Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Geração Automática de Ontologias para a Web Semântica

Carlos de Oliveira Bravo

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Célia Ghedini Ralha

Brasília
2010

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenadora: Prof.ª Dr.ª Alba Cristina Magalhães Alves de Melo

Banca examinadora composta por:

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora) — CIC/UnB
Prof.ª Dr.ª Marisa Brascher — CID/UnB
Prof.ª Dr.ª Renata Vieira — FACIN/PUC-RS

CIP — Catalogação Internacional na Publicação

Bravo, Carlos de Oliveira.

Geração Automática de Ontologias para a Web Semântica / Carlos de Oliveira Bravo. Brasília : UnB, 2010.
151 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2010.

1. semântica, 2. web, 3. ontologia, 4. léxico, 5. gramática

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Geração Automática de Ontologias para a Web Semântica

Carlos de Oliveira Bravo

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora)
CIC/UnB

Prof.ª Dr.ª Marisa Brascher
CID/UnB

Prof.ª Dr.ª Renata Vieira
FACIN/PUC-RS

Prof.ª Dr.ª Alba Cristina Magalhães Alves de Melo
Coordenadora do Mestrado em Informática

Brasília, 5 de março de 2010

Dedicatória

Dedico este trabalho à minha esposa, Luciana, e aos meus filhos, Humberto e Marília.

Agradecimentos

Muitas pessoas me apoiaram neste projeto, que não poderia ter sido concluído, de outro modo. Entre elas destacam-se meus pais, Januário Guarany Guagliani Bravo e Marieta de Oliveira Bravo, pela presença constante e ajuda incondicional em todas as situações; e minha esposa, Luciana de Souza Pereira Bravo, pelo apoio de sempre e pelos comentários que se mostraram decisivos para o desenvolvimento do trabalho.

Não posso deixar de agradecer a Rui Oscar Dias Janiques (SSASEN/Prodasen) e Rosemary Schietti Assumpção (SSASEN/Prodasen), pelo apoio nos momentos mais importantes desde que este trabalho foi iniciado.

Meus agradecimentos ao amigo José Maurício Nunes Mendes, pelas dicas de trabalho e pelo exemplo de competência e profissionalismo.

Agradeço também a Weliton Moreira Bastos, grande guerreiro, pelo trabalho sério que executamos em conjunto durante as tarefas relacionadas às disciplinas do mestrado, especialmente em Teoria da Computação.

Meus agradecimentos a Wanda Conceição de Sousa, pela inestimável ajuda.

Um agradecimento especial ao Prof. Dr. José Carlos Ralha (CIC/UnB), cujas sugestões foram de grande valia desde que o trabalho foi iniciado.

Agradeço também à minha orientadora, Prof.ª Dr.ª Célia Ghedini Ralha (CIC/UnB), pelas valiosas orientações fornecidas durante todo o trabalho.

À Prof.ª Dr.ª Marisa Brascher (CID/UnB) e à Prof.ª Dr.ª Renata Vieira (FACIN/PUC-RS), meus agradecimentos pelas oportunas observações acerca deste trabalho.

Abstract

The current Semantic Web specifications provide the necessary support for a broad range of applications. Most of these applications require direct or indirect ontology handling. Therefore the ontologies play a crucial role in the universe of the Semantic Web, working trustful sources of knowledge, from which one can establish and validate relationships between conceptual elements related to applications.

The task of developing an ontology is not trivial, as it requires the work of people with reasonable degree of knowledge in the considered application field. In the Web´s context, this task becomes more difficult due to the large dynamism in the generation of content related to the applications.

This work describes a process and a prototype to automatically obtain ontologies through the combination of grammatical elements present in texts written in Portuguese language, allowing the merge to preexisting ontologies. The idea is to start from the syntactic analysis of Web texts, where the tool structures the elements using a syntactical analyser (PALAVRAS) and a basic Portuguese language ontology to associate to definitions found in preexisting ontologies. The resulting ontology is a useful artifact to the Semantic Web, once it reflects the original syntactic structure of the texts from different perspectives of concept relationships, which are allowed by the preexisting ontologies merged to the preliminar ontology.

Keywords: semantics, web, ontology, lexical, grammar

Resumo

As atuais especificações da Web Semântica fornecem o suporte necessário para uma ampla variedade de aplicações. Essas aplicações necessitam, em sua grande maioria, da manipulação direta ou indireta de ontologias. As ontologias desempenham, portanto, um papel crucial dentro do universo da Web Semântica, funcionando como fontes confiáveis de conhecimento a partir das quais é possível estabelecer e validar relações entre os elementos conceituais tratados pelas aplicações.

A tarefa de elaboração de uma ontologia não é trivial, pois requer o trabalho de pessoas com razoável grau de conhecimento nas áreas de aplicação envolvidas. No contexto da Web, essa tarefa torna-se mais difícil devido ao grande dinamismo na geração de conteúdo a ser tratado pelas aplicações.

Este trabalho descreve um processo e um protótipo implementado para obtenção automática de ontologias baseadas na combinação de elementos gramaticais presentes em textos da língua portuguesa, permitindo uma operação de fusão com ontologias pré-existentes. A idéia é que a partir da análise sintática de textos da Web a ferramenta estruture, de acordo com uso de um analisador sintático (PALAVRAS) e uma ontologia base da língua portuguesa, os elementos gramaticais identificados no texto, associando-os às definições encontradas em ontologias pré-existentes. A ontologia resultante do processo constitui-se em um artefato útil para a Web Semântica, por manter a estrutura do texto original sob diferentes perspectivas de relacionamentos entre os conceitos, os quais são viabilizados pelas ontologias pré-existentes fundidas à ontologia preliminar.

Palavras-chave: semântica, web, ontologia, léxico, gramática

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Abreviaturas

ABox	*Assertion box*.
AL	*Attributive Language*.
API	*Application Programming Interface*.
DAML	*DARPA Agent Markup Language*.
DAML+OIL	*DARPA Agent Markup Language (DAML) + Ontology Inference Layer (OIL)*.
DAML-ONT	*DARPA Agent Markup Language (for ontologies)*.
DARPA	*Defense Advanced Research Projects Agency*.
DL	*Description Logics*.
GRDDL	*Gleaning Resource Descriptions from Dialects of Languages*.
GWA	*Global WordNet Association*.
HTML	*HyperText Markup Language*.
IA	Inteligência Artificial.
IRI	*Internationalized Resource Identifier*.
LD	Lógica Descritiva.
LPO	Lógica de Primeira Ordem.
MP	*Modus Ponens*.
MT	*Modus Tollens*.
OCR	*Optical Character Recognition*.
OIL	*Ontology Inference Layer*.
OWL	*Web Ontology Language*.
OWL 2	*Web Ontology Language - Revision 2*.
PICS	*Platform for Internet Content Selection*.
PLN	Processamento de Linguagem Natural.
POWDER	*Protocol for Web Description Resources*.
RDF	*Resource Description Framework Core Model*.
RDFa	*RDF in Attributes*.

RDFS	*RDF Schema*.
RIF	*Rule Interchange Format*.
SAWSDL	*Semantic Annotations for WSDL and XML Schema*.
SKOS	*Simple Knowledge Organization System*.
SPARQL	*SPARQL Protocol and RDF Query Language*.
TBox	*Terminology box*.
Turtle	*Terse RDF Triple Language*.
URI	*Uniform Resource Identifier*.
W3C	*World Wide Web Consortium (W3C)*.
WSDL	*Web Services Description Language*.
XCES	*Corpus Encoding Standard for XML*.
XHTML	*eXtensible HyperText Markup Language*.
XML	*eXtensible Markup Language*.

Capítulo 1

Introdução

Depois do rádio e da televisão, meios de comunicação unidirecionais, contemplamos a chegada da Internet e da Web. Muito mais do que um meio de comunicação, a Web facilitou a interatividade entre pessoas, organizações e máquinas.

Além da interatividade, o sucesso da Web deve-se, em parte, à sua arquitetura descentralizada e heterogênea. Mas a Web de hoje é, sem dúvida, voltada exclusivamente para seres humanos. Na maior parte dos casos as máquinas operam apenas nos bastidores da rede, armazenando informações e executando tarefas de suporte. Entretanto, o grande volume de dados disponíveis na rede nos obriga a exigir das máquinas uma forma de processamento mais ativo e inteligente: o tratamento semântico das informações.

No âmbito da Web, não podemos ignorar a possibilidade de vermos, em um futuro muito próximo, máquinas desempenhando funções mais complexas. Filtrando, classificando e tratando semanticamente as informações, as máquinas poderão executar tarefas cada vez mais sofisticadas. Essas tarefas deverão ser executadas em um nível de abstração mais alto do que o atual.

É nesse cenário que desponta a Web Semântica, uma extensão da Web proposta por Berners-Lee et al. [2001] destinada a prover os recursos necessários para um tratamento mais elaborado da informação e dessa maneira favorecer a interação entre pessoas e computadores. Como extensão, a Web Semântica não substitui a Web atual, mas a complementa.

No contexto da Web Semântica, usuários e máquinas co-existirão em uma rede repleta de serviços especializados, os chamados "serviços da Web". Esses serviços poderão ser utilizados tanto por pessoas quanto por máquinas, desde que estejam continuamente em conformidade com a estrutura atualizada da Web Semântica.

A atual especificação da Web Semântica (W3C [2002]) baseia-se em uma plataforma organizada em camadas, preparadas para suportar uma necessidade fundamental do processamento automatizado de dados semânticos: um padrão para a representação do conhecimento.

A representação de conhecimento necessária para a Web Semântica difere da normalmente considerada em sistemas de Inteligência Artificial (IA). Esses sistemas precisam contemplar diversos aspectos do conhecimento importantes não apenas para viabilizar a sua representação, mas também para solucionar questões muito complexas, como as referentes ao aprendizado automatizado e à resolução de problemas. Na Web Semântica, por outro lado, o foco é concentrado em uma forma confiável de representação de co-

nhecimento que viabilize o compartilhamento e o processamento automatizado de dados semânticos.

As ontologias, estruturas de representação do conhecimento nas quais as informações possuem significados bem definidos, transformaram-se, nos últimos anos, em peças essenciais para o desenvolvimento da Web Semântica.

Ontologias devem ser construídas em bases sólidas e bem definidas. Por esse motivo, o trabalho de elaboração de uma ontologia geralmente é conduzido por especialistas. Trata-se de um processo complexo, sempre passível de revisão.

A complexidade inerente ao desenvolvimento das ontologias, aliada ao grande dinamismo na produção de informações para a Web, evidenciam a importância de recursos que facilitem o processo de elaboração de ontologias.

Este trabalho apresenta um processo automatizado de geração destinado a mapear o conteúdo textual disponível na Web na forma de ontologias para auxílio à Web Semântica. Esse processo baseia-se na análise sintática de textos escritos em língua portuguesa para produzir artefatos que podem efetivamente ser aproveitados em aplicações específicas para a Web Semântica.

1.1 Motivação

Além da definição de um processo para geração de ontologias a partir de textos da Web, podemos citar as outras motivações que impulsionam este projeto:

- trata-se de um trabalho que utiliza como alvo de estudo a língua portuguesa, cuja gramática é bastante rica, mas ainda com carência de trabalhos em escala mundial;
- possibilidade de utilizar as ferramentas de manipulação de ontologias disponíveis atualmente para realizar operações de inferência sobre as informações contidas nas ontologias geradas;
- facilidade de identificação de conceitos e relações através da análise das ontologias geradas;
- possibilidade de adaptação do processo em aplicações que envolvam a interligação de conteúdo da Web e a busca semântica.

Outro aspecto importante é que o conteúdo textual representado em uma ontologia pode ser utilizado de diversas formas, garantindo o reuso dessas ontologias. Essa característica fica mais clara quando examinamos, por exemplo, transcrições da linguagem falada, cujas representações escritas podem ter várias interpretações.

Texto enfatizado	Interpretação
Eu vou comprar	"Pode deixar que eu compro, mais ninguém."
Eu **vou** comprar	"Comprarei, com certeza."
Eu vou **comprar**	"Eu vou comprar; não fabricarei nem pedirei emprestado."

Tabela 1.1: Exemplo de diferentes interpretações de um texto.

A Tabela 1.1 ilustra esse exemplo para a frase "*Eu vou comprar.*", onde a ênfase das palavras está destacada nos textos para ilustrar as diferentes interpretações. As

interpretações são diferentes, mas todas elas compartilham o fato de que existe um agente, "*Eu*" (o sujeito da frase), que executará uma ação no futuro, "*vou*" (verbo auxiliar, tempo verbal futuro), e que essa ação é "*comprar*". Muitas outras interpretações são possíveis, dependendo do contexto, sendo que a estrutura sintática do texto está presente em todas elas. Uma ontologia representativa do texto é igualmente útil, o que motiva a sua adoção em diversas aplicações.

1.2 Objetivos

O objetivo principal deste trabalho é descrever um processo que seja capaz de gerar, automaticamente, ontologias a partir de conteúdo, em língua portuguesa, disponível na Web dentro de páginas elaboradas com *HyperText Markup Language* (HTML).

A partir desse objetivo geral, derivam os seguintes objetivos específicos:

- preservar, nas ontologias geradas pelo processo, a estrutura sintática dos textos, evitando efetuar atribuições semânticas com base no significado das palavras;
- identificar as classes gramaticais mais importantes para o mapeamento da estrutura sintática dos textos;
- definir uma *ontologia base* que contenha essas classes gramaticais selecionadas e que possa mapear todos os elementos constituintes das ontologias geradas.
- identificar as classes gramaticais dos elementos estruturais dos textos e criar, nas ontologias geradas, os vínculos necessários para preservar as estruturas identificadas;
- descrever cada etapa do processo e seus resultados intermediários;
- implementar um protótipo de geração automática de ontologias que ilustre cada etapa descrita.

1.3 Justificativa

Ontologias geradas automaticamente são importantes para o trabalho de construção de ontologias de forma geral, funcionando não apenas como elos de referência entre o que está disponível na Web e a sua interpretação, mas também como mapas para a informação semântica contida nos textos.

Domínios que sofrem freqüente atualização de dados podem ser beneficiados com a adoção do processo de geração de ontologias aqui descrito. Isso é especialmente verdadeiro para os sítios da Web que são atualizados diariamente, pois seria muito difícil manter, nesses sítios, ontologias constantemente sincronizadas com o conteúdo modificado apenas com processos puramente manuais ou interativos.

Por último, a construção de ontologias é um tema relevante na Web Semântica, e a definição de um processo automatizado de geração de ontologias é um objeto de pesquisa intrigante e rico.

1.4 Trabalhos Correlatos

A pesquisa e torno da construção de ontologias a partir de conteúdo textual já conta com uma variedade de trabalhos destinados à obtenção de ontologias de forma automática ou semi-automática. Destacamos três projetos que compartilham, com este trabalho, muitas características: *Text-to-Onto*, *Artequakt* e *OntoLP*.

Text-to-Onto

Text-to-Onto (Maedche and Staab [2000]) é um ambiente de construção de ontologias que procura obter estruturas conceituais a partir de textos da língua inglesa de forma semi-automática, conforme mostrado na Figura 1.1.

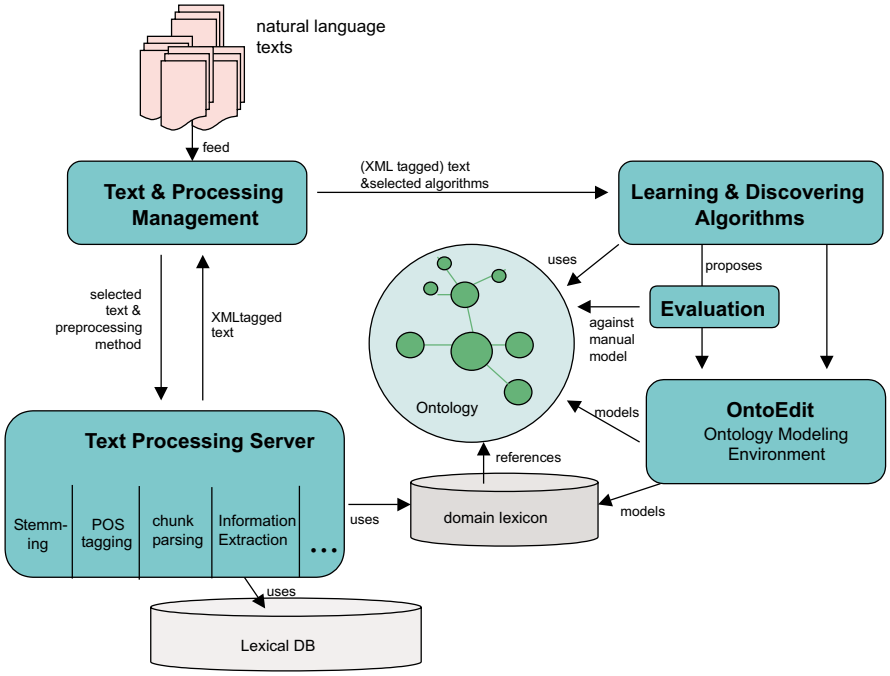

Figura 1.1: Arquitetura do sistema *Text-to-Onto* (Maedche and Staab [2000]).

No sistema *Text-to-Onto*, textos em linguagem natural são submetidos a processos de mapeamento léxico e análise estatística, os quais contam com a ajuda de um banco de dados léxico com cerca de 120 mil palavras. O resultado desses processos é um conjunto de textos com anotações em formato XML, o qual é submetido a um processo de mapeamento conceitual sobre taxonomias pré-existentes. A partir desse ponto são aplicados algoritmos para a obtenção de informações conceituais adicionais e a ontologia resultante passa por um processo de avaliação que depende de intervenção humana.

Em contraste com este trabalho, as ontologias processadas no ambiente *Text-to-Onto* são geradas de forma semi-automática, devido às fases de avaliação e edição das ontologias, ao longo do processo. Neste trabalho, as ontologias são geradas sem interação com o usuário, através de parâmetros que incluem um conjunto de ontologias pré-existentes. Esse conjunto de ontologias desempenha o mesmo papel das taxonomias pré-existentes

utilizadas no ambiente *Text-to-Onto*. Não obstante, as tarefas executadas pelo último são muito mais abrangentes, incluindo a avaliação interativa de resultados, realimentação de conteúdo e implementação de regras de extração de informações que caracterizam um processo de aprendizagem.

Artequakt

Artequakt (Alani et al. [2002]) é um sistema automatizado de extração de dados da Web a respeito de artistas, concebido com o propósito de construir uma base de conhecimento útil para a elaboração automática de biografias.

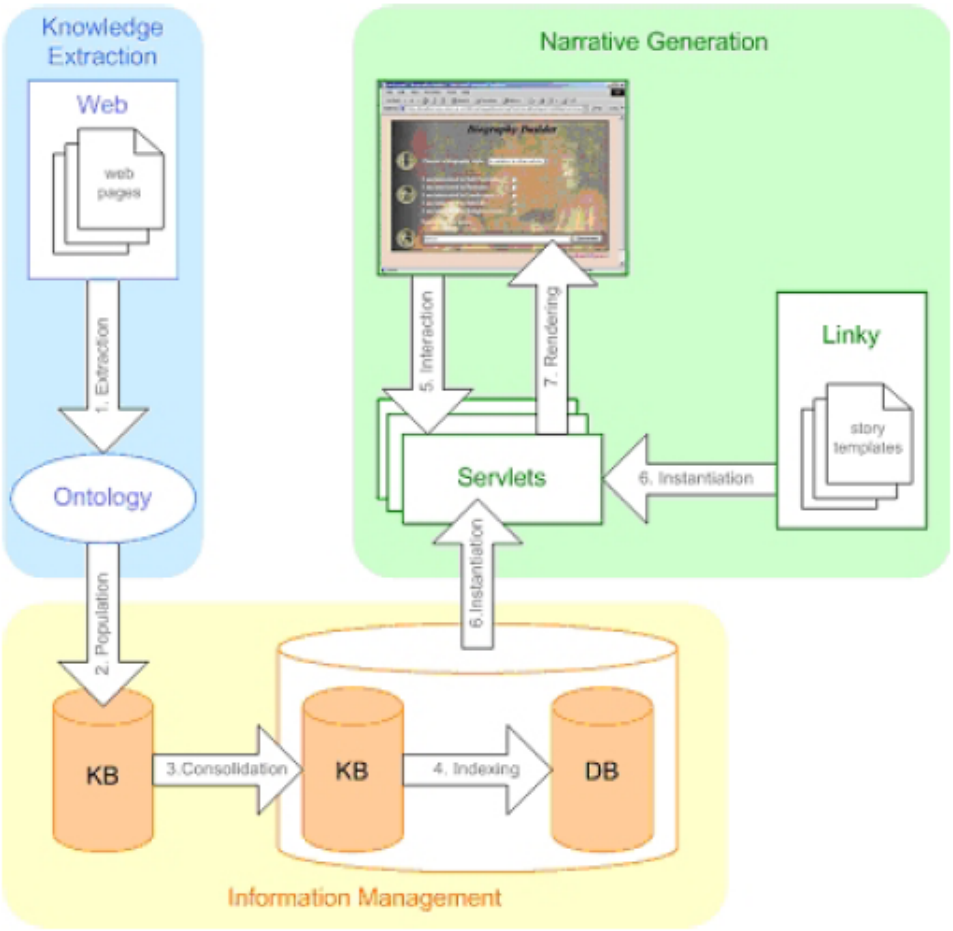

Figura 1.2: Arquitetura do sistema *Artequakt* (Alani et al. [2002]).

A arquitetura desse sistema é mostrada na Figura 1.2. Após uma seleção prévia de documentos da Web, o sistema quebra cada documento em parágrafos e depois em sentenças. Cada parágrafo é analisado sintaticamente e semanticamente para a identificação de qualquer conhecimento relevante. Os principais componentes das sentenças são então identificados com a ajuda de um dicionário, para depois serem gravados em uma base de conhecimento.

Embora o sistema *Artequakt* tenha seu domínio restrito a biografias de artistas, seus métodos de extração de informações e de geração de ontologias são, em muitos aspectos,

semelhantes aos adotados neste trabalho. No sistema *Artequakt*, os elementos extraídos a partir de conteúdos textuais são classificados a partir de uma consulta a uma base de conhecimento, presente em um servidor de ontologias. Neste trabalho, essa classificação se dá de forma semelhante, através da combinação do resultado da análise gramatical com o conteúdo de ontologias que classificam elementos previamente identificados. Entretanto, neste trabalho não existe um servidor de ontologias, de modo que estas devem ser elaboradas à parte, ou devem incorporar ontologias geradas previamente pelo processo.

OntoLP

OntoLP (Junior [2008]) é um *plug-in* para o editor *Protégé* (vide Seção 3.5) que auxilia o processo de construção de ontologias em língua portuguesa. O *plug-in* trabalha com um *corpora* (vide Seção 3.5) representado no formato XCES (Vassar College and LORIA/CNRS. [2010]). O *corpora* é submetido a um processo de extração de termos simples e complexos. Os termos extraídos passam então por uma etapa de organização hierárquica, a qual é responsável pela geração taxonomias que podem ser editadas posteriormente. A Figura 1.3 ilustra o funcionamento do *OntoLP*.

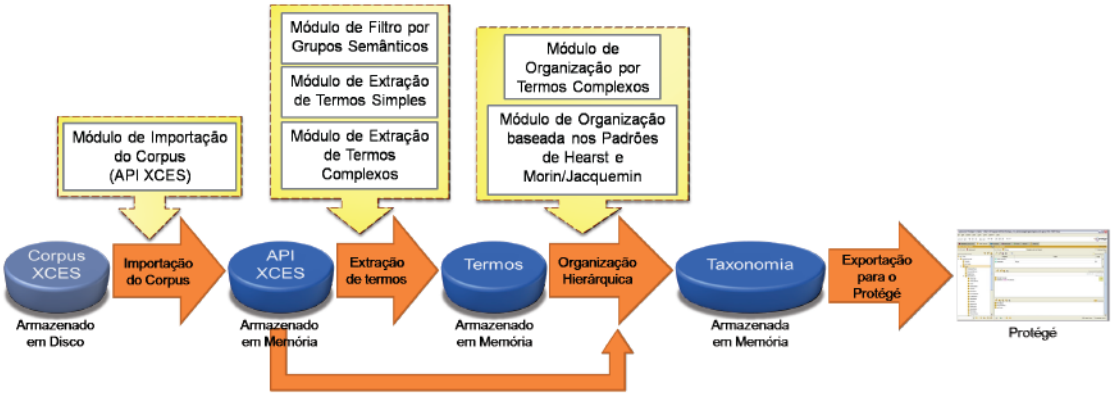

Figura 1.3: Visão geral do funcionamento do *OntoLP* (Junior [2008]).

De modo análogo a este trabalho, o *OntoLP* utiliza internamente o sistema PALAVRAS (vide Seção 5.4.2) para efetuar a extração de informações a partir de textos em língua portuguesa. Os processos implementados por ambos mantém, por esse motivo, muitas similaridades. Entretanto, o *OntoLP* destina-se ao uso interativo, dentro de um editor de ontologias, sendo sua principal característica a organização hierárquica de termos contidos nos textos de origem a partir de uma marcação semântica de alto nível. Este trabalho, por outro lado, destina-se ao uso não interativo e ao processamento em lote, necessitando que a classificação semântica dos termos presentes nos textos de origem estejam em um conjunto pré-existente de ontologias.

1.5 Estrutura do Documento

O restante deste documento está organizado da seguinte forma:

- o Capítulo 2 mostra os padrões já estabelecidos para a Web Semântica, alguns exemplos de uso dessa nova tecnologia e outros assuntos correlatos;

- o Capítulo 3 discorre sobre a estrutura das ontologias, seus elementos constituintes e demais tópicos a elas relacionados;
- o Capítulo 4 apresenta uma breve introdução às lógicas formais relacionadas às ontologias;
- o Capítulo 5 mostra os aspectos relacionados à análise sintática de conteúdos textuais e apresenta alguns trabalhos importantes na área de tratamento de linguagem;
- o Capítulo 6 descreve o processo de geração automática de ontologias definido e desenvolvido neste trabalho;
- o Capítulo 7 encerra este documento apresentando as principais conclusões acerca do trabalho desenvolvido, incluindo sugestões de possíveis trabalhos futuros com novas implementações e expansões.

A Figura 1.4 mostra a relação de dependência entre os capítulos.

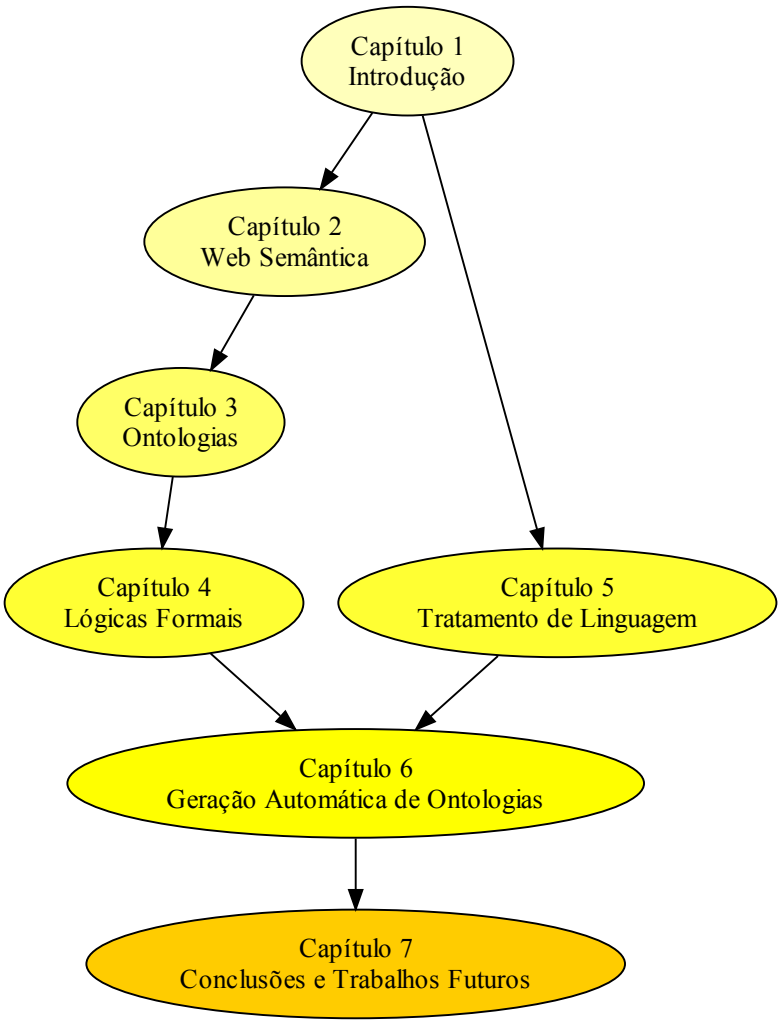

Figura 1.4: Estrutura do documento e dependência entre os capítulos.

Capítulo 2

Web Semântica

Segundo Berners-Lee et al. [2001], a Web Semântica, também conhecida como *Web 3.0*, é uma extensão da Web atual, na qual é dada à informação um significado semanticamente definido, permitindo que computadores e pessoas trabalhem em cooperação.

Do ponto de vista prático, a Web Semântica funcionará em conjunto com a Web para expor conceitos e suas relações intrínsecas, enriquecendo o conteúdo existente com informações implícitas, resumos e ligações entre conceitos. Os mecanismos necessários para essa nova realidade basear-se-ão sobretudo no processamento automatizado das informações disponíveis na Web.

Esse processamento automatizado não pode existir quando as informações têm múltiplos significados ou não estão definidas claramente. De acordo com Breitman [2005], na Web Semântica a informação terá significado bem definido através de linguagens de marcação semântica. Caminhamos nesse sentido, atualmente, na medida em que utilizamos a plataforma proposta do *World Wide Web Consortium* (W3C) para a Web Semântica.

De acordo com Berners-Lee [2008], as estruturas necessárias para o crescimento da Web Semântica já estão bem fundamentadas, sendo que a Web Semântica está sendo vista de maneira diferente por diferentes grupos sociais. Desse modo, o autor classifica a Web Semântica como sendo um vasto conjunto de tecnologias utilizadas de diferentes formas por diferentes comunidades.

Até chegar a um patamar de confiabilidade aceitável, no qual seres humanos e máquinas podem interagir e trocar informações representadas semanticamente através da Web, o desafio da Web Semântica consistirá, por um lado, na interpretação automatizada de conteúdo para exposição de informações semânticas e, por outro lado, na recuperação de informações semânticas a partir de conteúdos previamente estruturados. Segundo Iskold [2008], esses dois caminhos caracterizam, respectivamente, abordagens *bottom-up* e *top-down* (Figura 2.1).

A abordagem *bottom-up* é a chamada abordagem clássica, adotada pelo W3C desde o princípio. Ela parte da idéia de que o conteúdo pode ser "anotado" com informações semânticas e que essas anotações servem de base para a representação do conhecimento necessário para as aplicações da Web Semântica. Um grande arsenal de ferramentas já está disponível para trabalhos nesse sentido.

O caminho inverso caracteriza a abordagem *top-down*, uma nova linha de pensamento que amplia o leque de opções para o pleno desenvolvimento da Web 3.0. A idéia consiste em obter dados semânticos e relacionamentos diretamente a partir do conteúdo já dis-

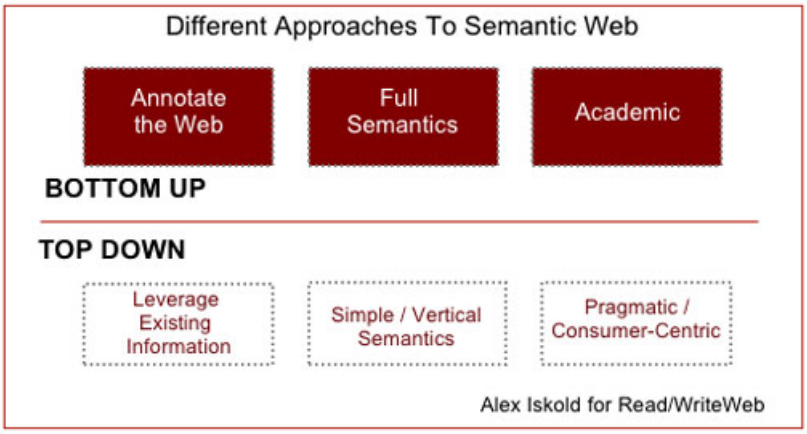

Figura 2.1: Abordagens bottom-up e top-down da Web Semântica (Iskold [2008]).

ponível na Web, sem entretanto realizar processamentos sofisticados como os vistos em algoritmos de IA para o reconhecimento de linguagem natural (vide Capítulo 5).

A Web Semântica tem evoluído bastante com a utilização de ontologias (vide Capítulo 3), que agora desempenham um papel central, permitindo a troca de informações semânticas referentes aos conteúdos disponíveis na Web e funcionando como fontes de termos precisamente definidos (Horrocks [2002]).

A sistemática de utilização de ontologias amplia sensivelmente o leque de opções de aplicações possíveis na Web, não apenas pela capacidade de manter organizados os dados não estruturados, mas também por permitir a acomodação ou mapeamento de dados estruturados disponíveis em bancos de dados subjacentes.

Como se vê, diferentes abordagens prestam-se a diferentes objetivos, não sendo possível definir um propósito único e menos genérico para a Web Semântica do que o de simplesmente ampliar as possibilidades de utilização da Web atual. O ponto chave é que o tratamento semântico da informação será crucial para transformar a Web atual em um ambiente ainda mais produtivo para o ser humano (Alesso and Smith [2006]).

2.1 História

A idéia de adicionar informações semânticas ao conteúdo da Web remonta à proposta original da própria Web, como se pode ver no diagrama mostrado na Figura 2.2 (Berners-Lee [1989]).

Embora o foco, na época, fosse a criação de um sistema distribuído de hipertexto, alguns conceitos diretamente relacionados à Web Semântica já estavam presentes naquela visão inicial. O uso de metadados (vide Seção 2.2.1) para descrever documentos mostra alguns traços do que viria a ser a proposta de uma Web Semântica. Além disso, já existia a noção de conceitos (chamados de "nós") e a adoção de algumas relações básicas entre esses conceitos.

Posteriormente, em 1999, Tim Berners-Lee apresentou a seguinte declaração em Berners-Lee and Fischetti [1999]:

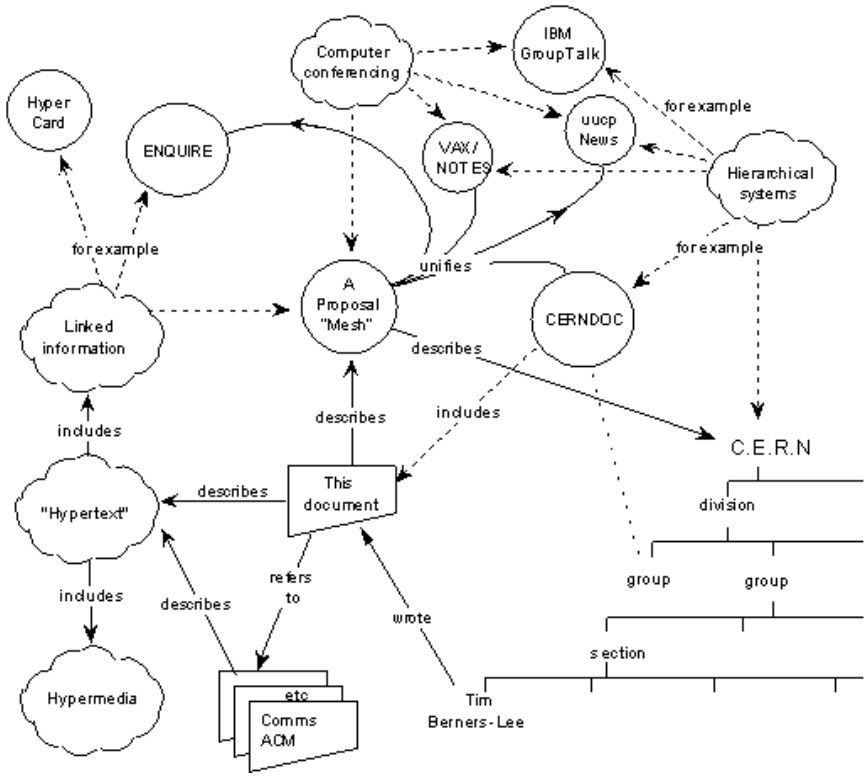

Figura 2.2: Proposta inicial da Web (Berners-Lee [1989].

> *I have a dream for the Web in which computers become capable of analyzing all the data on the Web - the content, links, and transactions between people and computers. A "Semantic Web", which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialize.*

Naquele momento, a proposta da Web Semântica já apresentava a estrutura mostrada na Figura 2.3.

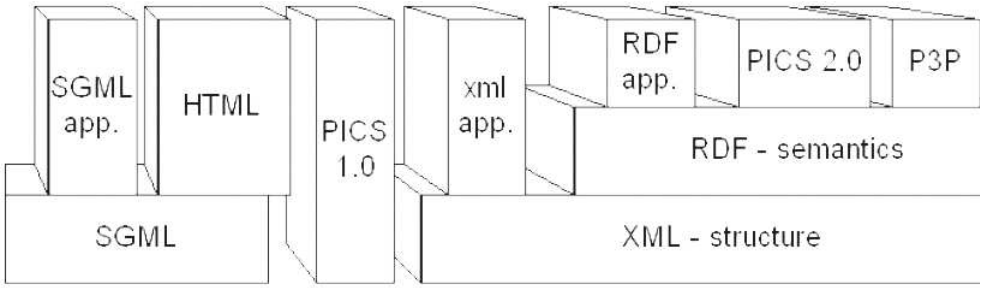

Figura 2.3: Web Semântica em 1997 (Veltman [2004]).

As questões e debates da época giravam em torno do uso de HTML, *Platform for Internet Content Selection* (PICS) e RDF ((vide Seção 2.2)) na descrição dos conteúdos dos sites. O formato HTML, apesar de ter sofrido uma notável evolução desde o seu

surgimento, 9 anos antes (Palmer [2004]), não contemplava as principais necessidades de uma Web Semântica. Os recursos do PICS (vide W3C [2009h]) eram interessantes, mas não suficientes para a descrição do conteúdo da Web. Surgiu então o RDF, que de fato é uma extensão do PICS (Boye [1998b]). O desenvolvimento do formato RDF foi iniciado em 1997 (Boye [1998a]) e vem servindo de base para novos desenvolvimentos até hoje.

2.2 Estrutura Atual

A estrutura atual da Web Semântica é a mostrada na Figura 2.4.

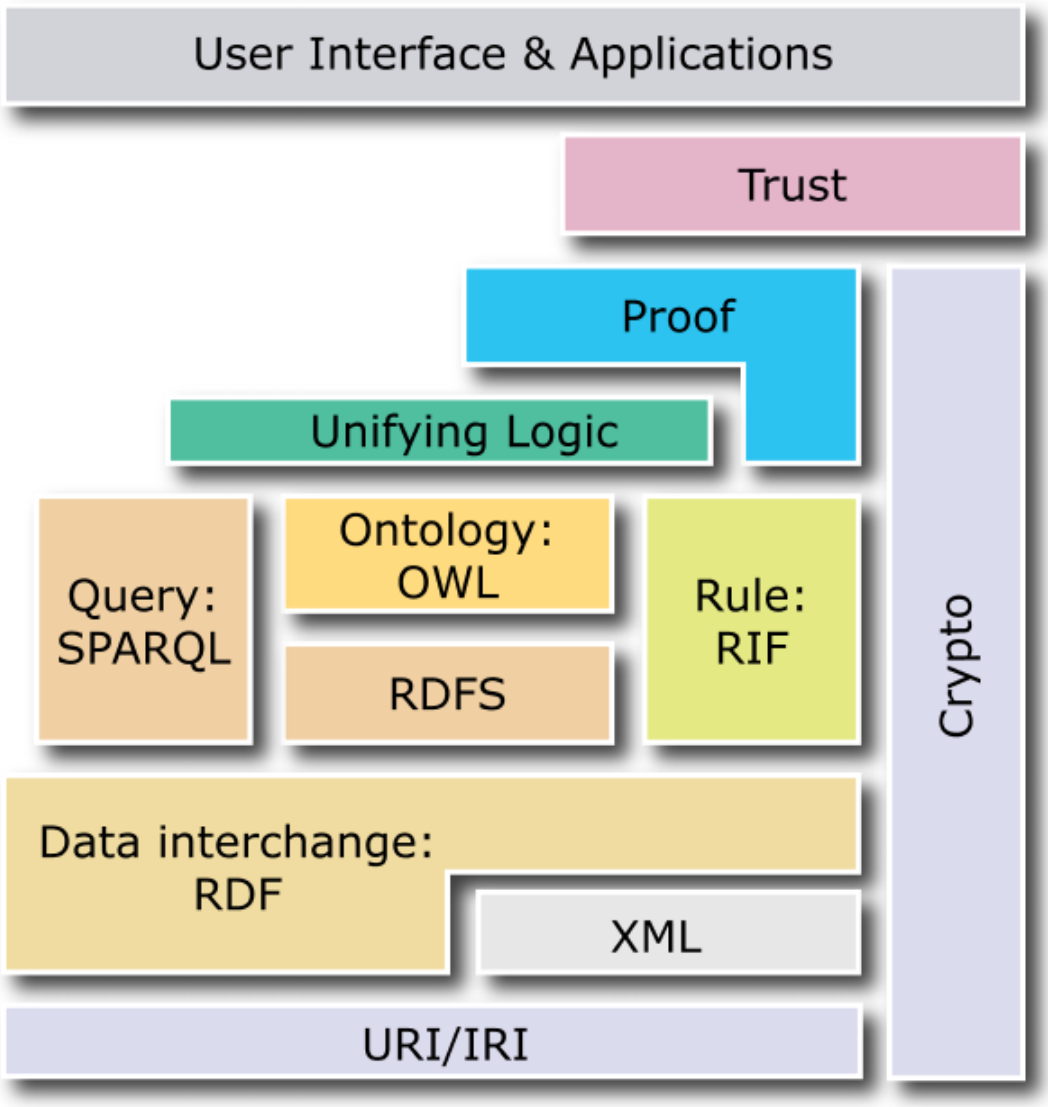

Figura 2.4: Estrutura atual da Web Semântica (W3C [2010]).

As camadas que compõe essa estrutura estão em constante evolução, de modo que a própria estrutura tem sofrido alterações razoavelmente freqüentes, na medida em que alguns componentes são substituídos por outros mais abrangentes ou mais eficazes. Segue uma breve descrição da estrutura:

- a camada "*URI/IRI*" estabelece o uso de *Uniform Resource Identifiers* (URI) e *Internationalized Resource Identifiers* (IRI) como padrões para assegurar o uso de padrões de codificação de caracteres internacionais e prover meios de identificar recursos na Web Semântica;
- a camada "*XML*" possui definições que permitem a integração das definições feitas na Web Semântica com outros padrões baseados em XML;
- as camadas "*RDF*" e "*RDFS*" permitem descrever objetos com URIs, definir vocabulários que também podem ser referenciados por URIs e atribuir definições de tipo para recursos;
- a camada "*Ontology*" suporta a evolução de vocabulários através da manutenção de ontologias;
- a camada "*Query*" permite a busca de informações na Web Semântica;
- a camada "*Rule*" permite a troca de informações entre sistemas baseados em regras;
- a camada vertical "*Crypto*", fornece os padrões de assinatura digital e criptografia necessários para garantir a segurança na troca de informações da Web Semântica;
- a camada "*Unifying Logic*" é destinada a prover os padrões necessários para a definição de linguagens lógicas com poder de expressividade suficiente para a implementação de novos mecanismos de inferência na Web Semântica;
- a camada "*Proof*" está destinada a prover serviços apropriados para a validação de definições feitas na Web Semântica;
- a camada "*Trust*" está destinada a oferecer confiabilidade à Web Semântica através de mecanismos que permitam, por exemplo, definir políticas de segurança e regras de acesso a fontes de informações;
- a camada "*User Interface & Applications* é formada pelos sistemas que utilizam a Web Semântica como plataforma de execução.

As camadas "*Unifying Logic*", "*Proof*" e "*Trust*" ainda estão em desenvolvimento e são consideradas importantes plataformas de pesquisa da Web Semântica.

Segundo W3C [2009l], os padrões atuais estabelecidos para a Web Semântica são:

- *Resource Description Framework Core Model* (RDF), uma linguagem de descrição de metadados (vide Seção 2.2.1) e de conteúdo semântico (W3C [2004a]);
- *Web Ontology Language* (OWL), uma linguagem de definição de ontologias (W3C [2009a]);
- *SPARQL Protocol and RDF Query Language* (SPARQL), a linguagem de pesquisa utilizada para recuperar e manipular informações gravadas em formato RDF ((W3C [2008b]));
- *RDF in Attributes* (RDFa), uma especificação que permite o uso de conteúdo XHTML para extrair informações de RDF (W3C [2008a]);

- *Simple Knowledge Organization System* (SKOS), um modelo de dados que fornece um padrão de migração de bases de conhecimento para a Web Semântica, com a ajuda de uma linguagem própria para a definição de conteúdos de modelos conceituais diversos (W3C [2009k]);
- *RDF Vocabulary Description Language 1.0: RDF Schema* (RDFS), uma linguagem de especificação para descrever vocabulários RDF e definir classes e propriedades de aplicações específicas (W3C [2004b]);
- *Gleaning Resource Descriptions from Dialects of Languages* (GRDDL), uma especificação de elementos que podem ser adicionados aos documentos disponíveis em outros formatos diferentes de RDF (tipicamente XML e XHTML), de modo que as informações contidas nesses documentos possam ser capturadas e tratadas satisfatoriamente (W3C [2007a]);
- *Protocol for Web Description Resources* (POWDER),um protocolo de definição de conteúdo que permite sua seleção prévia (W3C [2009i]);
- *Rule Interchange Format* (RIF), uma especificação que pode ser utilizada para o compartilhamento de informações entre sistemas baseados em regras (W3C [2009j]);
- *Semantic Annotations for WSDL and XML Schema* (SAWSDL), um padrão para a especificação de atributos semânticos que podem ser associados a especificações WSDL existentes. (W3C [2007b])

2.2.1 O RDF e a Definição de Metadados

O RDF provê uma forma de representar metadados, que são informações acerca de outras informações ("dados sobre dados"), em XML (Breitman [2005]). O principal objetivo do RDF é facilitar a troca de informações na Web. A especificação da sintaxe dos elementos RDF é descrita em W3C [2004c].

Um documento RDF é composto de uma série de declarações sobre recursos da Web. Cada recurso tem um determinado número de propriedades e cada propriedade tem um valor associado.

Com a ajuda das declarações RDF, aplicativos distintos podem obter informações detalhadas sobre os recursos compartilhados na Web.

Declarações sobre recursos são representadas através das chamadas "triplas" RDF, caracterizadas pelos seguintes elementos:

- sujeito - identificação do recurso, devendo ser expresso unicamente por um URI, como por exemplo: "http://www.wikipedia.org";
- predicado - nome de uma propriedade do sujeito (recurso), de um determinado tipo de dados (texto, números e outros), visto também como um atributo do sujeito;
- objeto - valor de um predicado (atributo) do sujeito.

Como exemplo de caracterização de uma tripla RDF, podemos considerar a seguinte declaração: *"O criador da página http://www.exemplo.org/index.html é Fulano de Tal"*. A Figura 2.5 mostra um grafo representativo dessa declaração.

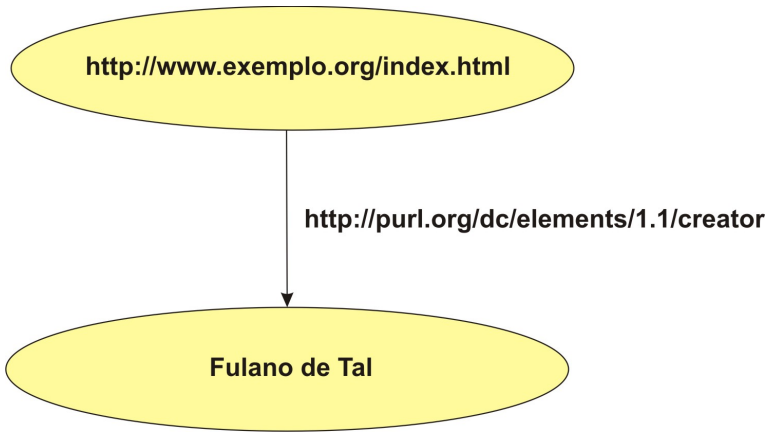

Figura 2.5: Exemplo de um grafo representando uma tripla RDF.

Nesse exemplo, no papel de sujeito temos a página "http://www.exemplo.org/index.html", que tem um predicado chamado "*creator*" (criador), descrito em "http://purl.org/dc/elements/1.1/creator, cujo objeto (valor) é "Fulano de Tal".

A representação dessa tripla em RDF permite a identificação, por parte dos computadores, de todos os elementos descritos, uma vez que para cada item há uma definição formal estabelecida de forma única. O predicado "*creator*", por exemplo, tem sua definição disponível em ''http://purl.org/dc/elements/1.1/creator. A Figura 2.6 mostra uma representação em RDF do exemplo descrito.

```
1 <?xml version=''1.0''?>
2   <rdf:RDF xmlns:rdf=''http://www.w3.org/1999/02/22-rdf-syntax-ns#''
3            xmlns:dc=''http://purl.org/dc/elements/1.1/''>
4   <rdf:Description rdf:about=''http://www.exemplo.org/index.html''>
5     <dc:creator>Fulano de Tal</dc:creator>
6   </rdf:Description\>
```

Figura 2.6: Descrição de uma tripla RDF.

O código em RDF declara que "http://www.exemplo.org/index.html" é um recurso (sujeito) já existente, e que o valor (objeto) do predicado (propriedade) "*dc:creator*" é "Fulano de Tal". O predicado "*dc:creator*" está definido em "http://purl.org/dc/elements/1.1/", conforme informado na declaração "*xmlns*" (*XML namespace*) "*xmlns:dc*", que aponta para o modelo de metadados do grupo Dublin Core (DCMI [2008]).

Outros predicados podem ser associados ao sujeito, conforme ilustrado na Figura 2.7.

O diagrama mostrado na Figura 2.7 mostra que foram adicionados ao exemplo inicial os predicados "*date*", cujo objeto (valor) é "2008-12-31", e "*language*", cujo valor é "pt-br" (Português do Brasil).

A especificação de sintaxe do RDF é normalmente utilizada em conjunto com o *RDF Schema* W3C [2004b].

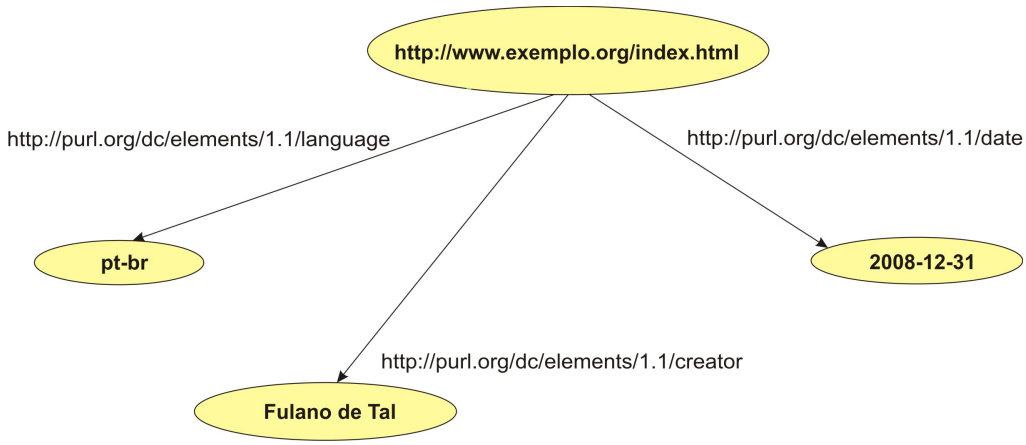

Figura 2.7: Exemplo de sujeito com vários predicados.

2.3 Aplicações Práticas

Recentes aplicações em Web Semântica mostram que a nova tecnologia está se concretizando. Algumas dessas aplicações são:

- *Powerset*, um motor de busca em linguagem natural baseado na Web Semântica, atualmente trabalhando com os dados existentes em Wikipedia [2001] (Boye [2005]);
- *BBC Semantic Music Project*, um projeto de interligação de anotações semânticas sobre música que utiliza a linguagem SPARQL para oferecer um inovador sistema de busca (BBC [2008]);
- *Hakia*, um motor de busca baseado na Web Semântica (Hakia [2007]).

Neste capítulo foram apresentados alguns aspectos referentes à Web Semântica, sua história, sua estrutura e algumas aplicações existentes. No próximo capítulo serão apresentadas as ontologias, que são elementos essenciais para a Web Semântica.

Capítulo 3

Ontologias

A idéia de ontologia nasceu na Grécia Antiga, sob uma perspectiva filosófica que englobava o estudo do ser, enquanto ser, e suas qualidades. A ontologia é um ramo fundamental da Filosofia, com muitas interpretações propostas ao longo do tempo, até hoje preservando a idéia central da busca de conhecimento da essência do ser através das suas manifestações físicas.

No terreno da Web Semântica, todavia, o conceito de ontologia tornou-se mais concreto e ao mesmo tempo mais restrito, deixando de ser uma ciência para tornar-se uma representação reduzida da realidade. De acordo com Gruber [1993], uma ontologia é uma especificação explícita de uma contextualização, que por sua vez é uma visão abstrata e simplificada do mundo a ser representado.

O papel de uma ontologia é o de capturar o conhecimento de um domínio para fornecer uma compreensão comumente aceita dele (Staab and Maedche [2000]).

Segundo Knublauch [2004], uma ontologia é composta por:

- *classes* - organizadas como uma taxonomia;
- *relações* - representam o tipo de interação entre os conceitos de um domínio;
- *axiomas* - usados para modelar sentenças sempre verdadeiras;
- *instâncias* - utilizadas para representar os dados.

Guarino [1998] define uma ontologia como sendo um artefato computacional, baseado em um vocabulário formal, utilizado para descrever uma conceitualização particular comumente aceita do mundo.

Uma formalização do conceito de ontologia é apresentada por Stumme et al. [2003]:

$$O := (C, \leq_C, R, \theta, \leq_R) \tag{3.1}$$

Na Equação 3.1, C e R são dois conjuntos disjuntos que contém, respectivamente, os conceitos e as relações; $\leq_C$ é uma ordem parcial chamada de hierarquia de conceitos ou taxonomia; a função $\theta : R \rightarrow C^+$ é chamada de assinatura; a ordem parcial $\leq_R$ é chamada de hierarquia de relações.

3.1 Classificações

De acordo com Guarino [1998], as ontologias podem ser classificadas de acordo com o nível de dependência em relação à uma determinada tarefa ou ponto de vista, conforme mostrado na Figura 3.1, na qual setas representam relacões de especialização.

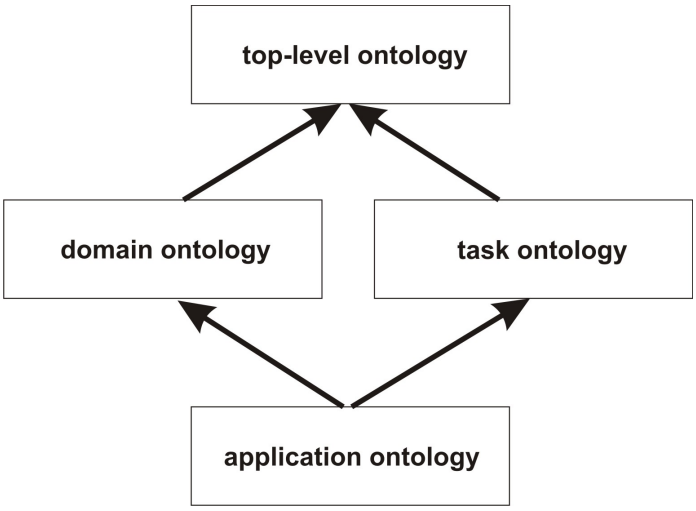

Figura 3.1: Tipos de ontologias (Guarino [1998]).

Sob essa perspectiva, as ontologias podem ser:

- de alto nível: descrevem conceitos muito gerais tais como espaço, tempo, assunto, objeto, evento e ação, os quais são independentes de algum problema em particular ou de algum domínio;
- de domínio: descrevem os vocabulários relacionados a domínios genéricos, tais como "medicina" e "automóveis", através da especialização de conceitos introduzidos nas ontologias de alto nível;
- de tarefa: descrevem atividades ou tarefas genéricas, tais como "diagnóstico" ou "vendas", através da especialização de conceitos introduzidos nas ontologias de alto nível;
- de aplicação: descrevem conceitos dependentes de uma ontologia de um domínio em particular ou de uma tarefa, sendo geralmente especializações dos dois tipos de ontologias ao mesmo tempo (domínio e tarefa), correspondendo normalmente a regras estabelecidas por entidades do domínio para a execução de atividades próprias, como por exemplo a descrição do processo de substituição de uma peça de um automóvel.

As ontologias de alto nível unificam conceitos utilizados por uma comunidade grande de usuários. Os outros tipos de ontologias são especializações daquelas, sendo as ontologias de aplicação as mais específicas.

Guarino [1998] acrescenta que embora uma ontologia de aplicação possa ser considerada uma base de conhecimento, existe uma clara diferença entre elas. Uma ontologia de aplicação define conceitos e fatos considerados verdadeiros por uma comunidade, qualquer

que seja a situação. Uma base de conhecimento pode conter informações dependentes de situações ou negócios diversos (informações dependentes do estado das coisas).

Van Heijst et al. [1997] classificam as ontologias em duas dimensões, sob a perspectiva da conceitualização descrita. A primeira refere-se à quantidade e ao tipo de estruturação da conceitualização, enquanto que a segunda diz respeito ao assunto da conceitualização. A primeira dimensão é separada em três categorias, como segue:

- ontologias terminológicas - especificam os termos usados para representar o conhecimento em um determinado domínio; uma ontologia com a descrição de todos os termos utilizados na área médica é um exemplo de ontologia terminológica.
- ontologias de informação - especificam a estrutura dos registros de banco de dados, tais como definições de esquema de bancos de dados; ontologias desse tipo provêem o necessário para o registro de dados sem informações adicionais sobre suas possíveis interpretações;
- ontologias de modelagem do conhecimento - especificam conceitualizações do conhecimento, fornecendo para tanto um conjunto mais rico de informações do que as ontologias de informação; ontologias de modelagem do conhecimento geralmente são direcionadas para um determinado uso particular do conhecimento que descrevem.

A segunda dimensão é dividida em quatro categorias, como segue:

- ontologias de aplicação - contém todas as definições necessárias para a modelagem do conhecimento necessário para uma aplicação em particular; as ontologias de aplicação geralmente são baseadas em uma combinação de ontologias de domínio e ontologias genéricas;
- ontologias de domínio - expressam conceitualizações que são específicas para um domínio em particular;
- ontologias genéricas - são similares às ontologias de domínio, mas os conceitos que definem são considerados genéricos em diversas áreas de conhecimento;
- ontologias de representação - explicam as conceitualizações em que se baseiam os formalismos de representação do conteúdo; as ontologias genéricas e as ontologias de domínio são normalmente baseadas nas ontologias de representação.

Do ponto de vista do grau de formalismo adotado na construção das ontologias, Uschold and Gruninger [1996] definem que estas podem ser:

- altamente informais - expressas com um vocabulário muito próximo da linguagem natural;
- semi-informais - expressa em uma forma restrita e estruturada de linguagem natural, aumentando enormemente a clareza das definições através da redução de ambigüidades;
- semi-formais - expressa em uma linguagem artificial definida formalmente;
- rigorosamente formal - expressa com termos meticulosamente definidos através de semântica formal, teoremas e provas.

Uschold et al. [1999] descrevem três cenários que caracterizam as ontologias a eles aplicáveis, como segue:

- autoria neutra - um artefato de informação é expresso em uma única linguagem e pode ser convertido em um formato diferente para uso em múltiplos sistemas; vantagens dessa abordagem incluem a reutilização do conhecimento, manutenção melhorada e retenção de conhecimento a longo prazo;
- livre acesso à informação - a informação é necessária por uma ou mais pessoas ou aplicações computacionais, mas é expressa com um vocabulário desconhecido ou está em um formato inacessível; a ontologia ajuda na tradução da informação por prover um conhecimento compartilhado dos termos usados ou pelo mapeamento entre conjuntos de termos; alguns benefícios dessa abordagem são a interoperabilidade e um reaproveitamento mais efetivo do conhecimento adquirido;
- indexação - a ontologia é utilizada como um mecanismo para indexar artefatos de informação; a grande vantagem dessa abordagem é o acesso rápido às informações importantes, o que possibilita uma reutilização eficaz do conhecimento previamente definido.

3.2 Operações

Devido à natureza distribuída da Web, espera-se, com o crescimento da Web Semântica, uma proliferação de ontologias. Essas ontologias sempre coexistirão em um ambiente de transformação contínua, sendo a interoperabilidade entre elas fundamental. Para que ontologias de origens distintas possam ser utilizadas em conjunto, pode ser necessário submetê-las a algumas operações.

Noy and Musen [2003] destacam algumas operações que podem ser realizadas com as ontologias, conforme mostrado na Figura 3.2:

- *merging*: fusão de ontologias com o propósito de criar uma nova; exemplos de ferramentas que executam essa operação são: iPrompt (Noy and Musen [2003]), Chimaera (McGuinness et al. [2000]) e OntoMerge (Dou [2004]);
- *transformation*: definição de uma função de transformação que converte uma ontologia em outra; OntoMorph (Chalupsky [2000]) é um exemplo de ferramenta que executa essa operação;
- *alignment*: definição de um mapeamento entre conceitos de duas ontologias pelo casamento de pares de conceitos relacionados; exemplos de ferramentas que suportam essa operação são: AnchorPrompt (Noy and Musen [2003]), GLUE (Doan et al. [2002]), OBSERVER (Mena et al. [2000]) e FCA-Merge (Stumme and Maedche [2001]);
- *articulation*: definição de regras de mapeamento para relacionar apenas partes relevantes das ontologias originais; ONION Mitra et al. [2001] é um exemplo de ferramenta que executa essa operação.

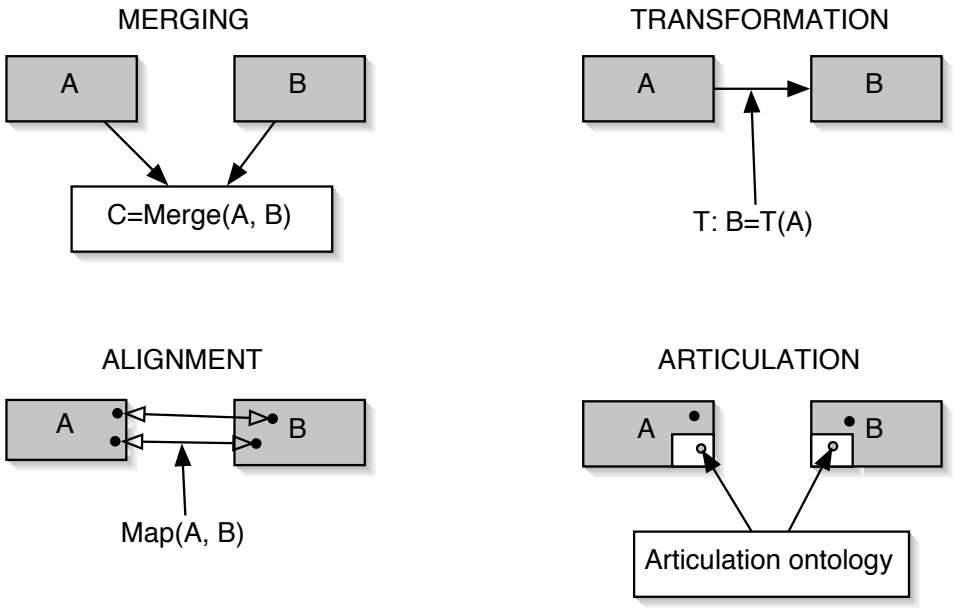

Figura 3.2: Operações sobre ontologias (Noy and Musen [2003]).

Pinto and Martins [2001] descrevem uma operação importante para a integração de múltiplas ontologias, cuja idéia é ilustrada na Figura 3.3. Esse processo de integração é uma forma de reutilização de ontologias alternativa ao processo de *merging* descrito por Noy and Musen [2003].

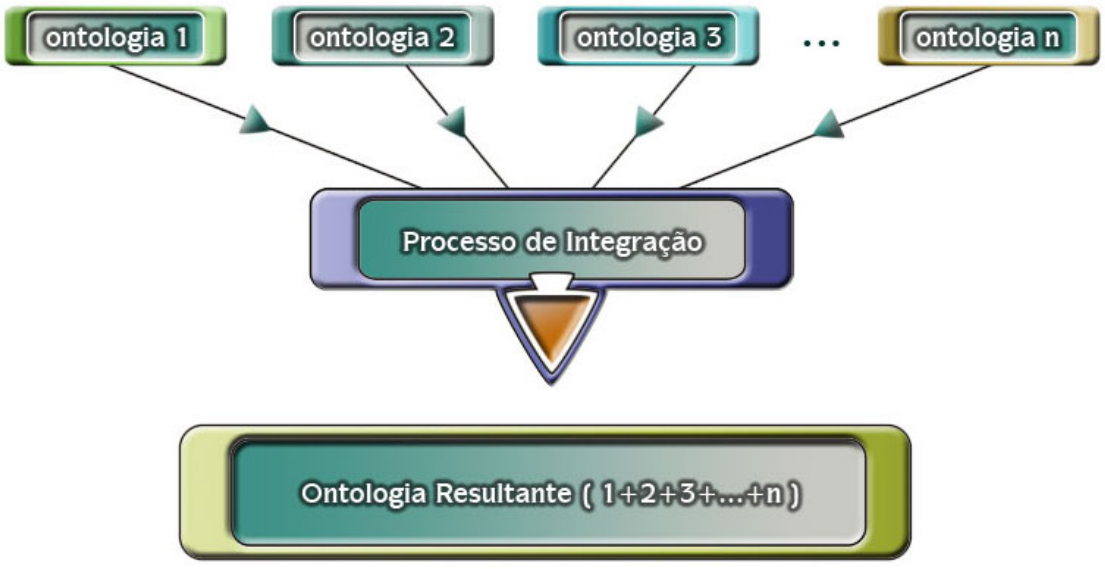

Figura 3.3: Integração de ontologias.

A integração é efetuada em vários estágios, a saber:

- seleção das ontologias passíveis de integração, por afinidade no domínio de aplicação;

- análise de compatibilidade entre os conceitos presentes nas ontologias, para verificar se é possível atender requisitos como completude e consistência;
- aplicação das operações de integração, conforme Farquhar et al. [1997], Borst et al. [1997], Pinto and Martins [2000] e Pinto [1999].
- análise, por especialistas do domínio de aplicação, da qualidade da ontologia resultante.

Os autores das ferramentas citadas justificam de forma unânime a necessidade dessas operações por serem elas importantes durante todo o processo de construção de ontologias, promovendo a reutilização de ontologias previamente elaboradas e facilitando a interoperabilidade entre elas.

3.3 Padrões

O uso de ontologias na Web Semântica partiu dos estágios iniciais de representação semântica em RDF para o uso de novas linguagens mais capazes de representar o conteúdo semântico da Web, sobretudo no que diz respeito à definição de relacionamentos lógicos entre conceitos. As linguagens mais recentes baseiam-se em lógica descritiva (vide Seção 4.3), fornecendo, portanto, semântica formal e suporte à inferência (Breitman [2005]).

Dentre as principais linguagens adotadas na Web Semântica após o surgimento do RDF, destacam-se:

- *Ontology Inference Layer* (OIL) - compatível com RDFS, combina o modelagem de ontologias baseada em *frames* com a capacidade de inferência;
- *DARPA Agent Markup Language* (DAML) - desenvolvida como uma extensão do XML e do RDF, ampliada posteriormente com os recursos emprestados da especificação OIL, constituindo o que se denominou DAML+OIL;
- DAML+OIL - combinação da linguagem DAML com OIL, desenvolvida para possibilitar raciocínio e inferência nas ontologias definidas;
- *Web Ontology Language* (OWL) - linguagem criada com o propósito de suprir as necessidades da Web Semântica no que se refere à definição de ontologias.

A seguir apresentamos a descrição de cada uma dessas linguagens.

3.3.1 OIL

A linguagem OIL é uma produção dos projetos Amsterdam [2000] e of Amsterdam et al. [2000], desenvolvida com o objetivo de prover um padrão de representação de ontologias baseado na Web com suporte à inferência .

OIL combina as primitivas de modelagem das linguagens baseadas em frames com a semântica formal da lógica descritiva e é compatível com RDFS. A linguagem foi elaborada de modo a atender os seguintes requisitos:

- oferecer primitivas de modelagem normalmente utilizadas nas ontologias baseadas em frames;

- manter a simplicidade, a clareza e uma semântica bem definida baseada em lógica descritiva;
- prover suporte ao raciocínio automatizado de uma forma computacionalmente eficiente.

A Figura 3.4 apresenta a estrutura atual da linguagem OIL, com as seguintes divisões:

- *Core OIL*: coincide quase na totalidade com o RDFS, permitindo que até mesmo agentes RDF muito simples sejam capazes de processar ontologias descritas OIL, até o limite de suas capacidades;
- *Standard OIL*: destinada a capturar as principais primitivas de modelagem que possibilitem ao mesmo tempo um poder de expressividade adequado e uma boa inteligibilidade, permitindo dessa maneira que elementos semânticos sejam definidos de forma precisa e que processos completos de inferência sejam viáveis;
- *Instance OIL*: inclui uma capacidade de integração individual mais completa do que a vista em *Standard OIL*, visto que é um superconjunto daquela com o poder adicional de definir instâncias de classes previamente definidas;
- *Heavy OIL*: pode incluir capacidades adicionais de representação e de raciocínio, estando a definição de sintaxe ainda em estudo.

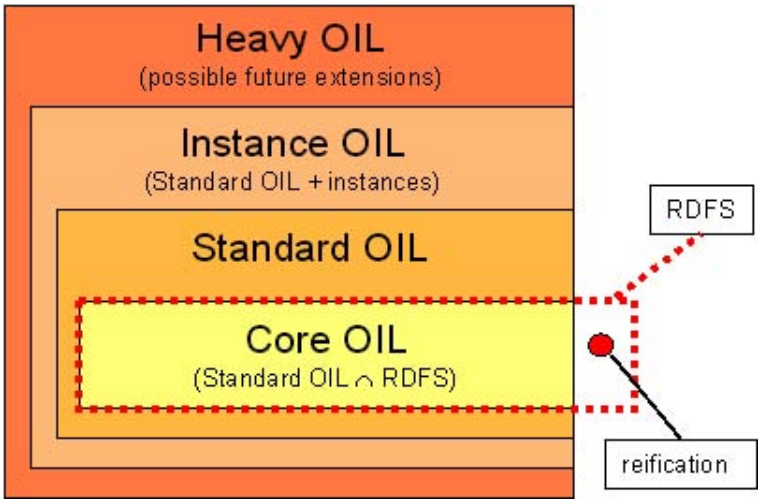

Figura 3.4: Estrutura atual do OIL (University et al. [2000]).

3.3.2 DAML

A linguagem DAML, ou DAML-ONT, como foi chamada inicialmente, foi desenvolvida a partir do ano 2000 para possibilitar a definição de ontologias com um grau de expressividade maior que a oferecida até então pelo RDFS. A partir do RDFS, a linguagem DAML-ONT implementa recursos adicionais tais como tipos de dados, enumerações e outras facilidades.

Após a criação de DAML-ONT, a linguagem foi combinada em 2001 com a linguagem OIL e dessa maneira foi possível aumentar o poder de expressividade da linguagem original, tendo em vista a capacidade de representação mais rica de fatos, em lógica descritiva, oferecida por aquela linguagem.

3.3.3 DAML+OIL

DAML+OIL é formada por um conjunto de triplas RDF, cada qual com um significado específico definido no vocabulário da linguagem. Uma ontologia DAML+OIL pode conter, além das triplas definidas no vocabulário, declarações RDF adicionais, mas as conseqüências semânticas de combinações dessa natureza são ignoradas pela linguagem (Connolly et al. [2001]).

As triplas RDF definidas para DAML+OIL formam a base de implementação dos vários componentes estruturais da linguagem. Alguns desses componentes são opcionais e outros podem aparecer várias vezes dentro de uma ontologia.

A estrutura de uma ontologia DAML+OIL é separada em duas partes disjuntas. A primeira compreende o domínio dos tipos de dados, todos importados do XML. A segunda parte constitui o domínio dos objetos, que são os membros das classes definidas na ontologia.

Uma ontologia DAML+OIL pode conter, opcionalmente, alguns cabeçalhos (*headers*) com definições de versão, *links* de importação de recursos e tipos de dados. Em seguida vêm as definições de classes, de atributos e de instâncias, que normalmente contituem a maior parte de uma ontologia DAML+OIL.

3.3.4 OWL

OWL, criada para substituir a linguagem DAML+OIL, é hoje a principal linguagem designada para a especificação de ontologias para a Web Semântica. Lançada inicialmente em 2004, conta com uma nova revisão atualmente denominada OWL 2. A última versão da linguagem tem uma estrutura muito semelhante à presente na primeira versão, com a inclusão de quase todos os seus elementos constituintes, com poucas mudanças, apresentando na maioria dos casos diferenças apenas de nomeclatura (W3C [2009a]). Ao utilizarmos aqui o termo OWL, estamos doravante nos referindo à versão mais nova da linguagem, OWL 2.

A linguagem OWL sintoniza-se com os objetivos da Web Semântica por ser destinada a facilitar o desenvolvimento de ontologias compartilháveis, tendo sido criada com o objetivo de tornar o conteúdo da Web mais acessível para as máquinas (W3C [2009a]).

As ontologias OWL podem ser expressas em várias sintaxes, a saber:

- *Functional* (W3C [2009c]), designada para fins de especificação e para prover fundamentos de implementação de ferramentas OWL, tais como APIs e raciocinadores (vide Seção 4.3.2);
- RDF/XML, uma tradução particular (W3C [2009e]) dos elementos OWL, sendo a única sintaxe obrigatória para qualquer ferramenta OWL;
- *Manchester* (W3C [2009d]), designada para facilitar a leitura das ontologias por profissionais não habituados com Lógica;

- OWL/XML (W3C [2009g]), a sintaxe OWL definida através do XML;
- *Turtle* (W3C [2009f]), bastante popular por ser concisa e de fácil entendimento.

Existem ferramentas próprias para a tradução entre sintaxes diferentes.

OWL é considerada como sendo uma linguagem de representação do conhecimento, designada para acomodar definições a respeito de um domínio em particular. Segundo W3C [2009b], os elementos básicos da linguagem são:

- *entidades*: elementos utilizados para referenciar objetos do mundo real;
- *axiomas*: declarações básicas de uma ontologia OWL, através das quais os fatos a respeito das entidades podem ser expressadas;
- *expressões*: combinações de entidades para formar descrições complexas a partir de definições básicas.

Nas ontologias OWL, as relações entre classes e as definições de axiomas são muito importantes. É através desses elementos que são realizados os processos de raciocínio sobre as ontologias (vide Capítulo 4).

3.4 Exemplo

A Figura 3.5 apresenta um fragmento de ontologia escrita em OWL (Knublauch [2007]).

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\#"
    ...
  xml:base="http://www.topbraid.org/2007/05/composite.owl">
  <owl:Ontology rdf:about="">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema\#string"
    >Created with TopBraid Composer by Holger Knublauch</owl:versionInfo>
  </owl:Ontology>
  <owl:ObjectProperty rdf:ID="child">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="parent"/>
    </owl:inverseOf>
    <rdfs:comment>The object is the child of the subject.</rdfs:comment>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="\#parent">
    <rdfs:comment>The object is the parent of the subject.</rdfs:comment>
    <owl:inverseOf rdf:resource="\#child"/>
  </owl:ObjectProperty>
</rdf:RDF>
```

Figura 3.5: Fragmento de ontologia OWL (Knublauch [2007]).

Podemos observar, nesse exemplo, a descrição de duas propriedades que modelam relacionamentos em uma família, mais precisamente entre pais e filhos, sendo que as duas propriedades são inversas. Desse modo, se um objeto tem o relacionamento "*parent*" em relação a outro objeto, este último terá automaticamente o relacionamento "*child*" em relação ao primeiro. Essas relações implícitas são automaticamente verificadas pelos raciocinadores.

3.5 Ferramentas

Existem diversas ferramentas para desenvolvimento e manutenção de ontologias, tais como:

- Protégé (Protégé [2006]) - um editor de ontologias gratuito que suporta vários formatos de ontologias;
- KAON (KAON [2001]) - uma infra-estrutura para a manutenção de ontologias que inclui um editor gratuito;
- OntoStudio (OntoStudio [2008]) - um ambiente de desenvolvimento de ontologias com capacidade de integração com bancos de dados e importação de bases de conhecimento a partir de diversos formatos.

Neste trabalho o editor Protégé foi utilizado em testes de validação das ontologias geradas, sendo escolhido devido aos recursos oferecidos para a utilização de raciocinadores a partir do ambiente do editor e à facilidade na importação e exportação de ontologias em diversos formatos.

Foram apresentados, neste capítulo, vários aspectos referentes às ontologias e às linguagens mais conhecidas, com destaque para a linguagem OWL. No próximo capítulo serão apresentados alguns conceitos introdutórios de lógicas formais, importantes para a compreensão dos mecanismos de inferência aplicados às ontologias.

Capítulo 4

Lógicas Formais

Entre a identificação de uma coleção de conceitos, sua representação intermediária e sua imagem final na forma de ontologia, está um processo conhecido como axiomatização. O estabelecimento dos relacionamentos entre os conceitos identificados constitui o resultado desse processo. Para que esses relacionamentos sejam passíveis de processamento automatizado, a estrutura da ontologia e seus axiomas baseiam-se propositalmente em uma linguagem lógica formal.

A importância da lógica formal pode ser resumida no fato de que seu uso facilita o equacionamento de problemas lógicos na forma de expressões sintaticamente manipuláveis.

A lógica formal possibilita a definição de argumentos que, de acordo com as premissas consideradas, podem ser formalmente avaliados. A Tabela 4.1 mostra um exemplo de um argumento que, embora correto do ponto de vista lógico, é notoriamente falso. Isso ocorre porque a primeira premissa não é verdadeira. Contudo, o provável questionamento dessa realidade não pode ser corroborado por um processo formal de validação.

Premissa 1	Um mentiroso nunca mente.
Premissa 2	João é um mentiroso.
Conclusão	João nunca mente.

Tabela 4.1: Exemplo de argumento questionável, mas formalmente correto.

Do ponto de vista formal, o argumento do exemplo da Tabela 4.1 não pode ser questionado porque as duas premissas são forçosamente supostas como sendo verdadeiras, mesmo que nosso julgamento discorde disso. No caso das ontologias, a especificação formal garante o compartilhamento do conhecimento nelas representado, mas não pode ser utilizada para verificar se esse conhecimento é correto ou não. Esse julgamento pode ser feito em um plano externo.

As ontologias podem conter definições de argumentos hipotéticos, dentro de uma visão específica de um domínio de aplicação. Essas definições, que fazem parte do processo de axiomatização, podem gerar conflitos, na medida em que os axiomas vão sendo definidos ao longo do tempo. Esses conflitos podem ser identificados com a ajuda de ferramentas de validação, ou raciocinadores. Quando as ontologias são obtidas através de um processo automatizado, os raciocinadores passam a ser ferramentas valiosas na medida em que ajudam a identificar possíveis problemas nos resultados obtidos.

4.1 Lógica Proposicional

A lógica proposicional trata de equacionar proposições que podem assumir apenas um dentre dois valores possíveis: verdadeiro ou falso. As proposições são combinadas em expressões a fim de formar os argumentos, que são premissas seguidas de uma "conclusão". Um argumento é considerado "válido" se a conclusão é uma conseqüência lógica das premissas, como no exemplo mostrado na Tabela 4.2.

Premissa 1	Sempre que trabalho demais, fico cansado.
Premissa 2	Trabalhei demais.
Conclusão	Logo, fiquei cansado.

Tabela 4.2: Exemplo de argumento válido.

As proposições são os elementos básicos das expressões da lógica proposicional. As expressões que fazem parte da lógica proposicional são as chamadas "expressões bem formadas". Uma expressão bem formada pode assumir as seguintes formas:

- as constantes de valores verdadeiro $\top$ ou falso $\bot$;
- os símbolos proposicionais, normalmente letras minúsculas do alfabeto latino (ex: p e q);
- a negação: se α é uma expressão, então sua negação, $\neg\alpha$, é uma expressão;
- a conjunção: se α e β são expressões, então a conjunção delas, $\alpha \wedge \beta$, é uma expressão;
- a disjunção: se α e β são expressões, então a disjunção delas é $\alpha \vee \beta$;
- a expressão condicional: se α e β são expressões, então a expressão condicional é $\alpha \rightarrow \beta$, onde α é denominada expressão "antecedente" e β é a expressão "conseqüente".

Os símbolos $\neg$, $\wedge$, $\vee$ e $\rightarrow$ são denominados conectores lógicos. A Tabela 4.3 mostra o uso desses conectores com duas proposições p e q.

$\mathbf{p}$	$\mathbf{q}$	$\mathbf{\neg p}$	$\mathbf{p \wedge q}$	$\mathbf{p \vee q}$	$\mathbf{p \rightarrow q}$
$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\top$	$\top$	$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\top$

Tabela 4.3: Valores possíveis de expressões.

A ordem de precedência dos conectores lógicos é a seguinte: negação $\neg$ (a maior precedência), conjunção $\wedge$, disjunção $\vee$ e condicional $\rightarrow$ (a menor precedência). Expressões podem ser colocadas entre parênteses, "(" e ")". As expressões entre parênteses são avaliadas em primeiro lugar, ou seja, têm maior precedência.

É interessante observar a semântica da expressão condicional, que só pode ser falsa se a expressão antecedente for verdadeira e a expressão conseqüente for falsa. A semântica desse operador é a mesma semântica que caracteriza o raciocínio dedutivo utilizado nos processos de inferência, o qual é normalmente orientado pelas seguintes regras:

- *Modus Ponens* (MP): se $\alpha \rightarrow \beta$ e α, então β;
- *Modus Tollens* (MT): se $\alpha \rightarrow \beta$ e $\neg\beta$, então $\neg\alpha$;
- Silogismo Hipotético: se $\alpha \rightarrow \beta$ e $\beta \rightarrow \gamma$, então $\alpha \rightarrow \gamma$.

A formação dos argumentos da lógica proposicional baseia-se na formalização das premissas como expressões proposicionais. A título de exemplo, vamos considerar a seguinte argumentação:

- Se o carro não tem combustível, o dono do carro está sem dinheiro.
- Se o carro tem combustível, então pode rodar.
- Se o carro pode rodar, a família pode viajar.
- A família não pode viajar.
- Logo, o dono do carro está sem dinheiro.

Podemos distinguir cada afirmação usando os seguintes símbolos proposicionais:

- $p =$ " carro tem combustível".
- $q =$ " dono do carro está sem dinheiro".
- $r =$ " carro pode rodar".
- $s =$ " família pode viajar".

Os símbolos obtidos podem compor as seguintes expressões do argumento:

- $\neg p \rightarrow q =$ "Se o carro não tem combustível, o dono do carro está sem dinheiro".
- $p \rightarrow r =$ "Se o carro tem combustível, então pode rodar".
- $r \rightarrow s =$ "Se o carro pode rodar, a família pode viajar".
- $\neg s =$ "A família não pode viajar".

4.2 Lógica de Primeira Ordem

A Lógica de Primeira Ordem (LPO) amplia a lógica de predicados com a inclusão dos operadores lógicos de quantificação: $\forall$ ("para todo") e $\exists$ ("existe"). A utilização desses operadores, dada uma variável x e uma sentença θ, é como segue:

- $\forall_x \theta$ significa "para todo x vale θ";
- $\exists_x \theta$ significa "existe x tal que vale θ".

Os operadores de quantificação aumentam o poder de expressividade da lógica proposicional permitindo declarações tais como: se "todo elemento divisível apenas por ele mesmo ou por 1 é um número primo" e se "existe x tal que x só pode ser dividido por 1 e por ele mesmo", então "x é um número primo".

Formalmente, a LPO pode ser aplicada em diversas áreas, constituindo linguagens de primeira ordem com notação apropriada a cada caso. A LPO aplicada à teoria dos conjuntos e a aplicada à Aritmética, por exemplo, possuem notações diferentes e constituem dessa forma linguagens distintas, porém semanticamente equivalentes. As operações possíveis em cada caso são as mesmas. Essas operações caracterizam o Cálculo de Predicados de Primeira Ordem, cuja notação e funcionamento podemos descrever a partir da escolha de qualquer uma das linguagens de primeira ordem existentes.

Uma linguagem de primeira ordem deve ter um *alfabeto*, um conjunto de *axiomas* e um conjunto de *regras* de derivação ou inferência, a partir das quais é possível desenvolver o cálculo de predicados de primeira ordem. Os axiomas são conjuntos de expressões bem formadas que, juntamente com as regras de inferência, servem para formar os teoremas de uma determinada área de aplicação, os quais são derivados do conjunto de axiomas a partir da aplicação dessas regras.

Um *alfabeto de primeira ordem* consiste dos seguintes elementos:

- variáveis
- constantes
- símbolos de função
- símbolos de predicado
- conectivos lógicos
- quantificadores lógicos
- símbolos de pontuação

As variáveis, constantes, símbolos de função e símbolos de predicado podem variar de acordo com a área de aplicação. Os conectivos lógicos, quantificadores e símbolos de pontuação são sempre os mesmos para qualquer alfabeto.

Os conectivos lógicos são os seguintes:

- $\neg$ - negação ("não");
- $\wedge$ - conjunção ("e");
- $\vee$ - disjunção ("ou");
- $\rightarrow$ - implicação ("... implica em ...");
- $\leftrightarrow$ - equivalência ("se e somente se").

Os quantificadores lógicos são os seguintes:

- $\forall_x \theta$ - universalidade ("para todo x temos θ");
- $\exists_x \theta$ - existência ("existe x tal que θ").

Os símbolos de pontuação podem ser:

- "," - vírgula, usada para separar parâmetros em expressões;

- "(" e ")" - parênteses, usados tanto para aninhar expressões quanto para agrupar parâmetros.

Barwise and Etchemendy [1999] descrevem uma linguagem de primeira ordem estabelecendo algumas definições de uso geral para constantes, variáveis, funções e predicados, como segue:

- uma *constante* é um símbolo, geralmente uma palavra em letras mínusculas, que representa um valor fixo, um indivíduo ou um nome de objeto em particular;
- uma *variável* é um elemento, normalmente representado por uma letra minúscula, que aparece nas expressões onde normalmente apareceria uma constante e que pode assumir um ou mais valores determinados;
- uma *função* funciona como um predicado, mas contém um ou mais parâmetros, que podem ser constantes, variáveis ou (também) funções;
- um *predicado* é um elemento utilizado para representar alguma propriedade ou para estabelecer alguma relação entre indivíduos, podendo receber parâmetros ou não.

O número de parâmetros que uma função ou predicado pode receber é chamado de *aridade*. Uma função que recebe n parâmetros tem, portanto, aridade n.

Um *termo* é definido como sendo uma constante, variável ou função. Os parâmetros de uma função podem ser termos. Se $t_1, t_2, t_3, ..., t_n$ são termos e f é uma função de aridade n, então $f(t_1, t_2, t_3, ..., t_n)$ é também um termo.

Uma expressão bem formada é chamada de *fórmula*, sendo definida da seguinte forma:

- se $t_1, t_2, t_3, ..., t_n$ termos e p é um predicado, então $p(t_1, t_2, t_3, ..., t_n)$ é uma fórmula, sendo nesse caso chamada de *fórmula atômica*, ou simplesmente *átomo*, e p, sem parâmetros, também é um átomo;
- se α e β são fórmulas, então são também fórmulas $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$ e $(\alpha \leftrightarrow \beta)$;
- se α é uma fórmula e x é uma variável, então $(\forall_x \alpha)$ e $(\exists_x \alpha)$ são também fórmulas.

As funções podem ser confundidas com os predicados, por que a sintaxe pode ser semelhante. Entretanto, um predicado não é um termo e uma função não pode constituir uma fórmula. Para evitar confusão, convém estabelecer convenções para diferenciar esses dois elementos. Considere, por exemplo, uma função utilizada para identificar o pai de um indivíduo e um predicado que indica que um indivíduo é feio. Estabelecendo que os predicados devem ter nomes iniciados com a primeira letra maiúscula e que as funções devem ter a primeira letra minúscula, chegaríamos ao exemplo da Tabela 4.4. Pela convenção adotada, os predicados podem ser facilmente diferenciados das funções.

Nesse exemplo, mesmo sem conhecermos os significados dos predicados e das funções, evitaríamos a construção de expressões mal formadas, tais como *Feio(Feio("João"))*, pois *Feio(x)* é, pela convenção adotada, um predicado. Embora nesse exemplo o significado das funções e dos predicados seja óbvio, em exemplos mais complexos ou em aplicações que utilizam alfabetos diferentes isso pode não acontecer.

Significado	**Expressão em LPO**
"o pai de João"	pai("João")
"João é feio"	Feio("João")
"o pai de João é feio"	Feio(pai("João"))
"o avô (pai do pai) de João é feio"	Feio(pai(pai("João")))

Tabela 4.4: Exemplo de predicado e função.

Na LPO, o funcionamento dos conectivos lógicos ($\neg$, $\wedge$, $\vee$ e $\rightarrow$) é idêntico ao da lógica proposicional. Adicionalmente, o conectivo lógico $\leftrightarrow$ nada mais é do que a implicação recíproca, isto é, $(F \leftrightarrow G)$ é o mesmo que $((F \rightarrow G) \wedge (G \rightarrow F))$.

Os quantificadores da LPO, não presentes na lógica proposicional, são os elementos viabilizadores do cálculo de predicados de primeira ordem, que torna possível a definição das teorias baseadas na lógica de primeira ordem, tais como as adotadas na Aritmética e na Teoria dos Conjuntos.

No contexto das ontologias, o cálculo de predicados de primeira ordem é importante para a compreensão dos mecanismos básicos utilizados nos processos de raciocínio automatizado.

4.3 Lógica Descritiva

A Lógica Descritiva (LD) é uma denominação geral para uma família de formalismos de representação do conhecimento, sobre os quais é possível estabelecer também métodos de raciocínio automatizado (Horrocks and Sattler [2002]). Segundo Nardi and Brachman [2003], a pesquisa no campo da representação do conhecimento e do raciocínio, pilares do estudo das ontologias, é normalmente centrada em métodos para a obtenção de descrições de alto nível para uso em aplicações inteligentes, capazes de inferir dados implícitos a partir das informações explicitamente representadas. Um outro aspecto importante da LD é sua tradução para a linguagem de predicados da LPO (Hustadt et al. [2004]).

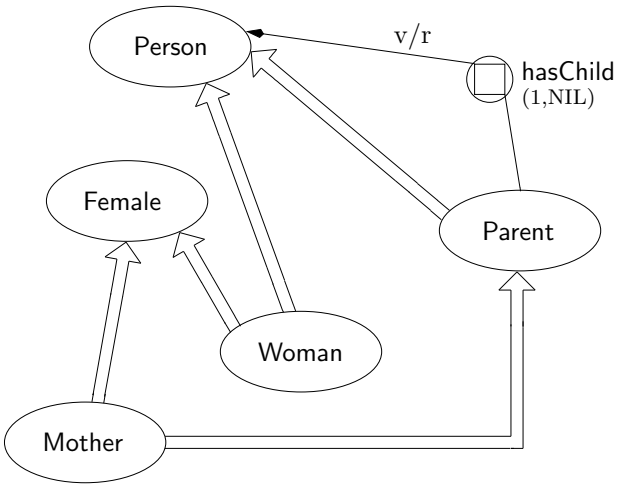

Figura 4.1: Exemplo de taxonomia (Nardi and Brachman [2003]).

Descrições de alto nível têm como ponto de partida relacionamentos hierárquicos entre conceitos. Tais relacionamentos caracterizam taxonomias, cujas estruturas baseiam-se essencialmente em ligações do tipo "é-um", como as mostradas no exemplo da Figura 4.1. Os conceitos, ou classes de indivíduos, representados dentro das elipses, são chamados de "nós". Eles aparecem conectados através de ligações hierárquicas. Esse tipo de ligação provê a base para a implementação de mecanismos de herança, pelos quais as propriedades dos conceitos mais gerais são propagados para os conceitos mais específicos (Brachman [1979]).

A representação de taxonomias é apenas uma das possibilidades da LD, que permite a definição de outros tipos de relacionamentos. Além disso, certas ligações entre conceitos podem apresentar características especiais, ou restrições. A Figura 4.1 apresenta, por exemplo, a propriedade $hasChild(1, NIL)$, que nesse caso indica que os indivíduos da classe $Parent$ são pessoas ($Person$) com pelo menos 1 filho, e que todos os seus filhos são pessoas ($Person$). Esse exemplo também mostra que um raciocinador automático pode deduzir, apesar de não estar explícito nas definições, que toda mãe ($Mother$) é também uma mulher ($Woman$). Os raciocinadores automatizados podem encontrar relacionamentos mais complexos do que esse através das regras de formação de uma determinada linguagem de LD adotada.

4.3.1 Representação do Conhecimento

Baader and Nutt [2003] descrevem o modelo de representação do conhecimento normalmente adotado em LD. A ilustração da Figura 4.2 mostra os dois principais aspectos considerados no estudo da LD: representação do conhecimento, à esquerda, e raciocínio automatizado, à direita, ambos ligados aos dois conjuntos de elementos básicos do modelo, o primeiro contendo a terminologia adotada no *Terminology Box* (*TBox*) e o segundo contendo as assertivas definidas no *Assertion Box* (*ABox*).

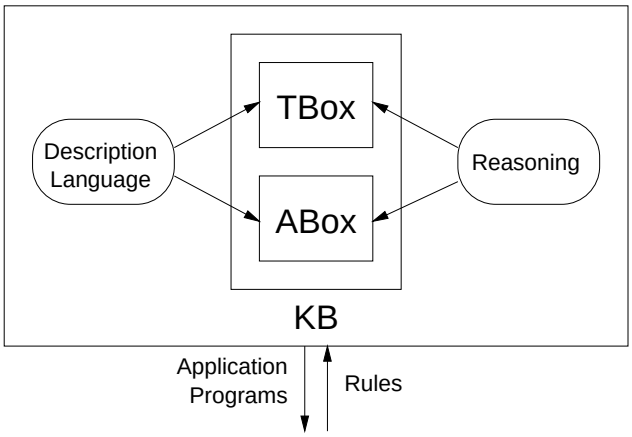

Figura 4.2: Representação do conhecimento em LD (Baader and Nutt [2003]).

O conjunto *TBox* contém toda a hierarquia de conceitos de uma estrutura de representação do conhecimento. Todas as definições mais gerais acerca dos conceitos identificados na estrutura ficam dentro desse conjunto. As informações desse conjunto tendem a permanecer estáveis ao longo do tempo.

O conjunto *ABox* contém o conhecimento estendido ou assertivo a respeito dos elementos presentes na estrutura (Nardi and Brachman [2003]). As informações desse conjunto podem sofrer mudanças ao longo do tempo, em níveis de intensidade e freqüência que podem variar de acordo com as circunstâncias.

O conhecimento é representado nos conjuntos *TBox* e *ABox* por intermédio de alguma *linguagem* de LD. Schmidt-Schaubßand Smolka [1991] definiram uma linguagem descritiva básica conhecida como *attributive language* ($\mathcal{AL}$), cuja sintaxe é apresentada na Figura 4.3.

$C, D \rightarrow A$	|	(atomic concept)
$\top$	|	(universal concept)
$\bot$	|	(bottom concept)
$\neg A$	|	(atomic negation)
$C \sqcap D$	|	(intersection)
$\forall R.C$	|	(value restriction)
$\exists R.\top$		(limited existential quantification)

Figura 4.3: Sintaxe da Linguagem AL (Schmidt-Schaubßand Smolka [1991]).

O *conceito universal* (verdadeiro) pode ser interpretado como o conjunto de todos os conceitos e o *conceito base* (falso) pode ser interpretado como o conjunto vazio, de nenhum conceito.

Voltando ao exemplo da Figura 4.1, podemos adicionar, de acordo com a sintaxe de $\mathcal{AL}$, as seguintes observações:

- $Person$, $Parent$, $Female$, $Woman$ e $Mother$ podem ser classificados como *conceitos atômicos*;
- a *negação* só é permitida sobre os conceitos atômicos, portanto apenas $\neg Person$, $\neg Parent$, $\neg Female$, $\neg Woman$ e $\neg Mother$ são permitidas;
- podemos definir, por exemplo, $Person \sqcap \forall hasChild.Female$ e $Person \sqcap \exists hasChild.\top$, denotando, respectivamente, todas as pessoas que têm pelo menos um filho e aquelas pessoas cujos filhos são todos do sexo feminino;
- não podemos declarar $Person \sqcap \exists hasChild.Female$, denotando aquelas pessoas que têm pelo menos um filho do sexo feminino, pois a sintaxe permite apenas o uso do quantificador existencial limitado;
- $Person \sqcap \forall hasChild.\bot$ representa todas as pessoas que não têm filhos.

A linguagem $\mathcal{AL}$ deu origem a diversas variações de linguagens descritivas, cada qual com regras de formação adicionais definidas para aumentar o poder de expressividade da linguagem original.

4.3.1.1 Definições em TBox

Um *conceito* é o elemento mais elementar em um TBox. Os conceitos são definidos a partir da definição de outros conceitos, formando uma estrutura hierárquica. Conside-

rando a Figura 4.1, podemos ter, por exemplo, o seguinte conceito, que define $Woman$ como sendo a classe de pessoas do sexo feminino:

$$Woman \equiv Person \sqcap Female$$

Segundo Nardi and Brachman [2003], essa relação de equivalência entre o novo conceito e a sua definição estabelece as condições necessárias e suficientes para o reconhecimento de indivíduos de uma nova classe, já que:

- para cada nome de conceito, apenas 1 definição é permitida, não sendo permitidas, portanto definições parciais e complementares de um mesmo conceito;
- não pode haver definições cíclicas (referências circulares) entre as definições, seja de forma direta ou indireta.

A Tabela 4.5 ilustra um conjunto TBox que contém as definições dos relacionamentos em uma família (Baader and Nutt [2003]). Esse conjunto foi definido com uma linguagem descritiva mais expressiva que $\mathcal{AL}$, pela qual é possível definir disjunções ($\sqcup$) e informações de cardinalidade ($\geqslant$, $\leqslant$, $>$, $<$).

$Woman$	$\equiv Person \sqcap Female$
Man	$\equiv Person \sqcap \neg Woman$
$Mother$	$\equiv Woman \sqcap \exists hasChild.Person$
$Father$	$\equiv Man \sqcap \exists hasChild.Person$
$Parent$	$\equiv Father \sqcup Mother$
$Grandmother$	$\equiv Mother \sqcap \exists hasChild.Parent$
$MotherWithManyChildren$	$\equiv Mother \sqcap \geqslant 3 hasChild$
$MotherWithoutDaughter$	$\equiv Mother \sqcap \forall hasChild.\neg Woman$
$Wife$	$\equiv Woman \sqcap \exists hasHusband$

Tabela 4.5: Exemplo de um TBox sobre família (Baader and Nutt [2003]).

Sistemas baseados em LD mais modernos oferecem um poder de expressividade maior ao permitirem a definição de axiomas gerais nos quais não há a restrição de obrigatoriedade de um conceito atômico do lado esquerdo da declaração, possibilitando definições do tipo:

$$C \sqsubseteq D$$

Por essa definição é possível definir que uma expressão - não apenas um conceito atômico - pode ser definida em função de outra mais geral. Esse tipo de definição pode levar a referências circulares, dificultando a implementação desses sistemas.

4.3.1.2 Definições em ABox

O conjunto de definições de um ABox compreende aquelas relacionadas principalmente aos indivíduos de um domínio, sendo normalmente utilizado para acomodar a informação

ou conteúdo propriamente dito. A seguinte declaração, por exemplo, aponta a existência de uma pessoa do sexo feminino cujo nome é "MARIA":

$$Female \sqcap Person(MARIA)$$

Por essa declaração deduz-se que "MARIA" pertence à classe $Woman$, embora isso não esteja explícito na declaração. Outra declarações sobre "MARIA" poderiam ser acrescentadas em ABox, tal como a existência de um filho chamado "ALFREDO":

$$hasChild(MARIA, ALFREDO)$$

A diferença entre essas duas declarações é que a primeira é uma definição *conceitual* a respeito de "MARIA", e a segunda estabelece um *relacionamento* entre "MARIA" e "ALFREDO".

A Tabela 4.6 mostra um exemplo de ABox baseado no TBox apresentado anteriormente. Através desse exemplo pode-se observar como um conjunto pequeno de definições pode conter muitas outras definições não declaradas explicitamente.

$MotherWithoutDaughter(MARY)$	$Father(PETER)$
$hasChild(MARY; PETER)$	$hasChild(PETER; HARRY)$
$hasChild(MARY; PAUL)$	

Tabela 4.6: Exemplo de um ABox sobre uma família (Baader and Nutt [2003]).

4.3.2 Raciocinadores

As declarações feitas nos conjuntos TBox e ABox não são apenas informações bem estruturadas, mas também as peças da estrutura maior que pode ser derivada dos dois conjuntos. Através de mecanismos de inferência, conduzidos por raciocinadores automatizados, informações adicionais podem ser deduzidas. Os raciocinadores são construídos com base nas regras estabelecidas pela linguagem descritiva adotada. Quanto maior o poder de expressividade da linguagem, mais complexo é o trabalho dos raciocinadores.

Existem diferentes tarefas de raciocínio normalmente executadas sobre uma estrutura de representação do conhecimento definidas em termos de TBox e ABox. Dentre elas destacam-se:

- verificação de *instância*, que consiste em verificar se um indivíduo pertence a um conceito específico;
- verificação de *consistência* da base, para verificar se cada conceito possui pelo menos um indivíduo associado;
- *identificação conceitual*, para encontrar o conceito que descreve em mais detalhe um determinado indivíduo e
- *busca de indivíduos* a partir de um determinado conceito.

Horrocks [2003] descreve algoritmos de otimização para a implementação de raciocinadores com o objetivo de oferecer um desempenho aceitável em aplicações típicas baseadas em LD. Soluções gerais são difíceis quando o nível de complexidade das linguagens caminha para a formação de problemas intratáveis. As melhorias obtidas nessa direção ao longo do tempo têm, contudo, possibilitado a construção de uma nova geração de raciocinadores altamente otimizados e abrangentes, tais como:

- *FaCT++*, um raciocinador gratuito, portável e eficiente;
- *RacerPro*, um raciocinador comercial (pago) muito abrangente e eficiente;
- *Pellet*, um raciocinador gratuito para uso com sistemas desenvolvidos com a linguagem Java;
- *MSPASS*, um provador de teoremas gratuito com aplicações em LD.

4.3.3 Modelagem Conceitual com DL

A utilização de LD em problemas de modelagem conceitual pode acontecer de diversas maneiras, pois isso envolve muitas questões que podem ser consideradas em maior ou menor grau de profundidade. Direcionada ao desenvolvimento de sistemas, por exemplo, a LD poderia ser utilizada para assegurar a consistência do modelo, ficando em um plano secundário o mapeamento dos indivíduos (dados) do domínio de aplicação. O mapeamento conceitual de um sítio de comércio da Web, por outro lado, pode enfatizar, por exemplo, a busca de indivíduos a partir de um determinado conceito que represente um produto específico à venda. Diferentes problemas podem projetar diferentes técnicas de modelagem, sendo que uma solução generalizada para esse problema é uma questão bastante complexa.

Borgida and Brachman [2003] sugerem, contudo, algumas orientações gerais para a modelagem conceitual baseada em LD:

- identificar os indivíduos do universo de discurso considerado (vocabulário);
- enumerar conceitos que agrupam ou descrevem esses indivíduos;
- fazer a distinção entre conceitos atômicos e relacionamentos, que pode não ser óbvia;
- desenvolver uma taxonomia de conceitos, sempre levando em consideração possíveis disjunções e conjunções de conceitos mais específicos;
- identificar qualquer indivíduo que possa ser associado a todos os contextos possíveis, considerando a possibilidade de associá-lo a um conceito geral;
- procurar sistematicamente relacionamentos do tipo "todo-parte", criando regras específicas para eles;
- identificar outras propriedades dos objetos, e então os relacionamentos mais gerais que podem ser definidos entre esses objetos;
- determinar limitações locais que possam envolver limites de cardinalidade ou restrições de valor, refletindo isso nas declarações já existentes;

- determinar restrições mais gerais nos relacionamentos identificados, reorganizando a hierarquia de conceitos conforme a necessidade;
- fazer a distinção entre propriedades essenciais e incidentais dos conceitos;
- considerar as propriedades dos conceitos que possam ser utilizadas na simplificação da estrutura por intermédio de técnicas tais como as descritas por Guarino and Welty [2000].

A estrutura resultante de um processo de modelagem conceitual baseado em LD sempre estará a mercê de um processo de validação capaz de aferir sua consistência, um trabalho que pode ser efetuado por raciocinadores automatizados.

Neste capítulo foram discutidos alguns conceitos introdutórios de lógicas formais, com destaque para as informações e orientações relevantes em um processo de definição de ontologias. No próximo capítulo serão apresentadas informações sobre tratamento de linguagem, especialmente úteis nos processos que envolvem extração de informação semântica a partir de conteúdo textual.

Capítulo 5

Processamento de Linguagem Natural

O tratamento de linguagem por intermédio de computadores, também conhecido como Processamento de Linguagem Natural (PLN), surgiu como objeto de pesquisa das áreas de IA e de Lingüística, tendo como principal meta a interação homem-máquina. O enfoque inicial do PLN era, portanto, a interpretação e a geração de respostas em linguagem natural. Isso implicava em capturar o significado do que era interpretado e produzir resultados em linguagem natural a partir do conhecimento adquirido.

O PLN é normalmente aplicado à linguagem escrita e à linguagem falada. Considerando apenas o caso da linguagem escrita e ignorando a necessidade de geração de resultados em linguagem natural, chegamos ao problema da interpretação de textos escritos, que é de grande interesse para a Web Semântica. Na Web Semântica, o conhecimento representado nas ontologias tem origem, em grande parte dos casos, em conteúdo textual expresso em linguagem natural.

Os recentes avanços no desenvolvimento da Web Semântica têm reforçado a tese de que, em uma visão alternativa à perseguida na área de IA, os computadores não precisam necessariamente incorporar, como resultado de um processo de tratamento de linguagem, o significado total ou parcial do conteúdo analisado. Segundo Alesso and Smith [2006], o objetivo da Web Semântica é diferente da maioria dos sistemas baseados em Lógica, tais como os vistos em IA. O objetivo da Web Semântica é criar um sistema unificado que contenha um universo restrito, no qual possa haver tratabilidade e eficiência necessárias para aplicações reais.

É importante separar os processos de representação do conhecimento dos destinados ao raciocínio automatizado, para que estruturas rígidas como ontologias possam acomodar conteúdos semânticos versáteis e passíveis de múltiplas interpretações. Em IA, a preocupação com a projeção direta do conteúdo interpretado nas estruturas de representação do conhecimento é um fator determinante para o sucesso. Outras aplicações, como as vistas em Web Semântica, por exemplo, podem exigir um pouco menos, limitando-se a constituir estruturas de representação do conhecimento sobre as quais não se procuram respostas, mas tão somente consistência e possibilidade de inferência. Não obstante, esses objetivos alternativos, embora mais amenos, não deixam de ser difíceis, dependendo da linguagem descritiva adotada (Zolin [2007]).

Nas ontologias os conceitos devem ser definidos de forma única. A interpretação da linguagem escrita e a estruturação do conhecimento por ela veiculado devem obedecer a essa restrição. Contudo, a natureza abstrata dos conceitos e das relações entre conceitos e indivíduos nos permite configurar estruturas muito versáteis, sobre as quais é possível traçar linhas distintas de inferência no tempo e no espaço. De fato, sob a luz da análise sintática, diferentes tipos de sintagmas podem ser representados nas ontologias de maneiras muito distintas, na forma de conceitos ou de relações, permitindo a construção de uma ampla gama de possibilidades de representação do conhecimento e, conseqüentemente, de mecanismos de raciocínio personalizados.

5.1 PLN na linguagem escrita

Um texto escrito pode ser o registro de muitas mensagens combinadas ou distintas que, na linguagem falada, podem ser expressadas de diversas formas, com entonações diferentes ou com linguagem figurada, por exemplo. Na linguagem escrita essas mensagens devem ser devidamente explicitadas, caso contrário pode haver uma falha na transmissão da mensagem. O problema toma proporções maiores quando consideramos todas as possibilidades de comunicação entre interlocutores e receptores, tais como expressões visuais, gestos e o contexto no qual se inserem as mensagens. Luger [2005] acrescenta que a comunicação em linguagem natural, seja ela escrita ou falada, depende muito do nosso conhecimento e das nossas expectativas dentro do domínio de discurso.

Para aplicações em Web Semântica, particularmente nas que envolvem extração de conteúdo semântico, o PLN pode ser efetuado atualmente sobre grande parte do conteúdo da Web: em páginas HTML, sobre documentos de texto em geral ou sobre documentos digitalizados que contenham texto acessível através de OCR.

Russell and Norvig [2003] descrevem os seguintes princípios fundamentais normalmente adotados pelo PLN:

- *linguagem formal*: conjunto possivelmente infinito de *cadeias*;
- *cadeia*: concatenação de *símbolos terminais*;
- *símbolo terminal*: símbolos elementares, como os apresentados na descrição da sintaxe da linguagem $\mathcal{AL}$ (vide Capítulo 4), onde "C", "D" e "$\sqcap$" são símbolos terminais e "$C \sqcap D$", por exemplo, é uma cadeia que (ao contrário de "$CD\sqcap$") faz parte da linguagem;
- *gramática*: conjunto finito de regras que especificam uma linguagem.

Esses princípios, normalmente aplicados no tratamento de linguagens formais, são também utilizados no PLN durante as primeiras fases de análise do texto, a saber:

- *análise sintática*: fase em que o texto tem sua estrutura sintagmática desmembrada através de um processo que geralmente começa com a identificação dos diversos tipos de sintagmas e seus elementos constituintes, e que termina com a geração de uma estrutura sintática do texto;
- *interpretação semântica*: fase que consiste no mapeamento, através de lógica formal, dos elementos identificados durante a análise sintática.

A Figura 5.1 mostra todas as fases típicas do PLN, conforme descrito por Luger [2005]. O processo inicia-se com a análise sintática, continua com a interpretação semântica e culmina na representação do conhecimento extraído do texto, que consiste no enriquecimento da estrutura obtida com os dados já existentes em uma base de conhecimento ou em um conjunto de modelos pré-definidos.

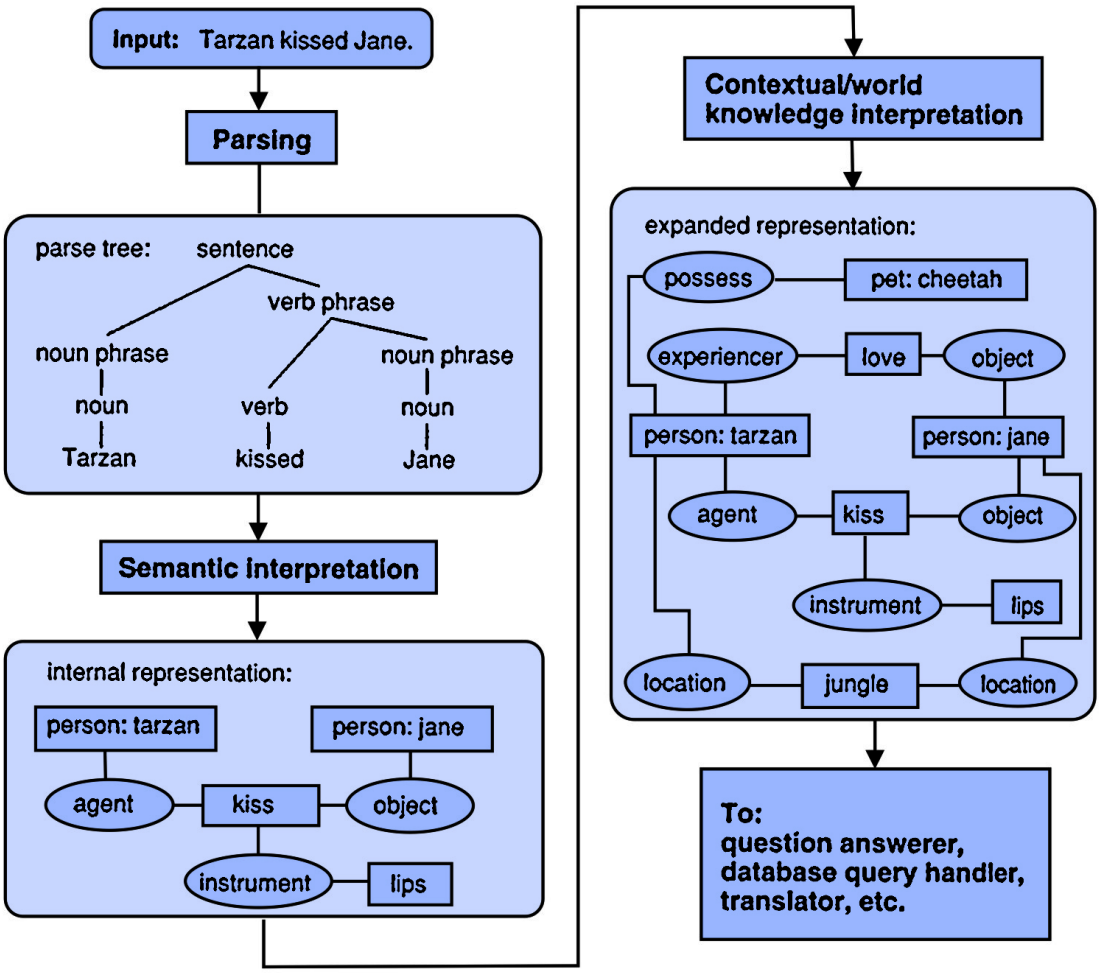

Figura 5.1: Fases típicas do PLN (Luger [2005]).

5.2 Análise Sintática

A análise sintática baseia-se no *reconhecimento* da estrutura do texto. A partir de uma sentença, deve-se identificar os sintagmas presentes e organizá-lo dentro de uma estrutura hierárquica. Para ilustrar esse processo, Russell and Norvig [2003] apresentam um fragmento da língua inglesa denominado ε_0, cuja lista de palavras permitidas (léxico) é mostrada na Figura 5.2.

Noun	→ **strench** | **breeze** | **glitter** | **nothing** | **agent**
	| **wumpus** | **pit** | **pits** | **gold** | **east** | **...**
Verb	→ **is** | **see** | **smell** | **shoot** | **feel** | **stinks**
	| **go** | **grab** | **carry** | **kill** | **turn** | **...**
Adjective	→ **right** | **left** | **east** | **dead** | **back** | **smelly** | **...**
Adverb	→ **here** | **there** | **nearby** | **ahead**
	| **right** | **left** | **east** | **south** | **back** | **...**
Pronoun	→ **me** | **you** | **I** | **it** | **...**
Name	→ **John** | **Mary** | **Boston** | **Aristotle** | **...**
Article	→ **the** | **a** | **an** | **...**
Preposition	→ **to** | **in** | **on** | **near** | **...**
Conjunction	→ **and** | **or** | **but** | **...**
Digit	→ **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

Figura 5.2: Léxico de ε_0 (Russell and Norvig [2003]).

Os sintagmas são formados por combinações de palavras do léxico. As regras de formação do sintagmas são especificadas através de uma *gramática*.

A gramática de ε_0 é mostrada na Figura 5.3. Um exemplo de sintagma aparece ao lado de cada regra.

S	→ *NP VP*	*I + feel a breeze*
	| *S Conjunction S*	*I feel a breeze + I smell a wumpus*
NP	→ *Pronoun*	*I*
	| *Name*	*John*
	| *Noun*	*pits*
	| *Article Noun*	*the + wumpus*
	| *Digit Digit*	*3 4*
	| *NP PP*	*the wumpus + to the east*
	| *NP RelClause*	*the wumpus + that is smelly*
VP	→ *Verb*	*stinks*
	| *VP NP*	*feel + a breeze*
	| *VP Adjective*	*is + smelly*
	| *VP PP*	*turn + to the east*
	| *VP Adverb*	*go + ahead*
PP	→ *Preposition NP*	*to + the east*
RelClause	→ **that** *VP*	*that + is smelly*

Figura 5.3: Gramática de ε_0 (Russell and Norvig [2003]).

A gramática de ε_0 define *sintagma* (*S*), *sintagma nominal* (*NP*), *sintagma verbal* (*VP*) e *sintagma preposicional* (*PP*), além da *cláusula relativa* (*RelClause*), que segue e modifica um sintagma nominal, sendo constituída pelo pronome relativo "*that*" seguido de um sintagma verbal. Os autores esclarecem que a gramática gera sentenças válidas em Inglês, tais como "*John is in the pit*", mas gera também sentenças que não

são gramaticalmente corretas, tais como "*Me go Boston*". Além disso, sentenças válidas, como "*I think the wumpus is smelly*", são rejeitadas pela gramática.

Existem muitos formalismos gramaticais utilizados para análises sintáticas. Chomsky [1957] definiu uma hierarquia de classes formalismos gramaticais, apresentadas aqui sob o ponto de vista de suas capacidade de reconhecimento de linguagens:

- as *gramáticas regulares*, que constituem a classe mais restrita, reconhecem determinadas expressões regulares, tais como a^+b^+ (uma seqüência de caracteres "a" seguida de uma seqüência de caracteres "b");
- as *gramáticas livres de contexto*, que incluem a gramática ε_0 da Figura 5.3, reconhecem, além das expressões reconhecidas pelas gramáticas regulares, expressões balanceadas tais como a^nb^n (uma seqüência de n "a"s seguidas da mesma quantidade de "b"s), sendo que a principal característica dessa classe de formalismo é o reconhecimento das cadeias independente do contexto, como acontece no exemplo de ε_0;
- as *gramáticas sensíveis ao contexto*, que reconhecem, além das expressões reconhecidas pelas gramáticas livres de contexto, expressões mais complexas, tais como $a^nb^nc^n$ (uma seqüência de n "a"s seguidas da mesma quantidade de "b"s e de "c"s), sendo que a principal característica dessa classe de formalismo é a possibilidade de definição de regras mais flexíveis pelas quais estabelecem-se contextos de aplicação;
- as *gramáticas recursivamente enumeráveis*, as mais amplas de todas, que podem reconhecem, além das expressões reconhecidas pelas gramáticas sensíveis ao contexto, expressões ainda mais complexas, pois permitem que as regras sejam especificadas de forma menos restrita.

Os formalismos gramaticais são utilizados na implementação das ferramentas utilizadas no desenvolvimento de *softwares* reconhecedores de sentenças de uma determinada linguagem. Entre essas ferramentas destacam-se o *Lex* e o *Yacc* (Levine et al. [1992]).

O processo de análise sintática de uma sentença normalmente resulta em uma estrutura conhecida como *árvore sintática*, composta por todos os sintagmas reconhecidos, dispostos na árvore de acordo com sua posição hierárquica, conforme especificado na gramática utilizada para o reconhecimento da linguagem. A Figura 5.4 mostra um exemplo de árvore sintática criada para a sentença de exemplo "*the wumpus is dead*".

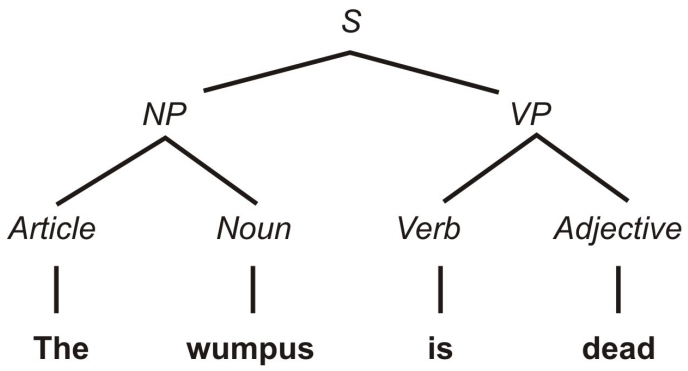

Figura 5.4: Árvore sintática típica (Russell and Norvig [2003]).

Luger [2005] observa que os *softwares* reconhecedores podem adotar muitas estratégias diferentes, mas todos utilizam algoritmos que podem ser classificados da seguinte forma:

- algoritmos que partem da identificação de palavras para reconhecerem os sintagmas, trabalhando de baixo para cima (algoritmos *bottom-up*);
- algoritmos que partem da definição dos sintagmas para obterem cadeias de caracteres que combinam com as sentenças de entrada (algoritmos *top-down*).

5.3 Interpretação Semântica

Após a fase de análise sintática, o PLN prossegue com a fase de interpretação semântica da árvore sintática previamente reconhecida. O objetivo dessa fase é obter sentenças lógicas a partir dos elementos sintáticos identificados.

Em IA, diversas abordagens tem sido adotadas para a obtenção de sentenças lógicas, sendo a LPO uma das principais linguagens de representação utilizadas. Nesse contexto, a interpretação semântica consiste na associação de um sintagma a uma expressão em LPO.

Para o exemplo da Figura 5.4, os autores sugerem que a sentença lógica *Dead(wumpus)* seria uma representação possível para a sentença "*the wumpus is dead*". Isso implicaria na definição do predicado *Dead* e na sua associação ao termo "*wumpus*". Outra possibilidade seria, por exemplo, a associação de "*wumpus*" ao conceito "*Death*" (morte). Numerosas implicações podem ser derivadas dessas associações, como por exemplo a associação da morte de "*wumpus*" a um evento ocorrido no passado, causa de sua condição atual.

O uso de lógica formal para a atribuição de significado a sentenças, no PLN, é apenas uma das muitas abordagens para o problema de interpretação semântica. Nirenburg and Raskin [2004] descrevem o histórico da busca pela definição das regras efetivas para a atribuição de significado a sentenças de texto em linguagem natural e constatam que o PLN deve necessariamente escolher uma dentre muitas possibilidades de interpretação possíveis, o que caracteriza um processo de simplificação da realidade.

A busca pelo significado das sentenças de texto é uma questão importante em Lingüística, Filosofia e PLN. A teoria de Katz and Fodor [1963] foi a primeira tentativa importante de se estabelecer o formato de uma teoria semântica para esse problema, cujos objetivos são fortemente compatíveis com os objetivos do PLN (Nirenburg and Raskin [2004]). Os autores basearam-se na construção de um dicionário e de um conjunto de regras de avaliação. As principais motivações da teoria foram as seguintes:

- determinação do número de significados possíveis de cada sentença analisada;
- determinação de cada significado propriamente dito;
- detecção de anomalias semânticas, tais como sentenças sem sentido;
- percepção de sentenças parafraseadas (paráfrases).

5.4 Recursos

O desenvolvimento de aplicações baseadas no PLN não poderia ser levado a efeito sem a existência de dados específicos sobre a linguagem processada, tais como dicionários e regras gramaticais. Ferramentas para a análise sintática são também imprescindíveis, mesmo quando essas ferramentas são utilizadas apenas para comparações ou para a avaliação do resultado do processamento.

Passaremos a expor algumas ferramentas úteis em PLN.

5.4.1 Wordnet

O WordNet [2006] é um banco de dados léxico, disponível em muitas línguas. Ele agrupa palavras em conjuntos de sinônimos chamados de *synsets*, os quais contém a descrição do significado das palavras e suas relações com outras. O WordNet [2006] é próprio para processamento de texto e aplicações em IA, por exemplo.

Os *synsets* são conectados entre si através de várias relações semânticas, dentre as quais destacam-se:

- *antonímia*, que relaciona as unidades lexicais opostas;
- *hiponímia*, que indica as unidades lexicais especializadas, como por exemplo "canino" e "cachorro", onde "cachorro" é um hipônimo de "canino";
- *hiperonímia*, que indica as unidades lexicais generalizadoras, como por exemplo "rosa" e "flor", onde "flor" é um hiperônimo de "rosa";
- *troponímia*, que indica os verbos relevantes relacionados (Fellbaum [1998]);
- *meronímia*, que indica as unidades lexicais componentes, como por exemplo "asa" e "avião", onde "asa" é uma parte (merónimo) de "avião";
- *holonímia*, que indica as unidades lexicais englobadoras, como por exemplo "prédio" e "janela", onde "prédio" é um holónimo de "janela";
- *acarretamento*, que indica relações de implicação, como por exemplo "comprar" e "pagar";
- *causa*, que indica relações de causa e efeito, como por exemplo "matar" e "morrer".

S: (v) **crash**	**(fall or come down violently)** "The branch crashed down on my car"; "The plane crashed in the sea"
S: (v) **crash**	**(move with, or as if with, a crashing noise)** "The car crashed through the glass door"
S: (v) **crash, ram**	**(undergo damage or destruction on impact)** "the plane crashed into the ocean"; "The car crashed into the lamp post"

Tabela 5.1: Fragmento de informações do WordNet (WordNet [2010])

.

A *Global WordNet Association* (GWA), entidade responsável pela padronização de *wordnets* para todas as linguagens do mundo (GWA [2000]), estima que atualmente já existem *wordnets* para mais de 45 línguas diferentes (GWA [2010]).

O banco de dados mais conhecido é o da língua inglesa. A Tabela 5.1 mostra um pequeno fragmento de informações disponíveis para a palavra "*crashed*".

Para a língua portuguesa, destaca-se o projeto do WordNet [2006] de Portugal (WordNet.PT [1998]). No Brasil existe um esforço para o desenvolvimento de uma *wordnet* específica para o Português brasileiro (da Silva et al. [2006]).

5.4.2 Sistema PALAVRAS

O sistema PALAVRAS (Bick [2000]) é um analisador gramatical da língua portuguesa. Ele extrai, a partir de um texto, os elementos gramaticais existentes, apresentando um rico conjunto de atributos que ajudam a identificar a função de cada elemento identificado.

O sistema PALAVRAS, disponível para utilização via Web em Palavras [2000], apresenta o resultado da análise gramatical de um texto informado pelo usuário em uma interface gráfica. A Figura 5.5 mostra a apresentação da árvore sintática obtida a partir da frase de exemplo "*O carro bateu na árvore.*".

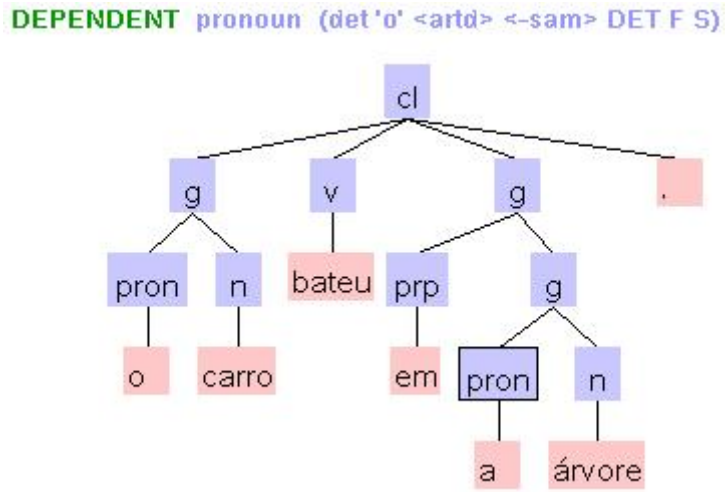

Figura 5.5: Árvore obtida com o sistema PALAVRAS (Palavras [2000]).

Além de uma interface gráfica de visualização das árvores sintáticas, o sistema oferece a possibilidade de fornecer, como resultado de processamento, representações textuais dessas árvores. Resultados assim representados podem ser utilizados por qualquer aplicação capaz de interpretar o formato utilizado pelo sistema PALAVRAS. Essa representação é também conhecida como "árvore sintática deitada" ou simplesmente "árvore deitada". A Figura 5.6 mostra um exemplo de resultado apresentado no formato textual para a frase de exemplo "*O cachorro, extremamente perigoso, mordeu o dono e depois atacou o vizinho*".

O resultado do processamento do sistema PALAVRAS contém a estrutura do texto e anotações de marcas gramaticais para os diversos elementos identificados no texto. O Apêndice A descreve todas as marcas gramaticais utilizadas pelo sistema PALAVRAS (VISL-Manual [2000]).

```
SOURCE: Running text
1. o cachorro, extremamente perigoso, mordeu o dono e depois atacou o vizinho.
A1
STA:fcl
S:np
=DN:pron("o"<artd> DET M <indf> S)                    o
=H:n("cachorro"M S)                                   cachorro
=,
=DNc:adjp
==DA:adv("extremamente"<quant>)                       extremamente
==H:adj("perigoso"M S)                                perigoso
,
X:par
=CJT:x
==P:v-fin("morder"<fmc> PS 3S IND VFIN)               mordeu
==Od:np
===DN:pron("o"<artd> DET M <indf> S)                  o
===H:n("dono"M S)                                     dono
=CO:conj-c("e"<co-fin> <co-fmc>)                      e
=CJT:x
==fA:adv("depois"<left>)                              depois
==P:v-fin("atacar"<fmc¿ PS 3S IND VFIN)               atacou
==Od:np
===DN:pron("o"<artd> DET M <indf> S)                  o
===H:n("vizinho"M S)                                  vizinho
.
```

Figura 5.6: Árvore em forma de texto do sistema PALAVRAS (Palavras [2000]).

5.4.3 Corporas

Corporas são grandes conjuntos de textos destinados a estudos diversos, podendo conter anotações linguísticas (marcas gramaticais) que facilitam a identificação dos elementos gramaticais dos textos neles contidos.

A utilização de *corporas* é importante para a realização de experimentações com grandes volumes de texto previamente anotado com marcas gramaticais. Existem diversos *corporas* em língua portuguesa disponíveis para consulta, com destaque para os seguintes:

- *Corpus de Extractos de Textos Electrónicos MCT/Público* (CETEMPúblico) - textos jornalísticos de Portugal com marcas gramaticais geradas pelo sistema PALAVRAS e revisados por lingüistas, contendo cerca de 180 milhões de palavras (Linguateca [2000]);
- *Corpus de Extractos de Textos Electrónicos NILC/Folha de S. Paulo* (CETEMFolha) - textos jornalísticos do Brasil com marcas gramaticais geradas pelo sistema PALAVRAS e revisados por lingüistas, contendo cerca de 24 milhões de palavras (Linguateca [2002]);
- *Corpus Internacional do Português* CINTIL - oferece uma interface de busca de exemplos de uso dos argumentos pesquisados pelo usuário em um universo de cerca de 1 milhão de palavras (CINTIL [2004]).

Embora esses *corporas* constituam um excelente acervo de pesquisa em língua portuguesa, suas aplicações em PLN são prejudicadas pelo fato dos textos neles contidos não possuírem suas estruturas hierárquicas (árvores sintáticas) explicitamente representadas. Entretanto, o Projeto Floresta Sintática (Freitas et al. [2008]), descrito adiante, possui subconjuntos desses *corporas* com árvores sintáticas incluídas.

5.4.4 Linguateca

O portal de recursos Linguateca (Linguateca [1998]) é um centro de recursos distribuídos para o processamento computacional da língua portuguesa. O portal destina-se a facilitar o acesso aos recursos da língua portuguesa já existentes através do desenvolvimento, em colaboração com usuários e pesquisadores interessados, dos recursos considerados mais importantes.

O Linguateca tem como principal diretriz a total abertura, ou seja, a disponibilização irrestrita do conteúdo do portal ao público em geral, evitando o desenvolvimento de recursos proprietários e favorecendo os recursos gratuitos.

Dentre os diversos recursos mantidos pelo Linguateca, destaca-se o Projeto Floresta Sintática (Freitas et al. [2008]), cujo acesso gratuito está disponível em Projeto Floresta Sintática [2002].

A "Floresta Sintática" é formada por um conjunto de *corporas* anotados pelo sistema PALAVRAS, contendo não apenas as marcas gramaticais dos elementos processados, mas também a estrutura hierárquica dos textos, possibilitando dessa maneira o reconhecimento das árvores sintáticas e a sua adoção em aplicações de PLN. Por terem sido revisados por lingüistas, esses *corporas* constituem uma acervo textual muito confiável.

Atualmente, o material da Floresta Sintática soma cerca de 6,7 milhões de palavras sintaticamente analisadas, sendo dividido da seguinte forma:

- *Bosque* - totalmente revisto por linguistas, composto por 9368 frases retiradas os primeiros 1000 extratos dos *corporas* CETEMFolha e CETEMPúblico;
- *Selva* - com *corporas* parcialmente revistos e cerca de 300 mil palavras dividida em diferentes gêneros textuais, do português de Portugal e do Brasil;
- *Floresta Virgem* - não revista por lingüistas, formada por cerca de 95 mil frases e 1,6 milhões de palavras retiradas das partes iniciais dos *corporas* CETEMFolha e CETEMPúblico;
- *Amazônia* - não revista por lingüistas, com cerca de 4,6 milhões de palavras (cerca de 275 mil frases), retiradas do sítio Overmundo [2006], com todo o conteúdo em português do Brasil.

Os *corporas* da Floresta Sintática que utilizam fragmentos dos *corporas* CETEMFolha e CETEMPúblico diferem destes pelo fato de possuírem marcas gramaticais mais detalhadas, com a inclusão das árvores sintáticas deitadas semelhantes à mostrada na Figura 5.6.

Neste capítulo foram apresentadas diversas informações sobre tratamento de linguagem e processos que envolvem análise sintática de conteúdo textual e interpretação semântica. No próximo capítulo será descrito o processo de geração automática de ontologias, objeto deste trabalho.

Capítulo 6

Descrição do Trabalho

Ao traçarmos um caminho entre o conteúdo textual de um documento da Web e sua representação ontológica, devemos preservar a legibilidade da informação, evitando encaminhar a estrutura resultante para um viés semântico. Vimos no Capítulo 1 que um texto pode ser a representação escrita de muitas interpretações, todas associadas à mesma estrutura sintática que encerra os elementos constituintes das frases e respectivas ligações internas, completamente identificáveis e passíveis de análise gramatical.

A representação ontológica da estrutura sintática do texto original pode constituir uma base de conhecimento que, quando combinada com ontologias específicas de um determinado domínio, possibilita o tratamento semântico da informação original em um nível de abstração muito mais alto do que o que seria necessário unicamente com a utilização do conteúdo textual.

Preservar, na medida do possível, a estrutura geral do conteúdo textual original na ontologia resultante foi o caminho escolhido para o processo de geração automática de ontologias descrito neste trabalho. Ao abrirmos mão de uma análise semântica mais elaborada nos estágios iniciais de desenvolvimento de ontologias, como a que seria natural em um PLN típico, estamos deixando o caminho livre para explorações posteriores que podem ser efetuadas sobre os resultados obtidos, através do uso destes como ponto de partida para ontologias derivadas ou como referência de pesquisa de conteúdo, ou ainda como subsídio para mecanismos de inferência personalizados.

Assim, o processo de geração automática de ontologias aqui descrito procura vincular conteúdos de documentos da Web a estruturas nas quais conceitos e relacionamentos são determinados exclusivamente a partir das estruturas gramaticais existentes e das respectivas ligações entre elas.

Não obstante, esse processo não está livre de erros, já que nem sempre é possível determinar quais são exatamente as ligações entre as estruturas gramaticais identificadas, seja por causa de ambigüidades inerentes à língua estudada, seja devido a problemas com os próprios textos de origem.

O problema de associação da informação presente nos textos de origem ao conhecimento existente no domínio de aplicação é abordado pelo processo através da combinação de ontologias pré-existentes com o resultado da análise sintática dos textos. Esse recurso permite a leitura, conforme a necessidade, de interpretações diversas a partir de um único texto de origem.

6.1 Histórico

Com foco na Web Semântica, considerada peça importante no ambiente interativo em que a Web se transformou, este trabalho iniciou-se com a modelagem de um sistema de busca semântica, cujo propósito era o de juntar as diversas peças que compõe o intrincado quebra-cabeças do tratamento semântico das informações livremente disponibilizadas na rede. A busca semântica é um exemplo de aplicação que envolve tanto a criação de conteúdo semântico quanto sua utilização efetiva, constituindo um ciclo típico de processamento na Web Semântica.

Avaliamos alguns *corporas* com a finalidade de identificar formas de mapear conceitos e relações a partir da análise das marcas gramaticais disponíveis. A representação do conhecimento através da interligação do conteúdo da Web passou a ser um objeto importante de pesquisa na medida em que os resultados intermediários começaram a aparecer. Um resumo desses resultados iniciais é descrito no Apêndice B. Mapas preliminares de interligação de conteúdo foram gerados a partir do conteúdo dos *corporas*. A transformação desses mapas em ontologias foi então considerada muito relevante por endereçar a motivação inicial de modelagem de um sistema de busca semântica sem a necessidade de implementação tão específica e ao mesmo tempo alinhar o projeto com a linguagem adotada na Web Semântica.

As estruturas de representação dos conceitos e suas relações obtidas inicialmente, quando materializadas na forma de ontologias, evidenciaram o fato de que a geração automática de ontologias seria de grande utilidade na Web Semântica, tendo em vista a facilidade na obtenção de ontologias derivadas para aplicações em domínios específicos.

Dessa forma iniciou-se, com base nos resultados iniciais, a definição de um processo de geração automática de ontologias e a construção de um protótipo para a sua implementação. pertinentes. A metodologia adotada neste trabalho é apresentada aqui através da descrição desse protótipo e das decisões que nortearam seu desenvolvimento.

6.2 Metodologia Adotada

O processo de geração automática de ontologias é dividido em quatro etapas. Na Seção 6.2.1 apresentamos uma visão geral do processo, seguida da descrição, nas seções subseqüentes, de cada etapa que o constitui, a saber: pré-processamento, análise sintática, interligação de conteúdo e geração de ontologias.

6.2.1 Visão Geral

A Figura 6.1 apresenta, como retângulos destacados, todas as etapas do processo de geração de ontologias, mostradas em um fluxo iniciado com a leitura de documentos da Web e terminado com a geração das ontologias que representam esses documentos.

O protótipo implementado é composto por módulos que executam cada etapa separadamente. Essa disposição facilita a análise de resultados intermediários e permite a execução de cada etapa isoladamente.

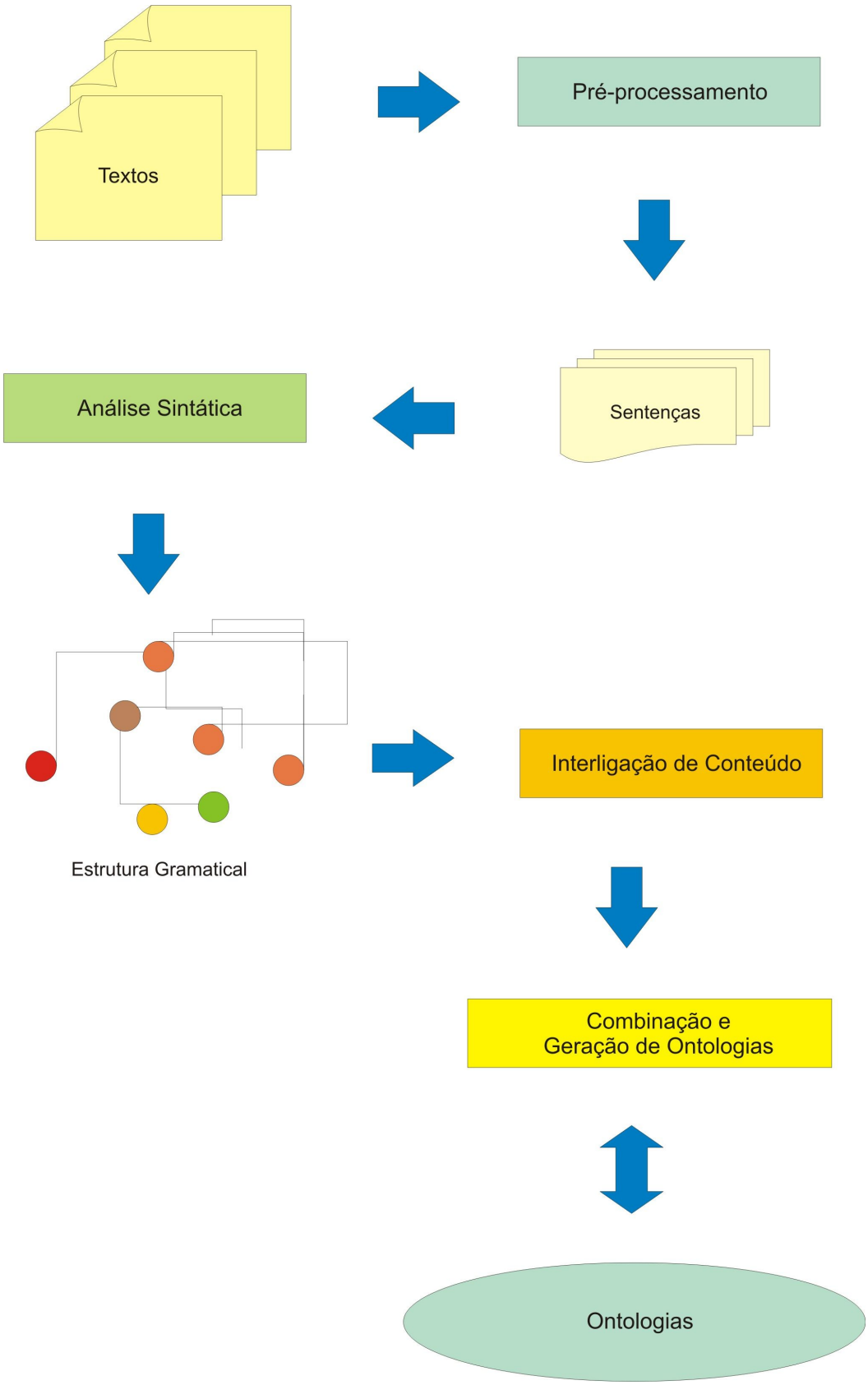

Figura 6.1: Visão geral do processo de geração automática de ontologias.

Segue uma breve descrição de cada etapa:

- *pré-processamento* - realiza a limpeza dos documentos de entrada, criando um fluxo de texto cuja unidade básica de processamento é uma sentença ou período (Bechara [2009]);
- *análise sintática* - executa uma análise sintática no conteúdo textual pré-processado, construindo uma estrutura gramatical que serve de base para as etapas posteriores;
- *interligação de conteúdo* - percorre a estrutura gramatical construída na etapa anterior, enriquecendo-a com os atributos obtidos durante a análise dos elementos gramaticais identificados e formando um mapa desses elementos com seus respectivos inter-relacionamentos;
- *combinação e geração de ontologias* - transforma o mapa obtido na etapa anterior em uma ontologia representativa do texto de origem, a qual pode ser associada a um contexto específico através da combinação do seu conteúdo com o de um conjunto de ontologias pré-existentes.

Pode-se observar que até a etapa de combinação e geração de ontologias há um fluxo unidirecional de informações, quando então ocorre a incorporação de um modelo de ontologia que encerra os conceitos abstratos de mais alto nível necessários para a representação dos elementos gramaticais extraídos dos textos. Esse modelo, descrito na Seção 6.2.4.2, é uma *ontologia base* criada ao longo deste trabalho para ser incorporada pelas ontologias geradas. A versão atual dessa ontologia encontra-se em Bravo [2010a].

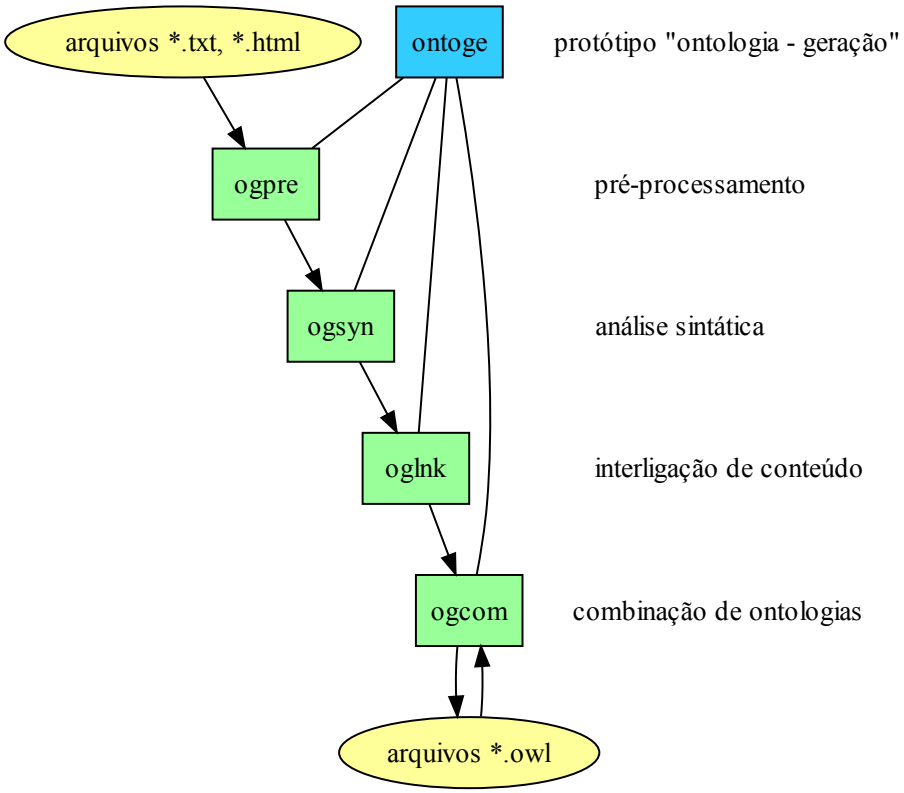

Figura 6.2: Visão geral do protótipo implementado.

O protótipo de geração automática de ontologias, cuja estrutura original é mostrada na Figura 6.2, teve sua implementação iniciada a partir de algumas escolhas fundamentais, de caráter geral, a saber:

- utilização de uma linguagem de programação capaz de manipular eficientemente conteúdos textuais, ao nível dos caracteres;
- organização interna baseada em classes e objetos, com o intuito de permitir um desenvolvimento orientado a objetos e ao mesmo tempo facilitar a manutenção do trabalho;
- capacidade de geração de código nativo, para melhores tempos de resposta no tratamento de grandes volumes de texto;
- capacidade de integração e de incorporação de ferramentas de terceiros;
- utilização de definições e de recursos que garantissem a portabilidade do código entre ambientes distintos.

A linguagem de programação C++ foi a que melhor atendeu a esses requisitos, tendo sido portanto a linguagem escolhida para o desenvolvimento do protótipo, cuja versão atualizada está disponível em Bravo [2010c].

Os módulos componentes do protótipo são programas independentes que trocam informações através de arquivos. O programa principal chama-se *ontoge*, cujo nome é uma abreviatura de "*ontologias - geração*".

Os demais componentes são os seguintes:

- *ogpre* - o pré-processador, que filtra arquivos de texto ou arquivos HTML para deixar apenas uma seqüência de texto puro, recodificando caracteres especiais, preservando a pontuação e garantindo que o texto pré-processado não fique em um formato que prejudique o processamento subseqüente;
- *ogsyn* - o analisador sintático, atualmente baseado no *sistema PALAVRAS* (vide Seção 5.4.2), conforme descrito adiante, responsável pela geração de uma uma árvore sintática deitada;
- *oglnk* - o interligador de conteúdo, que interpreta o arquivo disponibilizado pelo componente *opsyn* e monta uma árvore sintática em memória, cuja estrutura passa por várias transformações antes de ser disponibilizada para a etapa subseqüente;
- *ogcom* - o combinador de ontologias, que traduz as informações disponibilizadas pelo componente *oglnk* em uma estrutura representativa do texto, combinando-a então com o conteúdo de ontologias pré-existentes.

Podemos observar que cada componente do protótipo tem como função executar, de forma independente, uma etapa específica do processo de geração automática de ontologias.

Nas seções subseqüentes veremos cada uma dessas etapas separadamente, discutindo as transformações efetuadas por elas e analisando os resultados intermediários produzidos pelos componentes do protótipo.

6.2.2 Pré-processamento

A etapa de pré-processamento é a primeira a ser executada pelo processo de geração de ontologias. Nessa etapa são efetuadas as seguintes operações:

- identificação das partes componentes do documento, tais como título (HTML), parágrafos e frases;
- remoção de códigos de marcação;
- tradução de caracteres HTML especiais para a codificação ISO 8859-1;
- seleção de texto com base em parâmetros de limitação de escopo informados pelo usuário;
- gravação de um documento de texto que contém apenas frases.

A Figura 6.3 ilustra o funcionamento do programa *ogpre*, o componente do protótipo responsável pela etapa de pré-processamento.

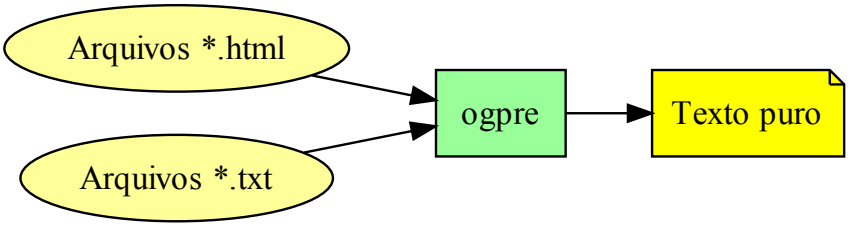

Figura 6.3: Componente *ogpre* (pré-processador) do protótipo.

Na etapa de pré-processamento, a parte do documento de entrada a ser tratada pode ser selecionada através de dois parâmetros informados pelo usuário: uma seqüência de caracteres que marca o início do texto desejado e outra seqüência que marca o fim. Dessa forma é possível excluir conteúdos irrelevantes ou destinados exclusivamente à visualização.

Os arquivos (HTML) podem ter seus conteúdos formatados de diversas maneiras, sendo que em alguns casos os textos neles contidos aparecem completamente fragmentados, favorecendo a visualização do conteúdo mas provocando, entretanto, o efeito colateral de dificultar o processamento automatizado da informação. Esforços como os promovidos por W3C [1999] têm incentivado a produção de documentos mais legíveis tanto para pessoas quanto para processos automatizados, mas mesmo nesses casos pode ser necessária a seleção prévia do conteúdo dos documentos.

Um exemplo de arquivo de entrada HTML para o pré-processador é mostrado na Figura 6.4. Podemos observar que o título do documento (*"A Igreja do Diabo"*) aparece duas vezes: uma na linha 4 e outra na linha 30. Estabelecendo como filtro o texto compreendido entre as cadeias "*A IGREJA*" e "*azul.*", selecionamos o texto que começa na linha 30 e obtemos do pré-processador o resultado mostrado na Figura 6.5, na qual é possível verificar, com a ajuda da numeração de linhas apresentada, que cada frase do resultado aparece em uma linha separada, com o objetivo de facilitar o trabalho da etapa subseqüente do processo.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD><META name="dc.creator" content="Machado de Assis">
<TITLE>A Igreja do Diabo</TITLE></HEAD>
<BODY bgcolor=white text=black link=blue vlink=purple alink=fushia>
<DIV class="Part">
<H4><B>A Igreja do Diabo, de Machado de Assis</B></H4>
<P>Fonte:</P>
<P>ASSIS, Machado de. Volume de contos. Rio de Janeiro : Garnier, 1884.</P>
</DIV>
<DIV class="Part">
<H4><B>Texto proveniente de:</B></H4>
<P>A Biblioteca Virtual do Estudante Brasileiro
&lt;http://www.bibvirt.futuro.usp.br</B> A Escola do Futuro da Universidade de
S&atilde;o Paulo Permitido o uso apenas para fins educacionais.</P></DIV>
<DIV class="Part">
<H4><B>Texto-base digitalizado por:</H4>
<P></B>Edi&ccedil;&atilde;o eletr&ocirc;nica produzida pela Costa Flosi Ltda.
Revis&atilde;o: Sandra Flosi/Edi&ccedil;&atilde;o: Edson Costa Flosi e Nancy
Costa</P>
<P>Este material pode ser redistribu&iacute;do livremente, desde que n&atilde;o
seja alterado, e que as informa&ccedil;&otilde;es acima sejam mantidas. Para
maiores informa&ccedil;&otilde;es, escreva para
&lt;bibvirt@futuro.usp.br&gt;.</P>
<P>Estamos em busca de patrocinadores e volunt&aacute;rios para nos ajudar a
manter este projeto. Se voc&ecirc; quer ajudar de alguma forma, mande um e-mail
para &lt;bibvirt@futuro.usp.br&gt; e saiba como isso &eacute;
poss&iacute;vel.</P>
</DIV>
<DIV class="Part"><H4 align="center"></H4><B>A IGREJA DO DIABO</B></H4>
<P align="center">Cap&iacute;tulo I</P><P>De uma id&eacute;ia mir&iacute;fica</P>
<P></B>Conta um velho manuscrito beneditino que o Diabo, em certo dia, teve a
id&eacute;ia de fundar uma igreja. Embora os seus lucros fossem
cont&iacute;nuos e grandes, sentia-se humilhado com o papel avulso que
exercia desde s&eacute;culos, sem organiza&ccedil;&atilde;o, sem regras,
sem c&acirc;nones, sem ritual, sem nada. Vivia, por assim dizer, dos
remanescentes divinos, dos descuidos e obs&eacute;quios humanos. Nada fixo,
nada regular. Por que n&atilde;o teria ele a sua igreja? Uma igreja do Diabo
era o meio eficaz de combater as outras religi&otilde;es, e destru&iacute;-las
de uma vez.</P>
<P>&mdash; V&aacute;, pois, uma igreja, concluiu ele. Escritura contra
Escritura, brevi&aacute;rio contra brevi&aacute;rio. Terei a minha missa, com
vinho e p&atilde;o &agrave; farta, as minhas pr&eacute;dicas, bulas, novenas e
todo o demais aparelho eclesi&aacute;stico. O meu credo ser&aacute; o
n&uacute;cleo universal dos esp&iacute;ritos, a minha igreja uma tenda de
Abra&atilde;o. E depois, enquanto as outras religi&otilde;es se combatem e se
dividem, a minha igreja ser&aacute; &uacute;nica; n&atilde;o acharei diante de
mim, nem Maom&eacute;, nem Lutero. H&aacute; muitos modos de afirmar; h&aacute;
s&oacute; um de negar tudo. </P>
<P>Dizendo isto, o Diabo sacudiu a cabe&ccedil;a e estendeu os bra&ccedil;os,
com um gesto magn&iacute;fico e varonil. Em seguida, lembrou-se de ir ter com
Deus para comunicar-lhe a id&eacute;ia, e desafi&aacute;-lo; levantou os olhos,
acesos de &oacute;dio, &aacute;speros de vingan&ccedil;a, e disse consigo:
&mdash; Vamos, &eacute; tempo. E r&aacute;pido, batendo as asas, com tal
estrondo que abalou todas as prov&iacute;ncias do abismo, arrancou da sombra
para o infinito azul.</P></DIV>
</BODY></HTML>
```

Figura 6.4: Exemplo de HTML de entrada para o pré-processador (MEC [2008]).

O caractere "%" é utilizado pelo pré-processador para marcar quebras de parágrafos e para injetar no arquivo de saída algumas informações adicionais sobre a estrutura do documento. Atualmente a única informação propagada dessa maneira é a marca HTML "*<TITLE>*", que não apareceu no resultado mostrado na Figura 6.5 porque não fazia parte do fragmento de texto selecionado.

```
A IGREJA DO DIABO .
%
Capítulo I .
%
De uma idéia mirífica .
%
Conta um velho manuscrito beneditino que o Diabo, em certo dia, teve a idéia de fundar uma igreja .
Embora os seus lucros fossem contínuos e grandes, sentia-se humilhado com o papel avulso que exercia desde séculos, sem organização, sem regras, sem cânones, sem ritual, sem nada .
Vivia, por assim dizer, dos remanescentes divinos, dos descuidos e obséquios humanos .
Nada fixo, nada regular .
Por que não teria ele a sua igreja ?
Uma igreja do Diabo era o meio eficaz de combater as outras religiões, e destruí-las de uma vez .
%
- Vá, pois, uma igreja, concluiu ele .
Escritura contra Escritura, breviário contra breviário .
Terei a minha missa, com vinho e pão à farta, as minhas prédicas, bulas, novenas e todo o demais aparelho eclesiástico .
O meu credo será o núcleo universal dos espíritos, a minha igreja uma tenda de Abraão .
E depois, enquanto as outras religiões se combatem e se dividem, a minha igreja será única; não acharei diante de mim, nem Maomé, nem Lutero .
Há muitos modos de afirmar; há só um de negar tudo .
%
Dizendo isto, o Diabo sacudiu a cabeça e estendeu os braços, com um gesto magnífico e varonil .
Em seguida, lembrou-se de ir ter com Deus para comunicar-lhe a idéia, e desafiá-lo; levantou os olhos, acesos de ódio, ásperos de vingança, e disse consigo :
- Vamos, é tempo .
E rápido, batendo as asas, com tal estrondo que abalou todas as províncias do abismo, arrancou da sombra para o infinito azul .
```

Figura 6.5: Exemplo de saída do pré-processador (texto extraído de MEC [2008]).

O arquivo de resultado gerado pelo pré-processador é o ponto de partida da etapa de análise sintática descrita a seguir.

6.2.3 Análise Sintática

A etapa de análise sintática dos textos de entrada utiliza o sistema PALAVRAS (VISL-Manual [2000]) para a obtenção das árvores sintáticas dos textos. Essa escolha levou em consideração as seguintes observações:

- as árvores sintáticas deitadas fornecidas pelo sistema contém todas as informações necessárias para a reconstrução e mapeamento da estrutura do texto original;
- as anotações feitas pelo sistema são bastante ricas, o que favorece a manipulação da estrutura em etapas subseqüentes;
- a sintaxe adotada nas anotações já é bastante conhecida na área de processamento computacional da língua portuguesa e isso facilita a consulta a exemplos de uso e materiais de referência.

O sistema PALAVRAS é executado remotamente através da Web através do sítio VISL [2010], que é acessado por intermédio do *software* gratuito *wget* (GNU [2008]). A Figura 6.6 ilustra a interação entre o componente do protótipo responsável pela análise sintática, *ogsyn*, o sistema PALAVRAS e o *software wget*.

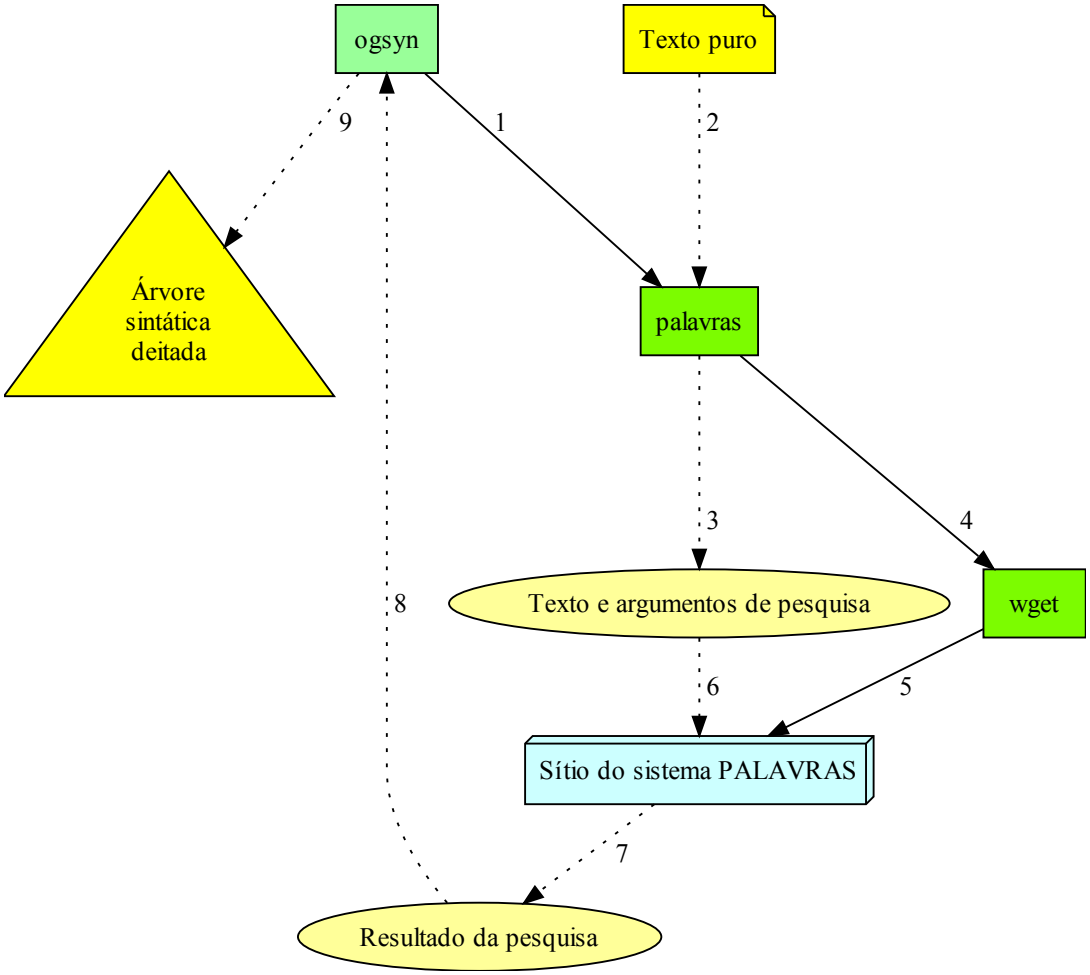

Figura 6.6: Interação entre o componente *ogsyn* e o sistema PALAVRAS.

Para a implementação do protótipo, além do componente *ogsyn*, que coordena a obtenção da árvore sintática do texto fornecido, foi desenvolvido o programa *palavras*, cuja função é preparar os parâmetros de comunicação e acessar remotamente o sistema PALAVRAS disponível no sítio VISL [2010], utilizando para isso o *software wget*.

Após a execução remota do sistema PALAVRAS, o resultado da pesquisa, retornado na forma de uma página HTML, é filtrado pelo componente *ogsyn*, que extrai a árvore deitada obtida e a disponibiliza em um arquivo de saída para a etapa subseqüente.

A vantagem da utilização remota do sistema PALAVRAS sobre a utilização de um *corpora* previamente anotado com marcas gramaticais está na possibilidade de realização de testes exaustivos sobre algum determinado aspecto gramatical específico que se deseja investigar em maior profundidade. Pode-se obter interativamente, com a utilização direta do sistema, toda a cadeia de resultados dos testes em questão, sendo essa opção melhor do que vasculhar *corporas* atrás de padrões específicos.

A Figura 6.7 mostra um arquivo de saída obtido com a frase de exemplo "*O calor está insuportável, mas vamos prosseguir*", após a filtragem, feita pelo componente *ogsyn*, do resultado fornecido pelo sistema PALAVRAS. As linhas precedidas pelo caractere "%" são linhas de comentários com informações adicionais fornecidas pelo sistema PALAVRAS e não fazem parte da árvore sintática do texto.

```
1  % SOURCE: Running text
2  % 1. o calor está insuportável, mas vamos prosseguir.
3  % A1
4  STA:fcl
5  S:np
6  =DN:pron("o" <artd> DET M <indf> S)      o
7  =H:n("calor" M S)        calor
8  X:par
9  =CJT:x
10 ==P:v-fin("estar" <fmc> PR 3S IND VFIN) está
11 ==Cs:adj("insuportável" M S)     insuportável
12 =,
13 =CO:conj-c("mas" <co-fin> <co-fmc>)      mas
14 =CJT:x
15 ==P:vp
16 ===VVaux:v-fin("ir" <fmc> PR 1P IND VFIN)        vamos
17 ===Vm:v-inf("prosseguir" <mv>)  prosseguir
18 .
```

Figura 6.7: Exemplo de árvore sintática deitada filtrada pelo componente *ogsyn*.

A Figura 6.8 mostra a representação inicial da árvore sintática desse exemplo. Alguns atributos dos elementos gramaticais identificados foram omitidos para facilitar a visualização. A descrição das marcas gramaticais anotadas pelo sistema PALAVRAS e todos os atributos relacionados estão descritos no Apêndice A.

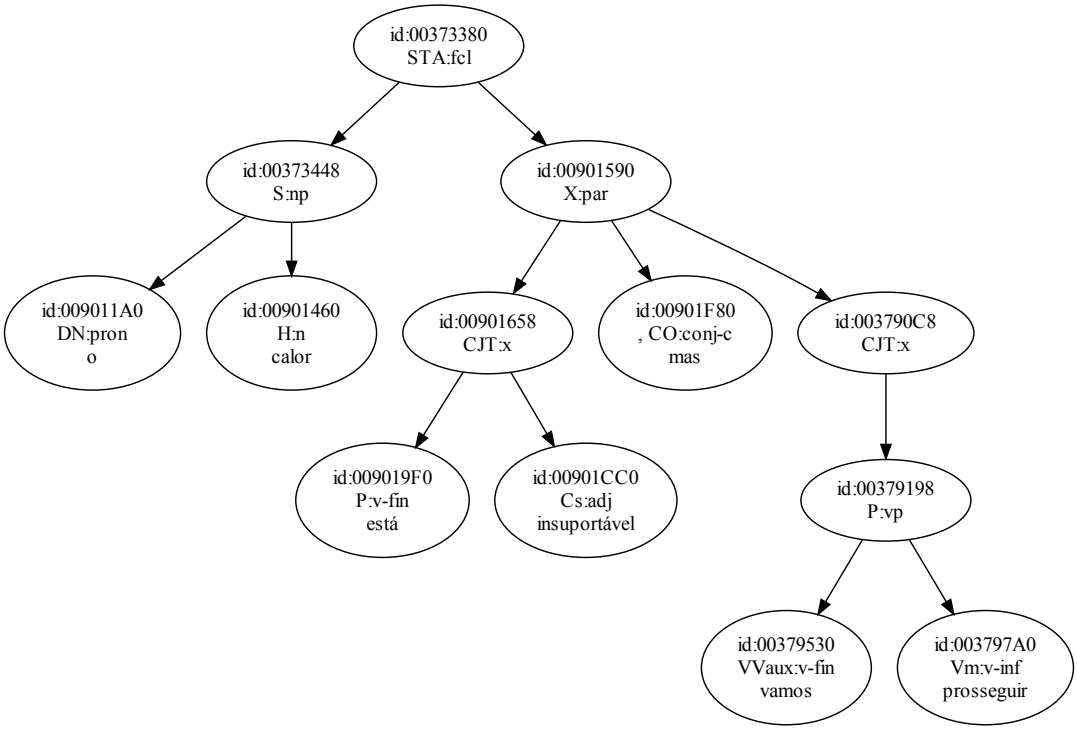

Figura 6.8: Representação inicial de uma árvore sintática.

A seguir é apresentada a etapa de interligação de conteúdo realizada a partir da árvore sintática obtida. São também descritas as transformações efetuadas na estrutura da árvore com o propósito de caracterizar os vínculos existentes entre os diversos elementos gramaticais identificados.

6.2.4 Interligação de Conteúdo

O ponto de partida da etapa de interligação de conteúdo é a árvore deitada obtida na etapa de análise sintática.

Os objetivos da etapa de interligação de conteúdo são:

- identificar instâncias de objetos que podem ser associadas às classes de alto nível definidas na *ontologia base* (vide Seção 6.2.4.2);
- criar, de acordo com a estrutura da árvore sintática, as propriedades de dados e de objetos necessárias para caracterizar os vínculos existentes entre as instâncias de objetos identificadas;
- gerar, em um formato próprio para a etapa subseqüente do processo, um arquivo de saída com toda a hierarquia de elementos identificados e criados.

Para atingir esses objetivos, as seguintes operações são executadas:

- interpretação da árvore deitada obtida na etapa de análise sintática e montagem da estrutura de representação inicial dessa árvore;
- associação dos elementos gramaticais existentes na árvore sintática às classes definidas na *ontologia base*;
- criação de listas hierárquicas de objetos e respectivas propriedades;
- unificação dos elementos gramaticais constituintes dos sintagmas existentes, em torno dos núcleos destes, através da junção dos elementos caracterizadores (adjetivos, advérbios, artigos e referências a outros sintagmas);
- criação das estruturas representativas das relações n-árias existentes nas orações, através da identificação dos elementos de ligação, tais como verbos, preposições, conjunções e pronomes relativos;
- criação das propriedades que explicitam as relações de transitividade presentes nas orações;
- gravação, em uma disposição hierárquica própria para a construção de uma ontologia, de um arquivo com as listas de objetos criados e respectivas propriedades.

Algumas dessas operações ocorrem simultaneamente. A Figura 6.9 ilustra a etapa de interligação de conteúdo em relação às operações realizadas. Durante a interpretação e carga da árvore deitada a partir do arquivo disponibilizado pela etapa de análise sintática, uma representação da árvore é construída em memória. A partir desse momento, são aplicadas, sobre essa árvore, operações de unificação de sintagmas e de associação de elementos a determinadas classes da *ontologia base* (vide Seção 6.2.4.2), sendo que durante essas operações são criadas listas de instâncias e de propriedades identificadas na árvore. Essas transformações resultam em uma árvore modificada, mais compacta, com nós interligados às listas de instâncias e propriedades. A árvore transformada é, então, utilizada na geração de propriedades adicionais que conectam os elementos das orações diretamente ou por transitividade.

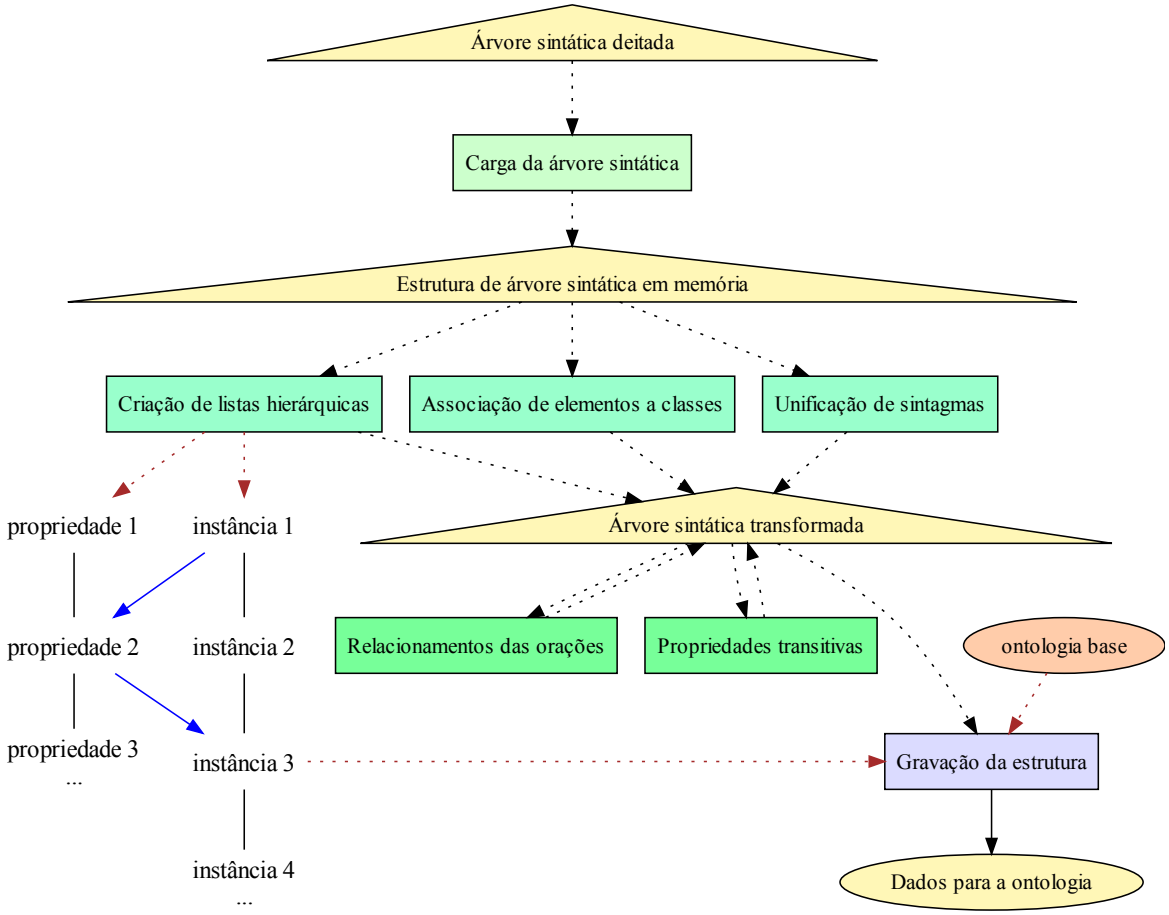

Figura 6.9: Operações realizadas na etapa de interligação de conteúdo.

O componente do protótipo responsável pela interligação de conteúdo, *oglnk*, realiza essas operações recebendo como entrada o arquivo disponibilizado na etapa anterior e gerando como resultado um arquivo que contém todas as informações necessárias para a etapa subseqüente (combinação de ontologias). A Figura 6.10 ilustra o funcionamento do componente *oglnk*.

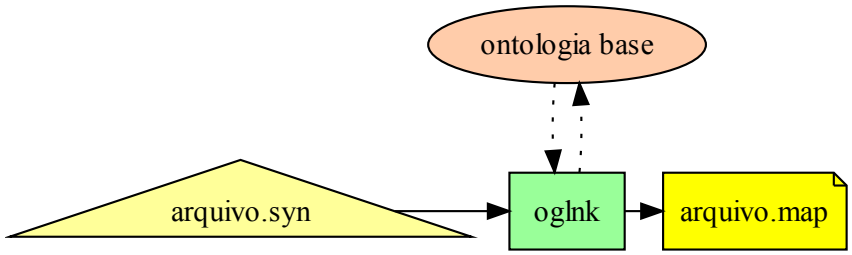

Figura 6.10: Funcionamento do componente *oglnk*.

O arquivo de entrada ("*arquivo.syn*") contém uma árvore sintática deitada. A interpretação e carga dessa árvore é descrita na Seção 6.2.4.1. A estrutura da *ontologia base* e a implementação das operações subseqüentes são descritas em seguida, e o formato do arquivo resultante ("*arquivo.map*") é mostrado na Seção 6.2.4.5.

6.2.4.1 Montagem da Árvore Sintática

A árvore sintática deitada contém, além dos atributos associados aos elementos gramaticais identificados, algumas informações sobre a estrutura do texto, sem as quais não seria possível carregar e reconstruir a árvore sintática em memória.

A Figura 6.11 mostra a árvore deitada para o arquivo gerado a partir do texto de exemplo "*As vítimas do terremoto começaram a saquear os supermercados da região*".

```
1  % SOURCE: Running text
2  % 1. as vítimas do terremoto começaram a saquear os supermercados da região.
3  % A1
4  STA:fcl
5  S:np
6  =DN:pron("o" <artd> DET F <indf> P)      as
7  =H:n("vítima" F P)       vítimas
8  =DN:pp
9  ==H:prp("de" <sam-> <np-close>) de
10 ==DP:np
11 ===DN:pron("o" <artd> <-sam> DET M <indf> S)     o
12 ===H:n("terremoto" M S) terremoto
13 P:vp
14 =VVaux:v-fin("começar" <fmc> PS/MQP 3P IND VFIN)         começaram
15 =PRT-Vaux:Vaux("a" <+top>)       a
16 =Vm:v-inf("saquear" <mv>)        saquear
17 Od:np
18 =DN:pron("o" <artd> DET M <indf> P)      os
19 =H:n("supermercado" M P)         supermercados
20 =DN:pp
21 ==H:prp("de" <sam-> <np-close>) de
22 ==DP:np
23 ===DN:pron("o" <artd> <-sam> DET F <indf> S)     a
24 ===H:n("região" F S)     região
25 .
```

Figura 6.11: Exemplo de árvore sintática deitada.

As linhas que não contém informações sobre as palavras do texto e seus atributos representam sub-árvores. As outras linhas representam os nós da árvore. Além disso, o caractere "=" é utilizado para indicar qual é o nível de profundidade de cada sub-árvore e de suas folhas. Dessa forma é possível identificar a estrutura da árvore.

A tarefa de reconhecimento e reconstrução da árvore em memória foi implementada, no componente *oglnk* do protótipo, com a ajuda de uma gramática livre de contexto, cuja descrição está no Apêndice D. Essa gramática foi desenvolvida com a ajuda das ferramentas *Lex* e *Yacc* (Levine et al. [1992]).

Para facilitar o desenvolvimento da gramática de reconstrução da árvore, as informações sobre o nível de profundidade das sub-árvores e das folhas foi ignorado em um primeiro momento. A árvore inicial construída de acordo com a gramática é, por causa disso, uma árvore degenerada, ou seja, uma árvore semelhante a uma lista encadeada. Para o texto do exemplo mencionado, a árvore inicial obtida é mostrada na Figura 6.12.

As informações sobre o nível de profundidade das sub-árvores e das folhas são utilizadas em um segundo momento, no qual a árvore é balanceada para refletir a sua real estrutura. O resultado dessa reorganização da árvore, para o exemplo em questão, é mostrado na Figura 6.13.

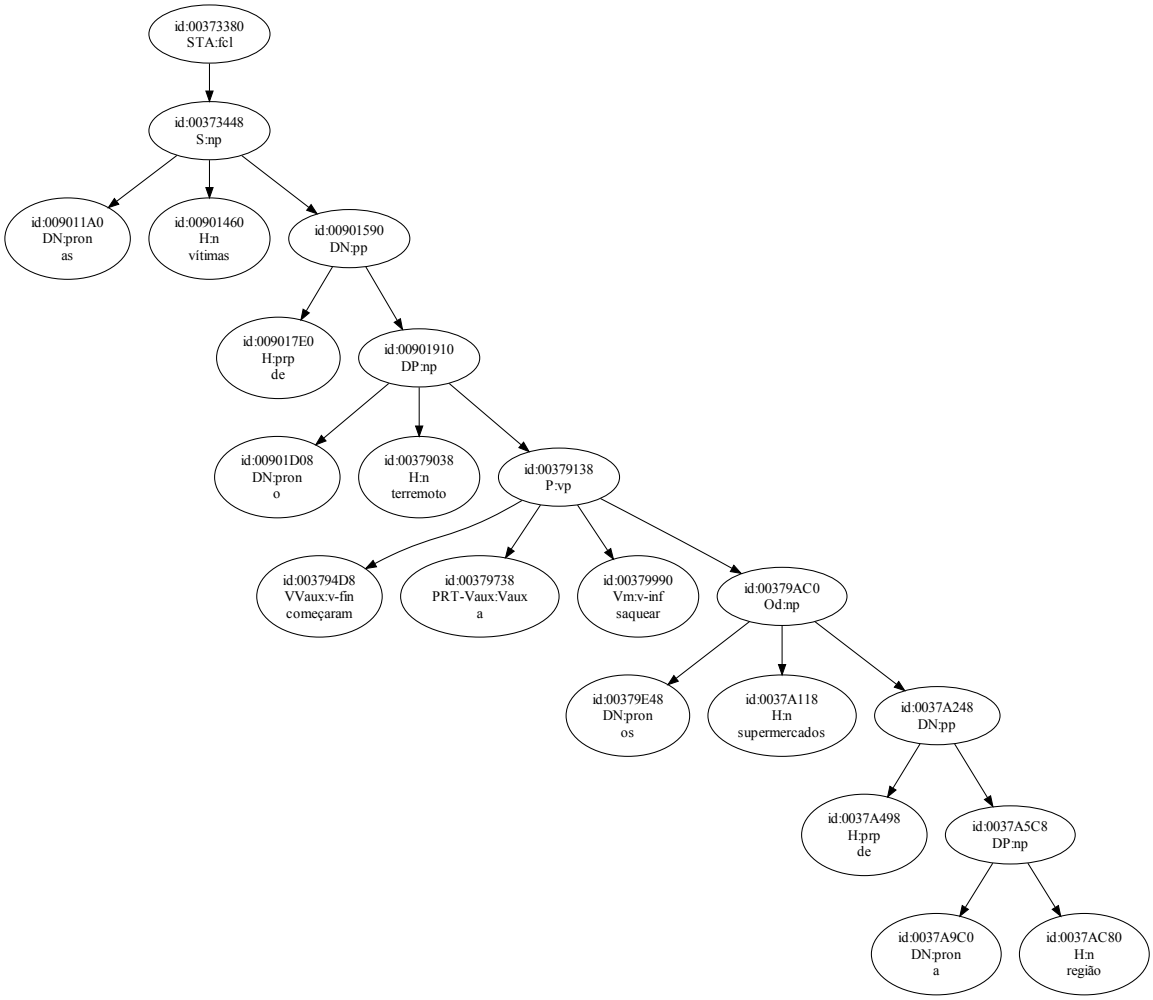

Figura 6.12: Árvore degenerada produzida inicialmente.

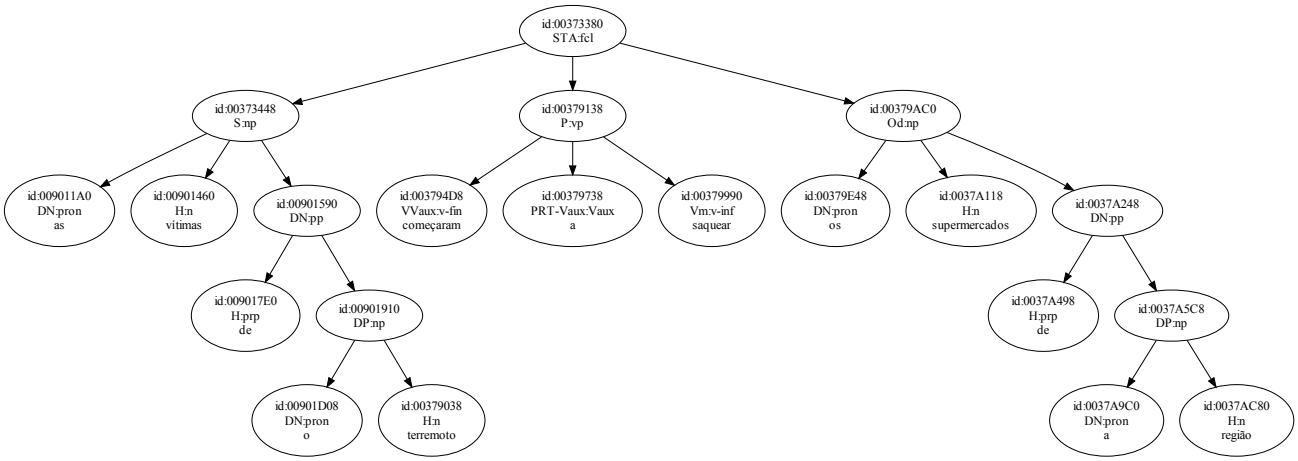

Figura 6.13: Árvore reorganizada para refletir a estrutura real.

A estrutura interna da árvore contém algumas ligações adicionais que possibilitam operações de reorganização e de unificação de sintagmas (vide Seção 6.2.4.3). Essas ligações são mostradas na Figura 6.14. Cada nível da árvore é uma lista duplamente encadeada e cada nó da lista aponta para o nó "pai", que possui ponteiros para o início e para o fim da lista. Essa estrutura permite que os nós sejam unidos ou movimentados para qualquer lugar da árvore.

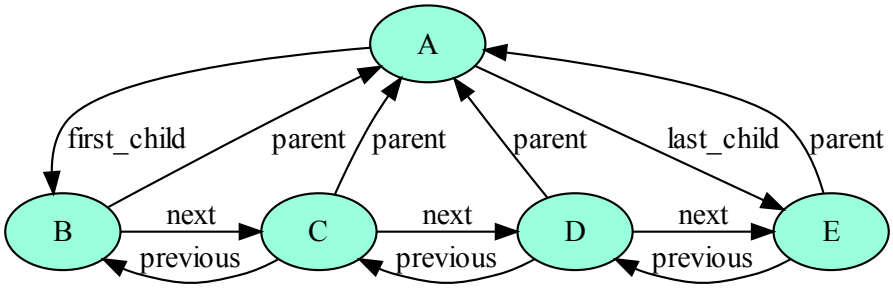

Figura 6.14: Ligações existentes em uma árvore.

A montagem da árvore sintática em memória é o primeiro passo da etapa de interligação de conteúdo. Antes de contemplarmos as demais operações efetuadas sobre a árvore sintática, veremos a seguir a definição da *ontologia base*.

6.2.4.2 Ontologia Base

A *ontologia base*, mostrada na Figura 6.15, foi definida ao longo deste trabalho com o propósito de mapear os elementos gramaticais considerados mais importantes para a representação da estruturas dos textos, através da organização desses elementos em uma hierarquia de classes gramaticais relevantes para o processo.

Os principais critérios estabelecidos para a criação da *ontologia base* foram os seguintes:

- seleção das classes gramaticais que categorizam e diferenciam as palavras, sem o quê não seria possível distigüir, por exemplo, o adjetivo "*bastante*" (como o usado em "*a comida não é bastante*") do advérbio "*bastante*" (como o usado em "*eles brigam bastante*");
- necessidade de seleção dos elementos de ligação, tais como verbos, preposições e conjunções como formas computacionalmente adequadas para a reconstrução e a representação de vínculos existentes nos textos;
- caracterização dos tempos e modos verbais, para acomodar as diversas conjugações verbais como vínculos distintos;
- caracterização de flexões de gênero, para diferenciar, por exemplo, "*o chefe*" de "*a chefe*";
- caracterização de flexões de número (plural e singular).

Esses critérios orientaram a definição da hierarquia de classes mostrada na Figura 6.15, definida conforme Bechara [2009]. A versão mais atual da *ontologia base* está disponível em Bravo [2010a]. O Apêndice C contém a versão original.

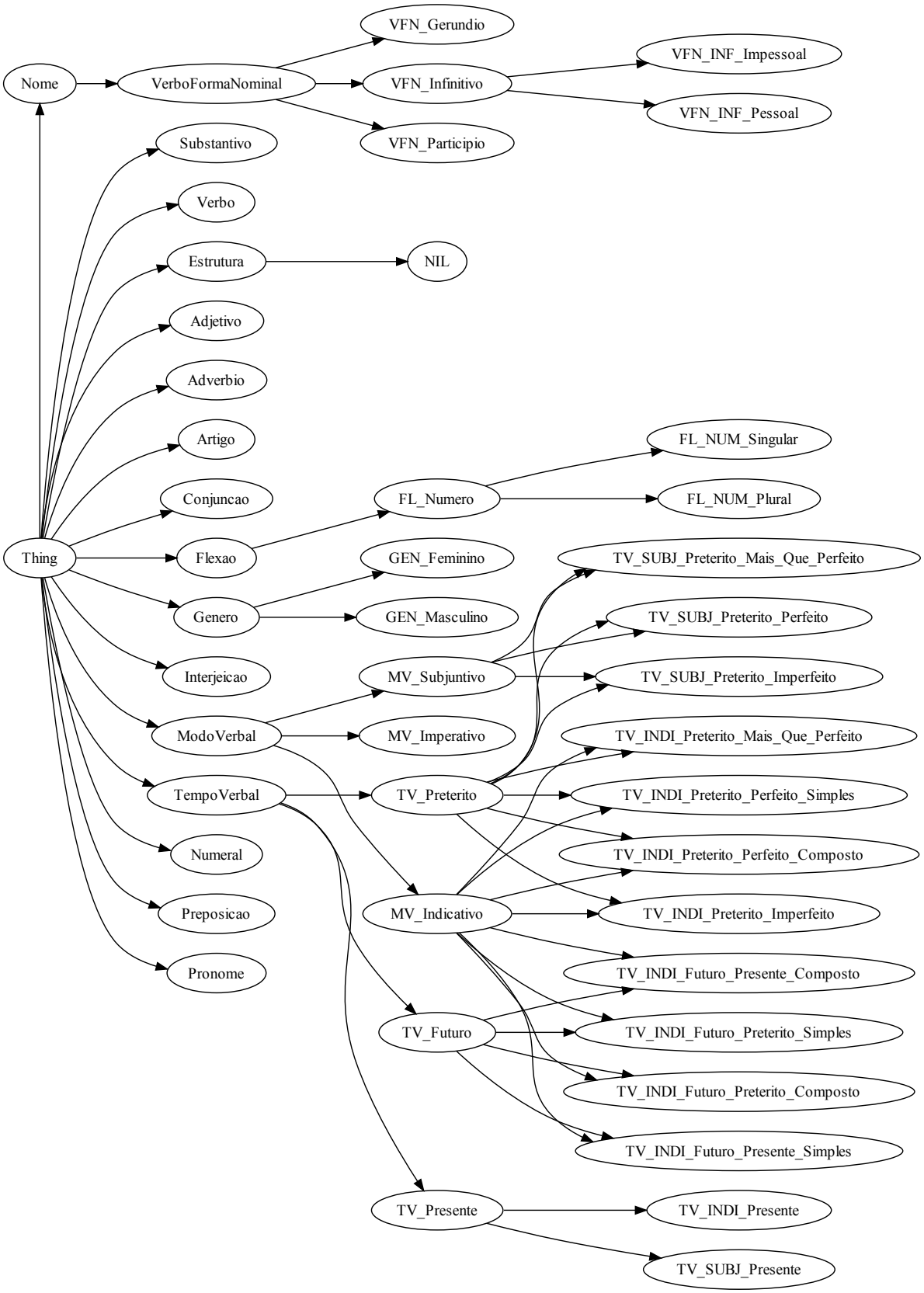

Figura 6.15: *Ontologia base* para a geração das ontologias da língua portuguesa.

A *ontologia base* é utilizada por todas as ontologias produzidas pelo processo de geração. Todas as instâncias e propriedades específicas dessas ontologias são associadas às classes definidas na *ontologia base*, através das operações de interligação de conteúdo, as quais estão descritas nas próximas seções.

6.2.4.3 Unificação de Sintagmas

Após a montagem da árvore sintática em memória (vide Seção 6.2.4.1), a etapa de interligação de conteúdo prossegue com a operação de unificação de sintagmas. O objetivo dessa operação é agregar todos os elementos constituintes dos sintagmas em torno dos seus núcleos, visando a formação de instâncias de objetos caracterizados por seus atributos. A operação fica mais clara sob a luz de um exemplo, tal como o mostrado na Figura 6.16, a qual contém a árvore sintática da frase "*O homem mau matou o homem bom.*".

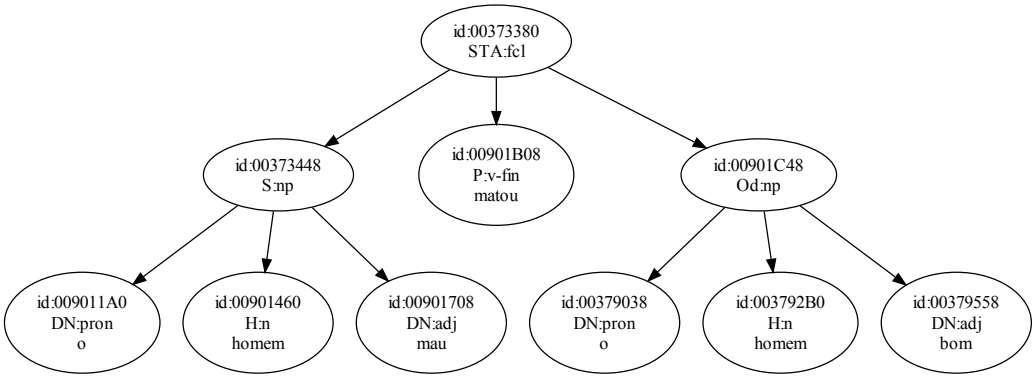

Figura 6.16: Árvore sintática inicial.

Cada sintagma pode ser representado por um conjunto de sub-árvores ou folhas, uma sub-árvore ou apenas uma folha. Com a operação de unificação, folhas vão sendo agrupadas até que reste apenas uma folha, ou até que nenhum agrupamento possa mais ser feito. Para o exemplo considerado, nós obtemos, após o agrupamento, a árvore mostrada na Figura 6.17. A árvore obtida não é exatamente essa porque cada elemento agrupado mantém, como parte do processo de geração, um vínculo com o elemento original, o qual não é descartado para ser incluído na ontologia gerada.

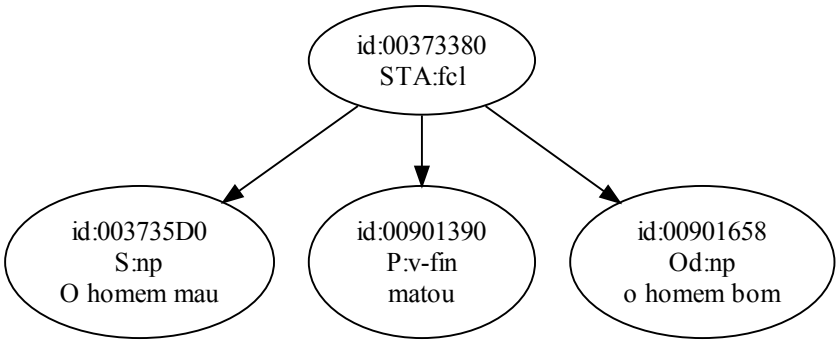

Figura 6.17: Árvore sintática com sintagmas unificados.

Esse exemplo ilustra a identificação de dois indivíduos distintos caracterizados pelos adjetivos a eles agrupados: "*o homem mau*" e "*o homem bom*". Na ontologia resultante, os elementos agrupados formam as instâncias mais específicas e os elementos constituintes são incluídos como instâncias agregadas por uma propriedade que representa esse relacionamento (vide Seção 6.2.4.4).

O componente *oglnk* efetua a operação de unificação de sintagmas levando em conta a proximidade de elementos dentro da mesma oração que não estão separados por elementos de ligação tais como verbos, preposições, pronomes relativos e conjunções.

A seguinte ordem de precedência, definida de acordo com as funções de modificação que advérbios, adjetivos e artigos desempenham, é considerada na operação:

- em primeiro lugar são anexados os advérbios, que atuam como modificadores de outros advérbios, de adjetivos, de verbos e, em alguns casos, de substantivos;
- em segundo lugar são anexados os adjetivos, que atuam como modificadores primários de substantivos;
- posteriormente são anexados os artigos, que funcionam como modificadores de outros elementos em gênero e número.

Os agrupamentos de advérbios são executados sobre a árvore sintática repetidamente até que nenhuma substituição seja mais possível. Cada folha agrupada por um advérbio preserva a classe gramatical original. Assim, adjetivos e verbos modificados por advérbios continuarão tendo funções de adjetivos e verbos, depois de anexados os advérbios.

Os agrupamentos de adjetivos vêm logo depois, executados uma vez só na árvore inteira. Cada nó agrupado por um adjetivo mantém a classe gramatical original.

Por fim são agrupados os artigos, os quais têm a função de caracterizar um substantivo em gênero e número, podendo modificar, portanto, a classe gramatical do elemento agrupado, como ocorre por exemplo em "*o entardecer*", onde o verbo, quando anexado ao artigo, passa a ter a função de substantivo.

A frase de exemplo "*O velho avião caiu e afundou muito rapidamente no mar profundo*", cuja árvore sintática inicial é apresentada na Figura 6.18, contém advérbios, adjetivos e artigos que, depois do processo de unificação, ficam agrupados aos elementos aos quais estão associados.

Nesse exemplo, inicialmente o sintagma adverbial "*muito rapidamente*" é agrupado, e a sub-árvore que o contém é eliminada. Em seguida, esse sintagma é agrupado ao verbo "*afundou*". Depois os adjetivos "*velho*" e "*profundo*" são agrupados, respectivamente, aos substantivos "*avião*" e "*mar*". Por fim são agrupados os artigos. A árvore obtida é mostrada na Figura 6.19.

Após a operação de unificação de sintagmas, resta uma árvore sintática transformada. É com base nessa árvore que o componente *oglnk* inicia a operação de criação, em memória, dos elementos representativos de instâncias e de propriedades que farão parte do resultado final. Essa operação é descrita na próxima seção.

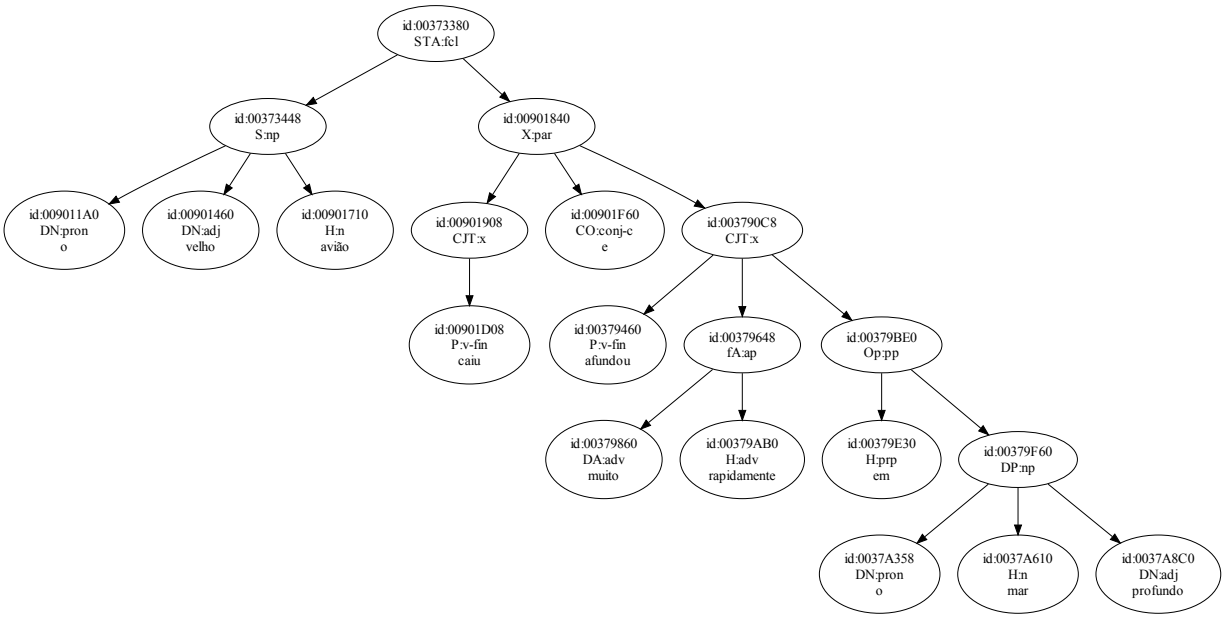

Figura 6.18: Árvore sintática inicial (exemplo 2).

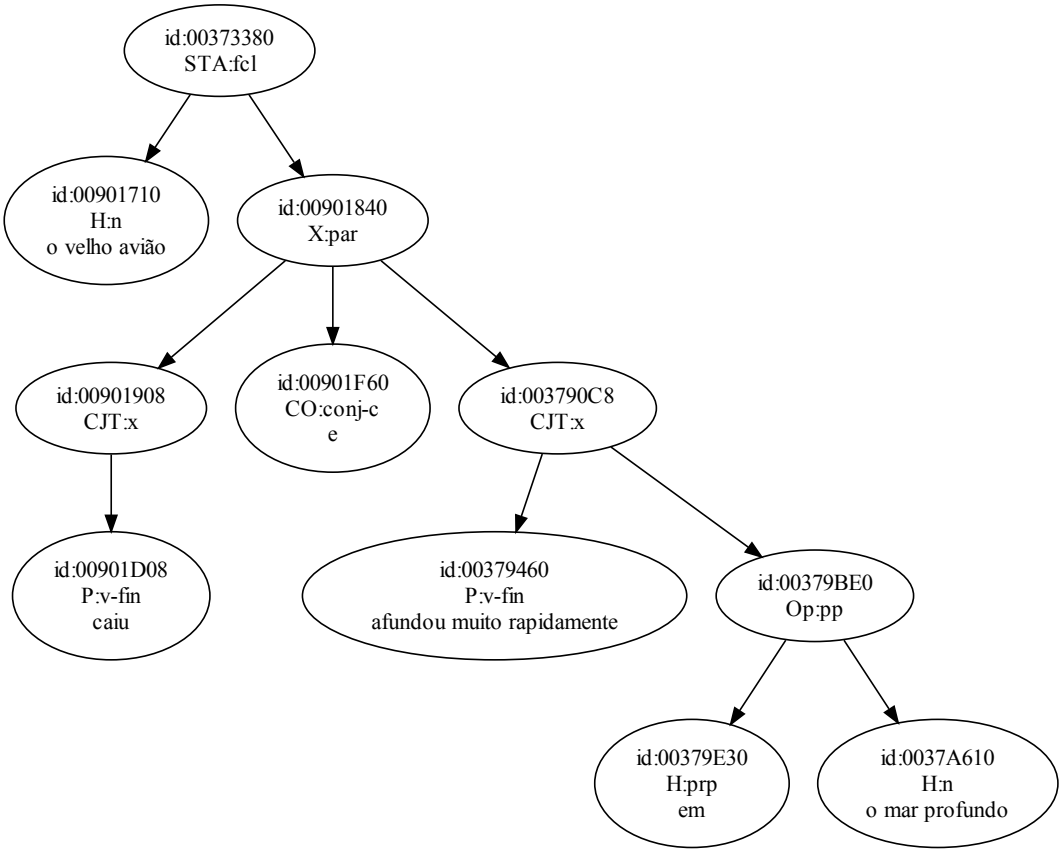

Figura 6.19: Árvore após unificação de sintagmas (exemplo 2).

6.2.4.4 Criação de Instâncias e Propriedades

As propriedades de objeto são geradas a partir dos elementos de ligação, tais como verbos, preposições, conjunções e pronomes. As seguintes regras de formação são adotadas:

- para os verbos, os nomes das propriedades incluem o prefixo "V_", o tempo verbal, o modo verbal, o verbo no infinitivo, o verbo conjugado e os advérbios, se existirem;
- para os demais elementos de ligação, o nome inclui um prefixo correspondente à classe gramatical do elemento: "PREP_", "PRON_", "CONJ_";
- demais referências entre elementos são prefixadas por "REF_", o prefixo utilizado para conectar verbos a nomes e adjetivos ou advérbios a outros elementos, dentro das orações.

Os artigos são transformados em propriedades de dados, e recebem o prefixo "ART_".

Propriedades de objeto formam hierarquias compostas por suas partes constituintes, permitindo, dessa forma, que uma ligação verbal entre instâncias possa ser caracterizada pelo verbo conjugado, pelo verbo no infinitivo, ou simplesmente pelo fato da ligação ser verbal.

Os verbos também aparecem como instâncias, formando os núcleos das orações, sendo que as propriedades de objeto originadas a partir deles são utilizadas na conexão das partes integrantes da oração, em uma relação n-ária (W3C [2006]). A Figura 6.20 mostra a estrutura de uma oração e suas partes constituintes para a frase de exemplo "*Maria deixou José na garagem*".

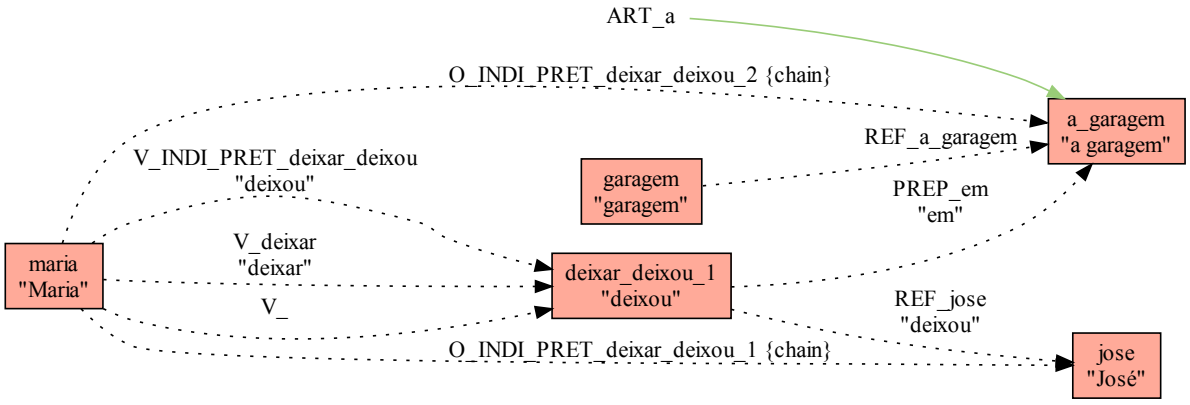

Figura 6.20: Representação ontológica de uma oração.

As orações são caracterizadas por relações transitivas implementadas na forma de cadeias de propriedades (*property chains*).

Todos os elementos identificados são associados às suas classes gramaticais, de acordo com a *ontologia base*. Números seqüenciais são associados às instâncias (exceto a primeira de cada repetição), fazendo parte de seus nomes. O texto original de cada palavra é preservado para posterior inclusão, em forma de anotação (*label*), na ontologia final.

A seguir é apresentado o formato do arquivo resultante da operação de criação de instâncias e propriedades.

6.2.4.5 Preparação do Resultado

Uma vez criados todos os elementos constituintes da ontologia a ser gerada, um arquivo é disponibilizado para a próxima etapa (combinação e geração de ontologias).

A Figura 6.21 mostra um exemplo de arquivo de saída para a oracão da Figura 6.20, assumindo que o texto original contém apenas a frase do exemplo utilizado.

```
1  %
2  % Maria deixou José na garagem.
3  %
4  @instance@
5  maria I "Maria" Nome 003735D0 S:prop
6  jose I "José" Nome 00901658 Od:prop
7  garagem I "garagem" Nome 003791E8 H:n
8  deixar_deixou_1 V "deixou" Verbo,MV_Indicativo,TV_INDI_Preterito_Perfeito_Simples
   00901390 P:v-fin
9  a_garagem J "a garagem" Nome ART_a garagem
10 %
11 @objprop@
12 V_
13 V_deixar V_
14 V_INDI_PRET_deixar_deixou V_deixar
15 REF_a_garagem
16 REF_jose
17 PREP_em
18 %
19 @dataprop@
20 ART_a A "a"
21 %
22 @chain@
23 O_INDI_PRET_deixar_deixou_1 V_INDI_PRET_deixar_deixou REF_jose
24 O_INDI_PRET_deixar_deixou_2 V_INDI_PRET_deixar_deixou PREP_em
25 %
26 @link@
27 maria V_ deixar_deixou_1
28 maria V_deixar deixar_deixou_1
29 maria V_INDI_PRET_deixar_deixou deixar_deixou_1
30 garagem REF_a_garagem a_garagem
31 deixar_deixou_1 PREP_em a_garagem
32 deixar_deixou_1 REF_jose jose
```

Figura 6.21: Exemplo de arquivo "*.map*" contendo objetos, propriedades e ligações.

O formato do arquivo foi idealizado para facilitar a carga do arquivo na etapa subseqüente do processo. Os seguintes grupos de dados podem ser gravados no arquivo:

- "*@instance@*" - lista de indivíduos e seus atributos;
- "*@objprop@*" - lista de propriedades de objeto;
- "*@dataprop@*" - lista de propriedades de dados;
- "*@chain@*" - lista de cadeias de propriedades (*property chains*);
- "*@link@*" - lista de relacionamentos.

Os grupos devem aparecer na ordem especificada: "*@instance@*", "*@objprop@*", "*@dataprop@*", "*@chain@*" e "*@link@*". Apenas o grupo "*@instance@*" é obrigatório. Os grupos "*@objprop@*" e "*@dataprop@*" podem se alternar e qualquer quantidade de grupos é possível, desde que a ordem especificada seja respeitada.

6.2.5 Combinação e Geração de Ontologias

A etapa de combinação de ontologias é a última a ser executada pelo processo de geração. Nessa etapa são efetuadas as seguintes operações:

- transformação do conteúdo do arquivo obtido na etapa anterior em uma estrutura representativa da ontologia a ser gerada;
- leitura do conteúdo das ontologias a serem incluídas no resultado final;
- gravação de um documento no formato RDF/XML ou OWL/XML contendo a ontologia resultante.

A Figura 6.22 mostra o funcionamento do componente *ogcom* do protótipo.

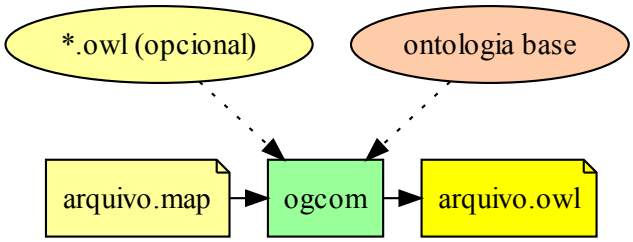

Figura 6.22: Funcionamento do componente *ogcom*.

O resultado da etapa de geração é um arquivo RDF/XML ou OWL/XML, sendo uma dessas sintaxes escolhidas como parâmetro de execução do protótipo. A sintaxe das ontologias incluídas no resultado final deve ser idêntica à escolhida para o formato do resultado final.

A etapa de geração funciona de acordo com uma gramática livre de contexto, a qual foi desenvolvida com a ajuda das ferramentas *Lex* e *Yacc* (Levine et al. [1992]). A descrição dessa gramática é apresentada no Apêndice E.

A combinação das ontologias é semelhante à operação de junção (*merging*) mencionada na Seção 3.2, exceto pela fato de que aqui as classes definidas em todas as ontologias passam a fazer parte do espaço de nomes da ontologia resultante. Isso ocorre para permitir que elementos previamente identificados possam ser devidamente caracterizados. A combinação das ontologias ocorre da seguinte forma:

- os cabeçalhos dos arquivos de origem são analisados e unificados através da extração de informações tais como documentos importados pelas ontologias e definições de localização (URI) de recursos, bem como definições de espaços de nomes utilizados;
- os conteúdos que não fazem parte dos cabeçalhos são copiados e as as referências aos espaços de nomes originais são substituídas por referências ao espaço de nomes da ontologia resultante.

Essa operação pode ficar mais clara à luz de um exemplo. Considerando a estrutura sintática obtida na etapa anterior, temos a configuração inicial mostrada na Figura 6.23, a qual difere da Figura 6.20 (vide Seção 6.2.4.5) apenas pelo fato de que aqui mostramos também as classes da *ontologia base* associadas às instâncias identificadas. Esse é o resultado obtido quando nenhuma ontologia adicional é apresentada ao processo de geração.

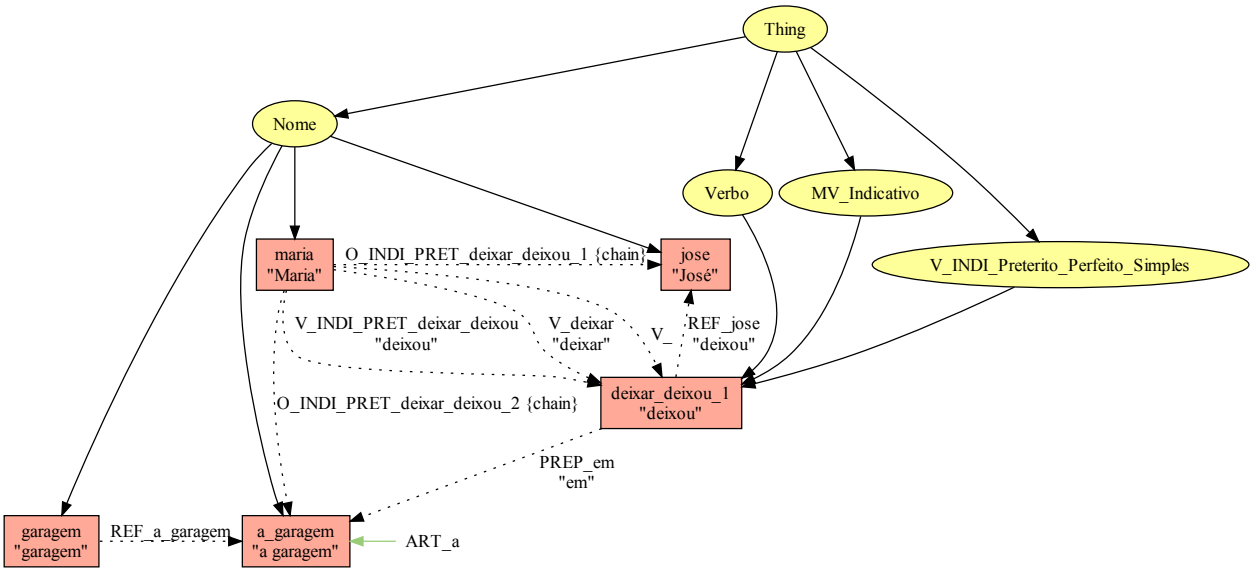

Figura 6.23: Representação de uma oração com a *ontologia base*.

Embora essa representação já forneça informações gramaticais úteis para um processo de raciocínio automatizado subseqüente, precisamos caracterizar apropriadamente os elementos identificados no texto para que ocorra uma integração dessas informações com o conhecimento representado no domínio, esteja ele presente em uma base de conhecimento ou não.

O processo de geração baseia-se exclusivamente na análise sintática do texto de origem, sem conduzir o resultado a nenhuma interpretação semântica específica. Não obstante, a combinação de ontologias no resultado final permite evidenciar uma determinada interpretação ou contexto. Conjuntos distintos de ontologias podem ser escolhidos para um determinado texto a cada processamento, permitindo, de forma flexível, que uma ou mais interpretações sejam associadas ao mesmo texto de origem.

Aproveitando o exemplo da Figura 6.23, podemos desejar, com base em um conhecimento prévio, descobrir que a frase "*Maria deixou José na garagem*" refere-se a uma mulher ("*Maria*"), que deixou seu filho ("*José*") em um local estranho ("*a garagem*"). Não poderíamos chegar a essa conclusão sem informações adicionais sobre "*Maria*", "*José*", ou "*a garagem*". Precisamos oferecer informações para que um raciocinador automatizado possa descobrir, por exemplo, que "*Maria*" é mãe de "*José*", uma vez que "*Maria*" é uma mulher e existe um relacionamento indicando que "*José*" é seu filho (vide Seção 4.3).

Além da classificação das entidades presentes no texto, informações sobre outras entidades podem ser adicionadas ao processo de geração. Nesse mesmo exemplo, poderíamos acrescentar informações conhecidas sobre o pai de "*José*". Essas informações podem ser oriundas de uma base de conhecimento ou de outros textos previamente processados. Continuando com esse mesmo exemplo, podemos acrescentar uma ontologia que inclui dados sobre o pai de "*José*", um funcionário da empresa "*Petrobrás S.A.*" chamado "*João*".

A Figura 6.24 mostra três ontologias criadas com o editor de ontologias *Protégé 4.0* (vide Seção 3.5) para representar nosso conhecimento a respeito dos elementos identificados nesse exemplo. Essas ontologias, para serem incluídas no resultado final, devem ser especificadas como parâmetros de execução para o processo de geração. O código-

fonte dessas ontologias, expresso nas sintaxes RDF/XML e OWL/XML, é apresentado no Apêndice F.

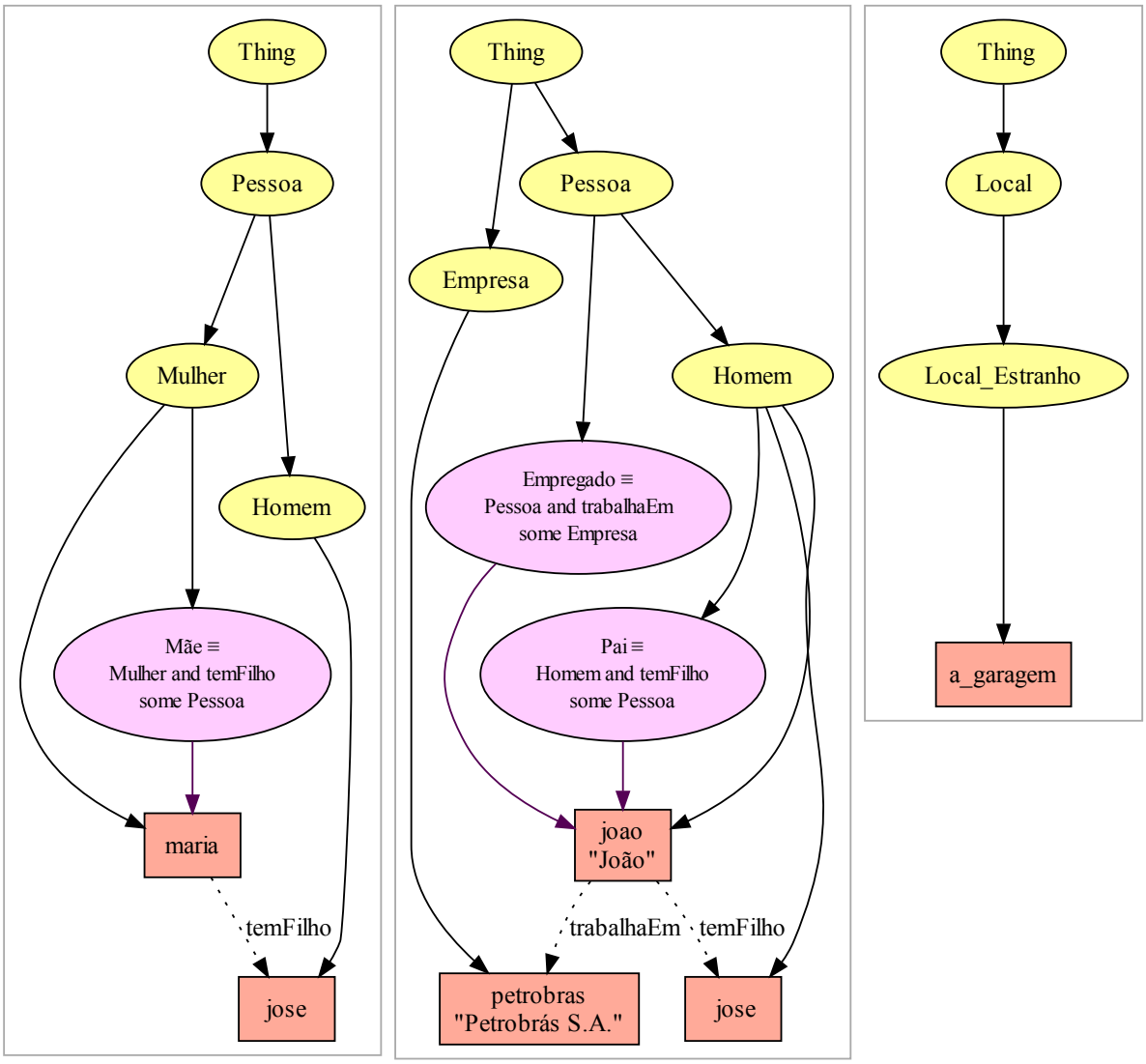

Figura 6.24: Visão esquemática das ontologias adicionadas ao processo.

Após a leitura do arquivo disponibilizado na etapa anterior e combinação dos elementos identificados no arquivo com as ontologias pré-existentes, a ontologia final é gerada. Para o exemplo aqui descrito, a ontologia resultante é a mostrada na Figura 6.25. O código-fonte dessa ontologia é apresentado no Apêndice G, nas versões RDF/XML e OWL/XML.

No exemplo em questão, as classificações de "*Maria*" e de "*João*" como pais de "*José*" são obtidas por inferência. O mesmo ocorre com a classificação de "*João*" como empregado da empresa "*Petrobrás S.A.*".

Podemos observar, com esse exemplo, que é possível adicionar ao processo de geração um contexto específico que inclua novas classes, novas instâncias e novos relacionamentos, além permitir a junção de instâncias presentes em espaços de nomes distintos.

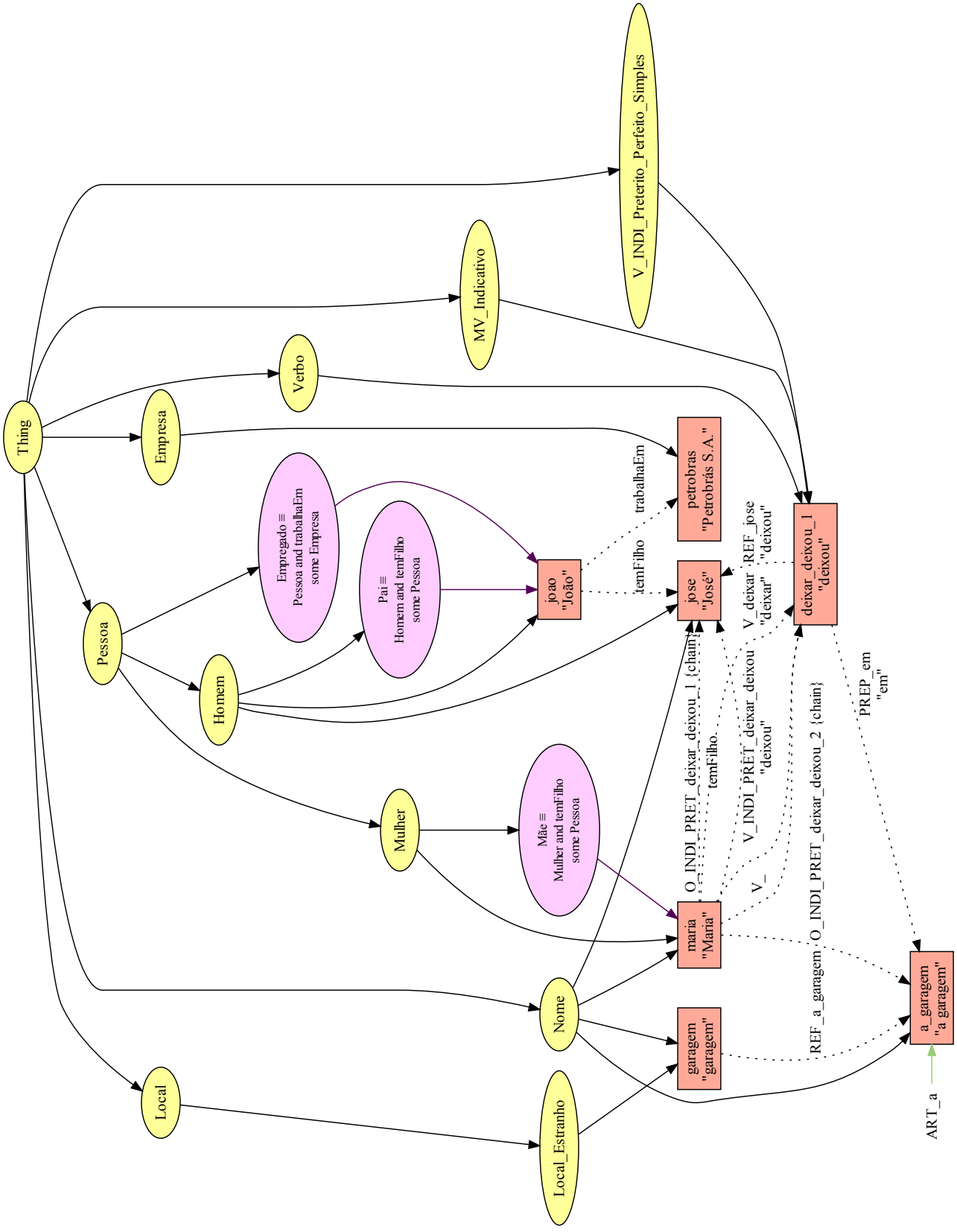

Figura 6.25: Ontologia resultante do processo de geração automática.

Em aplicações reais, as taxonomias pré-existentes podem apenas referenciar ontologias em comum, diminuindo a necessidade de redundância na definição das classes, tal como a encontrada nas ontologias apresentadas na Figura 6.24.

A combinação de ontologias efetuada pelo processo de geração oferece um meio de representar explicitamente uma contextualização da informação contida no texto original. Pode-se estender, através do processo, o conteúdo original com informações semânticas e, ao mesmo tempo, obter uma representação estrutural do texto que pode ser explorada concomitantemente.

Na medida em que novas ontologias são combinadas com ontologias pré-existentes, o processo de geração pode ser realimentado continuamente e resultados mais abrangentes e concretos podem ser obtidos. Esse aspecto do processo pode ser explorado durante a construção de uma base de conhecimento para o domínio considerado.

Neste capítulo foram apresentadas todas as etapas do processo de geração automática de ontologias implementadas neste trabalho. No próximo capítulo serão apresentadas as conclusões e sugestões para trabalhos futuros relacionados ao processo apresentado.

Capítulo 7

Conclusões e Trabalhos Futuros

Este trabalho descreveu um processo de geração automática de ontologias a partir de páginas HTML escritas em língua portuguesa.

O processo aqui descrito preserva, nas ontologias que produz, a estrutura sintática dos textos originais, através da representação dos elementos considerados estruturais, ou elementos de ligação, caracterizados ao longo deste trabalho. Os artefatos produzidos, úteis para a identificação de conceitos e respectivos relacionamentos, são, atualmente, coleções de orações identificadas nos textos de origem. Estes artefatos podem ser associados ao conhecimento presente em um domínio de aplicação pela combinação desse conhecimento com as informações textuais colhidas e estruturadas ao longo do processo.

A etapa final do processo de geração de ontologias implementa a combinação das informações obtidas a partir da estrutura dos textos de entrada com ontologias pré-existentes. A natureza das ontologias geradas está, portanto, sempre relacionada às ontologias incluídas na etapa final de geração. A combinação de ontologias na produção do resultado final favorece a interoperabilidade semântica e o reuso dessas ontologias. Acreditamos também que esse recurso possa ser útil na construção de bases de conhecimento.

No que se refere à etapa de combinação e geração de ontologias, esperamos, no futuro, continuar este trabalho com a análise das seguintes possibilidades de expansão:

- implementação de um repositório de ontologias que possa ser atualizado através da Web, com o objetivo de facilitar a construção de bases de conhecimento;
- estudo de formas adicionais de operações aplicáveis ao processo de geração, para ampliar desse modo as possibilidades de combinação de ontologias;
- definição de um método de mapeamento semântico baseado em regras, para viabilizar a implementação de mecanismos de classificação automática dos elementos identificados nos textos.

Uma *ontologia base* da língua portuguesa (vide Apéndice C) foi definida, ao longo do trabalho, de acordo com as classes gramaticais consideradas relevantes para a representação ontológica dos textos originais. Embora essa ontologia tenha sido suficiente para o processo de geração aqui descrito, sua abrangência deverá ser ampliada de acordo com as necessidades das aplicações que esperamos desenvolver a partir deste trabalho.

Dentre as diversas aplicações práticas podem ser derivadas deste trabalho, destacamos as que podem adotar o processo descrito em domínios que lidam com informações concretas, tais como:

- informações jornalísticas provenientes, por exemplo, de portais de notícias, pelas quais seja possível estabelecer vínculos entre entidades e acontecimentos;
- jurisprudências e outras informações do Poder Judiciário que normalmente necessitem de classificação automática de entidades e de partes envolvidas em processos judiciais;
- leis e outras informações legislativas a partir das quais seja necessária a obtenção automática de relações entre textos não-estruturados distintos.

O protótipo desenvolvido neste trabalho transformou-se em uma ferramenta útil de investigação do processo de geração, dada a sua característica de produzir resultados intermediários de fácil interpretação. A operação de unificação de sintagmas, executada durante a etapa de interligação de conteúdo, pode ser útil na tarefa de identificação de elementos conceituais. Os vínculos de relacionamento criados ao longo do processo refletem satisfatoriamente a estrutura interna das orações. A criação de vínculos entre orações distintas, entretanto, por requerer tratamento semântico detalhado, será objeto de estudo adicional, como extensão deste trabalho.

Cada uma das etapas do processo de geração de ontologias pode evoluir a partir do trabalho realizado até agora, sendo que em um futuro próximo há intenção de ampliar o protótipo implementado com as seguintes ações:

- iniciar o desenvolvimento de um analisador sintático leve, próprio para o processo descrito, com mais ênfase na estrutura do texto do que na morfologia das palavras, a qual poderia ser explorada em um passo seguinte, já sobre as ontologias geradas;
- implementar recursos que permitam a interligação de conteúdo entre ontologias geradas e, por conseguinte, entre recursos da Web;
- desenvolver uma interface gráfica que permita a execução interativa do protótipo e a visualização esquemática das ontologias geradas;
- integrar o protótipo a editores de ontologias, raciocinadores e ferramentas de manutenção de sítios da Web, através do desenvolvimento de interfaces apropriadas;
- implementar o suporte à leitura e geração de ontologias expressas em sintaxes diversas das sintaxes RDF/XML e OWL/XML, ampliando dessa forma o potencial de integração do protótipo com outras ferramentas e permitindo que especialistas familiarizados com outras sintaxes possam colaborar com o crescimento de bases de conhecimento alimentadas pelo processo de geração;
- desenvolver os mecanismos necessários para a interpretação e avaliação de um conjunto de regras de interligação de conteúdo que permita o aumento da granularidade de itens frasais para a criação de conexões entre sentenças, parágrafos e textos.

Por fim, acreditamos que o processo de geração de ontologias elaborado ao longo deste trabalho traçou um caminho, dentre os vários possíveis, sobre os diversos aspectos envolvidos no problema de classificação automática de conteúdo textual. Chegamos ao fim de uma etapa importante, mas esperamos continuar este trabalho, aprimorando o que foi feito até aqui e buscando novos horizontes nesse caminho de busca por uma Web Semântica plenamente operacional.

Referências

H. Alani, S. Kim, D. E. Millard, M. J. W. W. Hall, M. J. Weal, W. Hall, P. H. Lewis, and N. R. Shadbolt. Automatic ontology-based knowledge extraction and tailored biography generation from the web. *IEEE Intelligent Systems*, 18:14–21, 2002. xi, 5

H. P. Alesso and C. F. Smith. *Thinking on the Web: Berners-Lee, Gödel and Turing*. Wiley-Interscience, 2006. ISBN 0471768146. 9, 38

V. U. Amsterdam. http://www.ontoknowledge.org/, 2000. On-To-Knowledge: Content-driven Knowledge-Management through Evolving Ontologies. Acesso em: 15 set. 2009. 21

F. Baader and W. Nutt. Basic description logics. pages 43–95, 2003. xi, xiii, 32, 34, 35, 85, 86, 89

J. Barwise and J. Etchemendy. *Language Proof and Logic*. CSLI, 1999. 30

BBC. http://www.bbc.co.uk/music/, 2008. BBC Music Project. Acesso em: 13 ago. 2009. 15

E. Bechara. *Moderna Gramática Portuguesa*. Editora Nova Fronteira, Brasil, 2009. 51, 62

T. Berners-Lee. http://talis-podcasts.s3.amazonaws.com/twt20080207_TimBL.html, 2008. Sir Tim Berners-Lee Talks with Talis about the Semantic Web. Acesso em: 12 ago. 2009. 8

T. Berners-Lee. http://www.w3.org/History/1989/proposal.html, 1989. Information Management: A Proposal. Acesso em: 13 ago. 2009. xi, 9, 10

T. Berners-Lee. Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web. RFC 1630 (Informational), June 1994. URL http://www.ietf.org/rfc/rfc1630.txt. 89

T. Berners-Lee and M. Fischetti. *Weaving the Web : The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper San Francisco, September 1999. ISBN 0062515861. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0062515861. 9

T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284 (5):34–43, May 2001. 1, 8

E. Bick. *The Parsing System PALAVRAS: Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*. PhD thesis, Arhus University, Denmark, 2000. Disponível em http://beta.visl.sdu.dk/visl/pt/parsing/automatic/. 45

A. Borgida and R. J. Brachman. Conceptual modeling with description logics. pages 349–372, 2003. 36

P. Borst, H. Akkermans, and J. Top. Engineering ontologies. *Int. J. Hum.-Comput. Stud.*, 46(2-3):365–406, 1997. ISSN 1071-5819. doi: http://dx.doi.org/10.1006/ijhc.1996.0096. 21

J. Boye. http://www.irt.org/articles/js086/index.htm, 1998a. RDF - What's in it for us?. Acesso em: 13 ago. 2009. 11

J. Boye. http://www.powerset.com/, 2005. Powerset. Acesso em: 13 ago. 2009. 15

J. Boye. http://www.irt.org/articles/js086/#2, 1998b. A short history of RDF. Acesso em: 10 out. 2009. 11

R. J. Brachman. On the epistemological status of semantic networks. In N. Findler, editor, *Associative Networks: Representation and use of knowledge by computers*. Academic Press, New York, 1979. 32

C. Bravo. http://ontoge.org/ontology/ontoge/por/base.owl, 2010a. Base para a geração de ontologias da língua portuguesa. Acesso em: 5 mar. 2010. 51, 62, 102

C. Bravo. http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl, 2010b. Base para a geração de ontologias da língua portuguesa. Acesso em: 5 mar. 2010. 102

C. Bravo. http://ontoge.sourceforge.net/, 2010c. Ontoge - Protótipo de Geração Automática de Ontologias. Acesso em: 5 mar. 2010. 52

K. K. Breitman. *Web semântica: a internet do futuro*, volume 1. LTC, Rio de Janeiro, 2005. ISBN 85-216-1466-7. 8, 13, 21

R. Cailliau and D. Connolly. http://www.w3.org/History.html, 2000. A Little History of the World Wide Web. Acesso em: 14 out. 2009. 89

H. Chalupsky. Ontomorph: A translation system for symbolic knowledge. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR-2000)*, pages 471–482, Breckenridge (USA), 2000. Morgan Kaufmann Publishers. 19

N. Chomsky. *Syntactic Structures*. Mouton and Co, The Hague, 1957. 42

CINTIL. http://cintil.ul.pt/pt/, 2004. *Corpora* CINTIL - *Corpus Internacional do Português* - Universidade de Lisboa. Acesso em: 31 jul. 2008. 46

D. Connolly, F. van Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. http://www.w3.org/TR/daml+oil-reference, 2001. DAML+OIL (March 2001) Reference Description. Acesso em: 12 out. 2009. 23

B. C. D. da Silva, A. D. Felippo, and R. Hasegawa. Methods and tools for encoding the wordnet.br sentences, concept glosses, and conceptual-semantic relations. In *PROPOR*, pages 120–130, 2006. 45

DCMI. http://dublincore.org/documents/dces/, 2008. Dublin Core Metadata Element Set, Version 1.1. Acesso em: 21 ago. 2009. 14

A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 662–673, New York, NY, USA, 2002. ACM. ISBN 1-58113-449-5. doi: http://doi.acm.org/10.1145/511446.511532. 19

D. Dou. *Ontology translation by ontology merging and automated reasoning*. PhD thesis, New Haven, CT, USA, 2004. Director-Mcdermott, Drew V. 19

M. Duerst and M. Suignard. Internationalized Resource Identifiers (IRIs). RFC 3987 (Proposed Standard), Jan. 2005. URL http://www.ietf.org/rfc/rfc3987.txt. 87

A. Farquhar, R. Fikes, and J. Rice. Tools for assembling modular ontologies in ontolingua. In *In Proc. of AAAI 97*, pages 436–441. AAAI Press, 1997. 21

C. Fellbaum. *A semantic network of English verbs. In: WordNet. An Electronic Lexical Database.*, pages S. 69–104. MIT Press, Cambridge, Mass.; London, 1998. 44

C. Freitas, P. Rocha, and E. Bick. Floresta sintá(c)tica: Bigger, thicker and easier. In *PROPOR '08: Proceedings of the 8th international conference on Computational Processing of the Portuguese Language*, pages 216–219, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-85979-6. doi: http://dx.doi.org/10.1007/978-3-540-85980-2_23. 46, 47

GNU. http://www.gnu.org/software/wget/, 2008. GNU Wget - Free Software Foundation. Acesso em: 12 fev. 2010. 55

T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers. URL http://citeseer.ist.psu.edu/gruber93toward.html. 16

N. Guarino. Formal ontology and information systems. In *FOIS'98*, pages 3–15. IOS Press, 1998. xi, 16, 17

N. Guarino and C. Welty. Ontological analysis of taxonomic relationships. *Data and Knowledge Engineering*, 39:51–74, 2000. 37

GWA. http://www.globalwordnet.org/, 2000. Global WordNet Association. Acesso em: 14 dez. 2009. 45, 86

GWA. http://www.globalwordnet.org/gwa/wordnet_table.htm, 2010. Wordnets in the world. Acesso em: 14 jan. 2010. 45

I. Hakia. http://www.hakia.com/, 2007. Hakia Semantic Search. Acesso em: 24 ago. 2009. 15

I. Horrocks. Implementation and optimisation techniques. In *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 306–346. CUP, Jan. 2003. ISBN 0521781760. 36

I. Horrocks. DAML+OIL: a description logic for the semantic web. *IEEE Data Engineering Bulletin*, 25(1):4–9, 2002. URL http://citeseer.ist.psu.edu/578691.html. 9

I. Horrocks and U. Sattler. http://www.cs.manchester.ac.uk/~horrocks/Slides/ecai-handout.pdf, 2002. Description Logics—Basics, Applications, and More. 31

U. Hustadt, R. A. Schmidt, and L. Georgieva. A survey of decidable first-order fragments and description logics. *Journal of Relational Methods in Computer Science*, 1:2004, 2004. 31

A. Iskold. http://www.readwriteweb.com/archives/semantic_web_patterns.php, 2008. Semantic Web Patterns: A Guide to Semantic Technologies. Acesso em: 11 ago. 2009. xi, 8, 9

L. C. R. Junior. OntoLP: Construção Semi-automática de Ontologias a partir de Textos da Língua Portuguesa. Master's thesis, Universidade do Vale do Rio dos Sinos (UNISINOS, Av. Unisinos, 950 - B. Cristo Rei / CEP 93.022-000 - São Leopoldo (RS) - Brazil, 2008. xi, 6

KAON. http://kaon.semanticweb.org, 2001. FZI WIM and AIFB LS3. Acesso em: 7 jul. 2008. 25

J. J. Katz and J. A. Fodor. The structure of a semantic theory. *Language*, 39(2):170–210, Apr 1963. 43

H. Knublauch. http://www.topbraid.org/2007/05/composite.owl, 2007. TopBraid Composer. Acesso em: 01 ago. 2008. xi, 24

H. Knublauch. Ontology-driven software development in the context of the semantic web: An example scenario with protege/owl. In D. S. Frankel, E. F. Kendall, and D. L. McGuinness, editors, *1st International Workshop on the Model-Driven Semantic Web (MDSW2004)*, 2004. URL http://fparreiras/papers/OntologyDSDSemWebProtegeOWL.pdf. 16

J. Levine, T. Mason, and D. Brown. *lex & yacc, 2nd Edition (A Nutshell Handbook)*. O'Reilly, October 1992. ISBN 1565920007. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1565920007. 42, 60, 69, 107, 112

Linguateca. http://www.linguateca.pt/CETEMPublico/, 2000. *Corpora* CETEMPublico - Projecto Processamento Computacional do Português. Acesso em: 31 jul. 2008. 46

Linguateca. http://www.linguateca.pt/CETENFolha/, 2002. *Corpora* CETENFolha - Projecto Processamento Computacional do Português. Acesso em: 31 jul. 2008. 46

Linguateca. http://www.linguateca.pt/, 1998. Linguateca - Centro de recursos da língua portuguesa - Fundação para a Computação Científica Nacional. Acesso em: 31 jul. 2008. 47

G. F. Luger. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. ISBN 0321263189. xi, 39, 40, 43

A. Maedche and S. Staab. The TEXT-TO-ONTO Ontology Learning Environment. Software Demonstration at ICCS-2000 - Eight International Conference on conceptual Structures August 14-18, 2000, Darmstadt, Germany, available at: http://www.aifb.uni-karlsruhe.de/˜sst/ Research/Publications/Publications.htm, August 2000. xi, 4

D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. The chimaera ontology environment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 1123–1124. AAAI Press / The MIT Press, 2000. ISBN 0-262-51112-6. 19

MEC. http://machado.mec.gov.br/, 2008. Sítio sobre Machado de Assis - MEC - Ministério da Educação. Acesso em: 15 fev. 2010. xi, 54, 55

E. Mena, A. Illarramendi, V. Kashyap, and A. P. Sheth. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distrib. Parallel Databases*, 8(2):223–271, 2000. ISSN 0926-8782. doi: http://dx.doi.org/10.1023/A:1008741824956. 19

P. Mitra, G. Wiederhold, and S. Decker. A scalable framework for the interoperation of information sources. In *Stanford University*, pages 317–329, 2001. 19

D. Nardi and R. J. Brachman. An introduction to description logics. pages 1–40, 2003. xi, 31, 33, 34, 89

S. Nirenburg and V. Raskin. *Ontological Semantics*. MIT Press, Cambridge, MA, 2004. ISBN 978-0-262-14086-7. 43

N. F. Noy and M. A. Musen. The prompt suite: interactive tools for ontology merging and mapping. *Int. J. Hum.-Comput. Stud.*, 59(6):983–1024, 2003. ISSN 1071-5819. doi: http://dx.doi.org/10.1016/j.ijhcs.2003.08.002. xi, 19, 20

M. Obitko. http://www.obitko.com/tutorials/ontologies-semantic-web/frame-based-models.html, 2007. Frame Based Models. Acesso em: 2 set. 2009. 86

U. of Amsterdam, E. Open University, Milton Keynes, S. Spanish Council of Scientific Research (IIIA), Barcelona, G. Institute AIFB, University of Karlsruhe, U. Stanford University, I. S. C. S. A. Spain, and V. U. Amsterdam. http://hcs.science.uva.nl/projects/ibrow/home.html, 2000. An Intelligent Brokering Service for Knowledge-Component Reuse on the World-Wide Web. Acesso em: 15 set. 2009. 21

OntoStudio. http://www.ontoprise.de/en/home/products/ontostudio/, 2008. ontoprise GmbH. Acesso em: 10 fev. 2010. 25

I. S. Overmundo. http://www.overmundo.com.br/, 2006. Projeto Overmundo - Instituto Sociocultural Overmundo. Acesso em: 14 fev. 2010. 47

Palavras. http://beta.visl.sdu.dk/visl/pt/parsing/automatic/, 2000. University of Southern Denmark - SDU. Acesso em: 31 jul. 2008. xi, 45, 46, 101

S. B. Palmer. http://infomesh.net/html/history/early/, 2004. The Early History of HTML. Acesso em: 15 nov. 2009. 11

H. S. Pinto. Towards ontology reuse. In *Proceedings of AAAI99's Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends*, pages 7.1–7.12. AAAI, 1999. 21

H. S. Pinto and a. P. Martins, Jo˙A methodology for ontology integration. In *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture*, pages 131–138, New York, NY, USA, 2001. ACM. ISBN 1-58113-380-4. doi: http://doi.acm.org/10.1145/500737.500759. 20

H. S. Pinto and J. P. Martins. Reusing ontologies. In *In AAAI 2000 Spring Symposium on Bringing Knowledge to Business Processes*, pages 77–84. AAAI Press, 2000. 21

Projeto Floresta Sintática. http://www.linguateca.pt/Floresta/principal.html, 2002. Projeto Floresta Sintática - Fundação para a Computação Científica Nacional - Linguateca. Acesso em: 31 jul. 2008. 47

Protégé. http://protege.stanford.edu, 2006. Stanford Center for Biomedical Informatics Research. Acesso em: 7 jul. 2008. 25

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003. xi, 39, 40, 41, 42, 90

M. Schmidt-Schaubßand G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/0004-3702(91)90078-X. xi, 33, 85

S. Staab and A. Maedche. Ontology engineering beyond the modeling of concepts and relations. In *European Conference on Artificial Intelligence (ECAI) Workshop on Application of Ontologies and Problem-Solving Methods*, 2000. 16

G. Stumme and A. Maedche. Fca-merge: Bottom-up merging of ontologies. In B. Nebel, editor, *Proc. 17th Intl. Conf. on Artificial Intelligence (IJCAI '01)*, pages 225–230, Seattle, WA, USA, 2001. URL http://www.kde.cs.uni-kassel.de/stumme/papers/2001/IJCAI01.pdf. 19

G. Stumme, M. Ehrig, S. Handschuh, A. Hotho, A. Maedche, B. Motik, D. Oberle, C. Schmitz, S. staab, L. Stojanovic, N. Stojanovic, R. Studer, Y. Sure, R. Volz, and V. Zacharias. The Karlsruhe view on ontologies. Technical report, University of Karlsruhe, Institute AIFB, 2003. 16

S. University, V. U. A. Nederland, D. of Computer Science University of Manchester, S. University, B. L. Research, U. o. K. Institute AIFB, and M. I. of Technology. http://www.ontoknowledge.org/oil/index.shtml, 2000. Ontology Inference Layer. Acesso em: 15 set. 2009. xi, 22

M. Uschold and M. Gruninger. Ontologies principles, methods and applications. *Knowledge Sharing and Review*, 11(2), June 1996. 18

M. Uschold, V. R. Benjamins, B. Ch, A. Gomez-perez, N. Guarino, and R. Jasper. A framework for understanding and classifying ontology applications. In *Proceedings of the IJCAI99 Workshop on Ontologies*, pages 16–21, 1999. URL http://www.uni-leipzig.de/~tbittner/courses/GEOID/uschold99FrameworkUnderstandingOntology.pdf. 19

G. Van Heijst, A. T. Schreiber, and B. J. Wielinga. Using explicit ontologies in kbs development. *Int. J. Hum.-Comput. Stud.*, 46(2-3):183–292, 1997. ISSN 1071-5819. doi: http://dx.doi.org/10.1006/ijhc.1996.0090. 18

Vassar College and LORIA/CNRS. http://www.xces.org/, 2010. XCES Corpus Encoding Standard for XML. Acesso em: 10 fev. 2010. 6, 90

K. H. Veltman. Towards a semantic web for culture. *J. Digit. Inf.*, 4(4), 2004. xi, 10

VISL. http://visl.sdu.dk/visl/pt/parsing/automatic/, 2010. Machine Analysis. Acesso em: 2 fev. 2010. 55, 56

VISL. http://visl.sdu.dk/visl/pt/info/portsymbol.html, 2000. The Constraint Grammar category set of "Palavras" Semantic tags for nouns. Acesso em: 2 fev. 2010. 101

VISL-Manual. http://visl.sdu.dk/visl/pt/info/symbolset-manual.html, 2000. University of Southern Denmark - SDU. Acesso em: 31 jul. 2008. 45, 55, 91

W3C. http://www.w3.org/2001/sw/wiki/GRDDL, 2007a. Gleaning Resource Descriptions from Dialects of Languages. Acesso em: 10 jan. 2010. 13, 86

W3C. http://www.w3.org/2001/sw/wiki/OWL, 2009a. Web Ontology Language (OWL). Acesso em: 10 jan. 2010. 12, 23, 88

W3C. http://www.w3.org/TR/owl2-primer/, 2009b. OWL 2 Web Ontology Language Primer. Acesso em: 14 out. 2009. 24

W3C. http://www.w3.org/TR/2009/PR-owl2-syntax-20090922/, 2009c. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. Acesso em: 14 out. 2009. 23

W3C. http://www.w3.org/TR/2009/WD-owl2-manchester-syntax-20090611/, 2009d. OWL 2 Web Ontology Language Manchester Syntax. Acesso em: 14 out. 2009. 23

W3C. http://www.w3.org/TR/2009/PR-owl2-mapping-to-rdf-20090922/, 2009e. OWL 2 Web Ontology Language Mapping to RDF Graphs. Acesso em: 14 out. 2009. 23

W3C. http://www.w3.org/TeamSubmission/2008/SUBM-turtle-20080114/, 2009f. Turtle - Terse RDF Triple Language. Acesso em: 14 out. 2009. 24

W3C. http://www.w3.org/TR/2009/PR-owl2-xml-serialization-20090922/, 2009g. OWL 2 Web Ontology Language XML Serialization. Acesso em: 14 out. 2009. 24

W3C. http://www.w3.org/PICS/, 2009h. Platform for Internet Content Selection (PICS). Acesso em: 10 ago. 2009. 11, 88

W3C. http://www.w3.org/2001/sw/wiki/POWDER, 2009i. Protocol for Web Description Resources (POWDER). Acesso em: 10 jan. 2010. 13, 88

W3C. http://www.w3.org/2001/sw/wiki/RDF, 2004a. Resource Description Framework (RDF). Acesso em: 10 jan. 2010. 12, 88

W3C. http://www.w3.org/2001/sw/wiki/RDFS, 2004b. RDF Vocabulary Description Language 1.0: RDF Schema (RDFS). Acesso em: 10 jan. 2010. 13, 14, 88

W3C. http://www.w3.org/TR/rdf-syntax-grammar/, 2004c. RDF/XML Syntax Specification. Acesso em: 23 ago. 2009. 13

W3C. http://www.w3.org/2001/sw/wiki/RDFa, 2008a. RDF in Attributes (RDFa). Acesso em: 10 jan. 2010. 12, 88

W3C. http://www.w3.org/2001/sw/wiki/RIF, 2009j. Rule Interchange Format (RIF). Acesso em: 10 jan. 2010. 13, 88

W3C. http://www.w3.org/2001/sw/wiki/SAWSDL, 2007b. Semantic Annotations for WSDL and XML Schema. Acesso em: 10 jan. 2010. 13, 88

W3C. http://www.w3.org/2001/sw/wiki/SKOS, 2009k. Simple Knowledge Organization System (SKOS). Acesso em: 10 jan. 2010. 13, 89

W3C. http://www.w3.org/2001/sw/wiki/SPARQL, 2008b. SPARQL Query Language for RDF. Acesso em: 10 jan. 2010. 12, 89

W3C. http://www.w3.org/, 1994. World Wide Web Consortium (W3C). Acesso em: 14 out. 2009. 89

W3C. http://www.w3.org/2001/sw/, 2010. W3C Semantic Web Activity. Acesso em: 15 fev. 2010. xi, 11

W3C. http://www.w3.org/2001/sw/, 2002. W3C Semantic Web Activity. Acesso em: 10 fev. 2008. 1

W3C. http://www.w3.org/TR/swbp-n-aryRelations, 2006. Defining N-ary Relations on the Semantic Web. Acesso em: 2 fev. 2010. 67

W3C. http://www.w3.org/2001/sw/Specs.html, 2009l. Semantic Web Activity Statement. Acesso em: 10 jan. 2010. 12

W3C. http://www.w3.org/WAI/, 1999. Web Accessibility Initiative (WAI). Acesso em: 15 fev. 2010. 53

W3C. http://www.w3.org/TR/wsdl, 2001. Web Services Description Language (WSDL) 1.1. Acesso em: 10 jan. 2010. 89

W3C. http://www.w3.org/XML/, 1996. eXtensible Markup Language. Acesso em: 23 ago. 2009. 90

Wikipedia. http://www.wikipedia.org/, 2001. Wikipedia - Wikipedia Foundation. Acesso em: 13 ago. 2009. 15

WordNet. http://wordnet.princeton.edu/, 2006. Princeton University. Acesso em: 31 jul. 2008. 44, 45, 86, 89

WordNet. http://wordnetweb.princeton.edu/perl/webwn, 2010. WordNet Search 3. Princeton University. Acesso em: 15 jan. 2010. xiii, 44

WordNet.PT. http://cvc.instituto-camoes.pt/wordnet/, 1998. WordNetṖT Rede Léxico-conceptual do Português. Instituto Camões, Ministério dos Negócios Estrangeiros, Portugal. Acesso em: 15 jan. 2010. 45

E. Zolin. http://www.cs.manchester.ac.uk/~ezolin/dl/, 2007. Complexity of reasoning in Description Logics. 38

Glossário

ABox
Assertion box. Conjunto de definições utilizadas para descrever fatos e relacionamentos dentro de uma estrutura de representação do conhecimento (Baader and Nutt [2003]) . xiv, 32–35

AL
Attributive Language (linguagem descritiva $\mathcal{AL}$). Linguagem de lógica descritiva minimalista proposta por Schmidt-Schaubßand Smolka [1991]. Serviu de base para as diversas variações de linguagens descritivas da atualidade . xiv, 33, 34, 39

API
Application Programming Interface. Interface de programação de aplicativos. É um conjunto de definições e rotinas de um *software*, disponibilizadas para tornar recursos acessíveis para os aplicativos que deles necessitam. . xiv, 23

completude
Propriedade de um sistema lógico pela qual nenhum axioma precisa ser a ele adicionado para que todas as fórmulas válidas possam ser formalmente derivadas . 21

consistência
Propriedade de um sistema lógico pela qual uma fórmula precisa ser conseqüência lógica dos axiomas definidos e não pode causar contradições (refutar axiomas) . 21, 35–37

DAML
DARPA Agent Markup Language. Linguagem de marcação desenvolvida pela DARPA entre os anos 2000 e 2006 com o propósito de habilitar a criação de metadados e de ontologias. É uma extensão do XML e do RDF . xiv, 21, 22, 85

DAML+OIL
DARPA Agent Markup Language (DAML) + *Ontology Inference Layer* (OIL). Extensão da linguagem DAML original com os recursos do OIL, criada para viabilizar processos de raciocínio e inferência nas ontologias definidas de acordo com a especificação . xiv, 21, 23

DAML-ONT
DARPA Agent Markup Language (*for ontologies*). Primeira versão da linguagem XML . xiv, 22, 23

DARPA
Defense Advanced Research Projects Agency. É uma agência do Departamento de Defesa dos Estados Unidos responsável pelo desenvolvimento de novas tecnologias para uso militar . xiv, 85

DL
Description Logics. Formalismo lógico utilizado tanto para modelar estruturas de representação do conhecimento quanto para viabilizar o raciocínio automatizado (Baader and Nutt [2003]) . xiv, 87

expressões regulares
Expressões concisas utilizadas para identificar ou gerar cadeias de caracteres. A cadeia "abababababababaaaaaa", por exemplo, pode ser reconhecida ou gerada a partir da expressão regular "$(ab)^{+}a^{+}$", na qual os parênteses agrupam subcadeias e o símbolo "+" indica a ocorrência de 1 ou mais subcadeias. As expressões regulares são expressas através de uma linguagem formal que define outras operações além da repetição ("+") exemplificada aqui . 42, 87

frames
Modelos baseados em *frames*. Utilizam entidades denominadas *frames* e suas propriedades como uma primitiva de modelagem conceitual. A principal primitiva de modelagem é o *frame* e seus *slots*. Cada *slot* pode conter múltiplos valores de propriedades que são aplicáveis apenas ao *frame* ao qual o *slot* está ligado. Classes de *frames* podem ser definidas, e cada instância de uma classe é um *frame* individual, ou simplesmente indivíduo. Modelos baseados em *frames* possuem mecanismos de herança, sendo portanto apropriados para a implementação de ontologias e bases de conhecimento (Obitko [2007]) . 21

GRDDL
Gleaning Resource Descriptions from Dialects of Languages. Um mecanismo para extração de descrições de recursos a partir de outras linguagens (W3C [2007a]) . xiv, 13

GWA
Global WordNet Association. Entidade responsável pela padronização da WordNet [2006] para todas as línguas do mundo, através de iniciativas que incluem a divulgação de informações para o público em geral e a troca de recursos, *softwares* e ferramentas de apoio ao desenvolvimento das wordnets GWA [2000] . xiv, 45

HTML
Hypertext Markup Language. Linguagem de marcação utilizada para produzir páginas para a Web . xiv, 3, 10, 39, 52–54, 56, 74, 90

IA
Inteligência Artificial. Área de pesquisa da Ciência da Computação que tem como objetivo o desenvolvimento de máquinas inteligentes, capazes de resolver problemas e de interagir com o ser humano . xiv, 1, 9, 38, 43, 44, 88, 90

IRI
Internationalized Resource Identifier. Extensão do URI que permite uma especificação mais ampla de endereços da Internet através da definição de um conjunto maior de caracteres possíveis e de uma sintaxe melhorada (Duerst and Suignard [2005]) . xiv, 12

LD
Lógica Descritiva. Vide DL . 31–34, 36, 37

LPO
Abreviação de Lógica de Primeira Ordem. xiv, 28, 29, 31, 43

léxico
Lista de palavras permitidas de uma linguagem, cuja especificação pode incluir expressões regulares . 40, 41

Modus Ponens
Em latim, "modo de afirmar". Regra de inferência lógica estudada desde a Antigüidade, pela qual temos que se $\alpha \rightarrow \beta$, e sabemos que α é verdadeira, então deduzimos que β é necessariamente verdadeira . 28, 87

Modus Tollens
Em latim, "modo de negar". Regra de inferência lógica estudada desde a Antigüidade, pela qual temos que se $\alpha \rightarrow \beta$, e sabemos que β é falsa, então deduzimos que α é necessariamente falsa . 28, 87

MP
Abreviação de Modus Ponens . xiv, 28

MT
Abreviação de Modus Tollens . xiv, 28

OCR
Optical Character Recognition. Técnica de reconhecimento de caracteres a partir de uma imagem digitalizada de um texto. . xiv, 39

OIL
Ontology Inference Layer. Camada de representação e de interface de ontologias para a Web fortemente baseada em RDF . xiv, 21–23, 85

ontologias
Estruturas de representação do conhecimento nas quais cada conceito representado possui um significado bem definido. . vii, 2–7, 9, 12, 15–26, 31, 37–39, 48, 49, 51, 52, 58, 59, 64, 65, 67–71, 73–75, 85–87, 102, 117, 125

OWL
Web Ontology Language. Linguagem de descrição de ontologias própria para aplicações que necessitam processar o conteúdo da informação ao invés de apenas apresentá-lo

a humanos. Também denominada OWL 1, por já existir uma nova versão da linguagem denominada OWL 2. A linguagem possibilita uma melhor representação do conteúdo da Web do que a suportada por XML ou RDF (W3C [2009a]) . xiv, 12, 21, 23–25, 69, 71, 75, 88, 117, 125

OWL 2
Web Ontology Language - Revision 2. OWL versão 2, de 22 de setembro de 2009. Vide OWL (W3C [2009a]) . xiv, 23, 88

PICS
Platform for Internet Content Selection. Uma linguagem de descrição de metatados designada inicialmente para ajudar pais e professores a controlar o que crianças acessavam na Web, mas que também permitia seu uso, ainda que limitado, na descrição de conteúdo da Web (W3C [2009h]) . xiv, 10, 11

PLN
Processamento de Linguagem Natural. Conjunto de métodos formais, normalmente utilizados em IA, para a interpretação automatizada de linguagem falada ou escrita . xiv, 38–40, 43, 44, 46–48

POWDER
Protocol for Web Description Resources. Um protocolo de descrição de conteúdo que facilita a extração de informações necessárias para uma seleção prévia de recursos da Web, opcionalmente com uma verificação de autenticidade (W3C [2009i]) . xiv, 13

RDF
Resource Description Framework Core Model. Uma linguagem de descrição de metatados e de conteúdo semântico (W3C [2004a], W3C [2004b]) . xiv, 10–14, 21–23, 69, 71, 75, 85, 87–89, 117, 125

RDFa
RDF in Attributes. Uma especificação de atributos para expressar dados estruturados em XHTML (W3C [2008a]) . xiv, 12

RDFS
RDF Schema. Uma extensão do RDF para a definição de classes específicas de aplicações e respectivas propriedades (W3C [2004b]) . xv, 13, 21, 22

RIF
Rule Interchange Format. Um padrão para a troca de informações entre sistemas baseados em regras que podem ser compartilhadas entre sistemas distintos (W3C [2009j]) . xv, 13

SAWSDL
Semantic Annotations for WSDL and XML Schema. Padrão de definição de atributos utilizados para agregar informações semânticas a especificações WSDL (W3C [2007b]) . xv, 13

Silogismo Hipotético
Regra de inferência utilizada em lógica proposicional, pela qual temos que se $\alpha \rightarrow \beta$ e $\beta \rightarrow \gamma$, então por transitividade sabemos que $\alpha \rightarrow \gamma$. 28

SKOS
Simple Knowledge Organization System. Um modelo de dados para o compartilhamento e interconexão de bases de conhecimento através da Web (W3C [2009k]) . xv, 13

SPARQL
SPARQL Protocol and RDF Query Language. Linguagem de pesquisa utilizada para recuperar e manipular informações gravadas em formato RDF (W3C [2008b]) . xv, 12, 15

synsets
Agrupamentos de sinônimos utilizados no WordNet [2006]. Esses agrupamentos são conectados entre si através de uma série de relações semânticas, constituindo dessa forma um banco de dados léxico de grande valia para aplicações que utilizam alguma forma de representação do conhecimento . 44

TBox
Terminology box. Conjunto de definições utilizadas para descrever a terminologia hierarquicamente organizada em uma estrutura de representação do conhecimento (Baader and Nutt [2003]). Como essas definições normalmente são utilizadas para a definição de taxonomias, Nardi and Brachman [2003] sugerem que *TBox* pode ser considerada também uma abreviação de *Taxonomy box* . xv, 32–35

Turtle
Terse RDF Triple Language. Sintaxe baseada em RDF, porém mais intuitiva. . xv, 24

URI
Uniform Resource Identifier. Uma especificação para endereços da Internet que define a sintaxe dos identificadores de recursos da rede (Berners-Lee [1994]) . xv, 12, 13, 69, 87

W3C
World Wide Web Consortium (W3C). Organização fundadada em 1º de outubro de 1994 (Cailliau and Connolly [2000]) com o objetivo de supervisionar o desenvolvimento de padrões de protocolos para a Web e promover a sua interoperabilidade (W3C [1994]) . xv, 8, 90

WSDL
Web Services Description Language. Um formato XML para descrever serviços de rede e suas operações, bem como definições abstratas que podem ser associadas a protocolos de rede específicos (W3C [2001]) . xv, 13, 88

wumpus
Antigo jogo de computador cujo objetivo é caçar um monstro chamado wumpus, que vive em uma caverna e está sempre disposto a devorar qualquer aventureiro que se aproxime. Russell and Norvig [2003] utilizaram o mundo de wumpus como exemplo para o estudo de diversos assuntos em IA . 41, 42

XCES
Corpus Encoding Standard for XML. Padrão de codificação de *corporas* no formato XML (Vassar College and LORIA/CNRS. [2010]) . xv, 6

XHTML
eXtensible HyperText Markup Language. Uma reformulação em XML da linguagem HTML . xv, 12, 13, 88

XML
eXtensible Markup Language. Linguagem definida pelo W3C para a geração de linguagens de marcação (W3C [1996]) . xv, 4, 12, 13, 21, 23, 24, 69, 71, 75, 85, 88–90, 117, 125

Apêndice A

Marcas Gramaticais do Sistema PALAVRAS

As principais anotações gramaticais geradas pelo sistema PALAVRAS estão aqui separadas conforme as funções que desempenham nos textos. A referência completa de todas as anotações pode ser obtida em VISL-Manual [2000].

A.1 Termos das orações

Símbolo	Categoria	Exemplos
S @SUBJ> @<SUBJ	subject sujeito subjekt	Ninguém gosta de chuva Retomar o controle foi difícil. A cidade era toda de vidro. Seja quem for. Tem gente morrendo de fome no Brasil. Fugiram do zôo um hipopótamo e um crocodilo .
@SUBJ>>	pre-matrix verb subject	Nunca o vi jogar futebol. As inspecções , recorde-se, começaram em Abril
Od, Oacc @ACC> @<ACC	direct (accusative) object objeto direto (acusativo) direkte (akkusativ) objekt	Liga a luz ! Para combater as doenças do inverno, come vitaminos . Não tem onde morar . Sempre come um monte de folhas .
@ACC>>	pre-matrix verb object	ela se deixou levar. .., o que é quase dispensável a dizer
@ACC>-PASS @<ACC-PASS	passive 'se'-construction	
Oi, Odat @DAT> @<DAT	dative object objeto indireto pronominal indirekte (dativ) objekt	Lhe dou um presente. Preste- me a sua caneta, por favor! Me mostre seu hipopótamo!

Op, Opiv @PIV> @<PIV	prepositional object objeto preposicional præpositionsobjekt	Não me lembro dele . Falamos sobre a sua proposta . Gostava muito de passear ao longo do rio . Não sabe de nada . Pode contar comigo . Chamamos de objeto preposicional complementos indiretos não substituíveis por pronomes adverbiais.
As @ADVS> @<ADVS Ao @ADVO> @<ADVO	argument adverbial complemento adverbial adverbialargument [can be substituted by adverbial pronoun, valency bound, unlike adjuncts]	Durava muito tempo . (As) A vase caiu no chao . (As) Não mora mais aqui . Mora em São Paulo . (As) Voltamos ao nosso assunto . (As) Mandaram-nos para Londres . (Ao) Costuma custar mais de mil coroas . (As)
Cs @SC> @<SC	subject complement predicativo do sujeito subjektsprædikat(iv)	Está doente . Está com febre . A moça parece muito cansada . Nadava nua no mar. Andava zangado todo dia.
Co @OC> @<OC	object complement predicativo do objeto objektsprædikat(iv)	O acho muito chato . Tê-lo feito de propósito o faz um delito .
P	predicator predicador præddikator	Hipopótamo come folhas. Hipopótamo tem que dormir muito.
Vm @FMV @IMV	main verb verbo principal hovedverbum	Bebe muita cerveja. Todo dia mandava (1) o filho comprar (2) leite. Hipopótamo **tem que dormir** muito.
Vaux @FAUX @IAUX	auxiliary verbo auxiliar hjælpeverbum	A interface **foi feito** por uma equipe da WinSoft. **Estou lendo** um romance português. Hipopótamo **tem que dormir** muito.
@NPHR	top node noun phrase (especially headlines)	Clinton ainda no poder
@ADVL	top node adverbial (especially headlines)	Seis milhões em dez anos

A.2 Adjuntos

Símbolo	Categoria	Exemplos
fA @ADVL> @<ADVL	adjunct adverbial adjunto adverbial adverbialadjunkt	Sempre comiam cedo . As crianças jogavam no parque . Feito o trabalho temos tempo para mais uma cerveja. Entraram na vila quando amanheceu . O outro dia fugiu do zôo um hipopótamo.
@ADVL>AS<	adjunct adverbial in averbal clause	..., ainda que agora **de pouco valor**
fC @PRED> @<PRED	adjunct predicative adjunto predicativo prædikativadjunkt	Sempre nada nua . Cansado , se retirou.
fApass @<PASS	passive adjunct adjunto do passivo passivadjunkt	Era o herói do dia e foi elogiado pelo chefe do jardim zoológico .
fCsta @S<	statement predicative (sentence apposition) aposto da oração sætningsprædikativ	Morreu o cachorro da velha, o que muito a entristece .
@>S	complementizer adject	Só quando ele lhe surge na frente, se compenetra que era mesmo verdade
fCvoc @VOK	vocative adjunct constituinte vocativo vokativadjunkt	Me ajuda, Pedro !
FOC @FOC> @<FOC	focus marker marcador de foco fokusmarker	É a dançar que ela se entende. (focus bracket) Foi de sua música que gostei. (focus bracket) Gosta é de briga.
TOP @TOP	topic constituent constituinte de tópico topic-konstituent	A Maria , não quero convidá-la. (object topic) Esse rapaz , ele sabe dançar. (subject topic)

A.3 Subordinação e coordenação

Símbolo	Categoria	Exemplos
SUB @SUB	subordinator subordinador subordinator	Acho que um jardim zoológico sem hipopótamos não merece subsídios.
SUBcom @COM	comparative subordinator subordinador comparativo komparator	Esta fofoqueira fala como uma cachoeira .

SUBprd @PRD	predicative subordinator (role complementizer) subordinador predicativo rollejndleder	Trabalha como guía.
SUBaux @PRT- AUX<	auxiliary subordinator subordinador auxiliar (partículo auxiliar) auxiliarpartikel	**Voltou a molestá**-la no escritório. O outro ano **acabou de ensinar** inglês. Hipopótamo **tem que dormir** muito.
@#ICL-AUX< (not as simple @)	auxiliary complement complemento auxiliar auxiliarkomplement	Hipopótamo **tem que dormir** muito.
SUB< @AS<	[averbal] clause body tronco de oração [averbal] sætningsstamme (indlederkomplement)	Quando em Roma , faça como os romanos.
CO @CO	coordinator coordenador koordinator	Fugiram do zôo um hipopótamo e um crocodilo.
CJT	conjunct (elemento) conjunto konjunkt	Fugiram do zôo um hipopótamo e um crocodilo .

A.4 Termos de agrupamentos

Símbolo	Categoria	Exemplos
H - D	head <-> dependent núcleo <-> dependente hoved <-> dependent	uma grande árvore sem dinheiro devagar demais
D DN DNmod DNarg @>N, @N<	adnominal adject adjeto adnominal adnominaladjekt (H: noun or pronoun)	o (1) seu (2) grande (3) carro novo (4) (modifiers) a (1) mulher do amigo (2) (modifiers) um tanto (modifier) cacique Jerônimo (modifier) Manoel Neto (1) da Silva (2) (modifiers) a proposta de lhe ajudar (argument) combinaram a venda da casa . (argument) predisposição para diabete (argument)
DNapp @APP	(adnominal) apposition aposição (do substantivo) [epíteto de identidade] (nominal-) apposition	O grande cacique, Jerônimo , conhecia o seu país como mais ninguém.
DNc @N<PRED	predicative adject adjeto predicativo [epíteto predicativo] præ̈dikativadjekt	Jerônimo, um grande cacique , temia ninguém. com a mão na bolsa

DA DAmod DAarg @>A, @A<	adverbial adject adjeto adverbial adverbialadjekt (H: adjective, adverb or determiner)	muito devagar (modifier) devagar demais (modifier) rico em ouro (argument) receoso de lhe ter ofendido (argument)
@ADVL>A @A<ADVL @A<ADV @A<PASS @A<PIV @A<SC	functions in postnominal participal "clause"	o número de famílias já (ADVL>A) **abrangidas** pelo projecto (A<PASS) um equipamento periférico **denominado** "Audioman" (A<SC) um período **destinado** a testar a aplicação (A<PIV) uma discoteca **situada** em Albufeira (A<ADV)
@NUM<	numeral chain constituent	de **1** a 9 de Junho passado a criação de **30 mil** a 50 mil postos de emprego
DAcom @KOMP<	argument of comparative complemento comparativo komparativkomplement	é **mais** bonito do que um hipopótamo .
DP DParg DPmod @P<, @>P	argument of preposition argumento de preposição præpositionsargument [styrelse]	sem dinheiro nenhum (argument) quase sem dinheiro (modifier)
Dfoc	focus dependent dependente focalizardor fokus dependent	Até o rei gostava da peça. Comeu nem a sobremesa.

A.5 Enunciados

Símbolo	Categoria	Exemplos
UTT STA UTT QUE UTT COM UTT EXC	utterance enunciado ytring	Não faz nada. [statement] Já vás embora? [question] Espera! [command] Pobre de mim! [exclamation]
STA	statement enunciado declarativo udsagn	A terra é redonda. Gosta muito de elefantes. Sua vez. Às sete. Obrigado.
QUE	question enunciado interrogativo spørgsmål	Quem quer uma cerveja? Já ligou para o ministério? Quando?
COM	command enunciado imperativo ordre	Pára com isso! Venha pra cá! Fora!

EXC	exclamation enunciado exclamativo udråb	Deus! Que beleza! Quanta gente!

A.6 Sintagmas

Símbolo		Categoria	Exemplos
np	np propp pronp	noun phrase sintagma nominal nominalsyntagme (H: noun or pronoun)	Era um **homem** como um forte . (np) A velha **avó** dormia na rede. (np) Vou fazê-lo **eu** mesmo . (pronp) O seu nome era **Mário** Moreno dos Santos . (propp)
ap	adjp advp detp	adpositional phrase sintagma adposicional adpositionssyntagme (H: adjective, adverb or determiner)	As árvores no jardim eram muito **velhas** . (adjp) Foi um presidente um pouco **iconoclasta** . (adjp) Nesta saia, parece mais **jovem** do que as amigas . (adjp) Costuma falar muito **devagar** . (advp) Ainda **hoje** vivem de caça e pesca. (advp) Era muito **mais** vinho do que imaginava . (detp)
vp		verb phrase sintagma verbal verbalsyntagme (H: main verb [semantically] or auxiliary [dependency grammar])	Ele continua mexendo nas tarefas dos outros. Vem de lhes propor um acordo. Temos que dar -lhe mais dinheiro.
pp		prepositional phrase sintagma preposicional præpositionssyntagme (H: prepostion)	Abriu a janela da sala Gostou do que viu. Pedro da Silva Mudamos para São Paulo.

A.7 Orações

Símbolo		Categoria	Exemplos
cl	fcl @#FS-...	finite (sub)clause oração finita finit (led)sætning	Não acredito que seja verdade
	icl @#ICL-...	non-finite (sub)clause oração infinita infinit (led)sætning	Consertar um relógio não pode ser fácil

acl @#AS-...	averbal (sub)clause oração averbal averbal (led)sætning	Ajudou onde possível
cu	compound unit paratagma paratagme	ver Roma e viver a história era o seu sonho.

A.8 Classes de palavras

Símbolo	Categoria	Exemplos
n N	noun nome substantiv (nomen)	árvores **n**(F P) um oitavo **n**(<num> M S)
prop PROP	proper noun nome próprio proprium (egenavn)	Estados=Unidos **prop**(M P) Dinamarca **prop**(F S)
adj ADJ	adjective adjetivo adjektiv	belas **adj**(F P) terceiros **adj(<num> M P)**
v V v-fin VFIN	finite verb verbo finito finit verbum (bøjet i tid)	fizessem **v-fin**(IMPF 3P SUBJ)
v-inf INF	infinitive infinitivo infinitiv	fazermos **v-inf**(1P)
v-pcp PCP	participle particípio participium	comprados **v-pcp**(M P) [attributive] tem comprado **v-pcp** [verbal]
v-ger GER	gerund gerúndio gerundium	correndo **v-ger**
art DET <artd> DET <arti>	article artigo artikel	os membros **art**(<artd> M P) [definite] uma criança **art**(<arti> F S) [indefinite]
pron pron-pers PERS	personal pronoun pronome pessoal personligt pronomen	mim **pron-pers**(1S PIV) tu **pron-pers(2S NOM)**
pron-det DET	determiner pronoun pronome determinativo determinativt pronomen (adjektivisk pronomen)	estas **pron-det**(<dem> F P) [demonstrative] muita **pron-det**(<quant> F S) [indefinite] cujos **pron-det**(<rel> M P) [relative] quantos **pron-det**(<interr> M P) [interrogative] minhas **pron-det**(<poss 1P> F P) [possessive]

pron-indp SPEC	independent pronoun pronome independente independent pronomen (substantivisk pronomen)	isto **pron-indp**(<dem> M S) [demonstrative] algo, nada **pron-indp**(<quant> M S) [indefinite] os=quais **pron-indp**(<rel> M P) [relative] quem **pron-indp**(<interr> M S) [interrogative]
adv ADV	adverb advérbio adverbium	facilmente , devagar **adv** [modals] aqui , lá **adv** [pronominals] muito , imensamente **adv** [intensifiers] onde , quando , como **adv** [relatives, interrogatives] não , até , já **adv** [operators]
num NUM	numeral numeral numeralia	duas **num**(F P) 17 **num** (<cif> M/F P)
prp PRP	presposition preposição preposition	contra **prp** em=vez=de **prp**
intj IN	interjection interjeição interjektion	oi ! **in**
conj conj-s KS	subordinating conjunction conjunção subordinativa underordnende konjunktion	que **conj-s** embora **conj-s**
conj-c KC	coordinating conjunction conjunção coordenativa sideordnende konjunktion	e **conj-c** ou **conj-c**
pu (unused)	punctuation pontuação tegnsætningstegn	, **pu** [komma]

A.9 Marcas secundárias

<artd>	definite article	DET
<arti>	indefifnite article	DET
<card>, cf. <NUM-ord>	cardinal number	NUM

*<co-acc>, <co-advl>, <co-app>, <co-dat>, <co-fmc>, <co-ger>, <co-inf>, <co-oc>, <co-pcv>, <co-postad>, <co-postnom>, <co-pred>, <co-prenom>, <co-prparg>, <co-sc>, <co-subj>, <co-vfin>	what a co-ordinator co-ordinates (used internally for tree-generation) @ACC, @ADVL, @APP, @DAT, main clauses, GER, INF, @OC, PCP-@IMV, @A<, @N<, @PRED, @>N, @P<, @SC, @SUBJ, VFIN	KC
<dem>	demonstrative	DET, SPEC
<DERP>	derivation by prefixation	N, ADJ, V, ADV
<DERS>	derivation by suffixation	N, ADJ, V, ADV
<det>	determiner usage/inflection of adverb	ela estava toda nua
<diff>	differentiator	DET (o mesmo, o outro)
*<fmc>	finite main clause (used internally for tree-generation)	VFIN –> @FMV @#FS-STA (statement)
*<foc>	focus marker (redundant –> @FOC)	ADV (eis, eis=que, é=que, foi ... que, é)
<hyfen>	hyphenated word (left-marked)	acreditá-lo –> 'acreditar' V <hyfen> + 'o' PERS
<ident>	identifier	DET (ele mesmo, ele próprio)
<interr>, cf. <rel>	interrogative	DET (quanto, que), SPEC (quem, o=que), ADV (onde, porquê)
*<kc> (not disambiguated)	can be used as "conjuntional adverb"	ADV (pois, entretanto, mais)
<KOMP>	comparative "hook" (for @KOMP<)	DET, ADV (mais, menos, pior, tal, mesmo)
<ks>, cf. <prp>	used like a "subordinating conjunction"	ADV <rel> (como, onde, quando)
<NUM-ord>, cf. <card>	ordinal number (subclass of adjective)	ADJ (terceiro, quinto)
<poss 1S>, <poss 1P>, <poss 2S>, <poss 2P>, <poss 3S/P>	possessive	DET
<prop>	other word class used as "proper noun", i.e. with capital initial	N, ADJ, PCP

<prp>, cf. <ks>	used like a “preposition”	ADV <rel> (como, onde, quando)
<quant>	quantifier (indefinite)	DET, SPEC
<refl>, cf. <si>	reflexive	PERS (se, me, te, nos, vos, si)
<rel>, cf. <interr>	relative	DET (cujo), SPEC (quem, que, o=qual), ADV (onde, quando, como)
<sam->	1. part in contracted word	nisto –> em
<-sam>	2. part in contracted word	nisto –> isto
<si>, cf. <refl> and <poss>	reflexive usage of 3.person possessive	DET <poss> (seu, sua, seus, suas)
<SUP>	superlative	

Apêndice B

Testes Preliminares

Os testes iniciais de extração de conteúdo semântico a partir de textos concentraram-se na identificação de conceitos existentes nesses textos e a interligação entre esses conceitos, através do tratamento das marcas gramaticais existentes em *corporas*, com o objetivo de gerar mapas conceituais representativos dos textos.

O *corpora* CETEMPúblico foi escolhido para os testes porque contém marcas semânticas previamente anotadas pelo sistema PALAVRAS (Palavras [2000]). Segundo VISL [2000], essas marcas estão ainda em um estágio experimental. Sua utilização, entretanto, foi adequada para uma análise inicial do problema de interligação de conteúdo e identificação prévia de conceitos, mesmo considerando que esse *corpora* não contém a estrutura hierárquica do texto.

Os primeiros testes evidenciaram a necessidade de utilização de um dicionário para a análise do conteúdo completo do *corpora*. Sem esse dicionário, o tempo de execução em um Pentium 4 506 2.66 GHz com 1 GB de RAM foi de mais de 5 horas. Com o dicionário, implementado na forma de um *array* ordenado em memória, sobre o qual uma busca binária era efetuada, o tempo de resposta caiu para apenas cerca de 26 minutos (Tabela B.1).

Tempo total	00:25:32	(dicionário)
Linhas	232093994	
Extratos	1504081	
Títulos	655059	
Parágrafos	2571735	
Sentenças	7030479	
Nomes	36603868	
Nomes sem repetição	140073	0.38%
Nomes próprios	18994861	
Nomes próprios sem repetição	1049540	5.53%
Verbos	24683932	
Verbos sem repetição	25591	0.10%
Marcas semânticas	99618805	
Marcas semânticas sem repetição	515	$< 0.01\%$
Elementos não identificados	5185	

Tabela B.1: Resultados preliminares da análise do *corpora* CETEMPúblico.

Esses resultados foram considerados importantes para o dimensionamento das estruturas de dados internas necessárias para o tratamento dos textos.

Apêndice C

Ontologia Base

A seguir é apresentada a *ontologia base* original desenvolvida neste trabalho e utilizada no processo de geração automática de ontologias. Esta versão da ontologia está disponível em Bravo [2010b], enquanto que a versão atual está em Bravo [2010a].

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY ontoge "http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#" >
]>

<rdf:RDF xmlns="http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#"
     xml:base="http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:base="http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about="">
        <rdfs:label rdf:datatype="&xsd;string"
            >Ontoge v0.02 (por)</rdfs:label>
        <owl:versionInfo rdf:datatype="&xsd;string">0.02</owl:versionInfo>
        <dc:date rdf:datatype="&xsd;dateTime"
            >2010-03-05 15:00</dc:date>
        <dc:creator rdf:datatype="&xsd;string"
            >Carlos de Oliveira Bravo</dc:creator>
        <dc:description rdf:datatype="&xsd;string"
            >Classes b&#225;sicas para a modelagem de ontologias baseadas na l&#237;ngua
                portuguesa.</dc:description>
        <dc:title rdf:datatype="&xsd;string"
            >Ontoge v0.02 (por)</dc:title>
        <dc:publisher rdf:datatype="&xsd;string">ontoge.org</dc:publisher>
    </owl:Ontology>

    <!--
    ////////////////////////////////////////////////////////////////////////////////

    //
    // Annotation properties
    //
```

```
////////////////////////////////////////////////////////////////////////////////

 -->

<owl:AnnotationProperty rdf:about="&dc;date"/>
<owl:AnnotationProperty rdf:about="&dc;publisher"/>
<owl:AnnotationProperty rdf:about="&dc;title"/>
<owl:AnnotationProperty rdf:about="&dc;description"/>
<owl:AnnotationProperty rdf:about="&dc;creator"/>

<!--
////////////////////////////////////////////////////////////////////////////////

//
// Classes
//
////////////////////////////////////////////////////////////////////////////////

 -->

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#... -->
<owl:Class rdf:about="#...">
    <rdfs:subClassOf rdf:resource="#Estrutura"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Adjetivo -->
<owl:Class rdf:about="#Adjetivo"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Adverbio -->
<owl:Class rdf:about="#Adverbio"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Artigo -->
<owl:Class rdf:about="#Artigo"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Conjuncao -->
<owl:Class rdf:about="#Conjuncao"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Estrutura -->
<owl:Class rdf:about="#Estrutura"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#FL_NUM_Plural -->
<owl:Class rdf:about="#FL_NUM_Plural">
    <rdfs:subClassOf rdf:resource="#FL_Numero"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#FL_NUM_Singular
    -->
<owl:Class rdf:about="#FL_NUM_Singular">
    <rdfs:subClassOf rdf:resource="#FL_Numero"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#FL_Numero -->
<owl:Class rdf:about="#FL_Numero">
    <rdfs:subClassOf rdf:resource="#Flexao"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Flexao -->
<owl:Class rdf:about="#Flexao"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#GEN_Feminino -->
<owl:Class rdf:about="#GEN_Feminino">
    <rdfs:subClassOf rdf:resource="#Genero"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#GEN_Masculino -->
<owl:Class rdf:about="#GEN_Masculino">
    <rdfs:subClassOf rdf:resource="#Genero"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Genero -->
```

```
<owl:Class rdf:about="#Genero"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Interjeicao -->
<owl:Class rdf:about="#Interjeicao"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
    MV_IMPE_Afirmativo -->
<owl:Class rdf:about="#MV_IMPE_Afirmativo">
    <rdfs:subClassOf rdf:resource="#MV_Imperativo"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#MV_IMPE_Negativo
    -->
<owl:Class rdf:about="#MV_IMPE_Negativo">
    <rdfs:subClassOf rdf:resource="#MV_Imperativo"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#MV_Imperativo -->
<owl:Class rdf:about="#MV_Imperativo">
    <rdfs:subClassOf rdf:resource="#ModoVerbal"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#MV_Indicativo -->
<owl:Class rdf:about="#MV_Indicativo">
    <rdfs:subClassOf rdf:resource="#ModoVerbal"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#MV_Subjuntivo -->
<owl:Class rdf:about="#MV_Subjuntivo">
    <rdfs:subClassOf rdf:resource="#ModoVerbal"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#ModoVerbal -->
<owl:Class rdf:about="#ModoVerbal"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#NIL -->
<owl:Class rdf:about="#NIL">
    <rdfs:subClassOf rdf:resource="#Estrutura"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Nome -->
<owl:Class rdf:about="#Nome"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Preposicao -->
<owl:Class rdf:about="#Preposicao"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Pronome -->
<owl:Class rdf:about="#Pronome"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Substantivo -->
<owl:Class rdf:about="#Substantivo"/>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#TV_Futuro -->
<owl:Class rdf:about="#TV_Futuro">
    <rdfs:subClassOf rdf:resource="#TempoVerbal"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
    TV_INDI_Futuro_Presente_Composto -->
<owl:Class rdf:about="#TV_INDI_Futuro_Presente_Composto">
    <rdfs:subClassOf rdf:resource="#MV_Indicativo"/>
    <rdfs:subClassOf rdf:resource="#TV_Futuro"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
    TV_INDI_Futuro_Presente_Simples -->
<owl:Class rdf:about="#TV_INDI_Futuro_Presente_Simples">
    <rdfs:subClassOf rdf:resource="#MV_Indicativo"/>
    <rdfs:subClassOf rdf:resource="#TV_Futuro"/>
</owl:Class>
```

```
<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
    TV_INDI_Futuro_Preterito_Composto -->
<owl:Class rdf:about="#TV_INDI_Futuro_Preterito_Composto">
    <rdfs:subClassOf rdf:resource="#MV_Indicativo"/>
    <rdfs:subClassOf rdf:resource="#TV_Futuro"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
    TV_INDI_Futuro_Preterito_Simples -->
<owl:Class rdf:about="#TV_INDI_Futuro_Preterito_Simples">
    <rdfs:subClassOf rdf:resource="#MV_Indicativo"/>
    <rdfs:subClassOf rdf:resource="#TV_Futuro"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#TV_INDI_Presente
    -->
<owl:Class rdf:about="#TV_INDI_Presente">
    <rdfs:subClassOf rdf:resource="#MV_Indicativo"/>
    <rdfs:subClassOf rdf:resource="#TV_Presente"/>
    <dc:description
        >Indicativo - Presente</dc:description>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
    TV_INDI_Preterito_Imperfeito -->
<owl:Class rdf:about="#TV_INDI_Preterito_Imperfeito">
    <rdfs:subClassOf rdf:resource="#MV_Indicativo"/>
    <rdfs:subClassOf rdf:resource="#TV_Preterito"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
    TV_INDI_Preterito_Mais_Que_Perfeito -->
<owl:Class rdf:about="#TV_INDI_Preterito_Mais_Que_Perfeito">
    <rdfs:subClassOf rdf:resource="#MV_Indicativo"/>
    <rdfs:subClassOf rdf:resource="#TV_Preterito"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
    TV_INDI_Preterito_Perfeito_Composto -->
<owl:Class rdf:about="#TV_INDI_Preterito_Perfeito_Composto">
    <rdfs:subClassOf rdf:resource="#MV_Indicativo"/>
    <rdfs:subClassOf rdf:resource="#TV_Preterito"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
    TV_INDI_Preterito_Perfeito_Simples -->
<owl:Class rdf:about="#TV_INDI_Preterito_Perfeito_Simples">
    <rdfs:subClassOf rdf:resource="#MV_Indicativo"/>
    <rdfs:subClassOf rdf:resource="#TV_Preterito"/>
    <dc:description
        >Indicativo - Pret&#233;rito Perfeito Simples</dc:description>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#TV_Presente -->
<owl:Class rdf:about="#TV_Presente">
    <rdfs:subClassOf rdf:resource="#TempoVerbal"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#TV_Preterito -->
<owl:Class rdf:about="#TV_Preterito">
    <rdfs:subClassOf rdf:resource="#TempoVerbal"/>
</owl:Class>

<!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#TV_SUBJ_Presente
    -->
<owl:Class rdf:about="#TV_SUBJ_Presente">
    <rdfs:subClassOf rdf:resource="#MV_Subjuntivo"/>
    <rdfs:subClassOf rdf:resource="#TV_Presente"/>
    <dc:description
```

```
            >Subjuntivo - Presente</dc:description>
    </owl:Class>

    <!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
         TV_SUBJ_Preterito_Imperfeito -->
    <owl:Class rdf:about="#TV_SUBJ_Preterito_Imperfeito">
        <rdfs:subClassOf rdf:resource="#MV_Subjuntivo"/>
        <rdfs:subClassOf rdf:resource="#TV_Preterito"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
         TV_SUBJ_Preterito_Mais_Que_Perfeito -->
    <owl:Class rdf:about="#TV_SUBJ_Preterito_Mais_Que_Perfeito">
        <rdfs:subClassOf rdf:resource="#MV_Subjuntivo"/>
        <rdfs:subClassOf rdf:resource="#TV_Preterito"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#
         TV_SUBJ_Preterito_Perfeito -->
    <owl:Class rdf:about="#TV_SUBJ_Preterito_Perfeito">
        <rdfs:subClassOf rdf:resource="#MV_Subjuntivo"/>
        <rdfs:subClassOf rdf:resource="#TV_Preterito"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#TempoVerbal -->
    <owl:Class rdf:about="#TempoVerbal">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#VFN_Gerundio -->
    <owl:Class rdf:about="#VFN_Gerundio">
        <rdfs:subClassOf rdf:resource="#VerboFormaNominal"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#VFN_INF_Impessoal
         -->
    <owl:Class rdf:about="#VFN_INF_Impessoal">
        <rdfs:subClassOf rdf:resource="#VFN_Infinitivo"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#VFN_INF_Pessoal
         -->
    <owl:Class rdf:about="#VFN_INF_Pessoal">
        <rdfs:subClassOf rdf:resource="#VFN_Infinitivo"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#VFN_Infinitivo --
         >
    <owl:Class rdf:about="#VFN_Infinitivo">
        <rdfs:subClassOf rdf:resource="#VerboFormaNominal"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#VFN_Partic&#237;
         pio -->
    <owl:Class rdf:about="#VFN_Partic&#237;pio">
        <rdfs:subClassOf rdf:resource="#VerboFormaNominal"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#Verbo -->
    <owl:Class rdf:about="#Verbo"/>

    <!-- http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#VerboFormaNominal
         -->
    <owl:Class rdf:about="#VerboFormaNominal">
        <rdfs:subClassOf rdf:resource="#Nome"/>
    </owl:Class>

    <!-- http://www.w3.org/2002/07/owl#Thing -->
    <owl:Class rdf:about="&owl;Thing"/>
</rdf:RDF>
```

Apêndice D

Gramática para Leitura das Árvores Sintáticas

A seguir é apresentada a gramática utilizada durante o desenvolvimento do componente *oglnk*. Essa gramática foi desenvolvida com a ajuda das ferramentas *Lex* e *Yacc* (Levine et al. [1992]), sendo que o arquivo submetido ao **Lex** foi omitido aqui para facilitar a visualização da gramática.

```
%{
/*-------------------------------------------------------------------------
 *
 *  Ontoge - OgSem - Analisador Semântico
 *
 *  Autor - Carlos de Oliveira Bravo <carlos_bravo@terra.com.br>
 *
 *  oglnk.y - Gramática do Yacc (Bison) para a montagem da árvore sintática.
 *  Criado em: 2010-01-19 22:34
 *  Alterado em: 2010-03-20 19:47
 *
 */

#ifdef _MSC_VER
#pragma warning(disable:4273)
#pragma warning(disable:4065)
#endif

#include "tree.h"
#include "cScannerContext.h"
#include "cSyntaxTree.h"
#include <ontoge/ansiesc.h>
#include <stdio.h>
#include <string.h>
#include <iostream>

using namespace std;

#define scanner context.scanner

char *spaces(int n);
void show(int level, char *main_color, char *type, char *description, char *attrlist,
    char *word);
int yyerror(cScannerContext& context, cCollection<cSyntaxTree>& tree_array, char *msg);

%}

%pure-parser
%parse-param {cScannerContext &context}
%parse-param {cCollection<cSyntaxTree> &tree_array}
```

```
%lex-param   {yyscan_t scanner}

%union {
        struct {
                char *string;
                int level;
        }
        text;
        int value;
        void *data;
}

%token <text>  TYPE
%token <text>  DESCRIPTION
%token <text>  ATTRIBUTE
%token <text>  WORD
%token <text> PUNCTUATION
%token <text>  PARENTHESIS
%token <text>  COMMA

%%

text            : term_list PUNCTUATION
                  {
                          cSyntaxTree *tree;
                          t_node *node = new_node($2.level, $2.string, NULL, NULL, NULL,
                              NULL);

                          ((t_node *)$<data>1)->last->next = node;
                          node->previous = ((t_node *)$<data>1)->last;
                          ((t_node *)$<data>1)->last = node;
                          if(tree = new cSyntaxTree(context.sentence, (t_node *)$<data>1))
                              {
                                  tree_array.Add(*tree);
                          }
                          else {
                                  yyerror(context, tree_array, "Erro tentando criar nova
                                      árvore.");
                                  YYABORT;
                          }
                  }
                  | text term_list PUNCTUATION
                  {
                          cSyntaxTree *tree;
                          t_node *node = new_node($3.level, $3.string, NULL, NULL, NULL,
                              NULL);

                          ((t_node *)$<data>2)->last->next = node;
                          node->previous = ((t_node *)$<data>2)->last;
                          ((t_node *)$<data>2)->last = node;
                          if(tree = new cSyntaxTree(context.sentence, (t_node *)$<data>2))
                              {
                                  tree_array.Add(*tree);
                          }
                          else {
                                  yyerror(context, tree_array, "Erro tentando criar nova
                                      árvore.");
                                  YYABORT;
                          }
                  }
                | error
                  {
                          yyerror(context, tree_array, "Erro de sintaxe.");
                          YYABORT;
                  }
                ;


term_list       : term
                  {
```

```
                    // Primeiro nó da lista.
                    ((t_node *)$<data>1)->last = (t_node *)$<data>1;
                    $<data>$ = $<data>1;
                  }
                | term_list term
                  {
                    // Coloca o nó na lista.
                    ((t_node *)$<data>1)->last->next = (t_node *)$<data>2;
                    ((t_node *)$<data>2)->previous = ((t_node *)$<data>1)->last;
                    ((t_node *)$<data>1)->last = (t_node *)$<data>2;
                    // Propaga o primeiro nó na pilha.
                    $<data>$ = $<data>1;
                  }


term              : TYPE ':' DESCRIPTION '(' attribute_list ')' WORD
                  {
                    char dbg_attrlist[512];
                    // Cria um nó para o novo termo identificado.
                    $<data>$ = new_node($1.level, $1.string, $3.string, NULL, (t_strlist
                        *)$<data>5, $7.string);
                    strlist_to_char((t_strlist *)$<data>5, dbg_attrlist, sizeof
                        dbg_attrlist, ' ');
                    show($1.level, GREEN, $1.string, $3.string, dbg_attrlist, $7.string);
                  }
                | TYPE ':' DESCRIPTION '(' attribute_list ')' '(' attribute_list ')' WORD
                  {
                    char dbg_attrlist[512];
                    // Cria um nó para o novo termo identificado.
                    $<data>$ = new_node($1.level, $1.string, $3.string, (t_strlist *)$<
                        data>5, (t_strlist *)$<data>8, $10.string);
                    strlist_to_char((t_strlist *)$<data>8, dbg_attrlist, sizeof
                        dbg_attrlist, ' ');
                    show($1.level, GREEN, $1.string, $3.string, dbg_attrlist, $10.string);
                  }
                | TYPE ':' DESCRIPTION
                  {
                    $<data>$ = new_node($1.level, $1.string, $3.string, NULL, NULL, NULL);
                    show($1.level, GREEN, $1.string, $3.string, NULL, NULL);
                  }
                | TYPE ':' DESCRIPTION '(' attribute_list ')'
                  {
                    char dbg_attrlist[512];
                    $<data>$ = new_node($1.level, $1.string, $3.string, NULL, (t_strlist
                        *)$<data>5, NULL);
                    strlist_to_char((t_strlist *)$<data>5, dbg_attrlist, sizeof
                        dbg_attrlist, ' ');
                    show($1.level, GREEN, $1.string, $3.string, dbg_attrlist, NULL);
                  }
                | TYPE ':' DESCRIPTION '(' attribute_list ')' '(' attribute_list ')'
                  {
                    char dbg_attrlist[512];
                    $<data>$ = new_node($1.level, $1.string, $3.string, (t_strlist *)$<
                        data>5, (t_strlist *)$<data>8, NULL);
                    strlist_to_char((t_strlist *)$<data>8, dbg_attrlist, sizeof
                        dbg_attrlist, ' ');
                    show($1.level, GREEN, $1.string, $3.string, dbg_attrlist, NULL);
                  }
                | COMMA
                  {
                    $<data>$ = new_node($1.level, $1.string, NULL, NULL, NULL, NULL);
                    show($1.level, GREEN, $1.string, NULL, NULL, NULL);
                  }
                | PARENTHESIS
                  {
                    $<data>$ = new_node($1.level, $1.string, NULL, NULL, NULL, NULL);
                    show($1.level, GREEN, $1.string, NULL, NULL, NULL);
                  }
                | PARENTHESIS attribute_list
                  {
```

```
                    char dbg_attrlist[512];
                    $<data>$ = new_node($1.level, $1.string, NULL, NULL, (t_strlist *)$<
                        data>2, NULL);
                    strlist_to_char((t_strlist *)$<data>2, dbg_attrlist, sizeof
                        dbg_attrlist, ' ');
                    show($1.level, GREEN, $1.string, NULL, dbg_attrlist, NULL);
                  }
                ;

attribute_list : attribute
                  {
                    // A lista ainda não existe. Vamos criá-la e colocá-la na pilha.
                    $<data>$ = new_strlist();
                    /* Adiciona a string à lista. */
                    strlist_add((t_strlist *)$<data>$, $<text>1.string);
                  }
                | attribute_list attribute
                  {
                    // Adiciona a string à lista.
                    strlist_add((t_strlist *)$<data>1, $<text>2.string);
                    // Propaga a lista existente na pilha.
                    $<data>$ = $<data>1;
                  }
                ;


attribute      : ATTRIBUTE
                  {
                    // Guarda a string na pilha.
                    $<text>$ = $1;
                  }
                | '(' ATTRIBUTE ')'
                  {
                    int len;
                    char *temp;
                    // Coloca a string entre parênteses para preservar a informação.
                    len = strlen($2.string);
                    temp = (char *)malloc(len + 3);
                    temp[0] = '(';
                    strcpy(temp + 1, $2.string);
                    temp[len + 1] = ')';
                    temp[len + 2] = 0;
                    free($2.string);
                    $2.string = temp;
                    // Guarda a string na pilha.
                    $<text>$.string = temp;
                  }
                ;

%%

int yyerror(cScannerContext& context, cCollection<cSyntaxTree>& tree_array, char *msg) {

        if(strcmp(msg, "syntax error"))
                cerr << MAGENTA << msg << C_DEFAULT << '\n';
        return 1;
}


char *spaces(int n) {

        static char buffer[256];

        if(n < 0)
                n = 0;

        if(n >= sizeof buffer)
                n = (sizeof buffer) - 1;
```

```
        memset(buffer, '␣', (sizeof buffer) - 1);
        buffer[n] = 0;

        return buffer;
}


void show(int level, char *main_color, char *type, char *description, char *attrlist,
    char *word) {

        int n;

        fprintf(stderr, "%s", main_color);
        n = fprintf(stderr, "%s%s", spaces(level * 2), type);
        if(description) {
                fprintf(stderr, WHITE ":%s", main_color);
                n += fprintf(stderr, "%s", description) + 1;
        }
        if(attrlist) {
                fprintf(stderr, WHITE "(" YELLOW);
                ++n;
                n += fprintf(stderr, "%s", attrlist);
                fprintf(stderr, WHITE ")");
                ++n;
                if(!word)
                        fprintf(stderr, "%s", main_color);
                else
                        fprintf(stderr, WHITE "%s%s%s", n >= 60 ? "\t" : spaces(60 - n),
                            word, main_color);
        }
        fputc('\n', stderr);
}
```

Apêndice E

Gramática para a Combinação de Ontologias

A seguir é apresentada a gramática utilizada durante o desenvolvimento do componente *ogcom*. Essa gramática foi desenvolvida com a ajuda das ferramentas *Lex* e *Yacc* (Levine et al. [1992]), sendo que o arquivo submetido ao **Lex** foi omitido aqui para facilitar a visualização da gramática.

```
%{
/*-------------------------------------------------------------------------
 *
 *  Ontoge - OgCom - Combinação de Ontologias
 *
 *  Autor - Carlos de Oliveira Bravo <carlos_bravo@terra.com.br>
 *
 *  ogcom.y - Gramática do Yacc (Bison) para a leitura do mapa de elementos.
 *  Criado em: 2010-01-21 11:33
 *  Alterado em: 2010-03-22 10:17
 *
 */

#ifdef _MSC_VER
#pragma warning(disable:4273)
#pragma warning(disable:4065)
#endif

#ifdef _WIN32
#undef YY_NO_UNISTD_H
#define YY_NO_UNISTD_H
#endif
#include "ogcom.tab.h"  // Deve vir antes de "oglnk.lex.h".
#include "ogcom.lex.h"
#include <ontoge/ansiesc.h>
#include <ontoge/strlist.h>
#include <ontoge/owl/cOWLWriter.h>
#include <stdio.h>
#include <string.h>
#include <stdio.h>
#include <iostream>

using namespace std;

int yyerror(cOWLWriter& owlwriter, yyscan_t scanner, char *msg);

%}

%pure-parser
%parse-param {cOWLWriter &owlwriter}
```

```
%parse-param {yyscan_t scanner}
%lex-param {yyscan_t scanner}

%union {
        char *string;
        int value;
        void *data;
}

%token <value> INSTANCE_SECTION OBJPROP_SECTION DATAPROP_SECTION PCHAIN_SECTION
    LINK_SECTION
%token <string> STRING ARGUMENT
%token <value> COMMA NEW_LINE


%%


map_file        : section
                | map_file section
                | error
                  {
                        yyerror(owlwriter, scanner, "Erro de sintaxe.");
                        YYABORT;
                  }
                ;

section         : instance_section
                | objprop_section
                | dataprop_section
                | pchain_section
                | link_section
                ;

instance_section: INSTANCE_SECTION instance_list
                ;

instance_list   : instance
                | instance_list instance
                ;

instance        : ARGUMENT ARGUMENT STRING classes argument_list NEW_LINE
                  {
                        t_strnode *node;

                        // Declaração antecipada de classes (opcional).
                        node = ((t_strlist *)$<data>4)->first;
                        while(node) {
                                owlwriter.AddClass(node->string);
                                node = node->next;
                        }

                        // Declaração.
                        owlwriter.OpenInstance($1);

                        // Rótulo.
                        owlwriter.OpenInstanceLabel($1);
                        owlwriter << $3;
                        owlwriter.CloseInstanceLabel();

                        // Descrição.
                        owlwriter.OpenInstanceDescription($1);
                        owlwriter << '\"' << $3 << '\"';
                        node = ((t_strlist *)$<data>5)->first;
                        while(node) {
                                owlwriter << '_' << node->string;
                                node = node->next;
                        }
                        owlwriter.CloseInstanceDescription();
```

```
                        // Classes.
                        owlwriter.OpenInstanceClassAssertionList($1);
                        node = ((t_strlist *)$<data>4)->first;
                        while(node) {
                                owlwriter.AddInstanceClassAssertion(node->string);
                                node = node->next;
                        }
                        owlwriter.CloseInstanceClassAssertionList();

                        // Fecha a definição do elemento.
                        owlwriter.CloseInstance();
                  }
                ;

classes         : ARGUMENT
                  {
                        // A lista ainda não existe. Vamos criá-la e colocá-la na pilha.
                        $<data>$ = new_strlist();
                        /* Adiciona a string à lista. */
                        strlist_add((t_strlist *)$<data>$, $1);
                        //cerr   << "line " << __LINE__ << ": classes - ARGUMENT\n";
                  }
                | classes COMMA ARGUMENT
                  {
                        // Adiciona a string à lista.
                        strlist_add((t_strlist *)$<data>1, $3);
                        // Propaga a lista existente na pilha.
                        $<data>$ = $<data>1;
                        //cerr   << "line " << __LINE__ << ": classes - classes COMMA
                            ARGUMENT\n";
                  }
                ;

argument_list   : ARGUMENT
                  {
                        // A lista ainda não existe. Vamos criá-la e colocá-la na pilha.
                        $<data>$ = new_strlist();
                        /* Adiciona a string à lista. */
                        strlist_add((t_strlist *)$<data>$, $1);
                        //cerr   << "line " << __LINE__ << ": argument_list - ARGUMENT\n
                            ";
                  }
                | argument_list ARGUMENT
                  {
                        // Adiciona a string à lista.
                        strlist_add((t_strlist *)$<data>1, $2);
                        // Propaga a lista existente na pilha.
                        $<data>$ = $<data>1;
                        //cerr   << "line " << __LINE__ << ": argument_list -
                            argument_list ARGUMENT\n";
                  }
                ;

objprop_section : OBJPROP_SECTION objprop_list
                ;

objprop_list    : objprop
                | objprop_list objprop
                ;

objprop         : ARGUMENT NEW_LINE
                  {
                        // Declaração.
                        owlwriter.OpenObjectProperty($1);

                        // Rótulo apenas para a preposição (por enquanto).
                        if(strncmp($1, "PREP_", 5) == 0) {
                                owlwriter.OpenObjectPropertyLabel($1);
                                owlwriter << ($1 + 5);
                                owlwriter.CloseObjectPropertyLabel();
```

```
                        }

                        // Fecha a definição da propriedade.
                        owlwriter.CloseObjectProperty();
                }
                | ARGUMENT classes NEW_LINE
                {
                        t_strnode *node;

                        // Declaração.
                        owlwriter.OpenObjectProperty($1);

                        // Rótulo apenas para a preposição (por enquanto).
                        if(strncmp($1, "PREP_", 5) == 0) {
                                owlwriter.OpenObjectPropertyLabel($1);
                                owlwriter << ($1 + 5);
                                owlwriter.CloseObjectPropertyLabel();
                        }

                        // Propriedades "base".
                        owlwriter.OpenObjectPropertyParentList($1);
                        node = ((t_strlist *)$<data>2)->first;
                        while(node) {
                                owlwriter.AddObjectPropertyParent(node->string);
                                node = node->next;
                        }
                        owlwriter.CloseObjectPropertyParentList();

                        // Fecha a definição da propriedade.
                        owlwriter.CloseObjectProperty();
                }
                ;

dataprop_section: DATAPROP_SECTION dataprop_list
                ;

dataprop_list   : dataprop
                | dataprop_list dataprop
                ;

dataprop        : ARGUMENT ARGUMENT STRING NEW_LINE
                {
                        // Declaração.
                        owlwriter.OpenDataProperty($1);

                        // Rótulo
                        owlwriter.OpenDataPropertyLabel($1);
                        owlwriter << $3;
                        owlwriter.CloseDataPropertyLabel();

                        // Fecha a definição da propriedade.
                        owlwriter.CloseDataProperty();
                }
                ;

pchain_section  : PCHAIN_SECTION pchain_list
                ;

pchain_list     : pchain
                | pchain_list pchain
                ;

pchain          : ARGUMENT argument_list NEW_LINE
                {
                        t_strnode *node;

                        // Declaração.
                        owlwriter.OpenObjectPropertyChain($1);

                        // Propriedades componentes.
```

```
                        node = ((t_strlist *)$<data>2)->first;
                        while(node) {
                                owlwriter.AddObjectPropertyChainProperty(node->string);
                                node = node->next;
                        }

                        // Fecha a definição da propriedade.
                        owlwriter.CloseObjectPropertyChain();
                  }
                ;

link_section    : LINK_SECTION link_list
                ;

link_list       : link
                | link_list link
                ;

link            : ARGUMENT STRING ARGUMENT NEW_LINE
                  {
                        // Propriedade de dados.
                        owlwriter.AddDataPropertyAssertion($1, $2, $3);
                  }
                | ARGUMENT ARGUMENT ARGUMENT NEW_LINE
                  {
                        owlwriter.AddObjectPropertyAssertion($2, $1, $3);
                  }
                ;

%%


int yyerror(cOWLWriter& owlwriter, yyscan_t scanner, char *msg) {

        if(strcmp(msg, "syntax error"))
                cerr << MAGENTA << msg << C_DEFAULT << '\n';
        return 1;
}
```

Apêndice F

Arquivos Incluídos na Etapa de Combinação de Ontologias

Adiante são apresentadas as versões em RDF/XML e OWL/XML das ontologias mostradas na Figura 6.24. Essas ontologias foram criadas com o editor de ontologias *Protégé 4.0* (vide Seção 3.5) e posteriormente incluídas no processo de geração de ontologias para a frase de exemplo "*Maria deixou José na garagem*" (vide Apêndice G).

F.1 Versões em RDF/XML

F.1.1 Maria.owl (RDF/XML)

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY Maria "http://ontoge.org/ontology/test/Maria.owl#" >
]>

<rdf:RDF xmlns="http://ontoge.org/ontology/test/Maria.owl#"
     xml:base="http://ontoge.org/ontology/test/Maria.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:Maria="http://ontoge.org/ontology/test/Maria.owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>

    <!--
    ////////////////////////////////////////////////////////////////////////////////////

    //
    // Object Properties
    //
    ////////////////////////////////////////////////////////////////////////////////////

     -->

    <!-- http://ontoge.org/ontology/test/Maria.owl#temFilho -->
    <owl:ObjectProperty rdf:about="#temFilho"/>
```

```
    <!--
    ///////////////////////////////////////////////////////////////////////////////////

    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////

     -->

    <!-- http://ontoge.org/ontology/test/Maria.owl#Homem -->
    <owl:Class rdf:about="#Homem">
        <rdfs:subClassOf rdf:resource="#Pessoa"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/test/Maria.owl#Mae -->
    <owl:Class rdf:about="#Mae">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Mulher"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#temFilho"/>
                        <owl:someValuesFrom rdf:resource="#Pessoa"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>

    <!-- http://ontoge.org/ontology/test/Maria.owl#Mulher -->
    <owl:Class rdf:about="#Mulher">
        <rdfs:subClassOf rdf:resource="#Pessoa"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/test/Maria.owl#Pessoa -->
    <owl:Class rdf:about="#Pessoa"/>

    <!-- http://www.w3.org/2002/07/owl#Thing -->
    <owl:Class rdf:about="&owl;Thing"/>

    <!--
    ///////////////////////////////////////////////////////////////////////////////////

    //
    // Individuals
    //
    ///////////////////////////////////////////////////////////////////////////////////

     -->

    <!-- http://ontoge.org/ontology/test/Maria.owl#jose -->
    <owl:Thing rdf:about="#jose">
        <rdf:type rdf:resource="#Homem"/>
    </owl:Thing>

    <!-- http://ontoge.org/ontology/test/Maria.owl#maria -->
    <Mulher rdf:about="#maria">
        <rdf:type rdf:resource="&owl;Thing"/>
        <temFilho rdf:resource="#jose"/>
    </Mulher>
</rdf:RDF>
```

F.1.2 Joao.owl (RDF/XML)

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY Joao "http://ontoge.org/ontology/test/Joao.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<rdf:RDF xmlns="http://ontoge.org/ontology/test/Joao.owl#"
     xml:base="http://ontoge.org/ontology/test/Joao.owl"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:Joao="http://ontoge.org/ontology/test/Joao.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>

    <!--
    ///////////////////////////////////////////////////////////////////////////////////////

    //
    // Annotation properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////

     -->

    <owl:AnnotationProperty rdf:about="&dc;description"/>

    <!--
    ///////////////////////////////////////////////////////////////////////////////////////

    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////

     -->

    <!-- http://ontoge.org/ontology/test/Joao.owl#temFilho -->
    <owl:ObjectProperty rdf:about="#temFilho"/>

    <!-- http://ontoge.org/ontology/test/Joao.owl#trabalhaEm -->
    <owl:ObjectProperty rdf:about="#trabalhaEm"/>

    <!--
    ///////////////////////////////////////////////////////////////////////////////////////

    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////////

     -->

    <!-- http://ontoge.org/ontology/test/Joao.owl#Empregado -->
    <owl:Class rdf:about="#Empregado">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Pessoa"/>
                    <owl:Restriction>
```

```
                        <owl:onProperty rdf:resource="#trabalhaEm"/>
                        <owl:someValuesFrom rdf:resource="#Empresa"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Pessoa"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/test/Joao.owl#Empresa -->
    <owl:Class rdf:about="#Empresa"/>

    <!-- http://ontoge.org/ontology/test/Joao.owl#Homem -->
    <owl:Class rdf:about="#Homem">
        <rdfs:subClassOf rdf:resource="#Pessoa"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/test/Joao.owl#Pai -->
    <owl:Class rdf:about="#Pai">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#Homem"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#temFilho"/>
                        <owl:someValuesFrom rdf:resource="#Pessoa"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#Homem"/>
    </owl:Class>

    <!-- http://ontoge.org/ontology/test/Joao.owl#Pessoa -->
    <owl:Class rdf:about="#Pessoa"/>

    <!-- http://www.w3.org/2002/07/owl#Thing -->
    <owl:Class rdf:about="&owl;Thing"/>

    <!--
    ////////////////////////////////////////////////////////////////////////////////////

    //
    // Individuals
    //
    ////////////////////////////////////////////////////////////////////////////////////

     -->

    <!-- http://ontoge.org/ontology/test/Joao.owl#joao -->
    <owl:Thing rdf:about="#joao">
        <rdf:type rdf:resource="#Homem"/>
        <temFilho rdf:resource="#jose"/>
        <trabalhaEm rdf:resource="#petrobras"/>
    </owl:Thing>

    <!-- http://ontoge.org/ontology/test/Joao.owl#jose -->
    <owl:Thing rdf:about="#jose">
        <rdf:type rdf:resource="#Homem"/>
    </owl:Thing>

    <!-- http://ontoge.org/ontology/test/Joao.owl#petrobras -->
    <Empresa rdf:about="#petrobras">
        <rdf:type rdf:resource="&owl;Thing"/>
        <dc:description xml:lang="pt">Petrobr&#225;s S.A.</dc:description>
    </Empresa>
</rdf:RDF>
```

F.1.3 Garagem.owl (RDF/XML)

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY Garagem "http://ontoge.org/ontology/test/Garagem.owl#" >
]>

<rdf:RDF xmlns="http://ontoge.org/ontology/test/Garagem.owl#"
     xml:base="http://ontoge.org/ontology/test/Garagem.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:Garagem="http://ontoge.org/ontology/test/Garagem.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>

    <!--
    ////////////////////////////////////////////////////////////////////////////////////

    //
    // Classes
    //
    ////////////////////////////////////////////////////////////////////////////////////

     -->

    <!-- http://ontoge.org/ontology/test/Garagem.owl#Local -->
    <owl:Class rdf:about="#Local"/>

    <!-- http://ontoge.org/ontology/test/Garagem.owl#Local_Estranho -->
    <owl:Class rdf:about="#Local_Estranho">
        <rdfs:subClassOf rdf:resource="#Local"/>
    </owl:Class>

    <!-- http://www.w3.org/2002/07/owl#Thing -->
    <owl:Class rdf:about="&owl;Thing"/>

    <!--
    ////////////////////////////////////////////////////////////////////////////////////

    //
    // Individuals
    //
    ////////////////////////////////////////////////////////////////////////////////////

     -->

    <!-- http://ontoge.org/ontology/test/Garagem.owl#a_garagem -->
    <owl:Thing rdf:about="#a_garagem">
        <rdf:type rdf:resource="#Local_Estranho"/>
    </owl:Thing>
</rdf:RDF>
```

F.2 Versões em OWL/XML

F.2.1 Maria.owl (OWL/XML)

```
<?xml version="1.0"?>

<!DOCTYPE Ontology [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY Maria "http://ontoge.org/ontology/test/Maria.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<Ontology xmlns="http://www.w3.org/2006/12/owl2-xml#"
     xml:base="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:Maria="http://ontoge.org/ontology/test/Maria.owl#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     URI="http://ontoge.org/ontology/test/Maria.owl">
    <SubClassOf>
        <Class URI="&Maria;Homem" />
        <Class URI="&Maria;Pessoa" />
    </SubClassOf>
    <EquivalentClasses>
        <Class URI="&Maria;Mae" />
        <ObjectIntersectionOf>
            <Class URI="&Maria;Mulher" />
            <ObjectSomeValuesFrom>
                <ObjectProperty URI="&Maria;temFilho" />
                <Class URI="&Maria;Pessoa" />
            </ObjectSomeValuesFrom>
        </ObjectIntersectionOf>
    </EquivalentClasses>
    <SubClassOf>
        <Class URI="&Maria;Mulher" />
        <Class URI="&Maria;Pessoa" />
    </SubClassOf>
    <ClassAssertion>
        <Class URI="&Maria;Homem" />
        <Individual URI="&Maria;jose" />
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="&owl;Thing" />
        <Individual URI="&Maria;jose" />
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="&Maria;Mulher" />
        <Individual URI="&Maria;maria" />
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="&owl;Thing" />
        <Individual URI="&Maria;maria" />
    </ClassAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty URI="&Maria;temFilho" />
        <Individual URI="&Maria;maria" />
        <Individual URI="&Maria;jose" />
    </ObjectPropertyAssertion>
</Ontology>
```

F.2.2 Joao.owl (OWL/XML)

```
<?xml version="1.0"?>

<!DOCTYPE Ontology [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY Joao "http://ontoge.org/ontology/test/Joao.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<Ontology xmlns="http://www.w3.org/2006/12/owl2-xml#"
     xml:base="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:Joao="http://ontoge.org/ontology/test/Joao.owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     URI="http://ontoge.org/ontology/test/Joao.owl">
    <EquivalentClasses>
        <Class URI="&Joao;Empregado" />
        <ObjectIntersectionOf>
            <Class URI="&Joao;Pessoa" />
            <ObjectSomeValuesFrom>
                <ObjectProperty URI="&Joao;trabalhaEm" />
                <Class URI="&Joao;Empresa" />
            </ObjectSomeValuesFrom>
        </ObjectIntersectionOf>
    </EquivalentClasses>
    <SubClassOf>
        <Class URI="&Joao;Empregado" />
        <Class URI="&Joao;Pessoa" />
    </SubClassOf>
    <SubClassOf>
        <Class URI="&Joao;Homem" />
        <Class URI="&Joao;Pessoa" />
    </SubClassOf>
    <EquivalentClasses>
        <Class URI="&Joao;Pai" />
        <ObjectIntersectionOf>
            <Class URI="&Joao;Homem" />
            <ObjectSomeValuesFrom>
                <ObjectProperty URI="&Joao;temFilho" />
                <Class URI="&Joao;Pessoa" />
            </ObjectSomeValuesFrom>
        </ObjectIntersectionOf>
    </EquivalentClasses>
    <SubClassOf>
        <Class URI="&Joao;Pai" />
        <Class URI="&Joao;Homem" />
    </SubClassOf>
    <ClassAssertion>
        <Class URI="&Joao;Homem" />
        <Individual URI="&Joao;joao" />
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="&owl;Thing" />
        <Individual URI="&Joao;joao" />
    </ClassAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty URI="&Joao;temFilho" />
        <Individual URI="&Joao;joao" />
        <Individual URI="&Joao;jose" />
    </ObjectPropertyAssertion>
    <ObjectPropertyAssertion>
```

```
        <ObjectProperty URI="&Joao;trabalhaEm"/>
        <Individual URI="&Joao;joao"/>
        <Individual URI="&Joao;petrobras"/>
    </ObjectPropertyAssertion>
    <ClassAssertion>
        <Class URI="&Joao;Homem"/>
        <Individual URI="&Joao;jose"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="&owl;Thing"/>
        <Individual URI="&Joao;jose"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="&Joao;Empresa"/>
        <Individual URI="&Joao;petrobras"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="&owl;Thing"/>
        <Individual URI="&Joao;petrobras"/>
    </ClassAssertion>
    <EntityAnnotation>
        <Individual URI="&Joao;petrobras"/>
        <Annotation annotationURI="&dc;description">
            <Constant>Petrobr&#225;s S.A.</Constant>
        </Annotation>
    </EntityAnnotation>
</Ontology>
```

F.2.3 Garagem.owl (OWL/XML)

```
<?xml version="1.0"?>

<!DOCTYPE Ontology [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY Garagem "http://ontoge.org/ontology/test/Garagem.owl#" >
]>

<Ontology xmlns="http://www.w3.org/2006/12/owl2-xml#"
      xml:base="http://www.w3.org/2006/12/owl2-xml#"
      xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
      xmlns:Garagem="http://ontoge.org/ontology/test/Garagem.owl#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      URI="http://ontoge.org/ontology/test/Garagem.owl">
    <SubClassOf>
        <Class URI="&Garagem;Local_Estranho"/>
        <Class URI="&Garagem;Local"/>
    </SubClassOf>
    <ClassAssertion>
        <Class URI="&Garagem;Local_Estranho"/>
        <Individual URI="&Garagem;a_garagem"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="&owl;Thing"/>
        <Individual URI="&Garagem;a_garagem"/>
    </ClassAssertion>
</Ontology>
```

Apêndice G

Ontologia Resultante do Processo de Geração

A seguir são apresentadas as versões em RDF/XML e OWL/XML da ontologia resultante do processo de geração para a frase de exemplo "*Maria deixou José na garagem*". O componente *ogcom* do protótipo gerou essa ontologia combinando as ontologias apresentadas na Figura 6.24 (vide Apêndice F) com o resultado do processamento do arquivo mostrado na Figura 6.21.

G.1 Ontologia Resultante em RDF/XML

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY Garagem "http://ontoge.org/ontology/test/Garagem.owl#" >
    <!ENTITY Joao "http://ontoge.org/ontology/test/Joao.owl#" >
    <!ENTITY Maria "http://ontoge.org/ontology/test/Maria.owl#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY ontoge "http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#" >
    <!ENTITY nectar "http://ontoge.org/ontology/test/nectar.owl#" >
]>

<rdf:RDF xmlns="http://ontoge.org/ontology/test/nectar.owl#"
     xml:base="http://ontoge.org/ontology/test/nectar.owl#"
     xmlns:Garagem="http://ontoge.org/ontology/test/Garagem.owl#"
     xmlns:Joao="http://ontoge.org/ontology/test/Joao.owl#"
     xmlns:Maria="http://ontoge.org/ontology/test/Maria.owl#"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:ontoge="http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#"
     xmlns:nectar="http://ontoge.org/ontology/test/nectar.owl#" >

  <owl:Ontology rdf:about="">
    <dc:creator>Ontoge 0.02</dc:creator>
    <owl:imports rdf:resource="http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base
        .owl"/>
```

```
</owl:Ontology>

<owl:AnnotationProperty rdf:about="&dc;description"/>
<owl:AnnotationProperty rdf:about="&dc;creator"/>

<owl:Class rdf:about="&owl;Thing"/>

<!--     <owl:Ontology rdf:about=""/> -->

  <owl:ObjectProperty rdf:about="#temFilho"/>

  <owl:Class rdf:about="#Homem">
      <rdfs:subClassOf rdf:resource="#Pessoa"/>
  </owl:Class>

  <owl:Class rdf:about="#Mae">
      <owl:equivalentClass>
          <owl:Class>
              <owl:intersectionOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Mulher"/>
                  <owl:Restriction>
                      <owl:onProperty rdf:resource="#temFilho"/>
                      <owl:someValuesFrom rdf:resource="#Pessoa"/>
                  </owl:Restriction>
              </owl:intersectionOf>
          </owl:Class>
      </owl:equivalentClass>
  </owl:Class>

  <owl:Class rdf:about="#Mulher">
      <rdfs:subClassOf rdf:resource="#Pessoa"/>
  </owl:Class>

  <owl:Class rdf:about="#Pessoa"/>

  <owl:Class rdf:about="&owl;Thing"/>

  <owl:Thing rdf:about="#jose">
      <rdf:type rdf:resource="#Homem"/>
  </owl:Thing>

  <Mulher rdf:about="#maria">
      <rdf:type rdf:resource="&owl;Thing"/>
      <temFilho rdf:resource="#jose"/>
  </Mulher>

<!--     <owl:Ontology rdf:about=""/> -->

  <owl:AnnotationProperty rdf:about="&dc;description"/>

  <owl:ObjectProperty rdf:about="#temFilho"/>

  <owl:ObjectProperty rdf:about="#trabalhaEm"/>

  <owl:Class rdf:about="#Empregado">
      <owl:equivalentClass>
          <owl:Class>
              <owl:intersectionOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Pessoa"/>
                  <owl:Restriction>
                      <owl:onProperty rdf:resource="#trabalhaEm"/>
                      <owl:someValuesFrom rdf:resource="#Empresa"/>
                  </owl:Restriction>
              </owl:intersectionOf>
          </owl:Class>
      </owl:equivalentClass>
      <rdfs:subClassOf rdf:resource="#Pessoa"/>
  </owl:Class>

  <owl:Class rdf:about="#Empresa"/>
```

```
  <owl:Class rdf:about="#Homem">
      <rdfs:subClassOf rdf:resource="#Pessoa"/>
  </owl:Class>

  <owl:Class rdf:about="#Pai">
      <owl:equivalentClass>
          <owl:Class>
              <owl:intersectionOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Homem"/>
                  <owl:Restriction>
                      <owl:onProperty rdf:resource="#temFilho"/>
                      <owl:someValuesFrom rdf:resource="#Pessoa"/>
                  </owl:Restriction>
              </owl:intersectionOf>
          </owl:Class>
      </owl:equivalentClass>
      <rdfs:subClassOf rdf:resource="#Homem"/>
  </owl:Class>

  <owl:Class rdf:about="#Pessoa"/>

  <owl:Class rdf:about="&owl;Thing"/>

  <owl:Thing rdf:about="#joao">
      <rdf:type rdf:resource="#Homem"/>
      <temFilho rdf:resource="#jose"/>
      <trabalhaEm rdf:resource="#petrobras"/>
  </owl:Thing>

  <owl:Thing rdf:about="#jose">
      <rdf:type rdf:resource="#Homem"/>
  </owl:Thing>

  <Empresa rdf:about="#petrobras">
      <rdf:type rdf:resource="&owl;Thing"/>
      <dc:description xml:lang="pt">Petrobr&#225;s S.A.</dc:description>
  </Empresa>

<!--      <owl:Ontology rdf:about=""/> -->

  <owl:Class rdf:about="#Local"/>

  <owl:Class rdf:about="#Local_Estranho">
      <rdfs:subClassOf rdf:resource="#Local"/>
  </owl:Class>

  <owl:Class rdf:about="&owl;Thing"/>

  <owl:Thing rdf:about="#a_garagem">
      <rdf:type rdf:resource="#Local_Estranho"/>
  </owl:Thing>
<owl:Class rdf:about="&ontoge;Nome"/>

<owl:Thing rdf:about="#maria">
  <rdfs:label>Maria</rdfs:label>
  <dc:description>&quot;Maria&quot; 003735D0 S:prop</dc:description>
  <rdf:type rdf:resource="&ontoge;Nome"/>
</owl:Thing>

<owl:Thing rdf:about="#jose">
  <rdfs:label>Jos&#233;</rdfs:label>
  <dc:description>&quot;Jos&#233;&quot; 00901658 Od:prop</dc:description>
  <rdf:type rdf:resource="&ontoge;Nome"/>
</owl:Thing>

<owl:Thing rdf:about="#garagem">
  <rdfs:label>garagem</rdfs:label>
  <dc:description>&quot;garagem&quot; 003791E8 H:n</dc:description>
  <rdf:type rdf:resource="&ontoge;Nome"/>
```

```
</owl:Thing>

<owl:Class rdf:about="&ontoge;Verbo"/>

<owl:Class rdf:about="&ontoge;MV_Indicativo"/>

<owl:Class rdf:about="&ontoge;TV_INDI_Preterito_Perfeito_Simples"/>

<owl:Thing rdf:about="#deixar_deixou_1">
  <rdfs:label>deixou</rdfs:label>
  <dc:description>&quot;deixou&quot; 00901390 P:v-fin</dc:description>
  <rdf:type rdf:resource="&ontoge;Verbo"/>
  <rdf:type rdf:resource="&ontoge;MV_Indicativo"/>
  <rdf:type rdf:resource="&ontoge;TV_INDI_Preterito_Perfeito_Simples"/>
</owl:Thing>

<owl:Thing rdf:about="#a_garagem">
  <rdfs:label>a garagem</rdfs:label>
  <dc:description>&quot;a garagem&quot; ART_a garagem</dc:description>
  <rdf:type rdf:resource="&ontoge;Nome"/>
</owl:Thing>

<owl:ObjectProperty rdf:about="#V_"/>

<owl:ObjectProperty rdf:about="#V_deixar">
    <rdfs:subPropertyOf rdf:resource="#V_"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#V_INDI_PRET_deixar_deixou">
    <rdfs:subPropertyOf rdf:resource="#V_deixar"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#REF_a_garagem"/>

<owl:ObjectProperty rdf:about="#REF_jose"/>

<owl:ObjectProperty rdf:about="#PREP_em">
    <rdfs:label>em</rdfs:label>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:about="#ART_a">
    <rdfs:label>a</rdfs:label>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:about="#O_INDI_PRET_deixar_deixou_1">
    <owl:propertyChainAxiom rdf:parseType="Collection">
        <rdf:Description rdf:about="#V_INDI_PRET_deixar_deixou"/>
        <rdf:Description rdf:about="#REF_jose"/>
    </owl:propertyChainAxiom>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#O_INDI_PRET_deixar_deixou_2">
    <owl:propertyChainAxiom rdf:parseType="Collection">
        <rdf:Description rdf:about="#V_INDI_PRET_deixar_deixou"/>
        <rdf:Description rdf:about="#PREP_em"/>
    </owl:propertyChainAxiom>
</owl:ObjectProperty>

<owl:Thing rdf:about="#maria">
  <nectar:V_ rdf:resource="#deixar_deixou_1"/>
</owl:Thing>
<owl:Thing rdf:about="#maria">
  <nectar:V_deixar rdf:resource="#deixar_deixou_1"/>
</owl:Thing>
<owl:Thing rdf:about="#maria">
  <nectar:V_INDI_PRET_deixar_deixou rdf:resource="#deixar_deixou_1"/>
</owl:Thing>
<owl:Thing rdf:about="#garagem">
  <nectar:REF_a_garagem rdf:resource="#a_garagem"/>
</owl:Thing>
```

```
  <owl:Thing rdf:about="#deixar_deixou_1">
    <nectar:PREP_em rdf:resource="#a_garagem"/>
  </owl:Thing>
  <owl:Thing rdf:about="#deixar_deixou_1">
    <nectar:REF_jose rdf:resource="#jose"/>
  </owl:Thing>

</rdf:RDF>

<!-- Generated by Ontoge (version 0.02) -->
```

G.2 Ontologia Resultante em OWL/XML

```
<?xml version="1.0"?>

<!DOCTYPE Ontology [
    <!ENTITY Garagem "http://ontoge.org/ontology/test/Garagem.owl#" >
    <!ENTITY Joao "http://ontoge.org/ontology/test/Joao.owl#" >
    <!ENTITY Maria "http://ontoge.org/ontology/test/Maria.owl#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY ontoge "http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#" >
    <!ENTITY nectar "http://ontoge.org/ontology/test/nectar.owl#" >
]>

<Ontology xmlns="http://www.w3.org/2006/12/owl2-xml#"
      xml:base="http://www.w3.org/2006/12/owl2-xml#"
      xmlns:Garagem="http://ontoge.org/ontology/test/Garagem.owl#"
      xmlns:Joao="http://ontoge.org/ontology/test/Joao.owl#"
      xmlns:Maria="http://ontoge.org/ontology/test/Maria.owl#"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns:ontoge="http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl#"
      xmlns:nectar="http://ontoge.org/ontology/test/nectar.owl#"
      URI="http://ontoge.org/ontology/test/nectar.owl">

  <Import>http://ontoge.org/ontology/ontoge/0.02/por-2010-03-05/base.owl</Import>

  <Annotation annotationURI="&dc;creator">
    <Constant>Ontoge 0.02</Constant>
  </Annotation>

    <SubClassOf>
        <Class URI="#Homem"/>
        <Class URI="#Pessoa"/>
    </SubClassOf>
    <EquivalentClasses>
        <Class URI="#Mae"/>
        <ObjectIntersectionOf>
            <Class URI="#Mulher"/>
            <ObjectSomeValuesFrom>
                <ObjectProperty URI="#temFilho"/>
                <Class URI="#Pessoa"/>
            </ObjectSomeValuesFrom>
        </ObjectIntersectionOf>
    </EquivalentClasses>
    <SubClassOf>
        <Class URI="#Mulher"/>
```

```
    <Class URI="#Pessoa" />
</SubClassOf>
<ClassAssertion>
    <Class URI="#Homem" />
    <Individual URI="#jose" />
</ClassAssertion>
<ClassAssertion>
    <Class URI="&owl;Thing" />
    <Individual URI="#jose" />
</ClassAssertion>
<ClassAssertion>
    <Class URI="#Mulher" />
    <Individual URI="#maria" />
</ClassAssertion>
<ClassAssertion>
    <Class URI="&owl;Thing" />
    <Individual URI="#maria" />
</ClassAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty URI="#temFilho" />
    <Individual URI="#maria" />
    <Individual URI="#jose" />
</ObjectPropertyAssertion>
<EquivalentClasses>
    <Class URI="#Empregado" />
    <ObjectIntersectionOf>
        <Class URI="#Pessoa" />
        <ObjectSomeValuesFrom>
            <ObjectProperty URI="#trabalhaEm" />
            <Class URI="#Empresa" />
        </ObjectSomeValuesFrom>
    </ObjectIntersectionOf>
</EquivalentClasses>
<SubClassOf>
    <Class URI="#Empregado" />
    <Class URI="#Pessoa" />
</SubClassOf>
<SubClassOf>
    <Class URI="#Homem" />
    <Class URI="#Pessoa" />
</SubClassOf>
<EquivalentClasses>
    <Class URI="#Pai" />
    <ObjectIntersectionOf>
        <Class URI="#Homem" />
        <ObjectSomeValuesFrom>
            <ObjectProperty URI="#temFilho" />
            <Class URI="#Pessoa" />
        </ObjectSomeValuesFrom>
    </ObjectIntersectionOf>
</EquivalentClasses>
<SubClassOf>
    <Class URI="#Pai" />
    <Class URI="#Homem" />
</SubClassOf>
<ClassAssertion>
    <Class URI="#Homem" />
    <Individual URI="#joao" />
</ClassAssertion>
<ClassAssertion>
    <Class URI="&owl;Thing" />
    <Individual URI="#joao" />
</ClassAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty URI="#temFilho" />
    <Individual URI="#joao" />
    <Individual URI="#jose" />
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty URI="#trabalhaEm" />
```

```
        <Individual URI="#joao"/>
        <Individual URI="#petrobras"/>
    </ObjectPropertyAssertion>
    <ClassAssertion>
        <Class URI="#Homem"/>
        <Individual URI="#jose"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="&owl;Thing"/>
        <Individual URI="#jose"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="#Empresa"/>
        <Individual URI="#petrobras"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="&owl;Thing"/>
        <Individual URI="#petrobras"/>
    </ClassAssertion>
    <EntityAnnotation>
        <Individual URI="#petrobras"/>
        <Annotation annotationURI="&dc;description">
            <Constant>Petrobr&#225;s S.A.</Constant>
        </Annotation>
    </EntityAnnotation>
    <SubClassOf>
        <Class URI="#Local_Estranho"/>
        <Class URI="#Local"/>
    </SubClassOf>
    <ClassAssertion>
        <Class URI="#Local_Estranho"/>
        <Individual URI="#a_garagem"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class URI="&owl;Thing"/>
        <Individual URI="#a_garagem"/>
    </ClassAssertion>
  <Declaration>
    <Individual URI="&nectar;maria"/>
  </Declaration>
  <EntityAnnotation>
    <Individual URI="&nectar;maria"/>
    <Annotation annotationURI="&rdfs;label">
      <Constant>Maria</Constant>
    </Annotation>
  </EntityAnnotation>
  <EntityAnnotation>
    <Individual URI="&nectar;maria"/>
    <Annotation annotationURI="&dc;description">
      <Constant>&quot;Maria&quot; 003735D0 S:prop</Constant>
    </Annotation>
  </EntityAnnotation>
  <ClassAssertion>
    <Class URI="&ontoge;Nome"/>
    <Individual URI="&nectar;maria"/>
  </ClassAssertion>

  <Declaration>
    <Individual URI="&nectar;jose"/>
  </Declaration>
  <EntityAnnotation>
    <Individual URI="&nectar;jose"/>
    <Annotation annotationURI="&rdfs;label">
      <Constant>Jos&#233;</Constant>
    </Annotation>
  </EntityAnnotation>
  <EntityAnnotation>
    <Individual URI="&nectar;jose"/>
    <Annotation annotationURI="&dc;description">
      <Constant>&quot;Jos&#233;&quot; 00901658 Od:prop</Constant>
```

```
  </Annotation>
</EntityAnnotation>
<ClassAssertion>
  <Class URI="&ontoge;Nome"/>
  <Individual URI="&nectar;jose"/>
</ClassAssertion>

<Declaration>
  <Individual URI="&nectar;garagem"/>
</Declaration>
<EntityAnnotation>
  <Individual URI="&nectar;garagem"/>
  <Annotation annotationURI="&rdfs;label">
    <Constant>garagem</Constant>
  </Annotation>
</EntityAnnotation>
<EntityAnnotation>
  <Individual URI="&nectar;garagem"/>
  <Annotation annotationURI="&dc;description">
    <Constant>&quot;garagem&quot; 003791E8 H:n</Constant>
  </Annotation>
</EntityAnnotation>
<ClassAssertion>
  <Class URI="&ontoge;Nome"/>
  <Individual URI="&nectar;garagem"/>
</ClassAssertion>

<Declaration>
  <Individual URI="&nectar;deixar_deixou_1"/>
</Declaration>
<EntityAnnotation>
  <Individual URI="&nectar;deixar_deixou_1"/>
  <Annotation annotationURI="&rdfs;label">
    <Constant>deixou</Constant>
  </Annotation>
</EntityAnnotation>
<EntityAnnotation>
  <Individual URI="&nectar;deixar_deixou_1"/>
  <Annotation annotationURI="&dc;description">
    <Constant>&quot;deixou&quot; 00901390 P:v-fin</Constant>
  </Annotation>
</EntityAnnotation>
<ClassAssertion>
  <Class URI="&ontoge;Verbo"/>
  <Individual URI="&nectar;deixar_deixou_1"/>
</ClassAssertion>
<ClassAssertion>
  <Class URI="&ontoge;MV_Indicativo"/>
  <Individual URI="&nectar;deixar_deixou_1"/>
</ClassAssertion>
<ClassAssertion>
  <Class URI="&ontoge;TV_INDI_Preterito_Perfeito_Simples"/>
  <Individual URI="&nectar;deixar_deixou_1"/>
</ClassAssertion>

<Declaration>
  <Individual URI="&nectar;a_garagem"/>
</Declaration>
<EntityAnnotation>
  <Individual URI="&nectar;a_garagem"/>
  <Annotation annotationURI="&rdfs;label">
    <Constant>a garagem</Constant>
  </Annotation>
</EntityAnnotation>
<EntityAnnotation>
  <Individual URI="&nectar;a_garagem"/>
  <Annotation annotationURI="&dc;description">
    <Constant>&quot;a garagem&quot; ART_a garagem</Constant>
  </Annotation>
</EntityAnnotation>
```

```
<ClassAssertion>
  <Class URI="&ontoge;Nome"/>
  <Individual URI="&nectar;a_garagem"/>
</ClassAssertion>

<Declaration>
  <ObjectProperty URI="&nectar;V_"/>
</Declaration>

<Declaration>
  <ObjectProperty URI="&nectar;V_deixar"/>
</Declaration>
<SubObjectPropertyOf>
  <ObjectProperty URI="&nectar;V_deixar"/>
  <ObjectProperty URI="&nectar;V_"/>
</SubObjectPropertyOf>

<Declaration>
  <ObjectProperty URI="&nectar;V_INDI_PRET_deixar_deixou"/>
</Declaration>
<SubObjectPropertyOf>
  <ObjectProperty URI="&nectar;V_INDI_PRET_deixar_deixou"/>
  <ObjectProperty URI="&nectar;V_deixar"/>
</SubObjectPropertyOf>

<Declaration>
  <ObjectProperty URI="&nectar;REF_a_garagem"/>
</Declaration>

<Declaration>
  <ObjectProperty URI="&nectar;REF_jose"/>
</Declaration>

<Declaration>
  <ObjectProperty URI="&nectar;PREP_em"/>
</Declaration>
<EntityAnnotation>
  <ObjectProperty URI="&nectar;PREP_em"/>
  <Annotation annotationURI="&rdfs;label">
    <Constant>em</Constant>
  </Annotation>
</EntityAnnotation>

<Declaration>
  <DataProperty URI="&nectar;ART_a"/>
</Declaration>
<EntityAnnotation>
  <DataProperty URI="&nectar;ART_a"/>
  <Annotation annotationURI="&rdfs;label">
    <Constant>a</Constant>
  </Annotation>
</EntityAnnotation>

<Declaration>
  <ObjectProperty URI="&nectar;O_INDI_PRET_deixar_deixou_1"/>
</Declaration>
<SubObjectPropertyOf>
  <SubObjectPropertyChain>
    <ObjectProperty URI="&nectar;V_INDI_PRET_deixar_deixou"/>
    <ObjectProperty URI="&nectar;REF_jose"/>
  </SubObjectPropertyChain>
  <ObjectProperty URI="&nectar;O_INDI_PRET_deixar_deixou_1"/>
</SubObjectPropertyOf>

<Declaration>
  <ObjectProperty URI="&nectar;O_INDI_PRET_deixar_deixou_2"/>
</Declaration>
<SubObjectPropertyOf>
  <SubObjectPropertyChain>
    <ObjectProperty URI="&nectar;V_INDI_PRET_deixar_deixou"/>
```

```
      <ObjectProperty URI="&nectar;PREP_em"/>
    </SubObjectPropertyChain>
    <ObjectProperty URI="&nectar;O_INDI_PRET_deixar_deixou_2"/>
  </SubObjectPropertyOf>

  <ObjectPropertyAssertion>
    <ObjectProperty URI="&nectar;V_"/>
    <Individual URI="&nectar;maria"/>
    <Individual URI="&nectar;deixar_deixou_1"/>
  </ObjectPropertyAssertion>

  <ObjectPropertyAssertion>
    <ObjectProperty URI="&nectar;V_deixar"/>
    <Individual URI="&nectar;maria"/>
    <Individual URI="&nectar;deixar_deixou_1"/>
  </ObjectPropertyAssertion>

  <ObjectPropertyAssertion>
    <ObjectProperty URI="&nectar;V_INDI_PRET_deixar_deixou"/>
    <Individual URI="&nectar;maria"/>
    <Individual URI="&nectar;deixar_deixou_1"/>
  </ObjectPropertyAssertion>

  <ObjectPropertyAssertion>
    <ObjectProperty URI="&nectar;REF_a_garagem"/>
    <Individual URI="&nectar;garagem"/>
    <Individual URI="&nectar;a_garagem"/>
  </ObjectPropertyAssertion>

  <ObjectPropertyAssertion>
    <ObjectProperty URI="&nectar;PREP_em"/>
    <Individual URI="&nectar;deixar_deixou_1"/>
    <Individual URI="&nectar;a_garagem"/>
  </ObjectPropertyAssertion>

  <ObjectPropertyAssertion>
    <ObjectProperty URI="&nectar;REF_jose"/>
    <Individual URI="&nectar;deixar_deixou_1"/>
    <Individual URI="&nectar;jose"/>
  </ObjectPropertyAssertion>

</Ontology>

<!-- Generated by Ontoge (version 0.02) -->
```

Índice Remissivo

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Geração Automática de Ontologias para a Web Semântica

Carlos de Oliveira Bravo

Brasília
2010