Nota de tradução: Uma tradução alternativa aceitável poderia ser "Função".

Ver também: Phased Sortal, Mediation

Role Instance (Instância de Papel)

Termo alternativo de Qua Individual, conforme é proposto por Wieringa, Jonge e Spruit (1995) (GUIZZARDI, 2009).

Role Mixin (*Mixin* de Papel)

Na hierarquia da UFO, *Role Mixin* é um nome abreviado para entidades do tipo *Mixin* anti-rígido (Anti Rigid Mixin).

S

Semi Rigid Mixin (*Mixin* Semi-rígido)

Os *Mixins* Semi-rígidos representam o conjunto de propriedades que são essenciais para algumas de suas instâncias e acidentais para outras. Na hierarquia de conceitos da UFO, os Mixins denotam o estereótipo dos Semi Rigid Mixins.

Set (Conjunto)

Conforme Guizzardi (2005, p. 209), a ideia de *conjunto* na UFO é tratada de forma intuitiva e não se refere a nenhuma Teoria de Conjuntos particular. Na definição da

Ontologia, operações como pertence ($\in$), inclusão própria ($\subset$), união ($\cup$), intersecção ($\cap$) e diferença ($\setminus$) são utilizadas sem formalizações específicas. Ontologicamente, assume-se que conjuntos são distintos de Ur-Elementos (Urelement) e que o mundo pode ser fechado em construções conjuntistas. Conjuntos podem ser puros, quando seus membros são outros conjuntos, ou impuros, quando Ur-Elementos podem funcionar como membros do conjunto.

Ver também: Urelement

Shareable (Compartilhável)

Uma parte é *compartilhável* se ela não é uma Parte exclusiva (Exclusive part).

Ver também: Non-Shareable, Component of, Member of, Subcollection of, Subquantity of, Mereology

Simple DataType (*Datatype* simples)

Entidade da OntoUML que representa uma Dimensão de Qualidade (Quality Dimension).

Simple Quality (Qualidade Simples)

Uma Qualidade Simples é uma Qualidade (Quality) que não pertence a outras Qualidades. Na Figura 56, as Qualidades h, s e b são *Simple Qualities*. Formalmente, considerando o conceito de Inerência (Inherence) i, Guizzardi (2005, p. 232) apresenta:

$$simpleQuality(x) \coloneqq quality(x) \wedge \neg\exists y\ i(y, x)$$

Ver também: Indirect Quality, Complex Quality

Simple Quality Universal (Universal de Qualidade Simples)

Trata-se do Universal instanciado pelas Qualidades Simples (Simple Quality). *Simple Qualities Universals* sempre são associadas a Dimensões de Qualidade (Quality Dimension) e vice-versa (GUIZZARDI, 2005, p. 233).

Sortal

Como casos de Universal, os Sortais carregam o Princípio de aplicação (Principle of application), porém, apenas Sortais possuem o Princípio de identidade (Principle of identity), que os caracterizam. Conforme Guizzardi e Zamborlini (2014): *"Sortals are types that are able to carry a uniform principle of identity, persistence and counting for all its instances"*. Na categoria dos Sortais, é possível realizar ao menos duas

distinções referentes aos conceitos de Rigidez (Rigidity) e de Anti-rigidez (Anti Rigidity) (GUIZZARDI; WAGNER, 2010).
Nota de tradução: Sortal poderia ser traduzido para o português como "Contável", "Distinguível", "Ordenável" ou "Substantivo". Porém, opta-se por manter o termo original em inglês e, dessa forma, manter a característica técnica em que o conceito é usado. No entanto, o estrangeirismo é adotado no plural da palavra, que é definido como "sortais".

Ver também: Characterizing Universal

Sortal Universal (Universal de Sortal)

Ver: Sortal

Specialization relation (for relations) (Relação de especialização de relações)

A Relação de especialização de relações é uma relação taxonômica entre uma associação mais específica de outra mais geral. A relação específica herda as características da geral. Em contraste com a Relação de subconjunto (Subsetting relation) (Quadro 29), a relação de especialização entre relações é um constructo definido para as relações em si, e não para os terminais (COSTAL; GÓMEZ; GUIZZARDI, 2011a). Uma relação de especialização deve ser definida entre duas Relações Materiais (Material Relation) R_2 e R_1 ($R_1 \leftarrowtriangle R_2$) se essas duas relações são derivadas a partir de dois Relators RR_1 e RR_2, tal que RR_2 especializa RR_1 ($RR_1 \leftarrowtriangle RR_2$) (COSTAL; GÓMEZ; GUIZZARDI, 2011a, postulate 2).

Ver também: Subsetting relation, Redefinition relation

Strong Supplementation (Suplementação Forte)

Trata-se de um axioma que afirma que se um indivíduo y não é uma parte de outro indivíduo x, então há uma parte de y que não se sobrepõe com x (GUIZZARDI, 2005, p. 146). Formalmente, em que $\leq$ é definido em Mereologia Base (Ground Mereology) e $\int$ em Disjunção (Disjointness):

$$\forall x, y \ \neg(y \leq x) \rightarrow \exists z(z \leq y) \wedge (z \int x)$$

Strong Supplementation, juntamente com Mereologia Base (Ground Mereology), é usado na definição de Mereologia Extensional (Extensional Mereology).

Ver também: Weak Supplementation, Extensional Mereology, Minimal Mereology

Structuration Relation (Relação de estruturação)

Ver: Association Relation

Structured DataType (*Datatype* estruturado)

Entidade da OntoUML que representa uma Estrutura de Qualidade (Quality Structure).

Subcollection of (Subcoleção de)

Estereótipo usado na OntoUML para representar a Relação Subcoleção-Coleção (Subcollection-Collection Relation).

Ver também: Member of, Component of, Subquantity of

Subcollection-Collection Relation (Relação Subcoleção-Coleção)

Conforme Vieu e Aurnague (2007, p. 312), *"Subcollection-collection is the relation that holds between two plural entities – or collections constituted by such plural entities, all the atoms of the first being also atoms of the second."* Guizzardi (2005, p. 186-187) e Guizzardi (2010b), Guizzardi (2011) utilizaram a noção de Vieu e Aurnague para definir Relação Subcoleção-Coleção que relaciona dois Coletivos (Collective) e que possui o axioma Suplementação Fraca (Weak Supplementation). É importante observar que em Guizzardi (2005, p. 346), a relação é apresentada como *irreflexiva*, *anti-simétrica* e *transitiva*. Em Guizzardi (2005, p. 351), ela é apresentada como *não-reflexiva*. Já em Guizzardi (2011), a relação é apresentada como *irreflexiva*, *assimétrica* e *transitivia*. Por ser a obra mais recente, esta posição é mantida. Além disso, Relação Subcoleção-Coleção força uma Relação Membro-Coleção (Member-Collection Relation) ser transitiva quando ambas estão relacionadas do seguinte modo: seja $C(x, W)$ uma Relação Subcoleção-Coleção entre uma subcoleção x é um todo integral W, e seja $M(y, x)$ a Relação Membro-Coleção entre o membro y e o todo x. Dessa forma, $M(y, W)$ deve ser deduzível, conforme apresentado por Guizzardi (2011).
Todos os subcoletivos de um coletivo são Partes Inseparáveis (Inseparable Part) desse coletivo, conforme argumento apresentado na citação, ou seja, os subcoletivos não podem existir sem o "todo" do qual fazem parte. Ainda quanto à questão das Partes Inseparáveis e das Partes Essenciais (Essential Part), se um indivíduo x é uma Parte Essencial de um todo W, então todo W' que participar da relação $C(W, W')$ terá esse x como uma Parte Essencial, em que W' é uma subcoleção de W e W' é uma Parte Inseparável de W. Porém, se um coletivo W' é Parte Essencial de outro coletivo W, então não é o caso que os indivíduos de W' sejam também Partes Essenciais do todo W:

> *the structure of collectives is defined via specialization of the collective's unifying relation, i.e., the members of W' and W" are also members of W. This implies that all subcollectives of W are inseparable parts of it, i.e., W' and W" come to existence by refining the structure of W and by grouping the specific members of W.*

> *As a consequence, they cannot exist without that whole. A second observation we can make is that if there is an x which is a member of W', and x is an essential member of W then x must also be an essential member of W'. The argument can be made as follows. If x is an essential part of W then W cannot exist without x; If W′ is an inseparable part of W then W′ cannot exist without W; due to the transitivity of existential dependence (SIMONS, 1987), we have that W′ cannot exist without x, ergo, x is an essential part of W′. Finally, since we are admitting that collectives are not necessarily extensional entities, it is conceivable that a whole W has an essential part W′ composed of members which are not essential for either W or W′. For instance, suppose that by law, all juries must have at least two members which are older than sixty years old. Although this subcollective would be essential to the whole, it is conceivable that its individual members are exchangeable. By the same reasoning, one could admit a particular subcollective to be essential to a whole, without requiring the other collectives of that whole to be likewise essential (*GUIZZARDI, 2011, passim*).*

Ver também: Subcollection of, Part of, Member-Collection Relation, Mass-Quantity Relation, Component-Functional Complex Relation

Subcollective-Collective Relation (Relação Subcoletivo-Coleção)

Ver: Subcollection-Collection Relation

Subkind (Subespécie)

Um Subkind é um Sortal Rígido (Rigid Sortal) que herda seu princípio de identidade de uma Espécie (Kind), que é uma de suas classes superioras. Por exemplo, a entidade `Woman` é um *subkind* do *kind* `Person`.

Ver também: Kind

Subquantity of (Subquantidade de)

Estereótipo usado na OntoUML para representar a Relação Massa-Quantidade (Mass-Quantity Relation).

Ver também: Member of, Component of, Subcollection of

SubQuantity-Quantity (Subquantidade-quantidade)

Ver: Mass-Quantity Relation

Subsetting relation (Relação de subconjunto)

Subsetting é uma relação de subconjunto atribuída à relações que define que as instâncias relacionadas à relação de subconjunto são um subconjunto de instâncias

de outra relação. Uma relação de subconjunto deve ser definida entre duas relações materiais R_2 e R_1, para (R_2 *subsets* R_1) se, e somente se, essas relações são derivadas de Relators de tipos disjuntos e existe uma restrição de inclusão que inclui a extensão de R_2 na extensão de R_1 (COSTAL; GÓMEZ; GUIZZARDI, 2011a). Trata-se de uma relação definida para os terminais das relações.

Ver também: Redefinition relation, Specialization relation (for relations)

Substance (Substância)

Ver: Substantial Universal

Substance Sortal (Sortal de Substância)

Um Sortal de Substância é um Sortal Rígido (Rigid Sortal) e trata-se do único Sortal que provê um Princípio de identidade (Principle of identity) às suas classes concretas (GUIZZARDI, 2005, p. 100).

Ver também: Kind, Rigidity, Substantial Universal

Substantial (Substancial)

Ver: Substantial Universal

Substantial Object (Objeto Substancial)

Um Objeto Substancial é um Particular que exemplifica um Sortal de Substância (Substance Sortal). Na representação de modelos UML, trata-se de objetos concretos, que são instância de um tipo Universal Substancial (Substantial Universal) (GUIZZARDI, 2005, p. 317), (BENEVIDES, 2010, p. 36-37, 139-140).

Ver também: Universal, Particular

Substantial Universal (Universal Substancial)

Entidades substanciais, ou objetos – do inglês *objects* (GUIZZARDI, 2005, p. 216) – são existencialmente independentes (Independence), ou seja, não há nenhum outro indivíduo que precise existir para que um *substantial* exista. Conforme Guizzardi (2005, p. 95):

> *Substantials are entities that persist in time while keeping their identity (as opposed to events such as a kiss, a business process or a birthday party). Examples include physical and social persisting entities of everyday experience such as balls, rocks, students, the North Sea and Queen Beatrix.*

Uma definição formal é apresentada por Guizzardi (2005, p. 215): *"A substantial is an endurant that does not inhere in another endurant, i.e., which is not a moment."* Formalmente:

$$Substantial(x) \coloneqq Endurant(x) \wedge \neg Moment(x)$$

Em Guizzardi (2006), a noção de *substantial* (como indivíduo) é apresentada como:

> Substantials *are individuals that posses (direct) spatial-temporal qualities and that are founded on matter. Examples of Substances include ordinary objects of everyday experience such as an individual person, a dog, a house, a hammer, a car, Alan Turing and The Rolling Stones but also the so-called* Fiat Objects *such as the North-Sea and its proper-parts, postal districts and a non-smoking area of a restaurant. In contrast with moments, substantials do not inhere in anything and, as a consequence, they enjoy a higher degree of independence.* (grifos originais)

Na mesma obra, o *substantital* como Universal é descrito como *"A Substantial Universal is a universal whose instances are substances (e.g., the universal Person or the universal Apple)"*. Guizzardi e Zamborlini (2014) reafirmam os exemplos anteriores e ainda acrescentam:

> *To state this precisely we say that: an object x is* independent *of all other objects that are disjoint from x, i.e., that do not share a common part with x. This definition excludes the dependence between an object and its* essential *and* inseparable parts *(GUIZZARDI, 2005), and the obvious dependence between an object and its essential tropes.*

Ver também: Existential Dependence

T

Terminal (Terminal da relação)

A expressão *terminal da relação* refere-se aos objetos possíveis de serem ligados pela relação. Quando uma relação é funcional, seus terminais são o domínio e o contra-domínio.

Ver também: Redefinition relation

Topoid

No contexto de Mereologias (Mereology) e Relações Meronímicas (Meronymic relation), Topoids são regiões espaciais que possuem uma certa estrutura mereotopológica (GUIZZARDI, 2002).

Ver também: Subquantity of

Trope

Termo usado como sinônimo de Momento (Moment), que também é usado para designar os Momentos individuais (Moment individual). A ideia de *"trope"* é extraída de Williams (1953, p. 7), e está presente em Guizzardi, Masolo e Borgo (2006), Guizzardi (2009), Guizzardi e Wagner (2008), Guizzardi e Zamborlini (2014), entre outros.

Ver também: Externally Dependent Trope, Moment individual, Moment Universal, Bearer of a Trope, Relational Trope

Type (Tipo)

Ver: Universal

U

UFO

Acrônimo de Unified Foundational Ontology.

UFO-A

Trata-se de uma ontologia de Endurantes (Endurant), ou de Objetos sem partes temporais, que persistem no tempo e mantém sua identidade. A principal referência sobre UFO-A é a referida tese de Guizzardi, mas também alguns trabalhos posteriores como Guizzardi (2006), Guizzardi (2009), entre outros.

UFO-B

A UFO-B é uma ontologia que incrementa a UFO-A com a introdução da noção de Perdurantes (Perdurant), caracterizados como eventos com duração delimitada, compostos de partes temporais (GUIZZARDI; FALBO; GUIZZARDI, 2008).

UFO-C

Com base na UFO-A e na UFO-B, a UFO-C é uma ontologia de *entidades sociais* e intencionais, incluindo aspectos linguísticos. A UFO-C é desenvolvida essencialmente por Guizzardi (2006).

Unified Foundational Ontology (Ontologia de Fundamentação Unificada)

Em seu seminal trabalho de doutorado, Guizzardi (2005, p. 382)[20] propõe a *Unified Foundational Ontology* (UFO) como ontologia de fundamentação (ou ontologia fundacional) "centrada nas categorias ontológicas e Endurantes (Endurant) e de

[20] Uma das primeiras publicações sobre o nome "UFO" aparece em Guizzardi e Wagner (2005b).

endurantes universais" criada com o propósito específico de servir como uma teoria base para modelagem conceitual. Trata-se de desenvolvimento com abordagem interdisciplinar de Ontologia Formal, Lógica Filosófica, Linguística e Psicologia Cognitiva que é parte de um programa maior de ontologias de fundamentação, organizadas em três camadas: UFO-A, UFO-B e UFO-C. Conforme Guizzardi e Wagner (2005b), Guizzardi e Wagner (2005a) e Guizzardi e Wagner (2010, p. 175), UFO é derivada de uma síntese e do aperfeiçoamento de duas outras ontologias de fundamentação: GFO/GOL (DEGEN et al., 2001) (ver <http://www.onto-med.de/>) e OntoClean/DOLCE (WELTY; GUARINO, 2001; GUARINO; WELTY, 2002).

Universal

Os Universais são padrões de características independentes de espaço-tempo que são realizados em diferentes indivíduos (Particular). Todos os universais carregam um Princípio de aplicação (Principle of application).

> [...] *a universal can be considered simply as something (i) which can be predicated of other entities (or, in the Aristotelian sense, said of or attributed to other entities) and (ii) that can potentially be represented in language by* predicative terms. *In summary, as a synonymous of type* [...] (GUIZZARDI, 2005, p. 218)

A ideia de universal é bastante discutida pelo pesquisador em tela. Ele ressalta a diferença entre a ideia de universal adotada na UFO e as baseadas na teoria de conjuntos, que tradicionalmente são teorias extensionais que basicamente igualam um universal a ideia de um conjunto (GUIZZARDI, 2005, p. 219). Nesse sentido, diferente de outras teorias de universais com semânticas conjuntistas – como é o caso, por exemplo, da chamada *C-theory* e suas tentativas de aprimoramento propostas por Quinton (1957) na *Natural Class theory* – Guizzardi (2005, p. 220) estipula que os universais da UFO são definidos intencionalmente. Dessa forma, os universais são expressos por meio de especificação axiomática (Elementary specification), ou seja, um conjunto de axiomas que explicitamente reúnem um conjunto de Características (Feature) essenciais ao universal em questão, e que são a base de sua definição. Definido intencionalmente, a extensão de universal passa a ser o conjunto formado pelas instâncias desse universal, e apenas por elas.
Universais se subdividem em Universais Monádicos (Monadic Universal) e Relações (Relation).

Ver também: Urelement, glsextensionOfUniversal

Urelement (Ur-Elemento)

Em oposição à ideia de Conjunto (Set), os *Ur-Elemento* não podem ser desdobrados em outros elementos. Tratam-se de entidades que não são conjuntos. Conforme Guizzardi (2005, p. 209):

> *A fundamental distinction in this ontology is between the categories of the so-called* urelements *and* sets*. Urelements are entities that are not sets. They form an ultimate layer of entities without any set-theoretical structure in their build-up, i.e., neither the membership ($\in$) relation nor the set inclusion ($\subseteq$) relation can unfold the internal structure of urelements.*

Os *elementos-Ur* subdividem-se em Individuais (Individual) e Universais (Universal).

Ver também: Set

W

Weak Supplementation (Suplementação Fraca)

Uma crítica imediata da relação básica de ordem parcial provida pela Mereologia Base (Ground Mereology) retrata o fato de que ela permite a construção de modelos inconsistentes. É o caso, por exemplo, de um x que possui como parte própria apenas um y. Isso está em conformidade com os axiomas da Mereologia Base. Porém, se x possui uma parte própria, significa que ele precisa ter ao menos uma outra parte. Desse modo, Suplementação Fraca consiste no seguinte axioma, que pode ser usado como extensão à Mereologia Base. De dessa forma, adaptado de Guizzardi (2005, p. 145)[21]:

$$\forall x, y\ (y < x) \rightarrow \exists z\ ((z < x) \wedge (z \int y))$$

em que $<$ é definido na entrada Proper part e $\int$ é definido no verbete Disjointness.

Ver também: Strong Supplementation, Minimal Mereology

Whole (Todo)

Expressão usada para referenciar aquilo que agrega partes, ou que é composta de partes.

Ver também: Functional Complex, Inseparable Whole, Mandatory Whole

[21] Na obra original, a forma de *Weak Supplementation* é apresentada como: $\forall x, y\ (y < x) \rightarrow \exists z\ (z < x) \wedge (z \int y)$. Porém, na fórmula original z aparece livre, e não ligada a nenhum quantificador. Como supõe-se que o autor desejava apresentar z ligada ao quantificador existencial, apresentamos no glossário a fórmula já adaptada.

Apêndices

APÊNDICE A – Especificação da Metateoria de *Hypertypes*

O presente apêndice apresenta a especificação da Metateoria de *Hypertypes* (seção 5.6) embutida em toda teoria de Ontoprolog. A metateoria é especificada no Código 31 na sintaxe de Ontoprolog(Capítulo 7). O Código 32, por sua vez, contém as regras na forma `nrulemeta` (subseção 6.4.4) que complementam as regras de meta-propriedades bases.

Código 31 – Metateoria de *hypertypes*

```
%% ------------------------------------------------------------------------
%% HYPER TYPES
%% ------------------------------------------------------------------------

%%
%% Top level properties
%%

meta hypertype on property.

[slotProperty, multiplicityProperty] extends property.
meta hypertype
  on [ slotProperty,
       multiplicityProperty ].

meta check value_is_entity
  on transitive instances of slotProperty.

meta check value_is_cardinality
  on transitive instances of multiplicityProperty.

% instantiable properties

[domain,    codomain]    :: slotProperty.
[domain_cc, codomain_cc] :: multiplicityProperty.

meta required
  on [ instances of domain,
       instances of domain_cc,
       instances of codomain,
       instances of codomain_cc].

%%
%% Top level types
%%

meta hypertype on type.

relationship extends type.
binaryRelationship extends relationship.
[directedRelationship, associativeRelationship] extends binaryRelationship.
aggregationRelationship extends directedRelationship.

meta hypertype
  on [relationship,
      binaryRelationship,
      associativeRelationship,
      directedRelationship,
      aggregationRelationship].

meta abstract
  on relationship.

property [ domain,
           codomain,
           domain_cc,
```

```
          codomain_cc]
      on binaryRelationship.

meta symmetric
  on transitive instances of associativeRelationship.
```

Código 32 – Regras na forma `nrulemeta` da metateoria de *hypertypes*

```
%=============================================================================
% Negative constraint checking for meta properties
%=============================================================================

% ----
% General meta properties rules
% ----

% ncheck for meta abstract
% (Definição 5.6)
nrulemeta abstract
      for Abstract_T
  message ['"', Entity, '" should instantiate a non abstract entity directly
    besides "', Abstract_T, '"'] :-
        dio(Entity, Abstract_T),
        \+ ( dio(Entity, Non_Abstract_T),
            \+ dmo(abstract, Non_Abstract_T )
          ).

% ncheck for meta final
nrulemeta final
      for Final_T
  message ['"', Entity, '" should not extend the final entity "', Final_T, '"'] :-
        deo(Entity, Final_T).

% ----
% Rules for relationships
% ----

% ncheck for meta irreflexive
nrulemeta irreflexive
      for T
  message ['"', T, '" is not a (transitive) instance of "relationship" hypertype.']
    :-
        \+ hypertypesof(T, relationship).

% ncheck for meta irreflexive
nrulemeta irreflexive
      for Rel_Type
  message ['"rel(',Rel1,'::',Rel_Type,',', Type1,',', Type1,') is reflexive.'] :-
        instanceof(Rel1, Rel_Type),
        drel(Rel1, Type1, _, Type1, _).

% ncheck for meta symmetric
nrulemeta symmetric
      for T
  message ['"', T, '" is not a (transitive) instance of "relationship" hypertype.']
    :-
        \+ hypertypesof(T, relationship).

% ncheck for meta symmetric and meta asymmetric
nrulemeta symmetric
      for T
  message ['"', T, '" can not receive both "symmetric" and "asymmetric" meta
    properties.'] :-
        dmo(asymmetric, T).

% ncheck for meta asymmetric
nrulemeta asymmetric
      for T
  message ['"', T, '" is not a (transitive) instance of "relationship" hypertype.']
    :-
        \+ hypertypesof(T, relationship).

% ncheck for meta asymmetric
nrulemeta asymmetric
      for Rel_Type
  message ['"rel(',Rel1,'::',Rel_Type,',', Type1,',', Type2,') and
    "rel(',Rel2,'::',Rel_Type,',', Type2,',', Type1,')" are symmetric.'] :-
        instanceof(Rel1, Rel_Type),
```

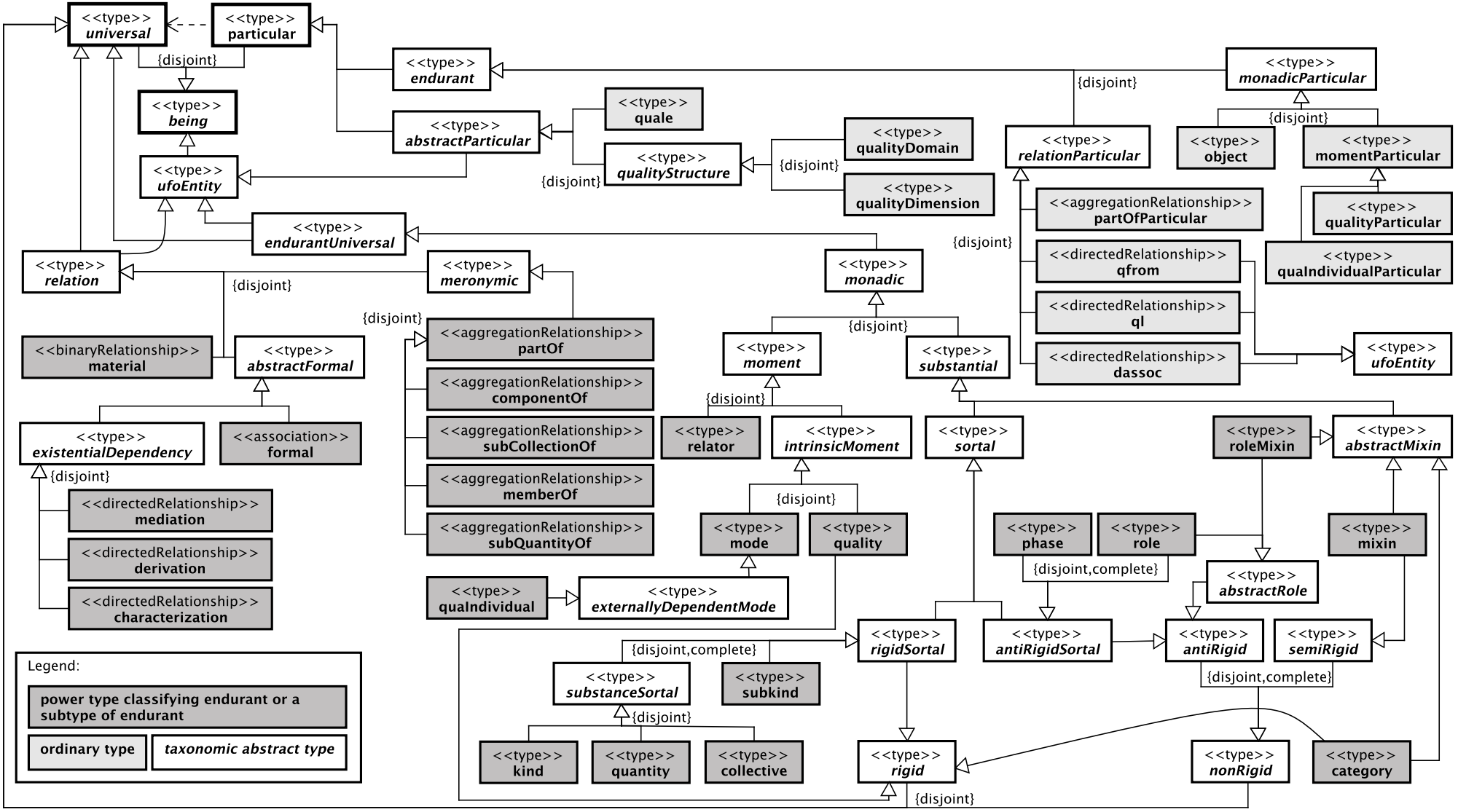
```
        instanceof(Rel2, Rel_Type),
        drel(Rel1, Type1, _, Type2, _),
        drel(Rel2, Type2, _, Type1, _).

% ncheck for meta transitive
nrulemeta transitive
      for T
  message ['"', T, '" is not a (transitive) instance of "relationship" hypertype.']
    :-
        \+ hypertypesof(T, relationship).

% ncheck for meta transitive and meta intransitive
nrulemeta transitive
      for T
  message ['"', T, '" can not receive both "transitive" and "intransitive" meta
    properties.'] :-
        dmo(intransitive, T).

% ncheck for meta intransitive
nrulemeta intransitive
      for T
  message ['"', T, '" is not a (transitive) instance of "relationship" hypertype.']
    :-
        \+ hypertypesof(T, relationship).

% ----
% Rules for properties
% ----

% ncheck for meta required
nrulemeta required
      for P at T
  message ['"', P at T, '" is not a (transitive) instance of "property"
    hypertype.'] :-
        entity(P),
        \+ hypertypesof(P, property).

% ncheck for meta required
nrulemeta required
      for Prop at Type
  message ['required property value for "', Prop, '" is not present at "',
    Particular, '::', Type, '".'] :-
        instanceof(Particular, Type),
        \+ dmo(abstract, Particular),
        \+ property_value(Prop at Particular, _).

% ----
% Check constraints for properties
% ----

nrulemeta check value_is_entity
      for Property
  message ['"', Value, '" of "', Property, ' at ', Entity, '" is not an entity in
    the theory.'] :-
        dpv(Property at Entity, Value),
        \+ entity(Value).

% checker for multiplicity/cardinality
nrulemeta check value_is_cardinality
      for Property
  message ['"', Value, '" of "', Property, ' at ', Entity, '" is not a cardinality
    constraint.'] :-
        dpv(Property at Entity, Value),
        \+ card(Value).

nrulemeta check lower_cc(Lower)
      for Property at Entity
  message ['It was expected that the lower limit "', CC_Lower, '" of "', Property,
    ' at ', Entity, '" were between "', Lower, '".'] :-
        dpv(Property at Entity, Value),
        card_lower(Value, CC_Lower),
        \+ card_includes(Lower, CC_Lower).

nrulemeta check upper_cc(Upper)
      for Property at Entity
  message ['It was expected that the upper limit "', CC_Upper, '" of "', Property,
    ' at ', Entity, '" were between "', Upper, '".'] :-
        dpv(Property at Entity, Value),
        card_upper(Value, CC_Upper),
        \+ card_includes(Upper, CC_Upper).
```

```
nrulemeta check including_cc(Check_CC)
      for Property at Entity
  message ['It was expected that the cardinality "', Value, '" of "', Property, '
    at ', Entity, '" were between "', Check_CC, '".'] :-
        dpv(Property at Entity, Value),
        \+ dmo(abstract, Entity),
        \+ card_includes(Check_CC, Value).

% ----
% Check extending/instantiating constraints
% ----

%% extension_disjoint_branches_for_check(+Type_List_In, -Type_List_Out)
% extension_disjoint_branches_for_check/2
% same as extension_disjoint_branches/2, but deals with :: operator
extension_disjoint_branches_for_check(Type_List_In, Type_List_Out) :-
        % remove :: operator
        findall(S,
                ( util_member(I, Type_List_In),
                  S :: _ = I
                ),
                SS),
        findall(I,
                ( util_member(I, Type_List_In),
                  S :: _ = I,
                  \+is_extensionof(S, SS)
                ),
                Type_List_Out).

% ncheck for meta check extending
nrulemeta check extending Cardinality_Constraint_Opt_Instance of Of_Type_Opt_List
      for Type
  message ['"', Type, '" have to extend "', Cardinality_Constraint, '" instance(s)
    (one or more) of "', Of_Type_Opt_List,
         '", but it extends "' , commas(Extension_List), '".'] :-
        syntactic_card_constraint(Cardinality_Constraint_Opt_Instance,
    Cardinality_Constraint),
        findall(S :: O,
              ( extensionof_reflexive(Type, S),
                util_member_opt_list(O, Of_Type_Opt_List),
                instanceof(S, O)
              ),
              Pre_Extension_List),
        extension_disjoint_branches_for_check(Pre_Extension_List, Extension_List),
        length(Extension_List, Extension_List_Length),
        \+ card_includes(Cardinality_Constraint, Extension_List_Length).


% ncheck for meta check instantitating
nrulemeta check instantiating Cardinality_Constraint_Opt_Instance of
    Of_Type_Opt_List
      for Type
  message ['"', Type, '" have to instantiate "', Cardinality_Constraint, '"
    instance(s) (one or more) of "', Of_Type_Opt_List,
         '", but it instantiates "' , commas(Instance_List), '".'] :-
        syntactic_card_constraint(Cardinality_Constraint_Opt_Instance,
    Cardinality_Constraint),
        findall(T :: O,
              ( instanceof(Type, T),
                util_member_opt_list(O, Of_Type_Opt_List),
                instanceof(T, O)
              ),
              Instance_List),
        length(Instance_List, Instance_List_Length),
        \+ card_includes(Cardinality_Constraint, Instance_List_Length).
```

APÊNDICE B – Figura da metateoria UFO

O presente apêndice apresenta o diagrama da Metateoria da UFO definida em Ontoprolog (subseção 5.7.1). A metateoria é apresentada na Figura 59 como um diagrama UML cujas entidades são conceitos da UFO e identificados por termos equivalentes a entradas no Glossário da UFO. O glossário, por sua vez, contém as definições dos termos e as referências bibliográficas correspondentes aos fundamentos ontológicos utilizados para construir cada conceito. A especificação formal dessa metateoria em Ontoprolog é apresentada pelo Apêndice C.

Figura 59 – Entidades da metateoria de Ontoprolog.



Fonte: Os autores

APÊNDICE C – Especificação da metateoria UFO em Ontoprolog

O presente apêndice apresenta a Metateoria da UFO definida em Ontoprolog (subseção 5.7.1). A metateoria é apresentada como uma Especificação Ontoprolog (Definição 7.7) Ontoprolog, conforme definido no Capítulo 7. O Código 33 contém a Especificação Ontoprolog representada em UML na Figura 59 do Apêndice B. Já o Código 34 contém as regras na forma `nrulemeta` (subseção 6.4.4) que complementam as regras de meta-propriedades específicas para a UFO.

Código 33 – Metateoria UFO de Ontoprolog

```
%%
%% REFERENCE ONTOLOGY
%%

being :: type.
meta abstract
  on being.

disjoint [particular, universal] :: type extend being.
meta abstract
  on [particular, universal].

particular :: universal.

%%
%% UFO-A meta theory
%%

ufoEntity :: type extends being.
meta abstract
  on ufoEntity.

% UNIVERSAL

endurantUniversal :: type extends [universal, ufoEntity].
meta abstract
  on endurantUniversal.

disjoint [monadic, relation] :: type.
relation extends [universal,ufoEntity].
monadic extends endurantUniversal.
meta abstract
  on [monadic, relation].
meta note('relation as poliadic relation')
  on relation.

% Endurants

% Relation (poliadic)

abstractFormal :: type.
meta abstract
  on abstractFormal.

material :: associativeRelationship relates
        many instances of monadic
        and many instances of monadic.
powertype material classifying relationParticular.

[material, abstractFormal] extend relation.

existentialDependency :: type.
meta abstract
  on existentialDependency.
```

```
formal :: associativeRelationship relates
        many instances of being
        and many instances of being.
powertype formal classifying relationParticular.

[formal, existentialDependency] extend abstractFormal.

% (Quadro 7 (Role), restrição 2, p. 173)
% (Quadro 10 (RoleMixin), restrição 3, p. 176)
% (Quadro 18 (Relator), restrição 1, p. 182)
% (Quadro 18 (Relator), restrição 2, p. 182)
% (Quadro 18 (Relator), restrição 3, p. 182)
% (Quadro 19 (Mediation), restrição 2, p. 183)
mediation :: directedRelationship relates
        1..many instance of relator
        to 2..many instance of abstractRole.

% (Quadro 21 (Derivation Relation), restrição 2, p. 185)
% (Quadro 23 (Material), restrição 1, p. 186)
derivation :: directedRelationship relates
        1 instance of relator
        to 1..many instance of material.

% (Quadro 17 (Mode), restrição 1, p. 181)
% (Quadro 18 (Relator), restrição 4, p. 182)
% (Quadro 20 (Characterization), restrição 2, p. 184)
% (Quadro 13 (Quality), restrição 2, p. 178)
% (Quadro 12 (Moment), restrição 1, p. 177)
characterization :: directedRelationship relates
        0..many instance of moment
        to 1..many instance of endurantUniversal.

[mediation,derivation,characterization] extend existentialDependency.

meta [ irreflexive, asymmetric ]
  on [ transitive instances of mediation,
       transitive instances of derivation,
       transitive instances of characterization ].

powertype [mediation,derivation,characterization] classifying relationParticular.

% Meronymic

meronymic :: type extends relation.
meta abstract on meronymic.

disjoint [meronymic, material, abstractFormal].  %%%%%

partOf :: aggregationRelationship extends meronymic relates
        many instance of monadic             % Weak supplementation(!)
        partof many instance of monadic.

meta irreflexive
  on transitive instances of partOf.

% (Quadro 25 (ComponentOf), restrição 1, p. 188)
componentOf :: aggregationRelationship relates
        many instance of substantial
        partof many instance of substantial.
% (Quadro 27 (SubCollectionOf), restrição 1, p. 190)
subCollectionOf :: aggregationRelationship relates
        many instance of collective
        partof many instance of collective.
% (Quadro 28 (MemberOf), restrição 1, p. 191)
memberOf :: aggregationRelationship relates
        many instance of substantial
        partof many instance of collective.
% (Quadro 26 (SubQuantityOf), restrição 1, p. 189)
subQuantityOf :: aggregationRelationship relates
        many instances of quantity
        partof 0..1 instances of quantity.  %% PENDENTE

meta note('componentOf relates functional complexes')
  on componentOf.

meta [ asymmetric,
       transitive ]
  on [ subQuantityOf,
       extensions of subQuantityOf,
       transitive instances of subQuantityOf,
```

```
       subCollectionOf,
       extensions of subCollectionOf,
       transitive instances of subCollectionOf].

meta [ asymmetric,
       intransitive ]
  on [ memberOf,
       extensions of memberOf,
       transitive instances of memberOf].

disjoint [componentOf,subCollectionOf,memberOf,subQuantityOf] extend partOf.

powertype [partOf, componentOf, subQuantityOf, subCollectionOf, memberOf]
    classifying partOfParticular.


% Monadic

disjoint [moment, substantial] :: type extend monadic.
meta abstract
  on [moment, substantial].
meta note('trope type')
  on moment.
meta note('object type')
  on substantial.

% Moment

disjoint [intrinsicMoment, relator] :: type extend moment.
meta abstract
  on intrinsicMoment.
powertype relator classifying momentParticular.
meta note('Relators as relational moments')
  on relator.

disjoint [quality, mode] :: type extend intrinsicMoment.
quality extends rigid.
powertype mode    classifying momentParticular.
powertype quality classifying qualityParticular.

externallyDependentMode :: type extends mode.
meta abstract
  on externallyDependentMode.

quaIndividual :: type extends externallyDependentMode.
powertype quaIndividual classifying quaIndividualParticular.

% Substantial

[sortal, abstractMixin] :: type extend substantial.
meta abstract
  on sortal.
meta [abstract,
      note('nonSortal')]
  on abstractMixin.

% rigid (new concepts from Guizzardi's hierarchy)

disjoint [rigid, nonRigid] :: type extends endurantUniversal.
disjoint [antiRigid, semiRigid] :: type cover nonRigid.
meta abstract
  on [rigid, nonRigid, antiRigid, semiRigid].

abstractRole :: type extends antiRigid.
meta abstract
  on abstractRole.

% Sortal

rigidSortal :: type extends [sortal,rigid].
meta abstract
  on rigidSortal.

disjoint [substanceSortal, subkind] :: type cover rigidSortal.
powertype subkind classifying object.
meta abstract
  on substanceSortal.

disjoint [kind, collective, quantity] :: type extend substanceSortal.
powertype [kind, collective, quantity] classifying object.
```

```
antiRigidSortal :: type extends [sortal, antiRigid].
meta abstract
  on antiRigidSortal.

disjoint [phase, role] :: type cover antiRigidSortal.
role extends abstractRole.
powertype [phase, role] classifying object.

% Mixin

[roleMixin, mixin, category] :: type extend abstractMixin.
powertype [roleMixin, mixin, category] classifying object.

roleMixin extends abstractRole.
mixin extends semiRigid.
category extends rigid.


% PARTICULAR


abstractParticular :: type extend [particular, ufoEntity].
endurant :: type extends particular.
disjoint [abstractParticular, endurant].
meta abstract
  on [abstractParticular, endurant].

disjoint [monadicParticular, relationParticular] :: type extend endurant.

meta note('abstractParticular')
  on endurant.

% Abstract Particular
% Quality Structures

disjoint [quale, qualityStructure] :: type extend abstractParticular.
meta abstract
  on qualityStructure.

disjoint [qualityDimension, qualityDomain] :: type extend qualityStructure.

% properties

property conceptualSpace on qualityDomain.

meta required
  on conceptualSpace at instances of qualityDomain.

meta check is_quality_space
  on transitive instances of conceptualSpace.

property qualityRegion on quale.

meta required
  on qualityRegion at instances of quale.

meta check is_quality_region
  on transitive instances of qualityRegion.

% Concrete Paricular

meta abstract
  on [monadicParticular, relationParticular].

% Monadic

disjoint [momentParticular, object] :: type extend monadicParticular.
disjoint [qualityParticular, quaIndividualParticular] :: type extend
    momentParticular.

% Relations

partOfParticular :: aggregationRelationship extends relationParticular relates
    0..many instances of particular
    partof 0..many instances of particular.

% (Quadro 14 (QualityDimension como SimpleDataType), restrição 2, p. 179)
% (Quadro 16 (Quale), restrição 1, p. 180)
qfrom :: directedRelationship relates
    0..many instances of quale
    to 1 instance of qualityStructure.
```

```
% (Quadro 16 (Quale), restrição 1, p. 180)
ql :: directedRelationship relates
    1..many instances of quale
    to 1 instance of qualityParticular.
% (Quadro 13 (Quality), restrição 1, p. 178)
% (Quadro 14 (QualityDimension como SimpleDataType), restrição 2, p. 179)
% (Quadro 15 (QualityDomain como StructuredDataType), restrição 2, p. 180)
dassoc :: associativeRelationship relates % (dassoc relates types)
    1..many instance of quality
    and 1..many instances of qualityStructure.

disjoint [qfrom, ql, dassoc] extend [relationParticular, ufoEntity].

meta [final, irreflexive]
  on [qfrom, ql, dassoc].


%%%%%
%%%%% Constraints (negative rules) over extensions or instances
%%%%% Based on UFO
%%%%%

% (Quadro 1 (Substance Sortal), restrição 1, p. 168)
meta check instantiating 1..many instances of [moment, relation, substanceSortal]
  on instances of monadicParticular.

% (Quadro 1 (Substance Sortal), restrição 2, p. 168)
meta check instantiating 0..1 instances of substanceSortal
  on instances of particular.

% (Quadro 1 (Substance Sortal), restrição 3, p. 168)
meta check extending 0..1 instance of substanceSortal
  on instances of universal.

% (Quadro 1 (Substance Sortal), restrição 4, p. 168)
% (Quadro 5 (Subkind), restrição 2, p. 171)
% (Quadro 6 (Phase), restrição 1, p. 172) (parte 1 de 2)
% (Quadro 7 (Role), restrição 1, p. 173) (parte 1 de 2)
% (Quadro 9 (Category), restrição 1, p. 175)
meta check extending 0 instances of antiRigid
  on instances of rigid.

% (Quadro 3 (Quantity), restrição 1, p. 169)
meta extensional
  on [ transitive instances of quantity,
       extensions of quantity ].

% (Quadro 5 (Subkind), restrição 3, p. 171)
meta check extending 1 instance of kind
  on instances of subkind.

% (Quadro 6 (Phase), restrição 1, p. 172) (parte 2 de 2)
meta check extending 0 instances of phase
  on instances of role.

% (Quadro 7 (Role), restrição 1, p. 173) (parte 2 de 2)
meta check extending 0 instances of role
  on instances of phase.

% (Quadro 8 (Mixin Class), restrição 1, p. 174)
% (Quadro 8 (Mixin Class), restrição 2, p. 174)
% (Quadro 9 (Category), restrição 2, p. 175)
% (Quadro 9 (Category), restrição 3, p. 175)
% (Quadro 10 (RoleMixin), restrição 1, p. 176)
% (Quadro 10 (RoleMixin), restrição 2, p. 176)
% (Quadro 11 (Mixin), restrição 2, p. 177)
meta [check extending 0 instance of sortal,
      abstract]
  on instances of abstractMixin.

% (Quadro 11 (Mixin), restrição 1, p. 177)
meta check extending 0 of antiRigid
  on instances of semiRigid.

% (Quadro 19 (Mediation), restrição 3, p. 183)
meta check lower_cc(1)
  on domain_cc at instances of mediation.
meta check lower_cc(1)
  on codomain_cc at instances of mediation.

% (Quadro 20 (Characterization), restrição 3, p. 184)
```

```
% (Quadro 20 (Characterization), restrição 4, p. 184)
% (Quadro 13 (Quality), restrição 2, p. 178)
meta check lower_cc(1..many)
  on domain_cc at instances of characterization.
meta check including_cc(1)
  on codomain_cc at instances of characterization.

% (Quadro 19 (Mediation), restrição 4, p. 183)
meta immutable
  on codomain at instances of mediation.

% (Quadro 20 (Characterization), restrição 5, p. 184)
meta immutable
  on codomain at instances of characterization.

% (Quadro 21 (Derivation Relation), restrição 3, p. 185)
% (Quadro 21 (Derivation Relation), restrição 5, p. 185) (parcial)
meta check including_cc(1)
  on domain_cc at instances of derivation.
meta check including_cc(1..many)
  on codomain_cc at instances of derivation.

% (Quadro 21 (Derivation Relation), restrição 4, p. 185)
meta immutable
  on domain at instances of derivation.

% (Quadro 23 (Material), restrição 2, p. 186)
meta derived
  on instances of material.

% (Quadro 23 (Material), restrição 3, p. 186)
meta check including_cc(1..many)
  on domain_cc at instances of material.
meta check including_cc(1..many)
  on codomain_cc at instances of material.

% (Quadro 24 (Meronymic), restrição 1, p. 187)
meta weak_supplementation
  on instances of partOf.

% (Quadro 26 (SubQuantityOf), restrição 2, p. 189)
meta check including_cc(1)
  on domain_cc at instances of subQuantityOf.

% (Quadro 26 (SubQuantityOf), restrição 3, p. 189)
% (Quadro 27 (SubCollectionOf), restrição 3, p. 190)
meta irreflexive
  on [ subQuantityOf,
       subCollectionOf,
       extensions of subCollectionOf,
       extensions of subQuantityOf ].

% (Quadro 26 (SubQuantityOf), restrição 4, p. 189)
meta nonshared
  on codomain at instances of subQuantityOf.

% (Quadro 27 (SubCollectionOf), restrição 2, p. 190)
meta check including_cc(1)
  on domain_cc at instances of subCollectionOf.

% (Quadro 27 (SubCollectionOf), restrição 4, p. 190)
meta inseparable
  on [ codomain at instances of subCollectionOf,
       codomain at extensions of subCollectionOf ].
meta immutable
  on [ domain at instances of subCollectionOf,
       domain at extensions of subCollectionOf ].

%%%%
%%%% Additional rules
%%%%

meta check extending 1 instance of rigidSortal
  on instances of antiRigidSortal.
```

Código 34 – Regras na forma `nrulemeta` da metateoria UFO

```
%=============================================================================
```

```
% Negative constraint checking for meta properties
%============================================================================

% ----
% quality spaces and quality regions
% ----

nrulemeta check is_quality_space
      for Property at Entity
  message ['"', Value, '" of "', Property, ' at ', Entity, '" is not a quality
    space.'] :-
        dpv(Property at Entity, Value),
        \+ syntactic_quality_space(Value).

nrulemeta check is_quality_region
      for Property
  message ['"', Value, '" of "', Property, ' at ', Entity, '" is not a quality
    region.'] :-
        dpv(Property at Entity, Value),
        \+ syntactic_quale_value(Value).

% ----
% extensional
% ----

% (Quadro 3 (Quantity), restrição 1, p. 169)
% (Quadro 4 (Collective), restrição 1, p. 170)
% "extensional" meta attribute present on wrong entities
nrulemeta extensional
      for Entity
  message ['It was expected that "', Entity, '" were an instance of "collective" or
    "quantity".'] :-
        \+ instanceof(Entity, collective),
        \+ instanceof(Entity, quantity).

% ----
% essential
% ----

% (Quadro 24 (Meronymic), restrição 2, p. 187)
% a) Essential {essential} cannot be applied on antiRigid entities
nrulemeta essential
      for domain at Meronymic_Rel_Type
  message ['It was expected that the whole "', Whole, '" in the relation
    "',Meronymic_Rel_Type,'" were an instance of "rigid".'] :-
        dwhole(Whole, Meronymic_Rel_Type),
        domain_type(Whole),
        instanceof(Whole, antiRigid).

% (Quadro 28 (MemberOf), restrição 3, p. 191)
% {essential} only if {extensional}
nrulemeta essential
      for Aggreg_Type
  message ['It was expected that the whole "', Whole, '" in the relation
    "',Aggreg_Type,'" had the "extensional" meta attribute.'] :-
         dpv(codomain at Aggreg_Type, Whole),
         \+ dmo(extensional, Whole).

% ----
% inseparable
% ----

% (Quadro 24 (Meronymic), restrição 3, p. 187)
% {inseparable} without whole minimum cardinality
% Message: Minimum cardinality should be 1
% meronymic_meta_inseparable_wrong_cardinality
nrulemeta inseparable
      for domain at Meronymic_Rel_Type
  message ['It was expected that the cardinality at whole side in the relation
    "',Meronymic_Rel_Type,'" were at least 1.'] :-
        dwhole(_, Meronymic_Rel_Type),
        dpv(codomain at Meronymic_Rel_Type, Whole_Cardinality),
        \+ card_greater_or_equals(Whole_Cardinality, 1).

% ----
% nonshared
% ----

% ncheck for meta nonshared
nrulemeta nonshared
      for P
```

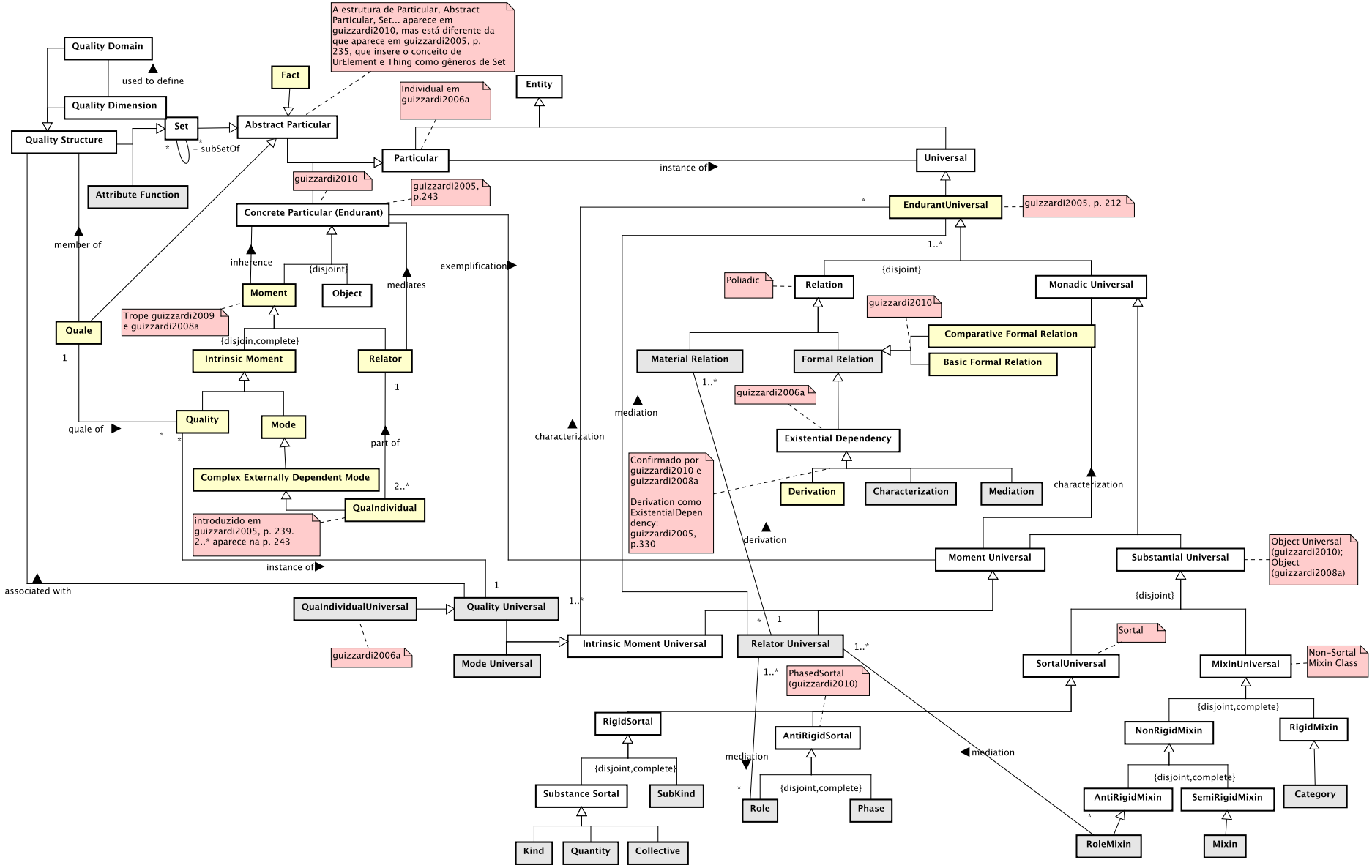
```
  message ['"', P, '" is not a (transitive) instance of "property" hypertype.'] :-
        entity(P),
        \+ hypertypesof(P, property).

nrulemeta nonshared
      for codomain at T
  message ['"', T, '" is not a (transitive) instance of "aggregationRelationship"
    hypertype.'] :-
        \+ hypertypesof(T, aggregationRelationship).

% (Quadro 24 (Meronymic), restrição 4, p. 187)
% Shared: an exclusive part: just one whole
% meronymic_meta_nonshared_broken
nrulemeta nonshared
      for codomain at T
  message Message :-
        nrulemeta check including_cc(0..1)
              for domaincc at T
          message Message.

% ----
% weak_supplementation
% ----

% (Quadro 24 (Meronymic), restrição 1, p. 187)
nrulemeta weak_supplementation
      for PartOf
  message ['The sum of the lower cardinality constraint of the partOf relations
    having "',
           Whole, '" as whole should be at least "2", but it is "', CC_Sum_Lower,
    '".'] :-
        property_value(codomain at PartOf, Whole),
        findall(P,
                ( dmo(weak_supplementation, P),
                  rel(P, _, _, Whole, _)
                ),
                PPs),
        util_to_set(PPs, PPss),
        findall(CC,
                ( util_member(M, PPss),
                  property_value(domain_cc at M, CC)
                ),
                CCs),
        card_sum(CCs, CC_Sum),
        card_lower(CC_Sum, CC_Sum_Lower),
        \+ card_greater_or_equals(CC_Sum_Lower, 2).
```

APÊNDICE D – Diagrama com visão da UFO-A original

O presente apêndice contém a Figura 60, que trata-se de um diagrama produzido pelos autores deste trabalho que representa uma visão geral da UFO-A, sendo a figura o mais fiel possível aos conceitos encontrados na bibliografia da UFO listada nas Referências. Apesar de o diagrama ser um dos produtos do estudo sistematizado das publicações referentes à UFO, ele não representa a metateoria incorporada às especificações Ontoprolog. Para isso, consultar o Apêndice B. Notas e outros comentários são mantidos na figura intencionalmente.

A Figura 9 (página 139) contém um diagrama produzido por Guizzardi e Wagner (2010) que, ao nosso ver, é um dos diagramas mais completos em respeito à quantidade de conceitos fundamentais produzido pelo autor da UFO-A.

Figura 60 – Diagrama com visão geral da UFO conforme discutida nos referenciais teóricos



Fonte: Os autores

APÊNDICE E – Operadores Prolog usados na definição da sintaxe de Ontoprolog

O presente apêndice apresenta o Código 35 e o Código 36. O primeiro contém a lista de operadores Prolog usados na definição das sintaxes de Ontoprolog discutidas neste texto, especialmente no Capítulo 7 e na seção 8.2. Já o Código 36 contém a definição do operador para anotação de modalidade discutido no Capítulo 9.

Código 35 – Operadores de Prolog usados nas definições das sintaxes de Ontoprolog

```
% --- nrulemeta operators  --- %

:- op(690,  fy, [nrulemeta]).
:- op(689, xfy, [for]).
:- op(688, xfy, [message]).

% --- UFO operators  --- %

:- op(670, xfy, [deriving]).
:- op(660, xfy, [mediates]).
:- op(650, yfx, [characterizes,characterize,inheres,inheresin,inhere,inherein]).

% --- base operators  --- %

:- op(646, xfy, [on, classifying, :=]).
:- op(645,  fy, [meta, powertype, property]).
:- op(644, yfx, [redefines, subsets]).
:- op(643, xfy, [at, value]).
:- op(640, yfx, [relates]).
:- op(630, yfx, [extends, extend, cover]).
:- op(487, xfy, [and, partof, to]).
:- op(486, xfy, [bearing]).
:- op(480, xfy, [::, =>]). %  :: is 1200 in MProlog
:- op(478,  fy, [instantiating, extending, check]).
:- op(477, yfx, [of]).
:- op(475,  yf, [instance, instances]).
:- op(474,  fy, [immutable]).
:- op(470, xfy, [<=]).
:- op(450,  fy, [disjoint, nonshared, quale, space]).

:- op(390, xfx, [..]).
:- op(385, xfy, [from, qua]).
:- op(379,  fy, [direct, transitive]).
:- op(375, xfy, [in]).

% --- explanation about operator associativity  --- %

% yfx infix left-associative     (+, -, *)
% xfy infix right-associative    (, (for subgoals))
% xfx infix non-associative      (=, is, < (i.e. no nesting))
% yfy makes no sense, structuring would be impossible
% fy prefix associative
% fx prefix non-associative      (- (i.e. --5 not possible))
% yf postfix associative
% xf postfix non-associative
```

Código 36 – Operador de Prolog usado na definição de anotação de modalidade

```
% --- base modal operators  --- %

:- op(900, xfx, [:>]).    % modalities
```

APÊNDICE F – Expansões das relações semânticas

Este apêndice apresenta as expansões das relações semânticas abordadas na seção 7.6.

As expansões são definidas pelo Código 37 com a linguagem usada em Prolog. No caso, o predicado unário *semantic_expansion*/1 é a definição das relações derivadas, em que cada instância β' em *semantic_expansion*(β') é uma relação derivada. *semantic_term*/1, em *semantic_term*(α), sucede para para qualquer relação semântica $\alpha \in \beta$, em que β o conjunto de relações semânticas da saída sem erros do algoritmo Σ definido na subseção 7.3.3. O conjunto de todos os β' é o conjunto $\mathcal{H}$ da estrutura $\mathcal{OT}$ (seção 6.2).

Já *semantic_expansion_dio_hierarchy*/2 e *semantic_expansion_deo_hierarchy*/2 denotam o fecho transitivo das relações semânticas *dio*/2 e *deo*/2, respectivamente, ambas incluídas no conjunto β e são definidos com base no Código 38.

Código 37 – Expansões das relações semânticas

```
%=============================================================================
% Entity
%
% entity/1     (direct disjunction)
%=============================================================================

%% entity/1    (direct disjunction)
semantic_expansion(entity(Entity)) :-
        semantic_term(entity(Entity)).

%=============================================================================
% Instance relation
%
% dio/2
%=============================================================================

%% dio/2        (direct instance of) with default rule
% Default rule: entities without instance relation (::) are
% read as instance of subkind (Quadro 5, restricao 1)
semantic_expansion(dio(Type, Meta_Type)) :-
        semantic_term(dio(Type, Meta_Type)).

%=============================================================================
% Defesiable dio/2: Complementing the definition of dio/2
%
% Non monotonic behaviour for instance relations based on defeasible/1
% provided from dec_sentence_case(property_default, ...)  (PROPERTY_ASSOCIATION)
%
% dio/2
%=============================================================================

semantic_expansion(dio(Type, property)) :-  % PROPERTY_ASSOCIATION
  semantic_term(entity(Type)),
  semantic_term(defeasible(dio(Type, property))),
  \+ semantic_term(dio(Type, _)).

%=============================================================================
% Properties for dmo/2
%
% dpv/2         (direct property value assignment)
```

```
% dpo/2         (direct property on)
%============================================================================

%% dpv/2        (direct property value assignment)
semantic_expansion(dpv(Property_At_Type, Value)) :-
        semantic_term(dpv(Property_At_Type, Value)).

%% dpo/2        (direct property on)
semantic_expansion(dpo(Property, On_Type)) :-
        semantic_term(dpo(Property, On_Type)).

%============================================================================
% Extension relation and semantic for powertype (pt/2)
%
% deo/2        (direct extension of)
% dcomplete/2  (direct complete subsumption)
%============================================================================

%% deo/2        (direct extension of)
semantic_expansion(deo(Type, Super_Type)) :-
        semantic_term(deo(Type, Super_Type)).

% Semantic for Powertype: Every instance of a metatype is an extension of another
    class
semantic_expansion(deo(Particular, Classified_Type)) :-
        semantic_expansion(dio(Particular, Meta_Type)),
        semantic_term(pt(Meta_Type, Classified_Type)).

%% dcomplete/2   (direct complete subsumption)
semantic_expansion(dcomplete(Super_Type, Type_List)) :-
        semantic_term(dcomplete(Super_Type, Type_List)).

%============================================================================
% Powertype
%
% powertype/2   (power type of)
%============================================================================

%% powertype/2  (power type of)
semantic_expansion(pt(Power_Type, Classified_Type)) :-
        semantic_term(pt(Power_Type, Classified_Type)).

%============================================================================
% Meta properties for dmo/2
%
% dmo/2         (direct meta on)
%============================================================================

%% dmo/2        (direct meta on)
semantic_expansion(dmo(Meta, Type)) :-
        semantic_term(dmo(M, T)),
        ( T = _ at _ % succeeds whether "at" is the main operator
          -> semantic_meta_expansion_at(dmo(M, T), dmo(Meta, Type))
           ; semantic_meta_expansion(dmo(M, T), dmo(Meta, Type))
        ).

% -- META EXPANSIONS

%-- single

semantic_meta_expansion(dmo(M, T), dmo(M, T)):-
        syntactic_entity_with_meta(T).

% -- extensions

semantic_meta_expansion(dmo(MetaProperty, extensions of Type), dmo(MetaProperty,
    Sub_Type)) :-
        semantic_expansion_deo_hierarchy(Sub_Type, Type).

% -- instances

semantic_meta_expansion(dmo(MetaProperty, instances of Meta_Type),
    dmo(MetaProperty, Instance_Type)) :-
        semantic_expansion(dio(Instance_Type, Meta_Type)).
semantic_meta_expansion(dmo(MetaProperty, instances of Super_Type),
    dmo(MetaProperty, Instance_Type)) :-
        semantic_expansion_deo_hierarchy(Sub_Type, Super_Type),
        semantic_expansion(dio(Instance_Type, Sub_Type)).

semantic_meta_expansion(dmo(MetaProperty, direct instances of Meta_Type),
    dmo(MetaProperty, Instance_Type)) :-
```

```
        semantic_expansion(dio(Instance_Type, Meta_Type)).

semantic_meta_expansion(dmo(MetaProperty, transitive direct instances of
    Meta_Type), dmo(MetaProperty, Instance_Type)) :-
        semantic_expansion_dio_hierarchy(Instance_Type, Meta_Type).

semantic_meta_expansion(dmo(MetaProperty, transitive instances of Meta_Type),
    dmo(MetaProperty, Instance_Type)) :-
        semantic_expansion_dio_hierarchy(Instance_Type, Meta_Type).
semantic_meta_expansion(dmo(MetaProperty, transitive instances of Super_Type),
    dmo(MetaProperty, Instance_Type)) :-
        semantic_expansion_deo_hierarchy(Sub_Type, Super_Type),
        semantic_expansion(dio(Instance_Type, Sub_Type)).

% -- META EXPANSIONS for AT MODIFIER

% -- single

semantic_meta_expansion_at(dmo(MetaProperty, Property at Type), dmo(MetaProperty,
    Property at Type)) :-
        syntactic_entity_with_meta(Type).

% -- extensions for AT MODIFIER

semantic_meta_expansion_at(dmo(MetaProperty, Property at extensions of Type),
    dmo(MetaProperty, Property at Sub_Type)) :-
        syntactic_entity_with_meta(Type),
        semantic_expansion_deo_hierarchy(Sub_Type, Type).

% -- instances for AT MODIFIER

semantic_meta_expansion_at(dmo(MetaProperty, Property at instances of Meta_Type),
    dmo(MetaProperty, Property at Instance_Type)) :-
        syntactic_entity_with_meta(Meta_Type),
        semantic_expansion(dio(Instance_Type, Meta_Type)).
semantic_meta_expansion_at(dmo(MetaProperty, Property at instances of Super_Type),
    dmo(MetaProperty, Property at Instance_Type)) :-
        syntactic_entity_with_meta(Super_Type),
        semantic_expansion_deo_hierarchy(Sub_Type, Super_Type),
        semantic_expansion(dio(Instance_Type, Sub_Type)).

semantic_meta_expansion_at(dmo(MetaProperty, Property at direct instances of
    Meta_Type), dmo(MetaProperty, Property at Instance_Type)) :-
        syntactic_entity_with_meta(Meta_Type),
        semantic_expansion(dio(Instance_Type, Meta_Type)).

semantic_meta_expansion_at(dmo(MetaProperty, Property at transitive direct
    instances of Meta_Type), dmo(MetaProperty, Property at Instance_Type)) :-
        syntactic_entity_with_meta(Meta_Type),
        semantic_expansion_dio_hierarchy(Instance_Type, Meta_Type).

semantic_meta_expansion_at(dmo(MetaProperty, Property at transitive instances of
    Meta_Type), dmo(MetaProperty, Property at Instance_Type)) :-
        syntactic_entity_with_meta(Meta_Type),
        semantic_expansion_dio_hierarchy(Instance_Type, Meta_Type).
semantic_meta_expansion_at(dmo(MetaProperty, Property at transitive instances of
    Super_Type), dmo(MetaProperty, Property at Instance_Type)) :-
        syntactic_entity_with_meta(Super_Type),
        semantic_expansion_deo_hierarchy(Sub_Type, Super_Type),
        semantic_expansion(dio(Instance_Type, Sub_Type)).

%=============================================================================
% Disjunction
%
% dd/1          (direct disjunction)
%=============================================================================

%% dd/1         (direct disjunction)
semantic_expansion(dd(Type_List)) :-
        semantic_term(dd(Type_List)).

% properties
semantic_expansion(dd(Direct_Property_List)) :-
        setof(P,
              semantic_expansion(dio(P, property)),
              Direct_Property_List).
semantic_expansion(dd(Direct_Property_List)) :-
        setof(P,
              (semantic_expansion_deo_hierarchy(P, Super_Type),
               semantic_expansion(dio(Super_Type, property))
               ),
```

```
              Direct_Property_List).

% top hypertypes
semantic_expansion(dd(Top_Hypertype_List)) :-
        findall(Top_HyperType,
              (semantic_expansion(dmo(hypertype, Top_HyperType)),
               \+ semantic_expansion(deo(Top_HyperType, _))
               ),
              Top_Hypertype_List).

%=========================================================================
% Defeasible semantc terms
%
% defeasible/1 (defeasible assertion)
%=========================================================================

%% defeasible/1
semantic_expansion(defeasible(Defeasible)) :-
        semantic_term(defeasible(Defeasible)).

%=========================================================================
% Ontological relations
%
% dso/2        (direct subset of)
% dro/2        (direct redefition of)
%=========================================================================

%% dso/2        (direct subset of)
semantic_expansion(dso(Type, Super_Type)) :-
        semantic_term(dso(Type, Super_Type)).

%% dro/2        (direct redefition of)
semantic_expansion(dro(Type, Super_Type)) :-
        semantic_term(dro(Type, Super_Type)).
```

Código 38 – Definições auxiliares das expansões das relações semânticas

```
%=========================================================================
% Semantic expansion helpers
%
% forassert_semantic_expansion_dio_hierarchy/3 (helper)
% forassert_semantic_expansion_deo_hierarchy/3 (helper)
%=========================================================================

% forassert_semantic_expansion_dio_hierarchy/3
forassert_semantic_expansion_dio_hierarchy(Type, _, Memo_List) :-
        util_memberchk(Type, Memo_List),
        !.
forassert_semantic_expansion_dio_hierarchy(Type, Meta_Type, _) :-
        semantic_expansion(dio(Type, Meta_Type)).
forassert_semantic_expansion_dio_hierarchy(Type, Meta_Type, Memo_List) :-
        semantic_expansion(dio(Type, X)),
        forassert_semantic_expansion_dio_hierarchy(X, Meta_Type, [Type|Memo_List]).

% forassert_semantic_expansion_deo_hierarchy/3
forassert_semantic_expansion_deo_hierarchy(Type, _, Memo_List) :-
        util_memberchk(Type, Memo_List),
        !.
forassert_semantic_expansion_deo_hierarchy(Type, Super_Type, _) :-
        semantic_expansion(deo(Type, Super_Type)).
forassert_semantic_expansion_deo_hierarchy(Type, Super_Type, Memo_List) :-
        semantic_expansion(deo(Type, X)),
        forassert_semantic_expansion_deo_hierarchy(X, Super_Type, [Type|Memo_List]).
```

APÊNDICE G – Extensão das expansões das relações semânticas para UFO

O presente apêndice complementa o Apêndice F com expansões semânticas específicas da metateoria UFO, conforme abordadas na subseção 8.2.3.

Código 39 – Expansões das relações semânticas para UFO

```
%=============================================================================
% Defesiable dio/2: Definition complementation for dio/2
% Non monotonic behaviour for instance relations based on defeasible/1
%
% dio/2
%=============================================================================

% The priority of defeasible dio/2 is:
% subkind < qualityDimension < quality < property
% defined on dec_sentence_case(monadic_type_quality_assoc_default, ...)
% (MONADIC_TYPE_QUALITY_ASSOC) (subseção 8.2.2.5)
semantic_expansion(dio(Type, quality)) :-
  semantic_term(defeasible(dio(Type, quality))),
  \+ semantic_term(defeasible(dio(Type, property))),
  \+ semantic_term(dio(Type, _)).
semantic_expansion(dio(Type, qualityDimension)) :-
  semantic_term(defeasible(dio(Type, qualityDimension))),
  \+ semantic_term(dio(Type, _)),
  \+ semantic_term(defeasible(dio(Type, property))),
  \+ semantic_term(defeasible(dio(Type, quality))).

% (Quadro 5 (Subkind), restrição 1, p. 171)
semantic_expansion(dio(Type, subkind)) :-
  semantic_term(entity(Type)),
  \+ semantic_term(dio(Type, _)),
  \+ semantic_term(dmo(hypertype, Type)),
  \+ semantic_term(defeasible(dio(Type, _))).

%=============================================================================
% Additional meta properties for dmo/2
%
% dmo/2        (direct meta on)
%=============================================================================

%% Rule for dmo/2
% (Quadro 24 (Meronymic), restrição 2, p. 187) b) {immutable} when {essential} is present
semantic_expansion(dmo(immutable, domain at Meronymic_Rel_Type)) :-
        semantic_expansion(dmo(essential, domain at Meronymic_Rel_Type)).

%% Rule for dmo/2
% (Quadro 4 (Collective), restrição 1, p. 170)
% (Quadro 24 (Meronymic), restrição 2, p. 187)
% c) {essential,immutable} when whole is {extensional}
% (Quadro 26 (SubQuantityOf), restrição 5, p. 189)
% (Quadro 28 (MemberOf), restrição 4, p. 191)
semantic_expansion(dmo(essential, domain at Aggreg_Type)) :-
        semantic_expansion(dmo(extensional, Whole)),
        semantic_term(dpv(codomain at Aggreg_Type, Whole)),
        semantic_expansion_dio_hierarchy(Aggreg_Type, meronymic).

% (Quadro 24 (Meronymic), restrição 5, p. 187)
semantic_expansion(dmo(mandatory, domain at Meronymic_Relation)) :-
        semantic_term(dpv(domain_cc at Meronymic_Relation, Card_Part)),
        semantic_expansion_dio_hierarchy(Meronymic_Relation, meronymic),
        card_greater_or_equals(Card_Part, 1).
semantic_expansion(dmo(mandatory, codomain at Meronymic_Relation)) :-
        semantic_term(dpv(codomain_cc at Meronymic_Relation, Card_Whole)),
        semantic_expansion_dio_hierarchy(Meronymic_Relation, meronymic),
        card_greater_or_equals(Card_Whole, 1).

%% Rule for dmo/2
% (Quadro 27 (SubCollectionOf), restrição 5, p. 190) transivity of {{essential}}
```

```
% IT IS NOT SAFE: it falls into a loop when there is a circular inheritance
semantic_expansion(dmo(essential, domain at Aggreg_SubType)) :-
        semantic_expansion_deo_hierarchy(Aggreg_SubType, Aggreg_Type),
        Aggreg_SubType \= Aggreg_Type,
        \+ semantic_term(dmo(essential, domain at Aggreg_SubType)),
        semantic_expansion(dmo(essential, domain at Aggreg_Type)).

%=============================================================================
% Definition complementation: drel/5
%
% Due to a syntactic sugar of quality particular declaration, the "qfrom" relation
% might be infered whether the Quality_Type of the quality particular is in a
% dassoc/2 relation to one and only one Quality_Structure_Type. Otherwise, qfrom/2
% have to be explicitally declared and a semantic crule/n might be triggered.
%
% dio/2
% drel/5
%=============================================================================

%% drel(Q_From, Quale_Particular, Quality_Structure_Type)
semantic_expansion(DERIVED_QFROM) :-
        semantic_term(dio(Quale_Particular, quale)),
        \+ ( semantic_term(dio(QFrom, qfrom)),
             semantic_term(dpv(domain at QFrom, Quale_Particular))
           ),
        semantic_term(dpv(domain at QL_Rel, Quale_Particular)),
        semantic_term(dpv(codomain at QL_Rel, Quality_Particular)),
        semantic_term(dio(QL_Rel, ql)),
        semantic_term(dio(Quality_Particular, Quality_Type)),
        setof(QS_Type,
              (
                 %semantic_expansion_deo_hierarchy(Quality_Type,
    Quality_Type_Super),
                 %Quality_Type_Super = Quality_Type,
                 qfrom_semantic_expansion_deo_hierarchy(Quality_Type,
    Quality_Type_Super),

                 semantic_term(dio(DAssoc_Rel, dassoc)),
                 semantic_term(dpv(domain at DAssoc_Rel, Quality_Type_Super)),
                 semantic_term(dpv(codomain at DAssoc_Rel, QS_Type))
              ),
              QS_Type_List),
        QS_Type_List = [Quality_Structure_Type],
        syntactic_concat_3(Quale_Particular, '_qfrom_', Quality_Structure_Type,
    Q_From_Rel),
        semantic_expansion_derived_QFROM(Q_From_Rel, Quale_Particular,
    Quality_Structure_Type, DERIVED_QFROM).

semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       entity(Q_From_Rel)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       dio(Q_From_Rel, qfrom)).
semantic_expansion_derived_QFROM(Q_From_Rel, Quale_P, _, dpv(domain at Q_From_Rel,
    Quale_P)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       dpv(domain_cc at
    Q_From_Rel, 1)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, QS_Type, dpv(codomain at
    Q_From_Rel, QS_Type)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       dpv(codomain_cc at
    Q_From_Rel, 1)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       entity(domain at
    Q_From_Rel)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       entity(domain_cc at
    Q_From_Rel)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       entity(codomain at
    Q_From_Rel)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       entity(codomain_cc at
    Q_From_Rel)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       dio(domain at Q_From_Rel,
    domain)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       dio(domain_cc at
    Q_From_Rel, domain_cc)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       dio(codomain at
    Q_From_Rel, codomain)).
semantic_expansion_derived_QFROM(Q_From_Rel, _, _,       dio(codomain_cc at
    Q_From_Rel, codomain_cc)).

%% qfrom_semantic_expansion_deo_hierarchy(Type, SuperType)
% qfrom_semantic_expansion_deo_hierarchy/2
% (Insecure) reflexive case for semantic_expansion_deo_hierarchy/2
qfrom_semantic_expansion_deo_hierarchy(T, T).
qfrom_semantic_expansion_deo_hierarchy(T, S) :-
     semantic_expansion_deo_hierarchy(T, S).
```

Índice