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

Proveniência de Dados em Workflows de Bioinformática

Renato de Paula

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Maristela Terto de Holanda

Brasília
2012

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

Coordenadora: Prof.ª Dr.ª Mylene Christine Queiroz de Farias

Banca examinadora composta por:

Prof.ª Dr.ª Maristela Terto de Holanda (Orientadora) — CIC/UnB
Prof.ª Dr.ª Maria Emilia M. Telles Walter — CIC/UnB
Prof. Dr. Sérgio Lifschitz — Departamento de Informática/PUC-RIO

CIP — Catalogação Internacional na Publicação

de Paula, Renato.

Proveniência de Dados em Workflows de Bioinformática / Renato de Paula. Brasília : UnB, 2012.
205 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2012.

1. Proveniência de Dados, 2. Bioinformática, 3. Modelo de Dados, 4. PROV-DM.

CDU 004.4

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

Proveniência de Dados em Workflows de Bioinformática

Renato de Paula

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof.ª Dr.ª Maristela Terto de Holanda (Orientadora)
CIC/UnB

Prof.ª Dr.ª Maria Emilia M. Telles Walter
CIC/UnB

Prof. Dr. Sérgio Lifschitz
Departamento de Informática/PUC-RIO

Prof.ª Dr.ª Mylene Christine Queiroz de Farias
Coordenadora da Pós-Graduação em Informática

Brasília, 11 de julho de 2012

Dedicatória

Dedico este trabalho à minha esposa Veronice que esteve carinhosamente ao meu lado nos momentos mais importantes da minha vida. Também ao meu filho Estevan, que muitas lições de vida tem me ensinado. Também a minha mãe Neusa e meu padrasto Fernando, por terem me incentivado a estudar continuamente e por me apoiarem em momentos importantes. Aos meus avós Maria Augusta, Damasceno, Iracema e Severino (em memória) que, apesar das dificuldades da vida, construíram suas famílias e nos dão valiosas lições de persistência e perseverança.

Agradecimentos

Em especial a Deus, por cuidar de cada um de nós, dando saúde, serenidade e, até mesmo, aquilo que nem sabemos que precisamos. À minha esposa Veronice e meu filho Estevan que estão ao meu lado todos os dias, dando força e alegria para vencer cada novo desafio. Ao meu sogro Bertoldo e minha sogra Irmgard, por terem acreditado em mim e pelo apoio nas horas difíceis. Aos meus irmãos Evandro, Márcio e Débora, que sempre me apoiaram e me incentivaram a novos desafios. À professora Maristela, pelo apoio, orientação e dedicação na construção desse trabalho. À professora Maria Emília, por compartilhar sua experiência conosco e pelas valiosas contribuições para o desenvolvimento desse trabalho. À professora Tainá, que nos ajudou com seus conhecimentos em biologia. Ao colega Ruben, por seu companheirismo, apoio, por sua contribuição na construção do artigo e com a execução dos experimentos. Ao colega João, pelas importantes informações do experimento do estudo de caso. Aos colegas Gisele, Fabiana, Lucinéia e José Luiz, pela amizade e companheirismo que tornaram menos árida a nossa caminhada. Também ao meu amigo e colega de trabalho Paulo Alexandre, que entre "resmungos", sapiência e orientação a objeto, contribuiu grandemente para a construção do simulador *Provenance*. Aos meus antigos professores, em especial Marcos Macedo e Elisiane, por me ensinarem a importância do conhecimento. Aos meus grandes amigos André, Mariane, Gustavo, Luciana, Paulo Sergio e Mylene, por compartilharem sua alegria e por me apoiarem nesta empreitada. Por fim, aos demais professores, familiares, amigos, colegas de trabalho e da UnB, pelas palavras de apoio e incentivo em cada passo desse trabalho.

Resumo

Avanços tecnológicos, tanto em equipamentos quanto em algoritmos, têm tornado a execução de experimentos científicos cada vez mais rápida e eficiente. Isso permite que os cientistas executem mais experimentos e possam compará-los entre si, o que traz maior acurácia às análises. Porém, a quantidade de dados que devem ser tratados aumenta a cada novo experimento executado, o que dificulta a identificação da origem dos dados e como os mesmos foram transformados em cada experimento. Assim, tem-se a necessidade de novas ferramentas que tornem possível preservar, não só as conclusões de um experimento científico, mas também a origem dos dados utilizados e as condições e parâmetros com os quais foram executados. Estudos recentes mostram que a utilização de modelos de proveniência de dados facilita o gerenciamento dos dados tanto em ambiente científico quanto naqueles disponibilizados pela internet. Uma importante área para o uso de proveniência de dados é o da bioinformática, principalmente em projetos genoma e transcritoma de alto desempenho, visto que seus experimentos geram grande volume de dados e seus processos podem ser executados diversas vezes com diferentes ferramentas, dados e parâmetros. Neste trabalho propomos a utilização de uma estrutura de proveniência de dados baseada no modelo PROV-DM para experimentos em projetos de bioinformática a fim de permitir que os cientistas possam trabalhar com seus experimentos em detalhes e, quando necessário, possam consultá-los e reexecutá-los de forma mais planejada e controlada.

Palavras-chave: Proveniência de Dados, Bioinformática, Modelo de Dados, PROV-DM.

Abstract

Technological Advances, both in equipment and algorithms, have made the execution of scientific experiments increasingly faster and more efficient. This allows scientists to execute more experiments and compare them, generating greater accuracy in analyses. However, the great quantity of data to be treated increases with each new experiment performed, which makes it difficult to identify the origin of data and how they were transformed in each experiment. Thus, there is a pressing need for new tools that make possible the preservation of, not only conclusions of scientific experiments, but also the origin of data used and the conditions and parameters with which each were performed. Recent studies show that the use of data provenance models facilitates the management of data, both in the scientific environment and those available on the internet. An important area for the use of data provenance is in bioinformatics, mainly in genome and high performance transcriptome projects, since these experiments generate a large volume of data and their process can be executed many times with different tools, data and parameters. In this work we propose the use of a data provenance structure based on the model PROV-DM for experiments in bioinformatics projects with the objective of allowing scientists to work with their experiments in fine detail, and, when necessary, consult them or re-execute them in a more planned and controlled way.

Keywords: Data Provenance, Bioinformatic, Data Model, PROV-DM.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A utilização de ferramentas computacionais vem se tornando imprescindível na realização de diversos tipos de experimentos científicos, principalmente naqueles que precisam processar grandes volumes de dados e exigem grande capacidade de processamento. À medida que novos equipamentos e aplicativos surgem para apoiar estes experimentos, estes crescem em tamanho e complexidade criando novos desafios para seus usuários. Não basta ser capaz de realizar tais experimentos, mas também há a necessidade de gerenciar suas execuções e seus resultados. Novas descobertas não surgem apenas da execução de novos experimentos, mas também da análise sistemática dos experimentos já executados.

Neste contexto, a proveniência de dados tem se tornado importante ferramenta de auxílio aos pesquisadores para o gerenciamento da execução de seus experimentos científicos. A proveniência de dados apóia os resultados de um experimento porque visa descrever os acontecimentos e insumos utilizados na geração de uma determinada informação. Segundo Buneman, Khanna e Tan (2001) proveniência de dados é *"... a descrição das origens de uma peça de dados e do processo pelo qual ela chegou em um banco de dados."* (tradução livre), ou seja, para garantir a proveniência dos dados se faz necessário guardar tanto a origem dos dados utilizados como matéria-prima, quanto os processos que transformaram esses dados no produto final.

Dessa forma, a proveniência de dados vem se tornando cada vez mais presente no ambiente científico, tanto para garantir a origem dos dados como para avaliar a sua acurácia. Pode-se citar diversos estudos com o objetivo de descrever detalhadamente proveniência de dados (TAN, 2004; CHENEY, 2009), classificar suas características (GLAVIC; DITTRICH, 2007), além de descrever aplicações práticas (ALDECO-PÉREZ; MOREAU, 2008; ORLANDI; PASSANT; CHAMPIN, 2010).

Porém, antes de definir quais informações são necessárias para garantir a proveniência e como serão gerenciadas, é preciso definir uma forma de organizá-las a fim de que se possa, posteriormente, recuperá-las e entendê-las. Com esse objetivo, encontra-se na literatura a definição de diversos modelos de proveniência de dados tais como *Provenir Ontology* (SAHOO; SHETH, 2009), *W7 Model* (RAM; LIU, 2007), *Provenance Vocabulary* (HARTIG; ZHAO, 2010), *Open Provenance Model* (MOREAU et al., 2010), PROV-DM (W3C, 2012c), entre outros. Estes modelos têm como objetivo tornar mais clara a idéia de proveniência de dados além de criar estruturas práticas para a organização e gerenciamento dos dados de proveniência. Neste contexto surgiu o conceito de Sistemas Gerenciadores de Proveniência (SGP) (MOREAU et al., 2008) voltados ao armazenamento, recuperação e gerenciamento

de dados de proveniência relacionados aos experimentos executados.

Grandes avanços tem sido obtidos nos estudos sobre proveniência desde a execução do *workshop* chamado *"The First Provenance Challenge"* ocorrido em 2006. A partir das conclusões obtidas neste *workshop* foi elaborado o texto entitulado *"Special Issue: The First Provenance Challenge"* (MOREAU et al., 2008) que serviu como base para mais três *workshops*, realizados especificamente para o tratamento de questões de proveniência de dados. No terceiro *workshop* chamado *"Third Provenance Challenge"*, ocorrido em 2009, foi avaliada a primeira versão do modelo *Open Provenance Model* (OPM), hoje na versão 1.1 (MOREAU et al., 2010) e utilizado em diversas aplicações como (CHAPMAN; BLAUSTEIN; ELSAESSER, 2010; BRAUN et al., 2010; CAO et al., 2008; GOMES, 2011; PAULA et al., 2012), entre outros. No quarto *workshop*, realizado em 2010, surgiu a iniciativa da criação do *W3C Provenance Working Group* que propiciou a proposta do modelo PROV-DM (W3C, 2012c), uma ampliação do modelo OPM. Apesar de ter sua primeira versão ainda em desenvolvimento, o modelo PROV-DM já se apresenta bastante desenvolvido e fornece condições de representar a proveniência de dados de forma mais detalhada se comparado aos outros modelos. Além disso, o modelo PROV-DM deve se tornar uma recomendação do W3C.

Por outro lado, projetos de bioinformática compreendem uma importante área de aplicação para SGP. Tais projetos consistem na execução de diversos experimentos com sequências de DNA ou RNA obtidas por sequenciadores de alto desempenho tais como Illumina (BENTLEY, 2006) ou 454/Roche (ROTHBERG; LEAMON, 2008). Estes equipamentos são capazes de gerar, em poucas horas, milhões de fragmentos de DNA (SCHUSTER, 2008) que são analisados com os mais diversos objetivos. Para tal são executados diferentes tipos de processos, envolvendo diferentes aplicativos, gerando terabytes de dados, que são armazenados em diferentes arquivos em diversos formatos.

Dessa forma, projetos de bioinformática utilizam diferentes ferramentas computacionais cujas configurações e parâmetros de entrada podem afetar fortemente os resultados obtidos. Portanto, os SGPs podem auxiliar projetos de bioinformática fornecendo a capacidade de armazenar e reconstituir cada passo executado em cada um dos experimentos. Tal reconstituição pode ser bastante detalhada, incluindo ambiente computacional, programas e suas versões e parâmetros utilizados, entre outros. Isso permite que, conhecendo-se a proveniência dos dados gerados em experimentos anteriores, estes dados podem ser reutilizados com um grau maior de confiabilidade em experimentos posteriores, além de permitir que os usuários possam rever conclusões e chegar a novas descobertas.

Um dos desafios de um SGP para o gerenciamento de proveniência de dados em projetos de bioinformática é que a diversidade de programas utilizados gera diferentes tipos de resultados, dependendo do tipo de análise que está sendo realizada no projeto. Por isso, armazenar e gerenciar essas informações, em formatos diversos depende de uma prévia padronização que seja flexível o suficiente para aceitar a grande maioria desses processos e formal o suficiente para permitir um gerenciamento adequado.

Neste trabalho, usou-se o modelo PROV-DM para o gerenciamento da proveniência. Alguns aspectos foram considerados para a aplicação desse modelo em projetos de bioinformática. O primeiro deles diz respeito à utilização dos próprios arquivos, originais e gerados em uma execução particular do *workflow*, para garantir a proveniência em um nível de granularidade mais fino, evita-se dessa forma a duplicação de dados. Outros aspectos importantes referem-se à simplicidade de uso e à independência de outras ferra-

mentas tais como Sistemas Gerenciadores de Workflow Científicos (SGWfC) e Sistemas Gerenciadores de Bancos de Dados (SGBD), por exemplo.

A fim de demonstrar a viabilidade da aplicação do modelo, parte das definições foi implementada em um simulador chamado *"Provenance"*, validado com informações reais de projetos de bioinformática executados no Laboratório de Biologia Molecular (BIOMOL) do Departamento de Biologia Celular (CEL) da Universidade de Brasília (UnB).

1.1 Problema

Projetos de bioinformática podem ser suportados por *workflows* que, por sua vez, podem ser executados várias vezes com parâmetros diferentes. Os resultados de cada execução são fortemente afetados tanto pelos dados quanto pelos programas e parâmetros utilizados. As saídas de cada passo do *workflow* não são registradas e não são mantidos históricos dessas execuções de forma automática.

1.2 Objetivos

O objetivo principal deste trabalho é gerenciar a proveniência de dados em *workflows* de projetos de bioinformática baseada no modelo PROV-DM. Este modelo deve evitar a duplicação de dados em diferentes estruturas, mantendo os arquivos originais disponíveis e causando o mínimo impacto em relação ao espaço de armazenamento. A implementação dessa solução deve ser multiplataforma, mantendo independência de outros programas tais como SGWfC, por exemplo.

Como objetivos específicos desse trabalho tem-se:

- Adotar o modelo PROV-DM para gerenciar a proveniência de dados em projetos de bioinformática;
- Implementar a proveniência de dados em um simulador;
- Realizar estudos de caso com dados reais;
- Avaliar os resultados obtidos em relação ao impacto da inclusão dos dados de proveniência.

1.3 Estrutura do Trabalho

Esta dissertação está estruturada em seis capítulos descritos a seguir.

O Capítulo 2 descreve conceitos importantes em Biologia e projetos de bioinformática, tais como principais fases de *workflows*, programas executados e dados utilizados e gerados durante os experimentos.

No Capítulo 3, são descritos os conceitos de proveniência de dados. Também são apresentados alguns modelos de proveniência de dados em mais detalhes e, no final do capítulo, encontra-se uma análise comparativa entre esses modelos.

No Capítulo 4 é proposto uma aplicação do modelo PROV-DM para projetos de bioinformática, a fim de garantir a proveniência de dados.

O Capítulo 5 descreve a aplicação do modelo PROV-DM utilizando o simulador *Provenance*, desenvolvido neste trabalho, com informações de dois estudos de caso de bioinformática com dados reais.

Por fim, o Capítulo 6 contém as conclusões deste trabalho e sugere trabalhos futuros.

Capítulo 2

Projetos de Bioinformática

Neste capítulo são apresentados conceitos relacionados aos projetos de bioinformática, especificamente os que envolvem *workflows* de sequenciamento de DNA ou RNA de alto desempenho. A Seção 2.1 traz uma breve história do surgimento dos estudos genômicos. A Seção 2.2 descreve DNA e RNA. A Seção 2.3 detalha projetos de bioinformática, principalmente os relacionados ao sequenciamento de DNA e RNA de alto desempenho.

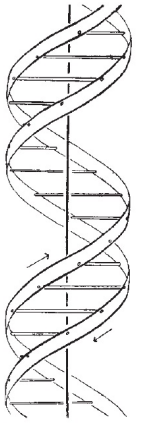
2.1 Uma Breve História

As pesquisas com DNA iniciaram na década de 1860, quando o bioquímico alemão Johann Friedrich Miescher (DAHM, 2008) descobriu compostos de natureza ácida no núcleo das células e deu o nome de "nucleína". Estes compostos são conhecidos como bases hidrogenadas e são os principais elementos que compõem o DNA. Mais de 60 anos depois, em 1929, o bioquímico lituano Phoebus Aaron Theodore Levene publicou o artigo *"The Structure of Thymonucleic Acid"* (LEVENE; LONDON, 1929), demonstrando como as quatro bases Adenina, Citosina, Guanina e Timina eram ligadas para formar o DNA.

Porém, a corrida para o mapeamento do DNA só foi iniciada a partir de 1953, quando os cientistas Watson e Crick descreveram a estrutura do DNA chamada "dupla hélice" (WATSON; CRICK, 1953) (Figura 2.1). Ainda se passariam 24 anos até que o primeiro organismo tivesse seu DNA completamente mapeado, o que foi realizado em 1977 por um grupo liderado pelo cientista Frederick Sanger (SANGER et al., 1977). O organismo mapeado foi o bacteriófago Phi-X174, cujo DNA é composto por cerca de 5000 bases, relativamente curto se comparado ao humano, com quase 3 bilhões de bases (SETUBAL; MEIDANIS, 1997).

Apesar dos avanços alcançados no mapeamento genético desde o descobrimento do DNA, o grande marco na história dos estudos genéticos aconteceu na década de 1990 quando iniciou-se o Projeto Genoma Humano. Este projeto tinha como objetivo mapear o DNA humano, feito este que somente foi conseguido em 2003, através da colaboração internacional de diversos laboratórios e instituições governamentais (HGSCI, Human Genome Sequencing Consortium International, 2004). Esse mapeamento foi feito utilizando o conhecido "método Sanger" (SETUBAL; MEIDANIS, 1997), desenvolvido por Frederick Sanger em 1975, porém amplamente utilizado até meados de 2004.

A partir de então, novas tecnologias tem sido desenvolvidas, diminuindo o tempo e reduzindo os custos de sequenciamento. Em 2004, novos equipamentos chamados sequen-



This figure is purely diagrammatic. The two ribbons symbolize the two phosphate—sugar chains, and the horizontal rods the pairs of bases holding the chains together. The vertical line marks the fibre axis

Figura 2.1: Dupla hélice de DNA (WATSON; CRICK, 1953).

ciadores de DNA de alto desempenho chegaram ao mercado, revolucionando os projetos de mapeamento de DNA (MARDIS, 2008). Entre eles pode-se citar o 454/Roche (ROTHBERG; LEAMON, 2008), que foi a primeira tecnologia após o Método Sanger a mapear um genoma humano, e o Illumina Genome Analyzer (BENTLEY, 2006). Esses equipamentos conseguem sequenciar milhares de fragmentos de DNA em poucas horas, além de garantir melhor qualidade em relação aos métodos anteriores.

2.2 DNA e RNA

O DNA, ou ácido desoxirribonucléico, é o código armazenado nas células de todos os seres vivos e que comanda o seu desenvolvimento e funcionamento. É formado pela sequência de nucleotídeos, cada um sendo uma das quatro bases nitrogenadas: Adenina (A), Citosina (C), Timina (T) e Guanina (G) (SETUBAL; MEIDANIS, 1997). O tamanho do DNA varia entre as espécies, sendo que os seres humanos possuem cerca de 3 bilhões de bases e, em geral, cada indivíduo tem DNA distinto dos demais, com raras exceções, como no caso de irmãos gêmeos idênticos, por exemplo.

Uma molécula de DNA forma uma dupla hélice, composta de duas fitas que possuem nucleotídeos complementares em cada fita, ou seja, se em uma fita existe um nucleotídeo Adenina, na outra encontra-se a Timina. O mesmo ocorre com os nucleotídeos Guanina e Citosina (Figura 2.2).

Outro fato importante é que, apesar do DNA de um indivíduo ser igual em todas as células do seu corpo, porções diferentes desse DNA podem ser expressas (transcritas) dependendo do local do organismo em que a célula se encontra (pêlos, rins, músculos, etc...). Além disso, essa expressão (transcrição) pode ser influenciada por fatores externos tais como infecções desenvolvidas, falta de alimentação, ingestão de medicamentos, entre outros. Assim, chama-se de DNA genômico ao DNA contido nas células de um indivíduo, e de transcritoma o conjunto dos DNA expressos (transcritos).

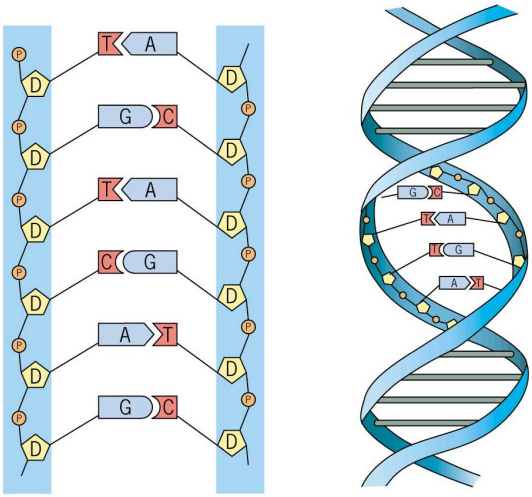


Figura 2.2: Representação da dupla hélice de DNA com as bases complementares em cada uma das fitas (JR; SASSON, 2002).

A formação de um transcrito ocorre quando uma enzima separa uma das fitas da hélice de DNA, que é copiada formando então o RNA. As partes do DNA copiadas dependem de regiões promotoras que indicam o início de um gene, a partir de onde o DNA é copiado. As moléculas de DNA e RNA são bastante semelhantes porém, algumas diferenças importantes devem ser citadas. A principal diferença é que o RNA possui a base Uracila (U) no lugar da Timina, presente no DNA. Além disso, o RNA não forma, em geral, uma dupla hélice, e possui funções variadas, dependendo do tipo de RNA. Uma finalidade importante do RNA é a produção de proteínas, necessárias no organismo de todos os seres vivos (SETUBAL; MEIDANIS, 1997).

O Dogma Central da Biologia Molecular (SETUBAL; MEIDANIS, 1997) (Figura 2.3) mostra a relação entre DNA, RNA e a produção de proteínas. A replicação é o processo que permite que o DNA seja replicado o que perpetua seu código dentro do organismo. O processo de transcrição cria o RNA, a partir de cópias de partes do DNA. O RNA, por sua vez, é utilizado para determinar quais proteínas devem ser produzidas através do processo de tradução. Existem diversos tipos de RNA dependendo da função executada por cada um. Os três principais tipos são: RNA Ribossômico (RNAr), RNA de Transferência (RNAt) e o RNA Mensageiro (RNAm).:

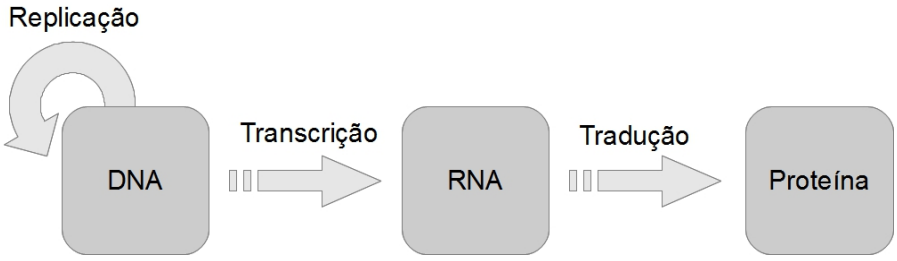


Figura 2.3: Os três processos que compõem o Dogma Central da Biologia Molecular.

2.3 Projetos de Bioinformática

A Informática fornece suporte para diversas áreas de pesquisa, tanto desenvolvendo modelos de dados como algoritmos. A Bioinformática surgiu da necessidade de utilização de ferramentas computacionais para as soluções de problemas da biologia tanto de processamento quanto de armazenamento e recuperação de dados.

Projetos de bioinformática estão ligados diretamente ao processamento de dados biológicos e, em geral, são relacionados aos processos de sequenciamento. Processos de sequenciamento de DNA ou RNA consistem na tarefa de descobrir, para um determinado organismo, qual é a sequência de bases que forma cada fragmento de DNA ou RNA que está sendo investigado. Assim, projetos genoma consistem em pesquisar DNA genômico (cromossomos), enquanto que projetos transcritoma consistem no estudo dos transcritos (RNA). A partir desses fragmentos, tanto de DNA quanto de RNA, diversos tipos de processos computacionais podem ser executados dependendo do objetivo final do projeto.

De forma geral, projetos genoma e transcritoma possuem suporte computacional, nos quais são projetados *workflows* que transformam fragmentos de entrada de tal forma a extrair destes informações como funções biológicas e localização dentro da célula. A Figura 2.4 mostra um exemplo de *workflow* dividido em três fases: filtragem, mapeamento/montagem e análise.

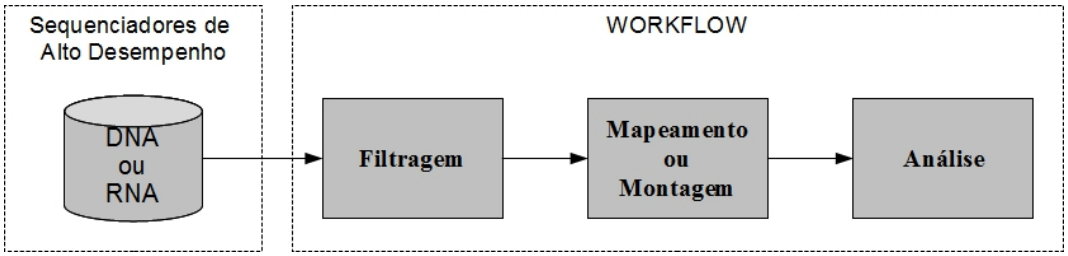


Figura 2.4: Exemplo de workflow para projetos genoma e transcritoma.

Inicialmente os biólogos realizam processos laboratoriais de coleta e replicação do material biológico. Depois esse material é enviado para o sequenciamento de pequenas porções do DNA ou RNA (dependendo do tipo do projeto), chamadas *reads*. O tamanho de cada *read* obtida depende do equipamento utilizado, podendo variar de 30 à 600 bases.

Estas *reads* geralmente tem um indicador de qualidade para cada base, o que pode ser usado para filtrar as *reads* com grande número de bases com qualidade baixa, ou seja, são aceitas para o projeto somente *reads* com certo grau de qualidade mínima. A etapa de filtragem pode também detectar contaminantes ou outros erros laboratoriais. A filtragem, por se tratar de uma tarefa simples, pode ser feita tanto por *scripts* desenvolvidos localmente pelo usuário, com características simples de filtragem, como também por programas mais completos com diversas possibilidades de uso. Como exemplo pode-se citar o pacote FASTX-Toolkit (HANNON, 2012). Esse pacote fornece programas para tratamentos de arquivos FASTA e FASTQ, comumente utilizados para armazenar *reads*. Entre esses tratamentos encontram-se filtros específicos e conversores de formato, muito úteis nessa fase. Os arquivos da fase de filtragem podem somar gigabytes de dados, incluídas as *reads* e informações adicionais, como o identificador para cada sequência e a qualidade de cada base. Esses dados constituem, em geral, parte considerável do espaço em disco ocupado em um experimento.

A fase de mapeamento tenta localizar, em um genoma de referência, as *reads* filtradas. O objetivo é encontrar o alinhamento para o maior número de *reads* possível e agrupá-las em porções maiores, que são analisados posteriormente. Essa técnica é usada para organismos que já possuem o seu DNA quase completamente sequenciado. A Figura 2.5 mostra um alinhamento de *reads*, que consiste em descobrir as suas posições em relação a um genoma de referência conhecido.

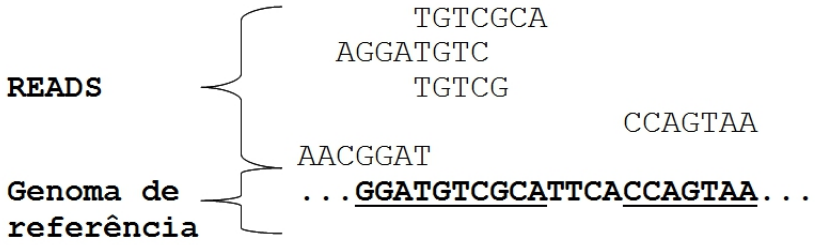


Figura 2.5: Exemplo do processo de mapeamento.

Quando não é conhecido o genoma de referência, pode-se alinhar as *reads* entre elas gerando sequências maiores chamadas *contigs*. Este processo é chamado de montagem. Na Figura 2.6 diversas *reads* são alinhadas e a sombra dessas *reads* mostra qual é a porção do cromossomo sequenciado.

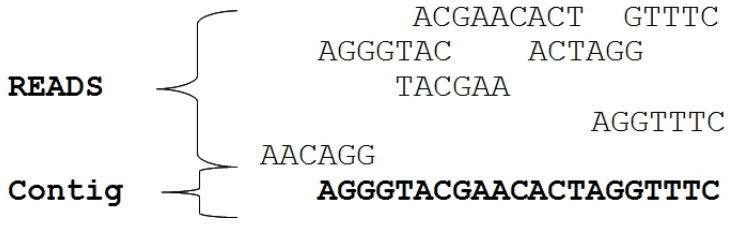


Figura 2.6: Exemplo do processo de montagem.

No caso de mapeamento de transcritos, a dificuldade ocorre a partir da existência de *splice junctions* nas *reads* sequenciadas. A Figura 2.7 mostra um exemplo do problema encontrado no processo de alinhamento de sequências de transcritos com *splice junctions*. Em (A) tem-se o processo de transcrição onde somente os *exons* são copiados para formar o transcrito. Em (B), quando a *read* com o transcrito é alinhada contra o genoma de referência, não encontra sua posição adequadamente porque existe um *intron* no genoma de referência. Quando isso ocorre, muitas *reads* podem ser descartadas por não encontrarem um alinhamento adequado.

Como pode ser percebido, a fase de mapeamento requer algoritmos mais complexos que a filtragem. Assim, diversos programas podem ser encontrados na literatura, oferecendo diferentes recursos e especialidades. O programa Bowtie (LANGMEAD et al., 2009), por exemplo, apresenta bons resultados com o mapeamento de *reads* pequenas. Além disso foi desenvolvido para tratamento eficiente de memória e foi projetado para utilizar mais de um processador durante o alinhamento, duas importantes características para o processo de alinhamento de grandes quantidades de *reads*. Outro exemplo é o TopHat (TRAPNELL; PACHTER; SALZBERG, 2009) que, apesar de utilizar o mesmo algoritmo do Bowtie, possui recursos específicos para o tratamento das *splice junctions*, tais como: identificar as

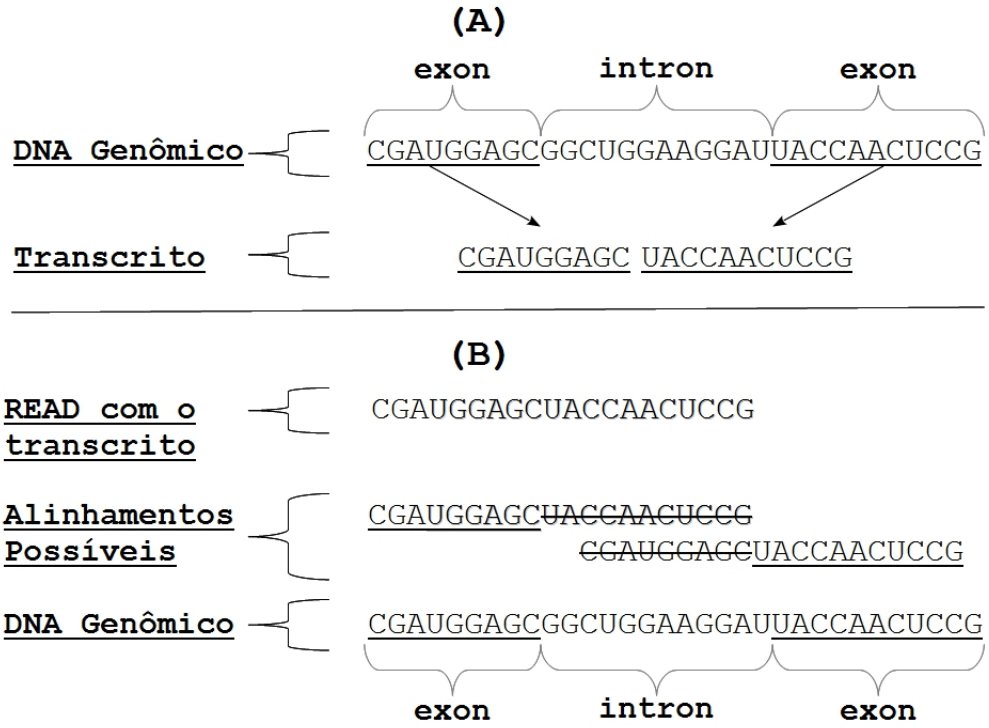


Figura 2.7: Exemplo do processo de transcrição (A) e do problema de alinhamento com *splice junctions* (B).

uniões *exon-exon* e criar um banco de dados para ser utilizado durante o alinhamento. Outro programa que também pode ser utilizado é o CUFFLINKS (ROBERTS et al., 2011), especializado em alinhamento de sequências de RNA.

A última fase do sequenciamento é a fase de análise. Nesta fase tem-se grandes porções do DNA ou RNA mapeadas que podem ser analisadas por diferentes processos dependendo do objetivo do experimento. Como exemplo pode-se citar a busca pela diferença da expressão genética entre humanos suscetíveis e não suscetíveis à infecção de um determinado vírus. Neste caso faz-se o mapeamento de diversas amostras de RNA de indivíduos suscetíveis e não suscetíveis, e na fase de anotação compara-se os transcritomas encontrados a fim de descobrir quais trechos expressos do DNA podem estar causando a deficiência dos indivíduos suscetíveis ou dando imunidade aos indivíduos não suscetíveis. A partir daí pode-se criar, por exemplo, formas de tratamento que ativem ou desativem a transcrição dessas porções de DNA.

Um exemplo de programa utilizado na fase de análise que pode ser citado é o R (R Development Core Team, 2009), que possui diversas ferramentas para análise estatística. Outro programa bastante utilizado nesta fase é o BLAST (ALTSCHUL et al., 1990) utilizado para descobrir a existência de um gene em um determinado genoma e a sua funcionalidade.

2.3.1 Formatos de Arquivos

Assim como existem diversos programas que podem ser usados em cada fase do processo de sequenciamento, também existem diferentes formatos de arquivos envolvidos. O tipo de arquivo depende do programa utilizado e da necessidade de cada fase. A seguir são discutidas algumas questões relacionadas aos tipos de arquivos mais comumente utilizados.

A fase de filtragem, conforme pode ser vista na Figura 2.4, trata os dados gerados pelos sequenciadores de alto desempenho, que são gravados em arquivos com as sequências de DNA ou RNA e que variam de formato conforme o equipamento utilizado. Pode-se citar como exemplo o formato *Standard Flowgram Format* (SFF) (NCBI, 2011) gerado pelo sequenciador 454/Roche, que armazena tanto as *reads* obtidas como sua qualidade e dados relevantes sobre o processo executado. Os arquivos SFF são codificados no formato binário.

Outro exemplo é o do sequenciador Illumina Genome Analyzer que armazena os dados no formato chamado *Illumina Sequence and Probability Files*. Este formato separa as informações em dois arquivos em formato texto, sendo um para os dados das *reads* e outro para a qualidade. Diversos outros formatos podem ser encontrados e geralmente são convertidos para um formato padronizado antes de serem processados.

Os formatos FASTA e FASTQ (COCK et al., 2010), ambos codificados em formato texto, são bastante utilizados em processos de sequenciamento. Na maior parte dos casos, os arquivos gerados nos sequenciadores são convertidos para os formatos FASTA ou FASTQ que são aceitos pela maioria dos programas próprios para filtragem, ordenação ou alinhamento das *reads*.

O formato FASTA armazena somente as *reads* enquanto que as qualidades, se existirem, ficam armazenadas em um arquivo separado. A Figura 2.8 mostra um arquivo no formato FASTA. Neste arquivo a linha que começa com o caractere "$>$" corresponde ao cabeçalho da sequência, as linhas seguintes são as sequências de DNA ou RNA. No caso da Figura 2.8, as sequências estão codificadas no padrão IUB/IUPAC (JCBN, 1983), para aminoácidos e ácidos nucléicos.

```
>sp|Q9JN46|AAM1_RHOSH Putative glycosidase OS=Rhodobacter
MPKMRMALAAPSAGRGNQRLCDRLSSHTRPRWGSGVTDRNGESGRMRPARALRAPKGPDS
EANRQGVAAATGAQDRRDGGEEALMRLADARVAIEGVNLEIDGGRFAAKVVAGWEVAVEA
DIFCDGHDSID
>sp|Q9Z3R8|AGLA_RHIME Probable alpha-glucosidase OS=Rhizobium
MTMNETTSSLLEPDRDWWRGAVIYQIYPRSFQDTNGDGIGDLQGITARLPHIAGLGADAI
WISPFFTSPMRDFGYDVSNYVDVDPIFGTLEDFDALIAEAHRLGLRVMIDLVLSHTSDRH
PWFVESRSSRS
```

Figura 2.8: Exemplo de arquivo no formato FASTA.

O formato FASTQ, por sua vez, é uma extensão do formato FASTA e permite armazenar as *reads* e as suas qualidades no mesmo arquivo. A Figura 2.9 mostra um exemplo de arquivo no formato FASTQ, onde as linhas em negrito são as *reads* obtidas e as linhas sublinhadas são as qualidades associadas a cada nucleotídeo. As demais linhas correspondem às informações complementares como nome da *read*, tamanho, entre outras.

```
@SRR002320.1 080226_CMLIVERKIDNEY_0007:1:1:112:735 length=36
GTGGTGGGGTTGGTATTTGGTTTCTCGTTTTAATTA
+SRR002320.1 080226_CMLIVERKIDNEY_0007:1:1:112:735 length=36
IIIIIIII"IIIII)I$I1%HII"I#./(#/'$#*#
@SRR002320.2 080226_CMLIVERKIDNEY_0007:1:1:114:564 length=36
GGATACTCAGGCTGGCCCAATTTCTGGGCGTGGGAA
+SRR002320.2 080226_CMLIVERKIDNEY_0007:1:1:114:564 length=36
IIII:>&<I;I%I88II1&+I:IF>II,&D:I-'),
@SRR002320.3 080226_CMLIVERKIDNEY_0007:1:1:109:558 length=36
GTAGAATTAGAATTGTGAAGATGATAAGTGTAGAGG
+SRR002320.3 080226_CMLIVERKIDNEY_0007:1:1:109:558 length=36
IIIIIIIIIIIIIIIIIIIIIII<IIAIIII6I?I:
```

Figura 2.9: Exemplo de arquivo no formato FASTQ.

Dependendo dos programas utilizados ou da fonte dos arquivos, outros formatos podem ser utilizados para o armazenamento das sequências, tais como: o formato EMBL (*European Molecular Biology Laboratory*), para arquivos disponibilizados pelo banco de dados na página do Instituto de Bioinformática Europeu (EBI); o formato GENBANK, para arquivos disponibilizados pelo banco de dados no site do Centro Nacional para Informações de Biotecnologia (NCBI); e o formato PHYLIP, utilizado pelo pacote de programas PHYLIP (FELSENSTEIN, 2009) para o tratamento filogenético das sequências.

Outros formatos são necessários para representar alinhamentos de sequências de DNA ou RNA tais como o *Stands for Sequence Alignment/Map* (SAM) e *Binary Alignment/Map* (BAM) (LI et al., 2009). Ambos os formatos armazenam as mesmas informações, sendo que o formato SAM é gravado no formato texto, enquanto que o BAM é codificado no formato binário, sendo comprimido para ocupar menos espaço de armazenamento. Tanto o formato SAM como o BAM, armazenam os dados das *reads* originais e dos alinhamentos feitos durante o processo.

A Figura 2.10 mostra um exemplo de arquivo no formato SAM. As duas primeiras linhas correspondem ao cabeçalho do arquivo com informações gerais sobre os alinhamentos, tais como nome da sequência de referência (*SN:ref*) e seu tamanho (*LN:45*), ordem dos alinhamentos (*SO:coordinate*), entre outras. As demais linhas trazem informações sobre cada alinhamento encontrado. As sequências que foram alinhadas estão sublinhadas.

```
@HD VN:1.3 SO:coordinate
@SQ SN:ref LN:45
r001 163 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002   0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003   0 ref  9 30 5H6M       *  0   0 AGCTAA   *   NM:i:1
r004   0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003  16 ref 29 30 6H5M       *  0   0 TAGGC    *   NM:i:0
r001  83 ref 37 30 9M         =  7 -39 CAGCGCCAT         *
```

Figura 2.10: Exemplo de arquivo no formato SAM (LI et al., 2009).

Outro formato utilizado, principalmente para alinhamento de sequências múltiplas, é o CLUSTAL W, também gravado em formato texto. Este formato permite demons-

trar o alinhamento de diversas sequências entre si, indicando as posições onde ocorreram discrepâncias. Em geral, esses arquivos possuem a extensão ALN (HIGGINS; SHARP, 1988).

A Figura 2.11 mostra um exemplo de um arquivo CLUSTAL W. A primeira linha é um cabeçalho indicando o tipo de arquivo e sua versão. As linhas seguintes correspondem às sete sequências alinhadas, codificadas no padrão IUB/IUPAC (JCBN, 1983).

```
CLUSTAL W (1.74) multiple sequence alignment

 seq1 ----------------------KSEYKDENGGNYFQLREDWWDANRETVWKAITCNA
 seq2 --------------YEGLTTANGXEYQDKNGGNFFKLREDWWTANRETVWKAITCGA
 seq3 ----KRIYKKIFKEIHSGLSTKNGVDYQN-DGDNYFQLREDWWTANRSTVWKALTCSD
 seq4 ----------------------SRYKD-DGGNYFQLREDWWTANRHTVWEAITCSA
 seq5 --------------------NVAALTYEK-DGQNFYQLREDWWTANRATIWEAITCSA
 seq6 ------FSKNIX--QIEELQDEWLLAYKD--TDNYYELREHWWTENRHTVWEALTCEA
 seq7 ------------------------------------------KELWEALTCSR
```

Figura 2.11: Exemplo de arquivo no formato CLUSTAL W (CBRG, 2011).

Diversos outros formatos podem ser usados para gravar informações de alinhamentos tais como o formato PSL (*Pattern Space Layout*), geralmente utilizado pelo pacote de programas BLAT (KENT, 2002) e o formato MAF (*Multiple Alignment Format*) utilizado por diversos programas para arquivos com alinhamentos múltiplos.

A fase de análise é bastante diversificada, pois diferentes análises podem ser feitas e os dados gerados podem ter diferentes formatos. Por exemplo, o programa R (R Development Core Team, 2009), bastante utilizado para análises estatísticas, permite que o usuário exporte os resultados em diversos formatos, inclusive formato texto, indicando quais campos deseja exportar.

Pode-se perceber que os dados gravados em cada fase podem ser bastante variados, dependendo dos programas utilizados ou da origem dos dados obtidos a partir de alguns bancos de dados.

Capítulo 3

Proveniência de Dados

Este capítulo descreve o conceito de proveniência de dados de forma geral e os principais modelos apresentados na literatura, sendo dividido nas seguintes seções. A Seção 3.1 descreve o conceito de proveniência de dados e apresenta algumas abordagens relacionadas a sua utilização. A Seção 3.2 apresenta quatro modelos de proveniência de dados, escolhidos por serem os mais citados na literatura. E, por fim, a Seção 3.3 traz uma comparação entre os modelos apresentados levando-se em conta as características desejadas neste trabalho.

3.1 Conceito de Proveniência de Dados

O termo proveniência de dados diz respeito à origem ou procedência de um determinado dado. Conceito semelhante e muito utilizado no ramo da arte para indicar a procedência de uma determinada peça. A proveniência, porém, guarda muito mais significado do que a simples identificação do local de procedência. No ramo da arte, além do local, identificação do artista, suas motivações, ou as técnicas utilizadas trazem um maior valor a uma obra.

De acordo com Davidson e Freire (2008), a proveniência de dados pode ser dividida em três tipos :

- *Prospectiva*: Trata-se da sequência de processos utilizados (receita) para a geração do dado;
- *Retrospectiva*: Trata-se das informações obtidas durante a execução dos processos de geração do dado. Compreende desde o tempo de duração de cada atividade executada até a origem dos dados de entrada. Além disso, não depende do tratamento da proveniência prospectiva para ser utilizado;
- *Dados Definidos Pelo Usuário*: Qualquer informação que o usuário julgar necessária para futura utilização. Como exemplo, pode-se citar anotações, conclusões a respeito do processo e, até mesmo, observações sobre parâmetros utilizados.

Além disso, a obtenção da proveniência pode seguir duas abordagens conforme descrito por Tan (2004):

- *Abordagem Preguiçosa (Lazy)*: Nesta abordagem a obtenção da proveniência é executada somente no momento que é solicitada. Dessa forma, apenas informações imprescindíveis para o rastreamento posterior da proveniência devem ser armazenadas. Porém, esse rastreamento se torna mais difícil, e em alguns casos, impossível, já que a informação original pode não estar mais disponível;
- *Abordagem Ansiosa (Eager)*: Nesta abordagem a proveniência é obtida durante a geração da informação e é armazenada para permitir futuras consultas. É uma abordagem muito importante já que a proveniência estaria prontamente disponível quando o usuário solicitasse.

Cabe ressaltar que as duas abordagens podem ser combinadas, fazendo com que algumas informações sejam obtidas pela abordagem preguiçosa e outras pela ansiosa.

Segundo Freire et al. (2008), quando a proveniência é capturada de forma automática, pode-se dividir o nível em que é feita a captura conforme segue:

- *Workflow*: Usado pela grande maioria das soluções com SGWfC. Neste caso o SGWfC deve ser adaptado para capturar os dados dos diferentes processos executados;
- *Atividade*: Pode ocorrer de duas formas. Na primeira, cada processo executado é alterado para capturar os dados de proveniência. Na segunda, podem ser criados programas específicos para monitorar a execução de um determinado processo e capturar os dados de proveniência;
- *Sistema Operacional*: Utiliza os dados fornecidos pelo próprio sistema operacional como insumo para a proveniência.

Pode-se melhor compreender o problema que envolve o estudo da proveniência de dados considerando os objetivos que se deseja alcançar. Entre esses pode-se citar:

- *Permitir que os dados sejam avaliados e reavaliados*: Com os dados de proveniência pode-se fazer uma melhor análise dos dados produzidos, entendê-los melhor, além de, possivelmente gerar mais conhecimento a partir destes. Ainda, pode-se fazer análises futuras em novos cenários;
- *Observar o uso dos dados*: Pode-se saber onde e de que forma o dado está sendo usado.

Para atender estes objetivos deve-se ir muito além das simples perguntas "o que?" e "onde?", isso será demonstrado com a descrição de alguns modelos de proveniência.

3.2 Modelos de Proveniência de Dados

Modelos de proveniência de dados tem como principal objetivo fornecer uma estrutura para que os dados de proveniência possam ser armazenados e recuperados, mantendo seu significado e potencializando os seus benefícios.

Diversos modelos de dados são encontrados hoje na literatura, com diferentes formatos e objetivos. A seguir é feita uma breve revisão sobre diferentes modelos de proveniência

de dados a fim de verificar os pontos relevantes de cada um, além de listar algumas áreas de aplicações. Os modelos apresentados foram escolhidos devido a sua relevância para a aplicação de projetos de bioinformática e por serem os mais utilizados na literatura.

3.2.1 Modelo W7

O modelo de proveniência de dados *W7* foi apresentado por Ram e Liu (2007) e tem como base a ontologia de Bunge (BUNGE, 1977), a qual objetiva descrever as propriedades de um objeto de caráter geral. A partir deste estudo, o modelo *W7* estruturou a proveniência de uma peça de dado através da resposta a 7 perguntas (ou dimensões): O que?, Quem?, Quando?, Onde?, Como?, Qual? e Por quê? (*"What?"*, *"Who?"*, *"When?"*, *"Where?"*, *"How?"*, *"Which?"* e *"Why?"*).

A Figura 3.1 mostra uma visão geral do modelo *W7*, onde pode-se perceber o *Event* (resposta para *what*?) como ponto central, com as demais dimensões ligadas a ele. Isso demonstra a preocupação com a identificação das principais características que definem uma determinada peça de dados.

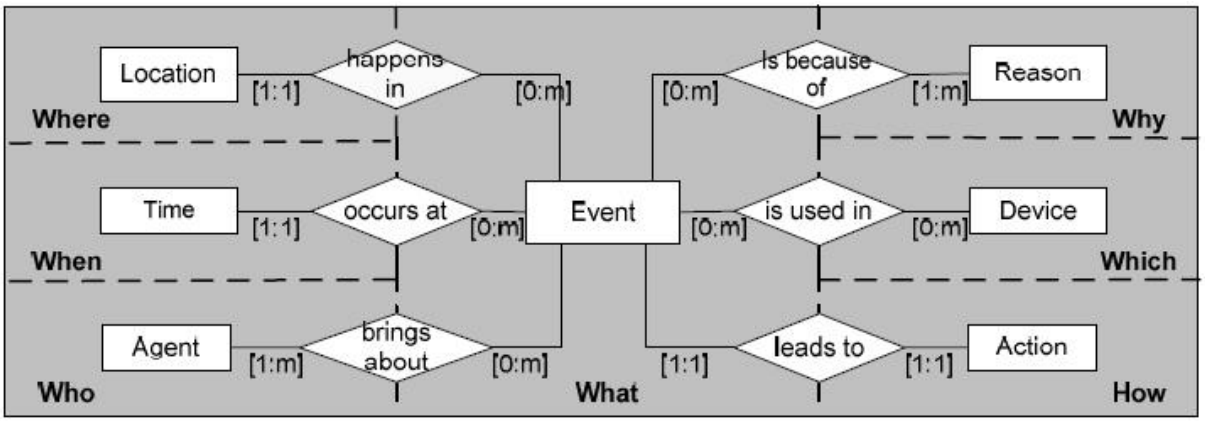


Figura 3.1: Visão geral do modelo W7 apresentando as 7 dimensões (RAM; LIU, 2007).

Visando maior flexibilidade, o modelo prevê ainda o detalhamento de cada dimensão. A Figura 3.2 mostra, por exemplo, a semântica das dimensões Tempo e Localização. O tempo pode ser verificado como um dado momento ou como uma duração, com um momento inicial e final. A localização, por sua vez, pode ser determinada como uma localização fonte ou destino e ainda ser expressa como uma localização física (ex.: coordenadas indicando um ponto em um mapa), geográfica (ex.: um país ou cidade) e lógica (ex.: um endereço IP ou o nome de um servidor). Dessa mesma forma, todas as outras dimensões são detalhadas no modelo, cada uma com a sua especificação.

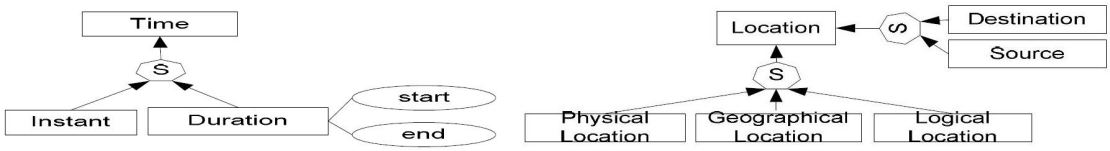


Figura 3.2: Semântica das dimensões "Quando?" e "Onde? " (RAM; LIU, 2007).

A Figura 3.3 mostra um exemplo de um grafo de proveniência baseado no modelo W7. No lado esquerdo superior do grafo pode-se ver uma retângulo com o texto *Terror_Report_Object* que representa um relatório sobre atividades terroristas. Ligado a este relatório está um evento e, a partir dele, pode-se ver as diferentes dimensões previstas no modelo.

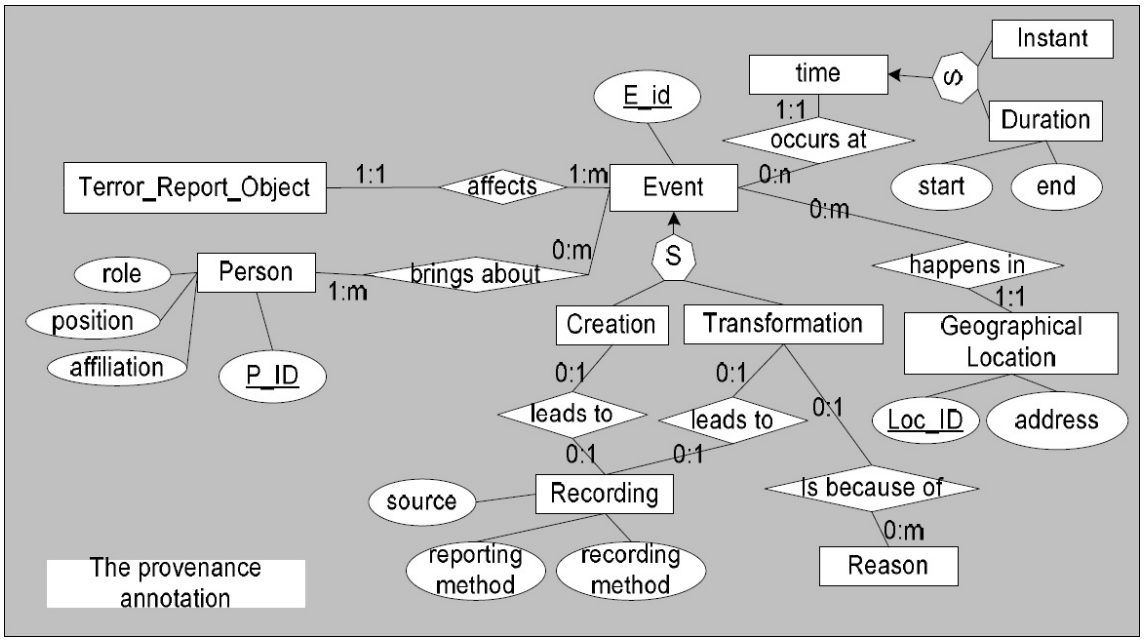


Figura 3.3: Exemplo de grafo de proveniência baseado no modelo W7 (RAM; LIU, 2007).

Trabalhos Relacionados e Avaliação

A seguir alguns exemplos de uso de proveniência baseados no modelo *W7* com uma breve explicação e suas referências:

- Marins et al. (2007) apresentou uma aplicação utilizada para buscar informações em computadores pessoais e catalogá-las a fim de facilitar a pesquisa destas pelo usuário;
- Liu e Ram (2009) utiliza a proveniência de dados para avaliar a qualidade de páginas da Wikipédia baseado na forma de colaboração dos usuários;
- Orlandi, Passant e Champin (2010) apresenta um modelo utilizado para captura de proveniência em páginas da Wikipédia a fim de mostrar informações de contribuição de usuários por páginas ou categorias.

O modelo *W7* tem uma representação bastante completa, visto que cada dimensão é descrita de forma a abranger a grande maioria dos dados de proveniência necessários a um determinado dado.

Por outro lado, tem uma representação gráfica confusa, principalmente ao demonstrar a sequência de processos que geraram um determinado objeto. Como pode ser visto na Figura 3.3, o objetivo não é demonstrar a sequência de acontecimentos e os dados consumidos para a criação de um dado, mas sim na caracterização deste dado através das sete dimensões propostas no modelo.

3.2.2 *Provenance Vocabulary*

O modelo *Provenance Vocabulary* descrito por Hartig e Zhao (2010) volta sua atenção para o problema da proveniência de dados publicados na web. A sua principal característica é fornecer classes e propriedades para que publicadores de dados para web possam armazenar, além dos dados publicados, também os metadados com informações úteis sobre a proveniência dos dados publicados.

Visando garantir maior flexibilidade às áreas de aplicação, o modelo prevê um núcleo central, e a possibilidade da criação e inclusão de extensões. Dessa forma, mantém o núcleo apenas com o essencial, sem comprometer a aplicação do modelo em sistemas mais complexos.

Levando em consideração o caráter aberto da web, o modelo prevê classes que propiciem metadados envolvidos tanto na criação como no acesso aos dados disponíveis, fechando assim o ciclo de oferta e consumo de dados na web (HARTIG; ZHAO, 2010).

Para melhor entender o modelo, pode-se destacar as três classes que fornecem a sua estrutura principal:

- *Artefato (Artifact)*: Podem ser tanto insumos como produtos de uma execução. Possui três subclasses, *DataItem* (um dado específico), *File* (uma coleção de *DataItem*) e *CreationGuideline* (regras ou mapeamentos para a criação de dados);
- *Execução (Execution)*: Representa a execução completa de uma ação ou processo. Possui duas subclasses, *DataCreation* (processo de criação de dados) e *DataAccess* (processo pelo qual um determinado dado foi acessado);
- *Ator (Actor)*: Representa uma entidade ativa que afeta a execução das ações ou processos. Possui duas subclasses, *HumanActor* (um ser social como uma pessoa ou organização) e *NonHumanActor* (representa um entidade não humana, como um serviço, por exemplo).

A Figura 3.4 mostra o núcleo central deste modelo, onde pode-se ver as classes Artefato, Execução e Ator com suas respectivas subclasses e propriedades. As linhas pontilhadas na figura representam as descendências entre as classes e as subclasses onde pode ser visto, por exemplo, que as subclasses *DataAccess* e *DataCreation* são descendentes da classe Execução. As linhas cheias, por sua vez, demonstram as ligações que podem ser feitas entre cada uma das classes. Por exemplo, um objeto da subclasse *NonHumanActor* pode ser operado por um objeto da subclasse *HumanActor* e essa relação é representada pela aresta *operatedBY*.

A Figura 3.5 mostra um exemplo de grafo de proveniência baseado no modelo *Provenance Vocabulary*. Neste grafo pode-se ver diversos nós ligados por arestas simbolizando os eventos e dados gerados. Pode-se citar, por exemplo, o nó chamado *Edit1* (centro da figura) que corresponde a uma Execução. Esse nó está ligado ao elemento *user1248* (Ator), por uma aresta chamada *performedBy* indicando que Execução *Edit1* foi executada pelo Ator *user1248*. O nó *Edit1* também está ligado ao nó chamado *node612986688v1* (Artefato) através de uma aresta *createdBy*. Assim, pode-se perceber que o Artefato *node612986688v1* foi criado pela Execução *Edit1*. A interpretação destas arestas e nós permite compreender a proveniência representada no grafo.

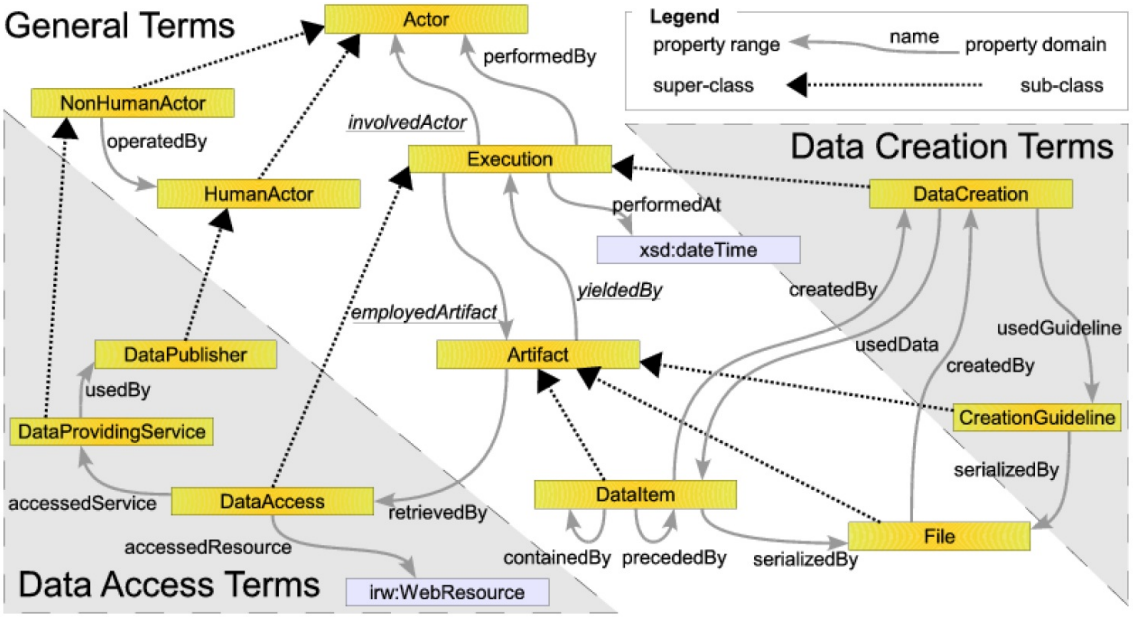


Figura 3.4: Modelo de classes e propriedades do modelo *Provenance Vocabulary* (HARTIG; ZHAO, 2010).

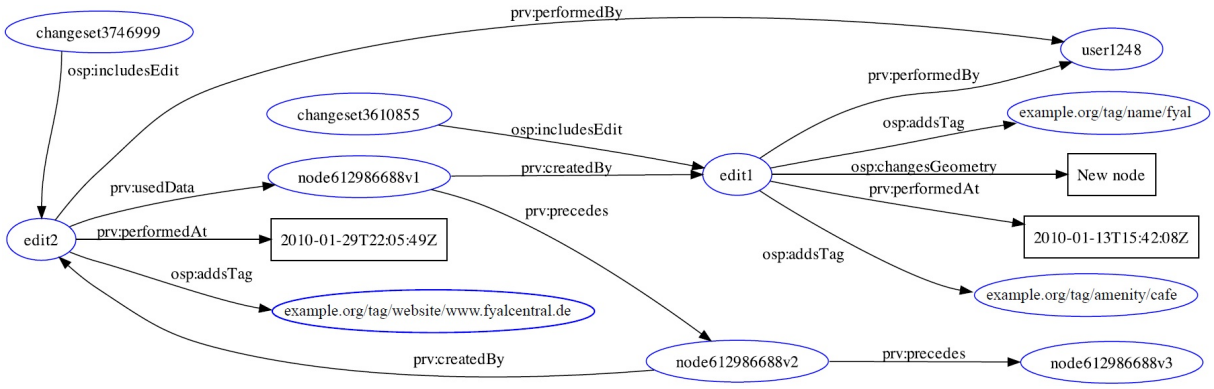


Figura 3.5: Exemplo de grafo de proveniência baseado no modelo *Provenance Vocabulary* (KEßLER; TRAME; KAUPPINEN, 2011).

Trabalhos Relacionados e Avaliação

A seguir alguns exemplos de utilização deste modelo:

- Hartig e Zhao (2009) apresentam uma aplicação que utiliza proveniência de dados na web para avaliação da qualidade das informações disponibilizadas;
- Omitola, Gibbins e Shadbolt (2010) apresentam uma extensão do modelo para permitir a integração de dados da web;
- Keßler, Trame e Kauppinen (2011) utiliza o *Provenance Vocabulary* como base para o tratamento de dados de proveniência no aplicativo *OpenStreetMap*.

O modelo *Provenance Vocabulary* demonstra os três elementos básicos fundamentais nos modelos de proveniência de dados (Artefato, Ator e Execução), porém suas ligações

representam a criação, publicação e acesso dos dados, principalmente em ambientes distribuídos. Dessa forma, suas características são mais voltadas ao rastreamento de dados publicados na internet. Além de possui uma representação gráfica confusa (como pode ser visto na Figura 3.5) já que não possui símbolos diferentes para cada elemento.

3.2.3 *Provenir Ontology*

O modelo descrito por Sahoo e Sheth (2009) foi desenvolvido para ser um modelo de proveniência de dados genérico, priorizando a interoperabilidade entre diferentes sistemas e sua adaptação para qualquer aplicação. Da mesma forma que no modelo *Provenance Vocabulary*, o modelo *Provenir Ontology* define um núcleo comum e permite a criação de módulos específicos para o domínio da aplicação desejada.

Para melhor entender este modelo são descritos dois conceitos primitivos para representar as classes de metadados. O primeiro é o conceito de *ocorrente* (*"occurrent"*) que é definido como entidades que ocorrem em sucessivas fases temporais. O segundo conceito é *continuo* (*"Continuant"*) e é definido como entidades que permanecem, ou continuam a existir, através do tempo, enquanto passam por diferentes tipos de mudanças, inclusive mudança de lugar. Estes dois conceitos são importantes porque a partir deles foram criadas três classes básicas, conforme descrito a seguir:

- *Continuo*:
  - *Dado (Data)*: Classe que representa tanto o material inicial, intermediário e o produto final de um experimento científico, além dos parâmetros que afetam a execução do processo científico;
  - *Agente (Agent)*: Classe que, de alguma forma, afeta um processo individual.
- *Ocorrente*:
  - *Processo (Process)*: Classe que demonstra uma ação que afeta (processa, modifica, cria, apaga, entre outras ações) dados individuais.

Além destas classes, o modelo básico prevê diversas subclasses conforme mostrado na Figura 3.6, onde podem ser vistas a hierarquia de classes e a forma de ligação das classes Dado, Processo e Agente. Por exemplo, a classe Dado possui duas subclasses, *Data_collection* e *Parameter*. Além das classes e subclasses também podem ser vistas as ligações entre cada uma delas. Estas ligações permitem criar o grafo de proveniência. Como exemplo, pode-se citar a ligação *has_parameter* que permite indicar que um Processo ou um Agente tenham parâmetros ligados a eles no grafo.

Trabalhos Relacionados e Avaliação

A seguir alguns casos de uso e extensões do modelo *Provenir Ontology*:

- Sahoo et al. (2009) apresenta uma extensão do modelo para tratamento de dados biológicos envolvidos no projeto que busca facilitar a identificação de vacina, diagnóstico e quimioterapia para o patógeno humano Trypanossoma Cruzi (T.cruzi);

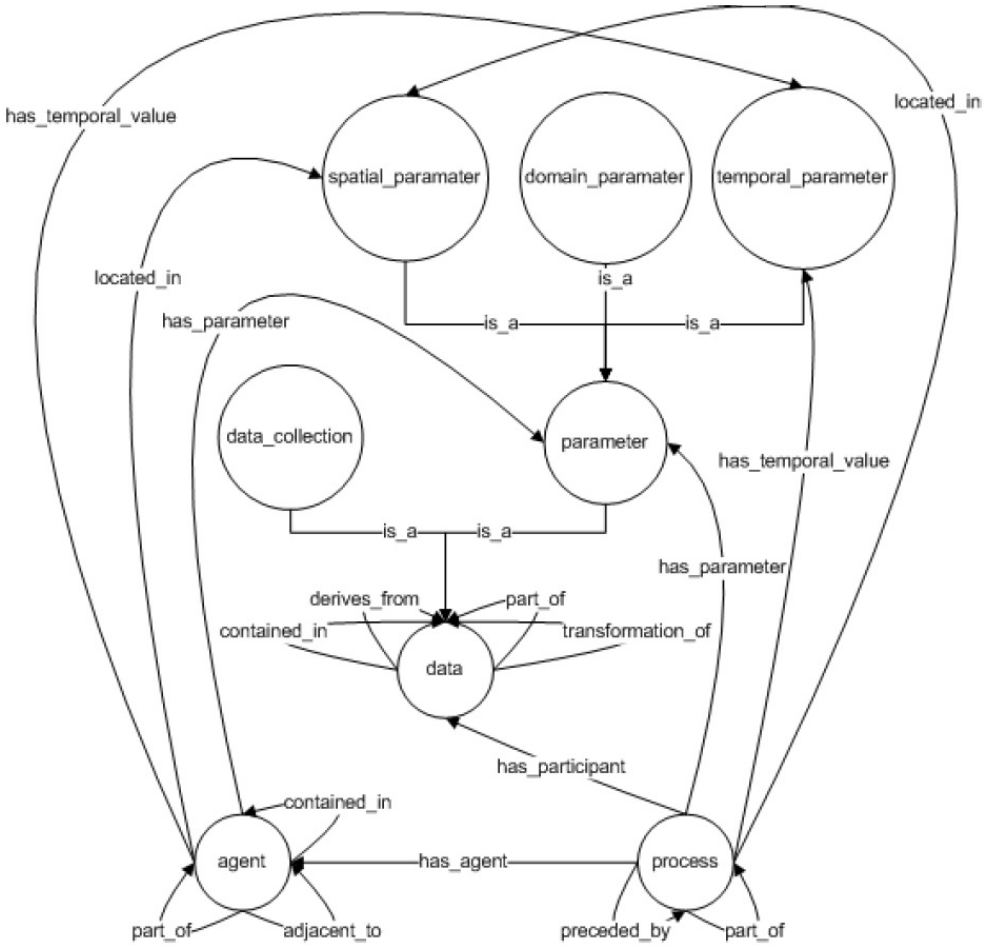


Figura 3.6: Esquema de classes e relacionamentos do modelo *Provenir Ontology* (MISSIER et al., 2010).

- Missier et al. (2010) apresenta uma aplicação que utiliza o modelo *Provenir Ontology* e um módulo extensão deste para dados biológicos visando a coleta de dados gerados pelo SGWfC Taverna. O exemplo utilizado é a busca de todos os relacionamentos conhecidos entre uma região específica no genoma do camundongo, conhecida como QTL (*Quantitative Trait Loci*) e os caminhos metabólicos envolvendo genes que estão presentes nesta região;
- Patni et al. (2010) apresenta um *framework* para armazenamento e consulta da proveniência de dados de sensores meteorológicos.

O modelo *Provenir Ontology*, assim como o *Provenance Vocabulary*, representa os três elementos básicos (Agente, Dado e Processo) além de tipos descendentes como Coleção de Dados e Parâmetro o que permite, juntamente com os diferentes tipos de ligações, representar o histórico de um determinado dado. Porém, da mesma forma que o *Provenance Vocabulary*, o modelo *Provenir Ontology* também não possui símbolos diferentes para cada elemento, o que torna o grafo de proveniência difícil de interpretar.

3.2.4 *Open Provenance Model* (OPM)

O *Open Provenance Model* começou a ser discutido em maio de 2006 no *Workshop* Internacional de Anotação e Proveniência realizado em Chicago no Estados Unidos e tem suas principais características moldadas por três encontros conhecidos como *Provenance Challenge* (Provenance Challenge Wiki, 2012). Esses desafios são abertos e tem como objetivo reunir pesquisadores interessados em proveniência de dados para que, reunidos em grupos, possam realizar trabalhos em torno de um mesmo problema de proveniência, sendo os resultados compilados a fim de direcionar a formação do modelo OPM.

Logo, o desenvolvimento do modelo OPM é definido como um projeto aberto, em que qualquer pessoa interessada pode sugerir mudanças. Isso pode ser visto no documento que descreve o modelo de governança do OPM em Moreau et al. (2009) na página wiki do modelo (OPM, 2011a).

Assim, foi projetado um modelo de proveniência de dados aberto voltado a caracterização da proveniência de qualquer objeto, material ou imaterial. O modelo OPM, descrito em Moreau et al. (2010), procura demonstrar a relação causal entre eventos que afetam objetos (digitais ou não) e descreve essa relação através de um grafo acíclico direcionado.

O modelo atualmente encontra-se na versão 1.1 e tem como base os seguintes objetivos:

- Permitir a troca de informações entre sistemas através de uma camada de proveniência;
- Permitir aos desenvolvedores construir e compartilhar ferramentas baseadas neste modelo;
- Definir proveniência de forma precisa;
- Fornecer uma representação digital de proveniência para qualquer "coisa";
- Permitir a coexistência de múltiplos níveis de descrição;
- Definir um conjunto central de regras que identifique inferências válidas que podem ser feitas pela representação de proveniência.

Da mesma forma, o documento do OPM (MOREAU et al., 2010) apresenta também quais não são os objetivos do OPM:

- Definir a forma como os sistemas armazenam ou manipulam as informações internamente;
- Definir uma linguagem de computador para o modelo;
- Definir protocolos para o armazenamento de informações de proveniência em repositórios;
- Definir protocolos para consulta em repositórios de proveniência.

Seu modelo básico tem como componentes do grafo três tipos de nós e cinco tipos de arestas (dependências), sendo permitidas extensões destes para melhor especificar cada tipo de aplicação.

Nós

Levando em consideração que objetos podem ser materiais, como um carro; digitais, como um arquivo de dados; ou imateriais, como uma decisão; e que estes objetos tem um estado que pode ser alterado por algum tipo de intervenção, foram definidos três tipos de nós (MOREAU et al., 2010):

- *Artefato (Artifact)*: Corresponde a um determinado objeto (digital ou não) em um determinado estado no tempo. Pode ser usado ou gerado por um Processo;
- *Processo (Process)*: Corresponde a uma ou mais ações executadas consumindo e/ou gerando Artefato;
- *Agente (Agent)*: Corresponde a uma entidade, em um determinado contexto, agindo como um catalisador do Processo, permitindo, facilitando, controlando, ou, de qualquer forma, afetando a sua execução.

Para exemplificar a utilização dos nós pode-se citar o processo de revisão de um automóvel em uma oficina mecânica. Assim tem-se o "carro na situação inicial" e o "carro revisado" como o mesmo objeto porém, sendo representado por dois Artefato diferentes já que encontram-se em situações diferentes em instantes diferentes. Ainda tem-se a revisão do carro como um Processo e o mecânico corresponderia ao Agente. Na Figura 3.7 podem ser vistos os respectivos símbolos para cada nó.

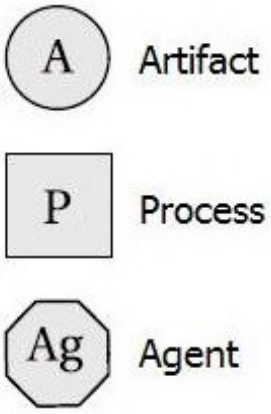


Figura 3.7: Representação gráfica dos nós do modelo OPM (MOREAU et al., 2010).

Dependências

Para que o grafo fique completo é necessário definir as arestas. No modelo OPM, as arestas são chamadas de dependências e representam as relações causais entre os Artefatos, Processos e Agentes. Para isso são definidos cinco tipos de dependências conforme descritos a seguir:

- *Used*: Indica que um determinado Artefato foi usado por um Processo;
- *WasGeneratedBy*: Indica que um Artefato foi gerado por um determinado Processo;
- *WasControlledBy*: Indica que um Processo foi controlado por um Agente;

- *WasTriggeredBy*: Indica que um Processo P1 foi acionado por um outro Processo P2;
- *WasDerivedFrom*: Indica que um Artefato A1 é derivado de um outro Artefato A2.

Devido ao fato do grafo ser direcionado e as dependências representarem relações causais entre os nós, pode-se observar que cada dependência tem uma origem e um destino. Para o OPM a origem de uma dependência representa o efeito da mesma e o destino corresponde a causa das dependências. Por exemplo, na aresta *WasGeneratedBy*, representada na Figura 3.8, o destino da aresta é um Processo, portanto, representa o efeito da dependência. Por outro lado, a origem da aresta é um Artefato que corresponde a causa da dependência. Na Figura 3.8 também são exibidas as demais dependências indicando suas origens e destinos.

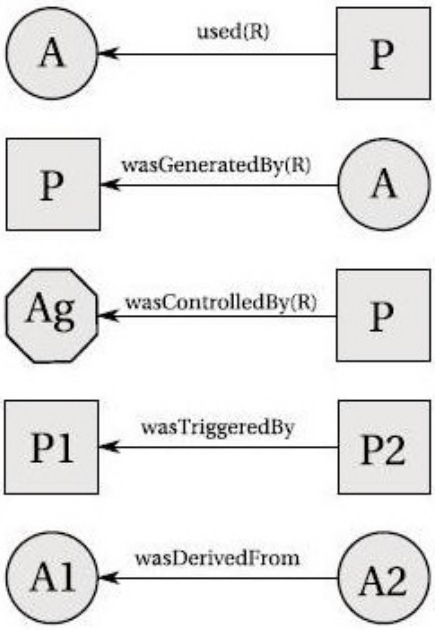


Figura 3.8: Representação gráfica das arestas do Modelo OPM (MOREAU et al., 2010).

Regras, Anotações e Contas

Em alguns casos existe a possibilidade de mais de uma relação de dependência do mesmo tipo para um determinado nó, como um Artefato sendo usado por mais de um Processo ou um Processo sendo controlado por mais de um Agente. Neste caso é necessário identificar estas relações e isto é feito através de regras (*Rules*). As regras definem a função do Artefato ou do Agente no Processo e dependem do domínio da aplicação. Como exemplificado na Figura 3.9, pode-se ver dois Agentes (Biólogo e Assistente) controlando o Processo e as regras indicam a função de cada um deles (Validação e Execução). Da mesma forma dois Artefatos são usados pelo Processo e as regras especificam o tipo de utilização (Sequências e Genoma de Referência). Conforme definido no modelo somente as dependências *Used*, *WasGeneratedBy* e *WasControlledBy* podem ter regras.

Outro recurso previsto pelo modelo OPM são as anotações (*Annotations*). Este recurso permite anexar informações adicionais em qualquer elemento do grafo de proveniência.

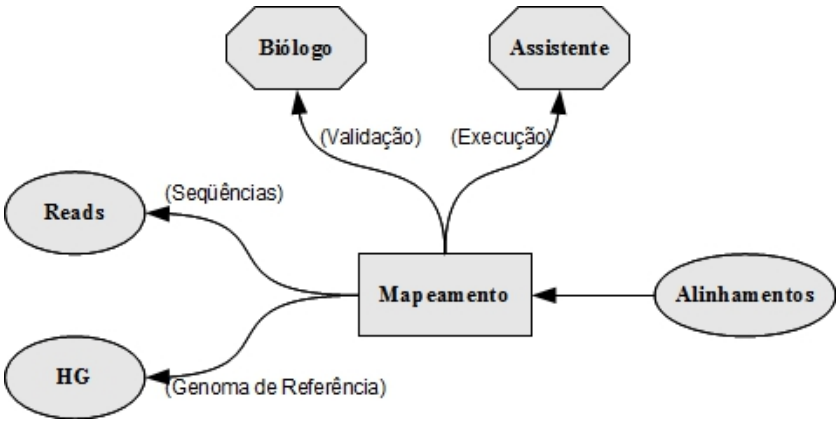


Figura 3.9: Exemplo da utilização de regras para definir o objetivo das arestas.

Com isso o usuário pode anexar, por exemplo, bibliografias, endereços de sites da internet, manuais de programas utilizados ou qualquer outro tipo de informação relevante em uma consulta futura.

Por fim, o modelo prevê Contas (*Accounts*) que permitem a partir de um mesmo grafo OPM, criar diferentes visualizações. Conforme pode ser visto na Figura 3.10, ambos os grafos de proveniência mostram a mesma operação mas representados por diferentes Contas. Na representação, um par de valores (2,6) teve o valor 1 adicionado em cada um dos termos, resultando no par (3,7). Na Conta da esquerda é mostrada uma visão mais simples onde um simples Processo chamado *add1ToAll* usa o Artefato *(2,7)* para gerar o Artefato *(3,7)*. A Conta da direita, por sua vez, representa a operação de maneira mais detalhada, onde um Processo desmembra cada um dos valores do par para então adicionar o valor 1 em cada um deles separadamente e então juntá-los novamente em outro processo.

Restrições

Quando existe a necessidade de representação temporal dos acontecimentos, começam a surgir questões importantes para que um determinado grafo de proveniência seja reconhecido como um acontecimento possível. No modelo OPM, tais situações são tratadas como restrições temporais (MOREAU et al., 2010) que devem ser verificadas a fim de validar um grafo de proveniência. A Figura 3.11 representa um grafo com as sete diferentes restrições temporais explicitadas pelo modelo OPM a fim de garantir que um grafo de proveniência seja válido.

Trabalhos Relacionados e Avaliação

A seguir alguns exemplos de aplicação do modelo OPM:

- Cao et al. (2008) apresentam um sistema que captura e gerencia dados de proveniência em experimentos científicos, baseado no modelo OPM;
- Marinho et al. (2009) descrevem a captura e gerenciamento de informações de proveniência em cenários de ambientes heterogêneos e distribuídos utilizando o formato OPM;

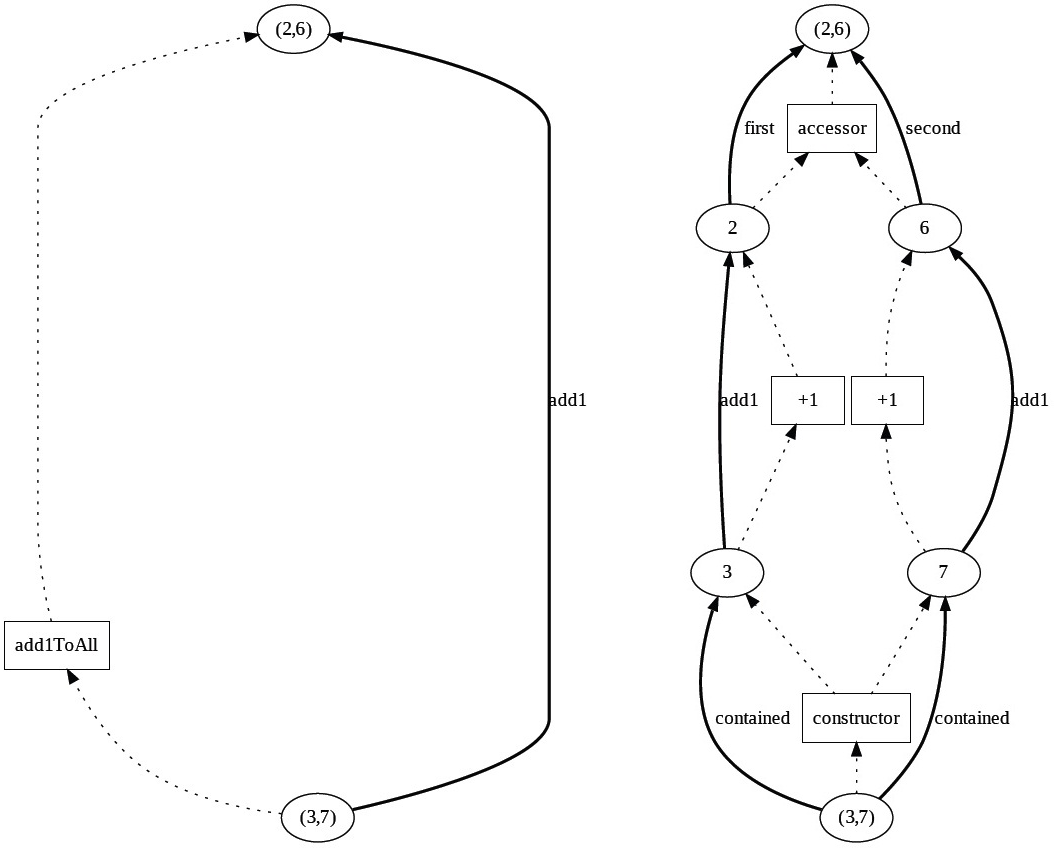


Figura 3.10: Exemplo da utilização de contas no modelo OPM (MOREAU et al., 2010).

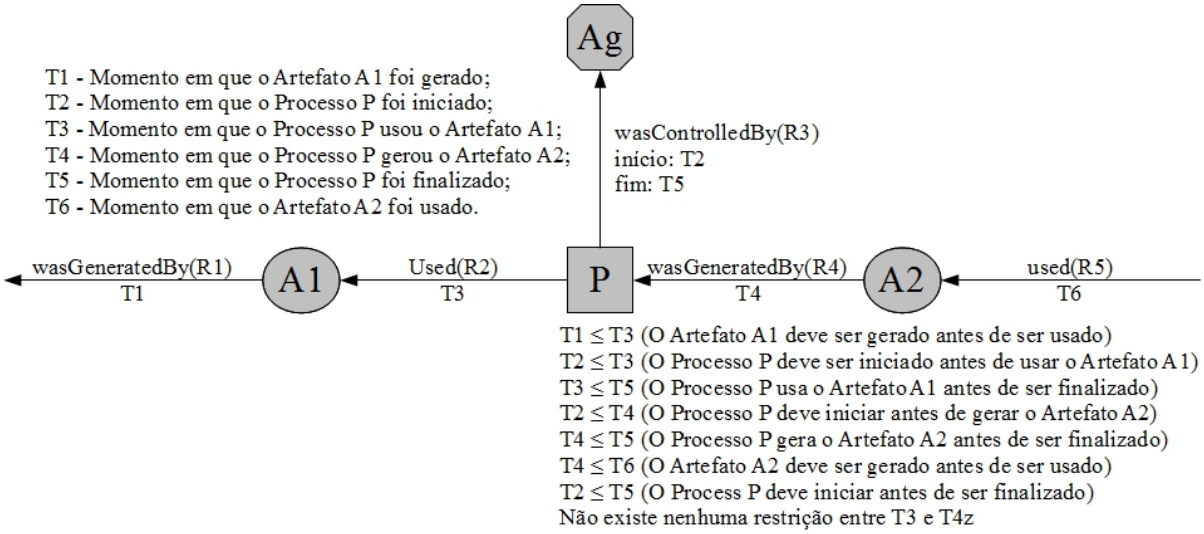


Figura 3.11: Exemplificação das restrições temporais do modelo OPM.

- Chapman, Blaustein e Elsaesser (2010) definem um modelo baseado no OPM possibilitando a avaliação da confiabilidade de um dado através das informações de proveniência;
- Braun et al. (2010) fazem a avaliação da interoperabilidade de dados de proveniência

entre os sistemas PASS (*Harvard Provenance Aware Storage System)* e o sistema MITRE's PLUS. Ambos trabalham com o modelo OPM;

- Gomes (2011) define um sistema para captura de dados de proveniência de *workflows* biológicos a partir de SGWfC e armazena estes dados em um banco de dados modelado conforme o modelo OPM.

Esta seção apresentou uma parte das definições disponíveis na especificação do modelo OPM. Assim, o modelo OPM apresenta-se bastante completo, representando, tanto lógica como graficamente, cada nó do grafo de forma clara, além de fornecer um conjunto de arestas suficiente para representar de forma genérica as dependências entre os nós. Também apresenta conceitos importantes como Regras, Anotações e Contas que tornam o grafo de proveniência mais rico em detalhes e permite diferentes visualizações do mesmo. As restrições são outra importante ferramenta que permite verificar os grafos gerados em relação à sua validade. Por fim, fornece também especificações XML e RDF, o que permite a troca de informações entre diferentes sistemas, além de diversas ferramentas de apoio.

Para a representação de conjunto de dados porém, o modelo OPM não apresenta nenhuma solução. Este elemento é de suma importância na representação de arquivos de dados em projetos de bioinformática.

3.2.5 PROV-DM

O PROV-DM teve a sua primeira versão desenvolvida em outubro de 2011 pelo grupo de trabalho do W3C. A sua versão mais nova foi publicada em maio de 2012 (W3C, 2012c), tratando-se assim de um modelo bastante recente. Este modelo tem como principal função descrever as pessoas, entidades e atividades envolvidas na produção de uma peça de dado ou de um objeto qualquer. Sendo assim, o modelo cria as condições para que a proveniência seja demonstrada e trocada entre diferentes sistemas.

Apesar de ser um modelo recente, já se encontra bastante robusto uma vez que utiliza os mesmo princípios do OPM, largamente aplicado em projetos apresentados na literatura. O PROV-DM possui um maior detalhamento, permitindo demonstrar a proveniência de forma mais precisa. Além disso, um dos objetivos do modelo PROV-DM é se tornar uma recomendação do W3C.

Visando essas características o grupo de trabalho do W3C produziu diversas especificações que apóiam o modelo PROV-DM em diferentes aspectos, conforme descrito a seguir:

- *PROV-DM*: Visão geral das principais características do modelo PROV-DM (W3C, 2012c);
- *PROV-CONSTRAINTS*: Define um conjunto de restrições aplicadas ao modelo PROV-DM (W3C, 2012a);
- *PROV-N*: Define uma notação para proveniência destinada para o uso em linguagem descritiva (W3C, 2012e);
- *PROV-O*: Define uma ontologia OWL-RL que permite mapear o modelo PROV-DM para o padrão RDF (W3C, 2012f);
- *PROV-AQ*: Define mecanismos de acesso e consulta de proveniência (W3C, 2012b);

- *PROV-PRIMER*: Apresenta uma introdução ao modelo de proveniência (W3C, 2012d);
- *PROV-SEM*: Define uma semântica formal do modelo PROV-DM (W3C, 2011a);
- *PROV-XML*: Esquema XML para o PROV-DM (W3C, 2011b).

O modelo PROV-DM tem como principal característica demonstrar a proveniência de qualquer objeto (real ou imaginário) através de um grafo direcionado. A raiz deste grafo representa a entidade cuja proveniência está sendo representada e as arestas são direcionadas para as atividades e entidades das quais foi originada.

O modelo é dividido em seis componentes que contêm tanto os elementos como as relações possíveis entre eles (W3C, 2012c).

- *Entidades e Atividades*: Entidades (*Entities*) podem representar qualquer objeto (real ou imaginário) e Atividades (*Activities*) representam os processos que usam e geram Entidades;
- *Agente e Responsabilidades*: Agentes (*Agents*) são Entidades que influenciam, direta ou indiretamente, a execução das Atividades, recebem atribuições de outros Agentes e podem ter algum tipo de ligação (posse, direitos, etc...) sobre outras Entidades;
- *Derivações*: Descreve a relação entre diferentes Entidades durante o ciclo de transformação executado pelas Atividades, permitindo demonstrar a dependência entre as Entidades usadas e geradas;
- *Alternativo*: Descreve a relação entre diferentes visões de uma mesma Entidade;
- *Coleções*: São Entidades que possuem membros, os quais são também Entidades, e podem ter a sua proveniência demonstrada de forma coletiva;
- *Anotações*: Fornece mecanismos para inclusão de anotações para os elementos do modelo.

Para melhor compreender o PROV-DM, tem-se a seguir um detalhamento separado dos tipos, suas relações e demais elementos do modelo.

Tipos

O modelo define dois tipos básicos que dão origem aos possíveis nós do grafo: Atividade e Entidade. Estes dois tipos dão origem a outros quatro subtipos. A hierarquia de tipos e subtipos é descrita com mais detalhes a seguir:

- *Atividade*: Representa os possíveis processos executados que deram origem ao objeto foco da proveniência;
- *Entidade*: Representa qualquer objeto para o qual se possa representar algum tipo de proveniência. Este, por sua vez, tem quatro subtipos que devem ser destacados: Agente, Coleção, Conta (*Account*) e Plano (*Plan*);
  - *Agente*: Representa qualquer entidade (pessoa, organização, software, etc...) que possa ter algum tipo de ação sobre uma Atividade ou possa ter algum tipo de responsabilidade (proprietário, direitos autorais, etc...) sobre uma Entidade;

– *Coleção*: Representa um conjunto de Entidades, e pode ter a sua proveniência representada como conjunto, independente da proveniência do seu conteúdo;
– *Conta*: Representa um conjunto de informações (tipos e relações) que compõem um grafo de proveniência;
– *Plano*: Representa um conjunto de ações ou passos que um Agente deve seguir para chegar a um determinado objetivo.

A Figura 3.12 mostra os símbolos utilizados pelo modelo para representar os diferentes nós do grafo. O símbolo da Entidade é utilizado para representar os tipos Coleção e Plano, uma vez que representam subtipos do tipo Entidade. O tipo Conta não tem símbolo, pois representa o próprio grafo de proveniência.

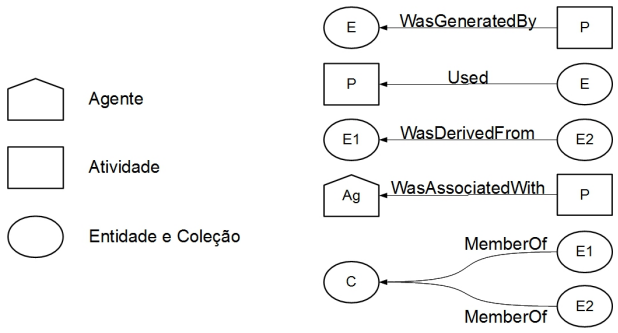


Figura 3.12: Representação gráfica dos diferentes tipos e relações no modelo PROV-DM.

As Coleções, por sua vez, possuem um subtipo chamado Dicionário (*Dictionary*). Esse subtipo é uma implementação do tipo Coleção que tem como objetivo formar uma estrutura lógica constituída pelo par chave/Entidade, ou seja, trata-se de uma Coleção cujos membros são indexados por uma chave.

Relações

As relações representam as arestas no grafo de proveniência que, por sua vez, indicam as relações possíveis entre cada nó. Além de descrever cada tipo que compõe o modelo, os seis componentes também detalham as relações que podem ocorrer entre cada um dos tipos. Na Tabela 3.1 podem ser vistas todas as relações possíveis entre os três tipos principais do modelo.

Tabela 3.1: Relações entre os tipos Entidade, Atividade e Agente.

	Entidade	**Atividade**	**Agente**
Entidade	wasDerivedFrom wasRevisionOf wasQuotedFrom hadOriginalSource alternateOf specializationOf	wasGeneratedBy wasInvalidatedBy	wasAttributedTo
Atividade	used wasStartedBy wasEndedBy	wasStartedByActivity wasInformedBy	wasAssociatedWith
Agente	—	—	actedOnBehalfOf

Como pode ser observado, o modelo PROV-DM possui uma quantidade maior de relacionamentos, o que permite expressar a proveniência de forma mais precisa. A seguir são descritas as relações mais relevantes para o nosso trabalho:

- *wasDerivedFrom*: Indica, de forma geral, que uma Entidade (original) foi usada, direta ou indiretamente, na geração de outra Entidade (derivada);
- *used*: Indica que uma Entidade foi usada por uma Atividade;
- *wasGeneratedBy*: Indica que uma Entidade foi gerada por uma Atividade;
- *wasAttributedTo*: Atribui algum tipo de responsabilidade a um Agente sobre uma Entidade;
- *wasAssociatedWith*: Atribui algum tipo de responsabilidade a um Agente sobre uma Atividade;
- *actedOnBehalfOf*: Indica que um Agente está agindo em nome de outro.

A Figura 3.13 demonstra, através da anotação UML, as relações entre os elementos Entidade, Atividade e Agente.

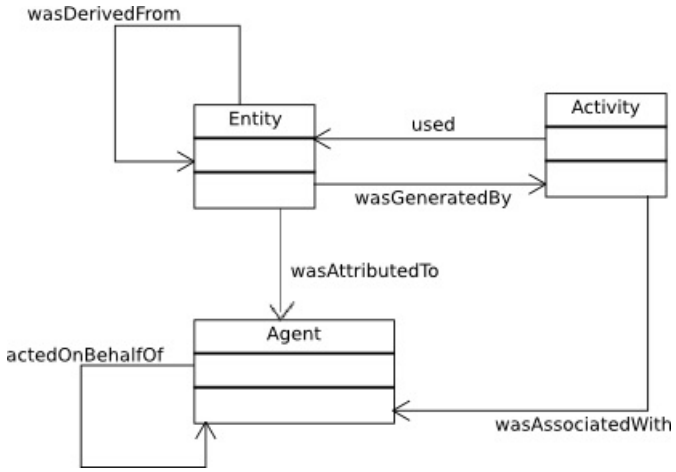


Figura 3.13: Relações possíveis entre os tipos Agente, Atividade e Entidade (W3C, 2012c).

A relação *wasDerivedFrom* pode representar todo o histórico da transformação dos dados brutos no produto final. A fim de permitir um histórico mais detalhado das derivações feitas durante o processo representado pela proveniência, a relação *wasDerivedFrom* possui diversos subtipos, conforme pode ser visto na Figura 3.14.

A seguir tem-se a descrição dos subtipos da relação *wasDerivedFrom*:

- *wasQuotedFrom*: Indica que a Entidade derivada foi gerada a partir da cópia de parte da Entidade original;
- *hadOriginalSource*: Indica que uma Entidade corresponde a fonte da informação que corresponde a Entidade derivada;
- *wasRevisionOf*: Indica que a Entidade derivada foi gerada a partir da revisão da Entidade derivada, sendo que tal responsabilidade da revisão pode ser atribuída a um Agente;

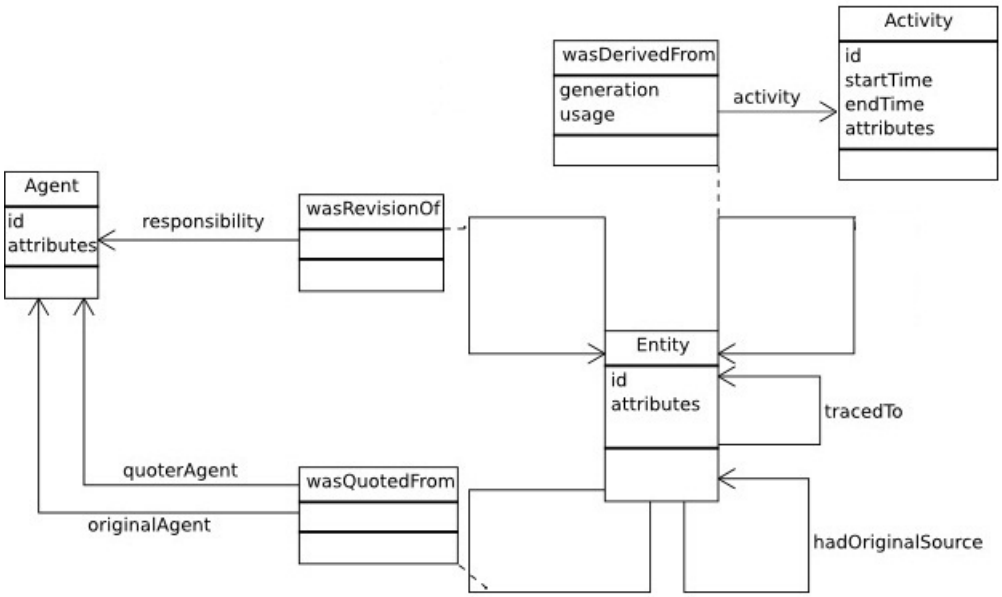


Figura 3.14: Definição dos subtipos de derivações (W3C, 2012c).

- *tracedTo*: Indica, de forma genérica, que existe uma relação de dependência entre duas Entidades, sem especificar qual é a derivada nem qual é a original.

Por fim, conforme pode ser visto na Figura 3.15, o subtipo Dicionário (tipo de Coleções cujos membros são indexados) possui três tipos de relações. Estas relações são demonstradas resumidamente na Tabela 3.2.

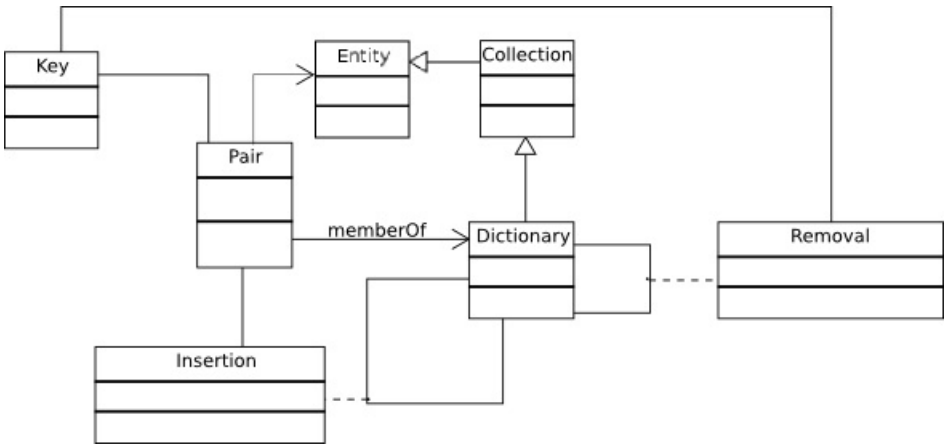


Figura 3.15: Relações possíveis entre os tipos Entidade e Dicionário (W3C, 2012c).

- *derivedByInsertionFrom*: Um caso particular da relação *wasDerivedFrom* que indica que um Dicionário foi derivado de um Dicionário original a partir da inserção de uma Entidade;

Tabela 3.2: Relações entre os elementos Entidade e Dicionário.

	Dicionário
Entidade	memberOf
Dicionário	derivedByInsertionFrom derivedByRemovalFrom

- *derivedByRemovalFrom*: Um caso particular da relação *wasDerivedFrom* que indica que um Dicionário foi derivado de um Dicionário original a partir da remoção de uma Entidade;
- *memberOf*: Indica que uma determinada Entidade é membro de um Dicionário.

Restrições

As restrições do modelo PROV-DM (W3C, 2012a) vêm sendo desenvolvidas em paralelo com o modelo PROV-DM e descrevem regras para a construção de grafos de proveniência. Tais restrições tem dois objetivos principais: (a) evitar a formação de grafos inválidos ou inconsistentes e (b) fazer inferências sobre os elementos e as relações entre eles. Por se tratar de um documento recente, este ainda possui algumas indefinições, porém alguns conceitos estão bem definidos e são descritos a seguir.

O tipo de restrição mais importante é chamada restrição de ordem dos eventos. Neste tipo de restrição pressupõem-se que os elementos do grafo de proveniência são dotados de características temporais representadas por propriedades como hora inicial ou final e que podem ser avaliadas e validadas. Assim, como no modelo OPM, tal validação pretende garantir que um grafo de proveniência represente uma ordem de acontecimentos possível.

As restrições são divididas em três grupos: restrições de Atividade, restrições de Entidade e restrições de Agente, e compreendem um total de 15 restrições. A seguir são descritas as restrições relevantes para o nosso trabalho:

- *Restrições de Atividade*: Restrições relacionadas à execução das Atividades:
  - *Início/Fim*: O início da execução de uma Atividade deve preceder o seu fim;
  - *Uso*: O uso de uma Entidade por uma Atividade deve ocorrer entre o início e o fim da sua execução;
  - *Geração*: A geração de uma Entidade por uma Atividade deve ocorrer entre o início e o fim da sua execução.
- *Restrições de Entity*: Restrições relacionadas ao ciclo de vida de uma Entidade:
  - *Geração/Uso*: A geração de uma Entidade deve preceder o seu uso;
  - *Derivação/Uso/Geração*: Para os casos em que existe uma derivação entre duas Entidades, por exemplo E2 é derivado de E1, e o uso de E1 é conhecido, então o uso de E1 deve preceder a geração de E2;
  - *Derivação/Geração/Geração*: Para os casos em que existe uma derivação entre duas Entidades, por exemplo E2 é derivado de E1, e o uso de E1 não é conhecido, então a geração de E1 deve preceder a geração de E2.

- *Restrições de Agente*: Restrição relacionada ao ciclo de vida de um Agente:
  - *Associação*: A associação entre um Agente e uma Atividade deve ocorrer entre o início e o fim da execução desta Atividade.

Anotações

O modelo PROV-DM fornece um recurso para adição de informações extras no grafo de proveniência através de um identificador chamado Nota (*Note*). Este identificador representa um conjunto de pares atributo-valor que permite ao usuário criar diversos tipos de anotações.

Cada Nota, por sua vez, pode ser conectado a qualquer tipo ou relação existente no grafo a partir de uma relação chamada *hasAnnotation*. A Figura 3.16 mostra um exemplo de grafo baseado no modelo PROV-DM, onde podem ser vistas quatro anotações, duas relacionadas às arestas *wasAssocitedWith* informando o papel de cada Agente na Atividade, outra relacionada com a Atividade e a última relacionada com a Entidade gerada.

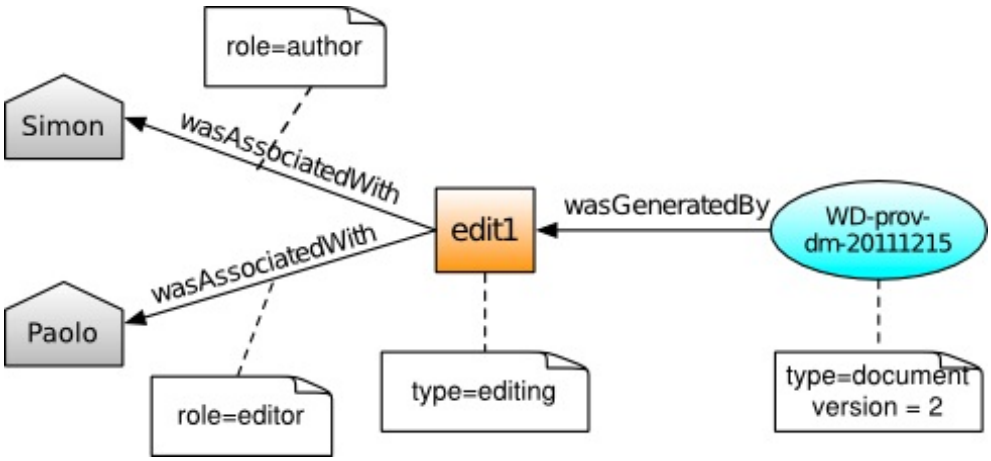


Figura 3.16: Exemplo de grafo do modelo PROV-DM que utiliza Nota (W3C, 2012c).

Trabalhos Relacionados e Avaliação

Como a especificação do modelo PROV-DM é recente, foram encontrados poucos trabalhos publicados na literatura com esse modelo. A seguir tem-se um exemplo de aplicação do modelo PROV-DM:

- Missier e Belhajjame (2012) apresenta uma aplicação de regras de dedução para verificação da validade de grafos de proveniência.

Somente parte do modelo PROV-DM foi descrito nesta seção, levando-se em conta as necessidades específicas deste trabalho. Porém pode-se perceber que o modelo possui as características necessárias (conforme descritas na Seção 3.3) para a representação de proveniência para *workflows* de bioinformática, inclusive Coleções para representar conjuntos de dados. Além disso, tem um conjunto de relações que permite demonstrar a proveniência de forma mais detalhada. Também fornece um conjunto de documentos com

especificações XML e RDF, que visa a troca de informações padronizada, e um conjunto de restrições que auxiliam na validação dos grafos gerados. Além disso, o modelo está sendo desenvolvido no *W3C*, indicativo importante de que pode tornar-se um modelo padrão em proveniência de dados.

Por outro lado, o modelo PROV-DM ainda possui diversas questões indefinidas, conforme descrito no documento do modelo (W3C, 2012c), e ainda está em construção. Por ser um modelo relativamente novo, possui poucas ferramentas disponíveis e poucos trabalhos desenvolvidos, utilizando o modelo como base.

3.3 Comparação Entre os Modelos

Além dos modelos para proveniência apresentados, pode-se citar outros estudos com definições de modelos particulares como Bowers et al. (2006), Groth et al. (2008) e Acar et al. (2010), dentre outros. Logo, pode-se perceber a diversidade de opiniões em relação ao modelo de proveniência, o que torna a escolha de um modelo um desafio.

A fim de melhor direcionar essa análise, pode-se definir os requisitos da aplicação do modelo em projetos de bioinformática. Dessa forma, pretende-se que o resultado deste trabalho seja capaz de produzir uma representação gráfica simplificada, para que o usuário possa rever seus experimentos de forma intuitiva, em qualquer nível de granularidade além de poder comparar estes experimentos entre si. O modelo deve estar bem desenvolvido a ponto de fornecer ferramentas que facilitem a sua utilização. Outra preocupação é a capacidade do modelo de propiciar o intercâmbio de informações entre diferentes sistemas. Isso está diretamente ligado à capacidade de difusão do modelo escolhido, podendo tornar-se um padrão para proveniência de dados.

Levando-se em conta as necessidades descritas, são apresentados alguns tópicos de avaliação entre os modelos discutidos neste trabalho, o que tornará mais factível a escolha do modelo de proveniência.

3.3.1 Objetivo do Modelo

O modelo *Provenance Vocabulary* foi desenvolvido com o objetivo de fornecer proveniência de dados publicados na web, o que traria uma maior dificuldade para sua adaptação em projetos de bioinformática.

Os modelos *W7*, *Provenir Ontology*, OPM e PROV-DM foram projetados para a descrição da sequência de passos na criação ou modificação de um dado, portanto, de acordo com os requisitos.

3.3.2 Representação Gráfica

O modelo *W7* carrega excessivamente a representação gráfica tornando dados acessórios (tempo, localização, etc..) parte principal dela. Além disso, essa sobrecarga causa problemas ao representar o encadeamento de eventos onde um determinado dado é criado por um evento e alterado por outro.

Tanto os modelos *W7* como *Provenir Ontology* e *Provenance Vocabulary* não tem símbolos diferentes para representar os diversos componentes (Dado, Agente, Processo,

etc...) de seus modelos. Isso torna a representação gráfica confusa e difícil de interpretar como pode ser visto nas Figuras 3.3 e 3.5.

Os modelos *Provenance Vocabulary*, *Provenir Ontology* e PROV-DM definem os elementos *File*, *File Collection* e *Colletion*, respectivamente, para representar conjuntos de dados. Estes elementos permitem representar grandes conjuntos de dados a partir de um único elemento, característica necessária para a aplicação em projetos de bioinformática.

Tanto o modelo OPM quanto o PROV-DM tem uma representação gráfica simplificada com diferentes símbolos para cada elemento principal e suas arestas indicam claramente a ligação entre eles. Os demais dados (tempo, localização, etc...) podem ser mostrados juntos com os elementos do gráfico ou ocultados, dependendo do nível de granularidade desejado.

3.3.3 Amadurecimento do Modelo e Ferramentas Disponíveis

Esta seção procura demonstrar o grau de amadurecimento dos modelos através de documentação disponibilizada para que se possa estudá-lo e utilizá-lo. Além da documentação com a especificação de cada modelo, foi feita uma pesquisa procurando sites que fornecessem informações mais detalhadas, principalmente um site específico do modelo, o que mostra maior possibilidade de disseminação e facilita o seu desenvolvimento. A partir dos sites encontrados, procurou-se por ferramentas disponíveis para que usuários em potencial pudessem ter apoio na utilização do modelo. Um resumo dessa investigação pode ser visto na Tabela 3.3.

Dos modelos investigados, apenas para o modelo W7 não foi encontrado uma página na Internet. Os modelos OPM e PROV-DM possuem, além do site específico, extenso material disponível e fóruns de discussão.

3.3.4 Considerações Finais

Diversas características dos modelos foram avaliadas, levando-se em conta a sua aplicação em projetos de bioinformática. Tais características foram divididas em duas tabelas apresentadas as seguir.

A Tabela 3.4 mostras algumas características técnicas avaliadas para os modelos estudados.

O primeiro critério avaliado foi o objetivo do modelo. Os modelos W7, *Provenir Ontology* e PROV-DM fornecem uma abordagem genérica enquanto que o modelo *Provenance Vocabulary* foi projetado com o objetivo de rastrear informações em ambiente distribuído, dessa forma, seus elementos e relações não fornecem características adequadas para demonstrar o histórico da geração de um determinado dado. Os demais modelos são voltados especificamente para esse fim, o que vem de encontro à necessidade da demonstração da proveniência em projetos de bioinformática.

O segundo critério diz respeito à capacidade da representação gráfica dos modelos. Somente os modelos OPM e PROV-DM possuem uma representação gráfica adequada para a construção de grafos de proveniência mais claros e de fácil visualização. Porém, o modelo OPM não possui elementos para a representação de coleções de dados. Em relação à quantidade de elementos e relações, todos os modelos apresentam características suficientes para a aplicação.

Tabela 3.3: Resumo da busca dos modelos na web.

Modelo	**Página**	**Ferramentas**	**Outros**
W7	Não Encontrada	Não Encontrada	-
Provenance Vocabulary	Sim[1]	Não Encontrada	-
Provenir Ontology	Sim[2]	Não Encontrada	-
OPM	Sim[3]	- OPM Toolbox[5] - Tupelo[6] - Taverna[6] - ProvenanceJS[6] - VisTrails[6] - Swift[6] - eBioFlow[5] - Karma[5] - ourSpaces[6] - OPMProv[5] - Kepler provenance listener[6] - PLIER[6] - Living Knowledge[6] - WebN+1[6]	Site Wiki[8] Tutoriais[9] Forum[10]
PROV-DM	Sim[4]	- ProvToolbox[7] - PROV-JSON toolbox[7]	Site Wiki[11] Forum[12]
[1]`http://trdf.sourceforge.net/provenance/ns.html` [2]`http://wiki.knoesis.org/index.php/Provenir_Ontology` [3]`http://openprovenance.org/` [4]`http://www.w3.org/TR/prov-dm/`			
[5]Ferramentas que implementam o modelo OPM [6]Ferramentas que permitem importar, exportar ou utilizar proveniência baseada no modelo [7]Ferramentas java e python que implementam funcionalidades do modelo			
[8]`http://twiki.ipaw.info/bin/view/OPM/` [9]`http://openprovenance.org/tutorial/` [10]`http://mailman.ecs.soton.ac.uk/pipermail/provenance-challenge-ipaw-info/` [11]`http://www.w3.org/2011/prov/wiki/Main_Page` [12]`http://lists.w3.org/Archives/Public/public-prov-wg/`			

A Tabela 3.5 mostra características não técnicas avaliadas para os modelos. Apesar de importantes, estas características trazem um certo grau de subjetividade em sua análise.

O primeiro critério refere-se ao material disponível. Somente os modelos OPM e PROV-DM tem definições específicas para o tratamento de restrições. Além disso, foi atribuída uma classificação do material disponível para os usuários. Também os modelos OPM e PROV-DM possuem extenso material relacionado à especificação dos modelos, além de diversos materiais de apóio como definições de ontologias, padrão XML e RDF, entre outros.

O segundo critério indica a capacidade de difusão dos modelos. Os modelos OPM e

Tabela 3.4: Resumo das características técnicas dos modelos.

	W7	**Provenance Vocabulary**	**Provenir Ontology**	**OPM**	**PROV-DM**
Adequação ao objetivo do modelo	Sim	Não	Sim	Sim	Sim
Símbolos gráficos adequados	Não	Não	Não	Sim	Sim
Representação para conjunto de dados	Não	Sim	Sim	Não	Sim
Definição de restrições	Não	Não	Não	Sim	Sim

Tabela 3.5: Resumo das características não técnicas dos modelos.

	W7	**Provenance Vocabulary**	**Provenir Ontology**	**OPM**	**PROV-DM**
Material disponível Nota de 0 a 5	1	3	2	5	5
Última versão	-	mar/2012	mai/2011	jul/2010	mai/2012
Fóruns de discussão	Não	Não	Não	Sim	Sim
Provável recomendação por órgão padronizador	-	-	-	-	W3C

PROV-DM possuem fóruns de discussão, que são ferramentas importantes para os usuários discutirem tanto aplicações dos modelos como contribuírem com a evolução do mesmo. Por fim, a provável recomendação pelo W3C demonstra uma importante incentivo ao uso do modelo além de torná-lo um provável padrão em modelos de proveniência.

O modelo PROV-DM destaca-se entre os demais pois possui as características necessárias para a aplicação em projetos de bioinformática. Além disso, este modelo está sendo desenvolvido pelo W3C, o que demonstra um importante indicador de qualidade do modelo.

Capítulo 4

Proveniência de Dados em Projetos de Bioinformática

Este capítulo descreve as implicações e a forma de utilização do modelo PROV-DM para a representação da proveniência de dados em projetos de bioinformática, sendo dividido nas seguinte seções. A Seção 4.1 descreve as principais características dos projetos de bioinformática relacionadas à proveniência de dados e quais as abordagens utilizadas para representação da mesma. A Seção 4.2 define a separação dos dados de proveniência em dois níveis. A Seção 4.3 explica a utilização dos tipos e das relações do modelo PROV-DM em projetos de bioinformática. A Seção 4.4 define as regras que devem ser obedecidas para a construção dos grafos de proveniência. Por Fim, a Seção 4.5 descreve as inferências que podem ser feitas, considerando as restrições.

A fim de tornar mais fácil a compreensão deste capítulo alguns termos devem ser melhor definidos. O termo processo é utilizado neste capítulo para representar a execução de um único programa dentro de um projeto de bioinformática. O termo experimento é utilizado para representar a execução de uma série de processos com um objetivo em comum.

4.1 Visão Geral

Projetos de bioinformática têm como uma de suas principais características o grande volume de dados processados. Além disso, a execução de um experimento pode percorrer um fluxo de dados entre diversos arquivos com diferentes formatos.

Outra característica importante dos projetos de bioinformática diz respeito à variedade de processos que podem ser executados, com diferentes opções de programas e diferentes formas de uso através de parâmetros e configurações.

Tais características afetam de forma significativa a análise dos resultados. Assim, manter a proveniência de dados em projetos de bioinformática requer uma solução que permita armazenar a ligação entre os dados processados, juntamente com as informações das execuções de cada processo e de seus resultados.

Em laboratórios onde diversos usuários podem estar trabalhando em diferentes projetos, torna-se importante o registro de quem atuou na execução de cada experimento. Tão importante quanto o nome do usuário, a função exercida por ele na execução do experi-

mento pode trazer maior confiabilidade aos resultados. Assim, pode-se ter, por exemplo, um assistente executando os processos e um biólogo validando os resultados obtidos.

Assim, os dados, os processos executados e as pessoas envolvidas formam os três pilares do modelo PROV-DM. Para completar o modelo, existem ainda as relações que ligam cada uma destas entidades, além do próprio grafo de proveniência, que representa um agrupamento de um ou mais nós e as suas relações.

O estudo da proveniência de dados fornece ferramentas para auxiliar na tarefa de manter e visualizar experimentos anteriormente executados. Para melhor entender a aplicação da proveniência de dados em projetos de bioinformática, pode-se dividí-la em três dimensões: captura, armazenamento e gerenciamento. Conforme pode ser visto na Figura 4.1, estas dimensões não são isoladas, mas se sobrepõem em muitos aspectos. Dessa forma, pode-se afirmar que os dados capturados afetam a forma de armazenamento que, por sua vez, afeta a forma de gerenciamento desses dados.

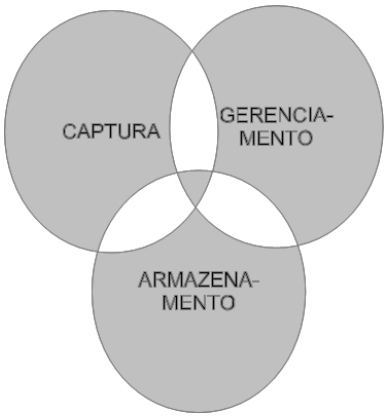


Figura 4.1: Sobreposição das dimensões do trabalho com proveniência de dados.

Alguns trabalhos apresentados na literatura, tais como Gomes (2011) e Marinho et al. (2009), definem a captura de dados de proveniência, de forma automática, através de módulos de importação específicos, executados a nível de *workflow* ou de atividade. Em nosso trabalho, considerando como público alvo projetos de bioinformática de porte pequeno e médio, a captura dos dados de proveniência será feita de forma manual a nível de atividade. Em relação a forma de obtenção, parte das informações é capturada pela abordagem ansiosa (*"Eager"*) enquanto outra será pela abordagem preguiçosa (*"Lazy"*).

Em relação à definição da forma de armazenamento dos dados de proveniência, deve ser levado em conta a grande quantidade de dados que são tratados em projetos de bioinformática. Dessa forma, o armazenamento dos dados de proveniência não podem agravar tal situação, criando duplicação de dados em qualquer outra plataforma. Assim, a exemplo do utilizado por Gomes (2011), foi definido que ligações serão mantidas com os próprios arquivos originais do experimento. Além disso, tais arquivos possuem informações importantes e podem ser usados posteriormente (abordagem preguiçosa) em futuras análises e experimentos. Também, a fim de manter maior independência em relação aos programas externos, os dados de proveniência serão armazenados em arquivos XML.

O gerenciamento dos dados de proveniência é a dimensão que mais afeta o usuário final, visto que é a partir dela que o usuário irá recuperar informações que poderão ser úteis em suas análises e direcionar novos experimentos. O primeiro ponto a ser abordado

é a representação gráfica do experimento. Utilizando o modelo PROV-DM como base, é possível criar uma forma simples de visualização do experimento, o que permitirá ao usuário tanto verificar o próprio experimento como compartilhá-lo com outros usuários.

4.1.1 Experimentos e Projetos

O modelo PROV-DM prevê somente a estrutura básica para representar a proveniência de dados. Para uma definição completa se faz necessária a inclusão de dados capazes de caracterizar cada elemento do modelo, além da inclusão de uma entidade capaz de manter agrupados diferentes experimentos que estejam relacionados entre si.

Um grafo de proveniência demonstra uma sequência de passos onde diversos processos foram executados e quais dados foram utilizados e gerados. Dessa forma, cada execução de um experimento é representado por um grafo de proveniência.

Projetos de bioinformática podem possuir diversos experimentos que, por sua vez, podem ser reexecutados diversas vezes com diferentes programas ou parâmetros. Além disso, o usuário pode sentir a necessidade de separar experimentos muito extensos em diferentes fases. Assim foi definido um novo elemento, não previsto no modelo PROV-DM, chamada *Project*. A criação deste elemento vem da necessidade de agrupar estas diferentes execuções de experimentos relacionados entre si, permitindo que o usuário crie agrupamento de execuções de experimento.

Pode-se dizer então que um projeto tem basicamente dois objetivos principais: (1) agrupar experimentos relacionados a um projeto específico, conforme pode ser visto na Figura 4.2; (2) caracterizar cada projeto através de seus atributos básicos conforme descrito na Tabela 4.1.

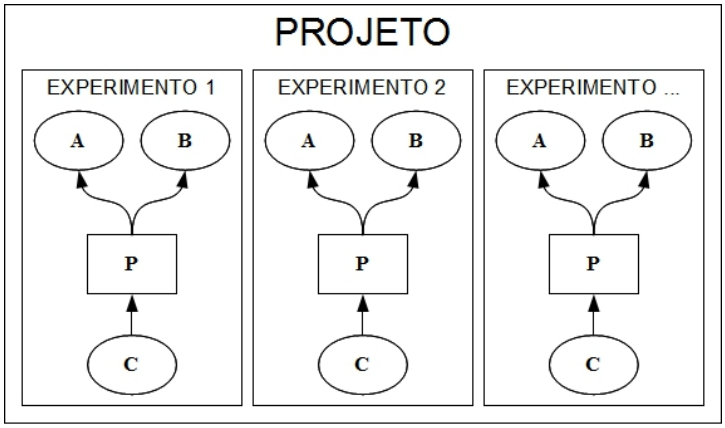


Figura 4.2: Um projeto agrupa diferentes experimentos executados.

4.2 Proveniência em Dois Níveis

Com o objetivo de manter os dados de proveniência estruturados, estes foram divididos em dois níveis.

O Nível 1 corresponde aos dados que representam o grafo de proveniência baseado no modelo PROV-DM. Além disso, esse nível conterá as informações detalhadas de cada nó representado no grafo de proveniência e de suas relações.

O Nível 2, por sua vez, fornecerá acesso aos dados brutos do experimento, tais como, sequências de DNA, alinhamentos encontrados, entre outros. Estes dados são recuperados a partir dos arquivos resultantes de cada processo na execução do experimento.

Dependendo do interesse e dos recursos disponíveis, o usuário poderá definir quais dados deverão ser mantidos. Considerando que o Nível 2 necessita de centenas de gigabytes de espaço em disco, dependendo do experimento executado, o usuário pode ter a necessidade de apagar arquivos intermediários desses experimentos. Contudo, a partir das informações do Nível 1, o usuário terá a capacidade de recuperar a informação de quais arquivos foram gerados e das ligações entre eles, mas não terá acesso aos dados de alguns desses arquivos, quando for tomada a decisão de apagá-los.

Apesar da separação das informações em diferentes níveis, ambos estão interligados, permitindo que seja possível navegar de forma transparente entre os dois níveis. Porém, para que isso seja possível, deve ser tomada a decisão de manter todos os dados armazenados. Caso seja necessário eliminar alguns dados por motivo de espaço em disco, ou por qualquer outro motivo, se o Nível 1 permanecer, toda a descrição do experimento ainda será mantida, além da ligação entre os diversos arquivos processados.

4.2.1 Nível 1: Informações de Alto Nível

O Nível 1 corresponde as informações referentes à formação do grafo de proveniência, ou seja, informações sobre coleções utilizadas, atividades executadas, agentes atuando no experimento, além das relações entre cada elemento. Também corresponde aos dados gerais do próprio experimento e do projeto ao qual ele pertence.

As informações do Nível 1 são obtidas através da abordagem ansiosa (*"Eager"*), ou seja, são armazenadas durante a execução do experimento, ficando disponível para consulta pelo usuário em tempo real.

Estas informações podem ser consultadas de duas formas, gráfica ou textual. A forma gráfica corresponde ao grafo de proveniência do experimento, enquanto que a forma textual corresponde às informações de cada elemento armazenadas através dos formulários preenchidos pelo usuário.

O primeiro problema que se encontra para a representação gráfica desse tipo de experimento é o grande volume de dados que podem ser representados como entidades (descritos na Seção 3.2.5). Isto tornaria impossível a visualização de qualquer experimento desse tipo uma vez que são processadas milhões de sequências de DNA ou RNA, cada uma potencialmente sendo uma entidade. Como estes dados estão originalmente agrupados em arquivos, propõe-se que cada arquivo seja representado por uma coleção. Uma coleção representa portanto, um conjunto de entidades ou até mesmo um conjunto de outras coleções. Para completar o modelo também é proposto a utilização da relação (ou aresta, para o grafo), chamada *memberOf*, para indicar a ligação entre uma coleção e o seu conteúdo. Assim, essa relação indica quais entidades ou coleções estão contidos em uma determinada coleção. Apesar da existência da relação *memberOf* no modelo PROV-DM, esta não será exibida no grafo de proveniência e será mantida nas informações de Nível 2, podendo ser recuperada através da utilização dos arquivos originais.

A ligação das demais arestas com as coleções funciona da mesma forma que com as entidades, porém algumas características devem ser observadas para seu completo entendimento nesta aplicação.

No caso das relações *Used* e *WasGeneratedBy*, considera-se que todo o conteúdo da coleção foi usado ou gerado, respectivamente, pela atividade correspondente.

A relação *WasDerivedFrom*, por sua vez, pode ser usada em qualquer combinação entre entidade e coleção. Porém, a ligação com uma coleção em uma das extremidades da relação, implica na necessidade de informações de Nível 2 a fim de permitir a relação exata entre as entidades originais e derivadas. Em outras palavras, as informações de Nível 1 apenas ligam as coleções com os processos e, para recuperar as entidades que compõem uma coleção, é necessário ler os arquivos correspondentes.

Isso é exemplificado na Figura 4.3, onde a Atividade 1 usou a Coleção 1 para a geração da Coleção 2. Dessa forma uma aresta *WasDerivedFrom* indica a derivação entre as duas coleções. Essas informações são armazenadas portanto, no Nível 1. A partir das informações do Nível 2, pode-se descobrir o conteúdo de cada coleção. De acordo com a Figura 4.3, a Entidade 1 e a Entidade 2 são membros da Coleção 1, a Entidade 3 e a Entidade 4 são membros da Coleção 2. Assim, unindo as informações de Nível 1 e 2 é possível descobrir a real derivação entre as entidades.

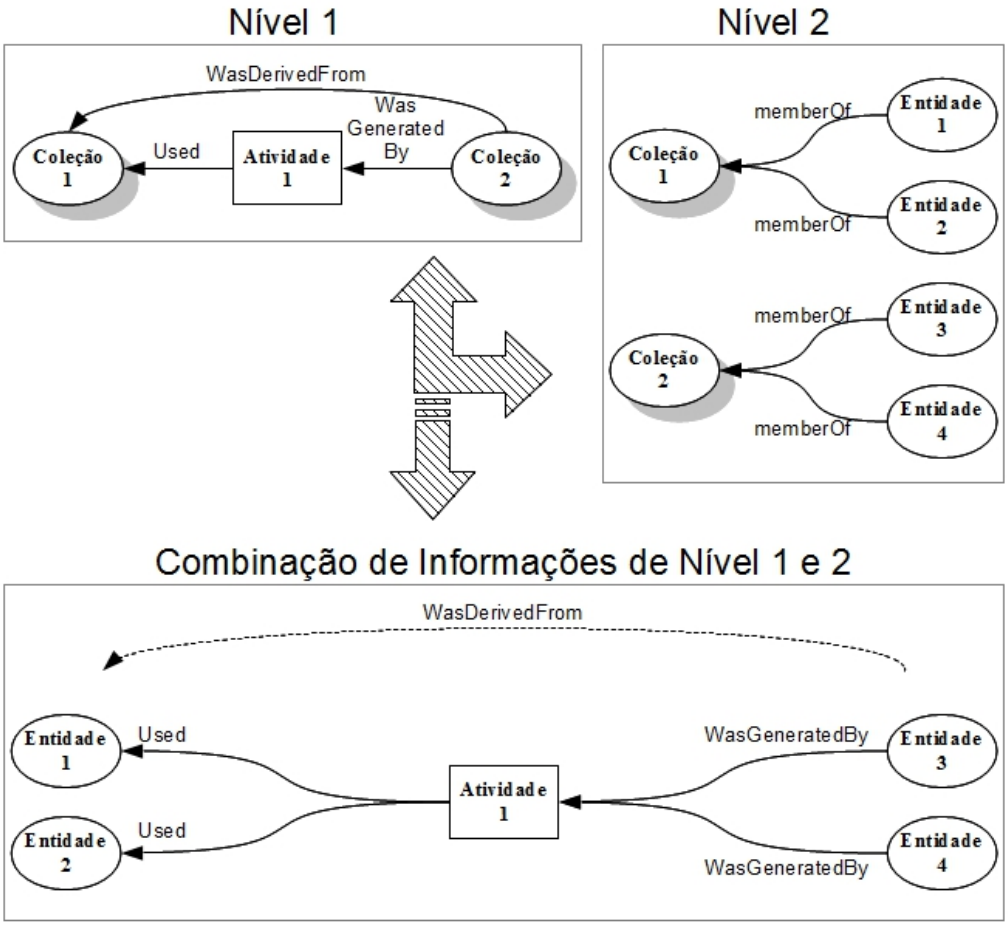


Figura 4.3: A ligação de coleções com a relação *WasDerivedFrom* necessita de informações de Nível 2 para a correta ligação de derivação entre as entidades.

Além da representação gráfica do grafo de proveniência, algumas informações adicionais são necessárias para entender completamente um determinado experimento. As-

sim, os elementos Projeto e Conta foram incluídos no modelo para representar, respectivamente, um projeto de Bioinformática e a execução de um experimento. Também foi incluído um conjunto mínimo de informações relacionadas a cada um dos elementos conforme detalhado na Tabela 4.1.

4.2.2 Nível 2: Conteúdo das Coleções

As informações do Nível 2 correspondem, em volume de dados, a totalidade do espaço em disco utilizado após a execução de um experimento genômico. A execução de cada processo, em um experimento, gera um grande volume de informações que pode, ou não, ser relevantes para futuras análises ou servir de insumo para outras execuções. Assim, quando uma particular execução é concluída, os biólogos têm que decidir quais informações devem ser mantidas e quais devem ser descartadas. Esta decisão torna-se muito importante para a manutenção das informações do Nível 2, uma vez que, para evitar gasto desnecessário de espaço em disco, os próprios arquivos do experimento são usados neste nível para garantir a proveniência.

A recuperação das informações do Nível 2 pode ser tanto a utilização de um arquivo inteiro, gerado durante a execução de um experimento, quanto a solicitação de dados específicos armazenados nestes arquivos. De qualquer forma, a recuperação será feita utilizando-se a abordagem preguiçosa (*"Lazy"*), ou seja, os dados somente serão tratados quando solicitados pelo usuário e podem levar algum tempo para serem preparados, dependendo da consulta requerida.

Outro possível uso para as informações de Nível 2 é a reexecução de um experimento, ou de parte dele. Assim, se os arquivos originais forem mantidos, juntamente com o grafo de proveniência descrevendo a execução do experimento, e ligando os arquivos, torna-se possível reexecutar o experimento de forma automática ou semi-automática.

4.3 Usando o PROV-DM

O modelo PROV-DM define uma série de elementos e relações. Dessa forma, algumas questões devem ser definidas para que a aplicação desse modelo seja completamente entendida. A fim de melhor descrever tais questões, estas serão divididas nos grupos elementos e relações, definidos a seguir.

4.3.1 Gerenciando Elementos

Serão utilizados neste trabalho os elementos Atividade, Agente, Entidade e Coleção que representam os possíveis nós do grafo de proveniência e o elemento conta que representa o próprio grafo de proveniência. Ambos são descritos no modelo PROV-DM. Apesar disso, algumas características relacionadas ao seu uso em aplicações de bioinformática devem ser tratadas.

O primeiro ponto a ser tratado refere-se ao elemento entidade. Este elemento possui três subtipos: o agente, a conta e a coleção. Em nossa aplicação, o elemento agente será utilizado para representar qualquer pessoa, instituição ou serviço que tenha algum tipo de ação sobre uma atividade. O elemento conta será tratado em nossa aplicação como a execução de um experimento, portanto representa um grafo de proveniência.

Tabela 4.1: Informações do grafo de proveniência para projetos de bioinformática.

PROJECT		**ACCOUNT**	
Nome	Nome do projeto	Nome	Nome do experimento
Descrição	Descrição do projeto	Descrição	Descrição do experimento
Instituições Financiadoras	Lista das instituições que financiaram o projeto	Local	Local de execução
		Data Inicial	Data inicial da execução
Instituições Participantes	Lista das instituições que participam do projeto	Data Final	Data final da execução
		Versão e Data	Número e data da versão gravada
Coordenador	Nome do coordenador do projeto	Anotações	Qualquer informação adicional sobre o experimento
Data Inicial	Data de início do projeto		
Data Final	Data final do projeto		
AGENT		**ACTIVITY**	
Nome	Nome do usuário	Nome	Nome da atividade
Instituição	Instituição do usuário	Programa	Nome do programa
Cargo	Cargo ou função	Versão	Versão do programa
Função	Função no experimento	Comando	Linha de comando com os parâmetros utilizados
Grupos	Grupos para filtrar o grafo de proveniência	Função	Descrição do que a atividade executou
Anotações	Qualquer informação adicional sobre o agente	Hora Inicial	Data e hora em que a atividade foi iniciada
		Hora Final	Data e hora em que a atividade foi concluída
		Ambiente	Descrição do ambiente computacional em que a atividade rodou
		Grupos	Grupos para filtrar o grafo de proveniência
		Anotações	Qualquer informação adicional sobre a atividade
COLLECTION		**ENTITY**	
Nome	Nome da coleção	Nome	Nome da entidade
Tamanho	Número de entidades contidas na coleção	Descrição	Descrição do conteúdo da entidade
Descrição	Descrição do conteúdo da coleção	Localização	Localização do arquivo ou banco de dados que guarda a entidade
Localização	Localização do arquivo ou banco de dados que guarda a coleção	Grupos	Grupos para filtrar o grafo de proveniência
Grupos	Grupos para filtrar o grafo de proveniência	Anotações	Qualquer informação adicional sobre a entidade
Anotações	Qualquer informação adicional sobre a coleção		

As coleções representam arquivos utilizados no experimentos e sua função principal é fazer a ligação entre as informações de Nível 1 e 2. Além de representar os arquivos no grafo de proveniência, as coleções também são utilizadas quando o usuário solicitar informações

referentes ao seu conteúdo. Neste caso, os arquivos devem ser lidos e traduzidos para representar o seu conteúdo no formato de entidades. Estas entidades podem ser usadas diretamente ou, quando possível, podem ser usadas para descobrir outras entidades que foram utilizadas na sua criação. Em outras palavras, podem ser usadas para descobrir de quais entidades elas derivaram.

Uma das solicitações dos biólogos ao desenvolver este trabalho foi a necessidade de ligar os dados que originaram um determinado resultado. Assim, diversos tipos de consultas relacionadas as coleções podem ser feitas, porém não serão tratadas especificamente neste trabalho. Pretende-se apenas demonstrar que, na maioria dos casos em projetos de bioinformática, as relações armazenadas no Nível 1, juntamente com os arquivos usados e gerados durante a execução do experimento, são suficientes para permitir a ligação entre os dados envolvidos.

A fim de exemplificar estas ligações pode-se citar a relação entre arquivos contendo sequências de DNA e um arquivo contendo os alinhamentos feitos entre elas. Conforme descrito na Seção 2.3.1, projetos de bioinformática inicialmente geram arquivos padronizados, por exemplo, FASTQ e SAM. Os arquivos FASTQ possuem identificadores para cada sequência contida dentro dele. Estes identificadores são passados para os arquivos SAM quando é feito o alinhamento entre diferentes arquivos, indicando quais sequências alinharam entre si. Dessa forma é possível rastrear as sequências alinhadas até os arquivos originais.

A Figura 4.4 demonstra a ligação com um exemplo onde dois arquivos de sequência (FASTQ1 e FASTQ2) são utilizados em um processo de alinhamento que, por sua vez, gerou um arquivo com os alinhamentos encontrados (SAM1). Unindo as informações de Nível 1 e 2 é possível descobrir as três sequências que foram alinhadas (A, B e D) fazendo a ligação de derivação corretamente. Da mesma forma é possível descobrir quais sequências não tiveram nenhum alinhamento (C, por exemplo). Para que isso seja possível na prática, três condições devem ser satisfeitas: (1) o arquivo derivado deve conter algum tipo de identificador que aponte para a entidade no arquivo original; (2) todos os arquivos envolvidos devem estar armazenados e acessíveis; e (3) devem ser desenvolvidos programas específicos para a leitura de cada tipo de arquivo.

Um exemplo do uso dessas ligações poderia ser a geração de uma nova coleção somente com as sequências que tiveram algum alinhamento no processo. Conforme pode ser visto na Figura 4.5, após a identificação das sequências que tiveram alinhamento, foi executada uma atividade de filtro, gerando uma nova coleção FASTQ3 somente com as sequências A, B e D. O Nível 1 representa o processo de criação desta coleção e o Nível 2 mostra o seu conteúdo.

O elemento entidade, por sua vez, será usado somente para representar um dado unitário como uma sequência de DNA ou um alinhamento, por exemplo. Na execução de experimentos em projetos de bioinformática, seu uso somente ocorrerá quando o usuário solicitar algum tipo de consulta sobre o conteúdo de uma coleção. De qualquer forma, em nenhum momento será possível exibir graficamente todas as entidades devido à grande quantidade, sendo possível apenas mostrar as coleções.

O outro elemento primário do modelo PROV-DM é a atividade. Na aplicação em projetos de bioinformática, uma atividade representa qualquer processo que possa ser executado em um experimento. Conforme mostrado na Tabela 4.1, este elemento indicará as propriedades do programa executado, incluindo linhas de comando, nome do programa,

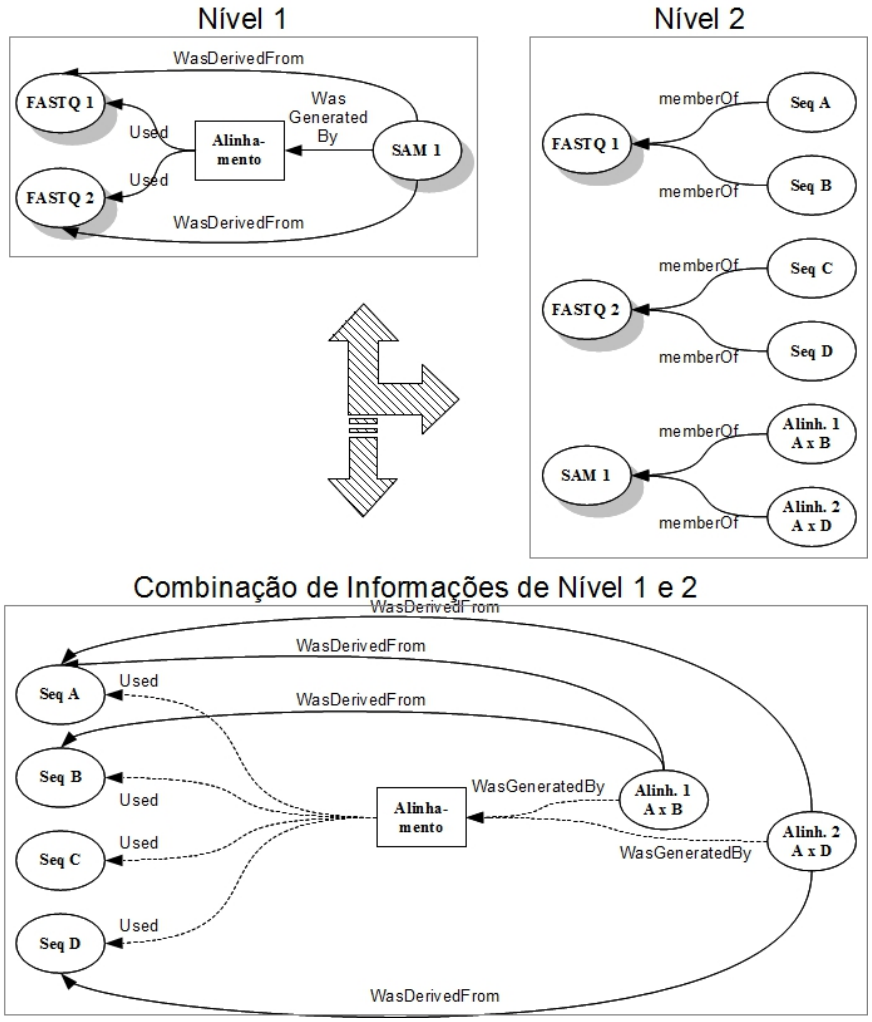


Figura 4.4: Exemplo de ligação entre arquivos de sequências (FASTQ) e alinhamento (SAM).

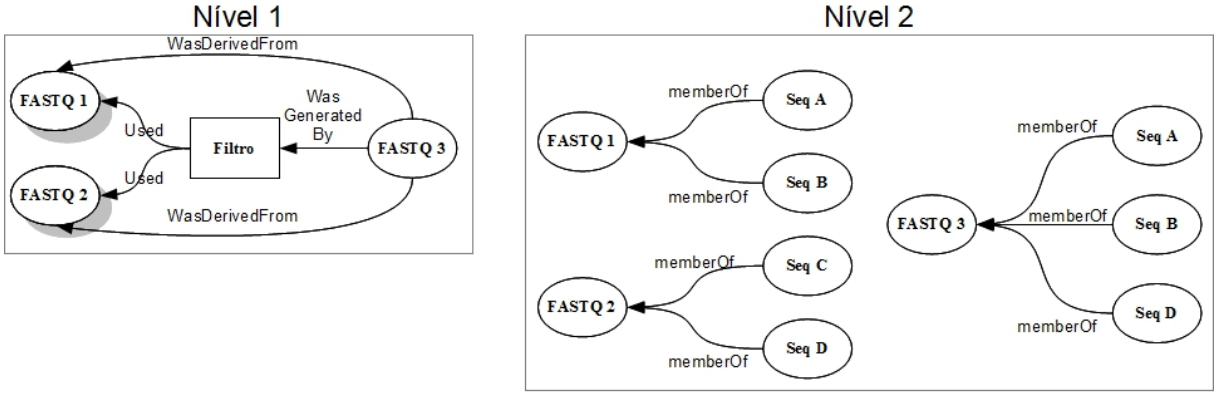


Figura 4.5: Exemplo de criação de uma nova coleção filtrando as sequências desejadas.

versão, entre outros. Também vale ressaltar que, dos elementos que compõem o grafo (coleção, agente, entidade e atividade), este é o único que possui características temporais

(hora inicial e hora final).

Por fim, com o objetivo de propiciar a criação de grafos de proveniência personalizados a partir dos experimentos executados, foi definido um atributo chamado *Grupo*. Este atributo está presente nos possíveis nós do grafo, ou seja, nos elementos coleção, entidade, agente e atividade. Os grupos são definidos pelo usuário durante o cadastramento de cada elemento e, por ser um atributo múltiplo, cada elemento pode fazer parte de diferentes grupos. Dessa forma, o usuário poderá solicitar a visualização de parte do grafo escolhendo os grupos desejados. Esse recurso é especialmente importante para grafos muito grandes onde o usuário pode querer visualizar somente parte dele.

4.3.2 Gerenciando Relações

As relações no modelo PROV-DM representam as arestas que ligam os diferentes nós do grafo de proveniência. Estas arestas são direcionadas e sempre partem do destino para a origem, assim demonstram a proveniência retrocedendo, a partir do objeto foco da proveniência, através dos eventos ocorridos na sua geração. O modelo PROV-DM descreve diversos tipos de relações, porém, visando aplicá-lo em projetos de bioinformática, algumas definições foram adotadas.

A primeira relação a ser considerada é a *Used*. Esta relação liga uma coleção ou uma entidade à atividade que está usando esta coleção ou entidade. Como uma aresta direcionada, sempre vai da atividade para a coleção ou entidade. Cada coleção ou entidade poderá ser usada em diversas atividades, da mesma forma que cada atividade poderá usar diversas coleções ou entidades. Isso está de acordo com os experimentos executados em projetos de bioinformática onde um determinado arquivo pode ser usado por diferentes processos, como no caso de um genoma de referência, por exemplo, que pode ser utilizado para o alinhamento com diferentes arquivos com *reads* em diferentes processos de alinhamento. Também um processo pode utilizar vários arquivos, como um processo de mesclagem, por exemplo, onde dois ou mais arquivos com *reads* podem ser unidos formando um arquivo único.

A relação *WasGeneratedBy*, por sua vez, indica qual atividade gerou uma determinada coleção ou entidade. Seu direcionamento, como aresta, parte da coleção ou entidade para a atividade que as gerou. Uma atividade poderá gerar diversas coleções ou atividades, porém cada coleção ou entidade somente pode ser gerada por uma única atividade. Dessa forma diversas relações *WasGeneratedBy* poderão ser feitas para uma atividade, porém somente uma poderá ser feita para cada coleção ou entidade. Essa definição corresponde aos processos executados em projetos de bioinformática, onde um determinado dado (ou arquivo) somente pode ter sido gerado pela execução de um único programa.

A relação *WasAssociatedWith*, por sua vez, indica que um agente teve algum tipo de ação sobre uma atividade. Seu direcionamento parte da atividade para o agente. Também são permitidas múltiplas ligações entre diferentes agentes e atividades.

A relação *WasDerivedFrom* indica a ligação de derivação, durante a execução do experimento, entre os dados utilizados e gerados. Segundo o modelo PROV-DM, esta relação possui diversos subtipos tais como *wasQuotedFrom* e *specializationOf*. Em projetos de bioinformática, diferentes processos podem ser executados, tais como filtros, ordenações, alinhamentos, entre outros. Além disso, alguns processos podem executar mais de uma função, como filtrar e ordenar um arquivo em uma mesma execução. Dessa forma, as

subdivisões previstas no modelo não são capazes de representar de forma adequada os processos utilizados.

Sendo assim, definiu-se que somente será utilizada a relação primária *WasDerivedFrom*, e será criado um atributo múltiplo chamado *Tipo de Derivação* ligado a esta aresta com as seguintes opções:

- *Filtro*: Indica que a coleção derivada representa parte dos dados da coleção original, separada através de um processo de filtragem;
- *Ordenação*: Indica que a coleção derivada possui os mesmos dados e formato da coleção original, porém foram rearranjados segundo algum critério de ordenação;
- *Mesclagem*: Indica que a coleção derivada foi criada a partir da união de duas ou mais coleções;
- *Conversão*: Indica que a coleção derivada representa o mesmo conjunto de dados da coleção original, mas seus dados foram convertidos em um formato diferente do original;
- *Outros*: Representa qualquer outro processo não previsto nos anteriores.

O atributo *Tipo de Derivação* pode receber mais de uma opção das citadas anteriormente. Este atributo pode ser útil quando o usuário solicita informações sobre o conteúdo das coleções e quando esta consulta necessita ler diferentes arquivos, retrocedendo no grafo de proveniência através das ligações de derivação para as coleções originais. Quando isso ocorre, pode-se ignorar algumas ligações do tipo filtro ou ordenação, por exemplo, já que os dados nas coleções originais e derivadas são os mesmos. Além disso, mesmo que o usuário apague esses arquivos intermediários, ainda assim seria possível, usando essas características das derivações, fazer a ligação até os arquivos originais. Dessa forma o processo de recuperação pode ser otimizado, tanto em tempo quanto em espaço em disco. Outra vantagem desse atributo refere-se a uma maior riqueza de detalhamento do grafo de proveniência, facilitando a sua interpretação.

A relação *WasDerivedFrom*, assim como as demais, representa uma aresta direcionada que aponta da coleção ou entidade derivada para a respectiva coleção ou entidade original. Também são permitidas múltiplas ligações entre coleções e entidades, já que uma coleção ou entidade pode ser derivada de diversas outras coleções ou entidades, assim como uma coleção ou entidade pode ser utilizada na geração de diversas outras coleções ou entidades.

Levando-se em conta que as relações *Used*, *WasGeneratedBy* e *WasAssociatedWith* podem ter múltiplas ligações, e que identificar essas ligações pode trazer informações importantes sobre o grafo de proveniência, foi incluído um atributo capaz de propiciar a identificação de cada uma destas ligações. Esse atributo é chamado de *Regra* (*Role*). Dessa forma, essas quatro relações tem associadas a elas o atributo regra, cujo valor será definido pelo usuário e indicará a natureza da ligação.

4.4 Restrições

A partir das definições descritas até aqui é possível a construção de diversos grafos de proveniência. Porém, não é possível garantir que estes grafos representem execuções de

experimentos possíveis no mundo real. O modelo PROV-DM prevê uma série de restrições genéricas, conforme descrito em W3C (2012a).

Estas restrições são importantes ferramentas que permitem a validação dos grafos de proveniência gerados e podem ajudar na construção de grafos mais confiáveis.

Algumas restrições já foram descritas brevemente neste capítulo, porém, nesta seção são reunidas todas as restrições utilizadas para a validação de grafos de proveniência em projetos de bioinformática neste trabalho. As restrições são divididas em três tipos, estruturais, temporais e funcionais, sendo descritas a seguir.

4.4.1 Restrições Estruturais

As restrições estruturais permitem a construção de um grafo de proveniência mais conciso, evitando ambiguidades e garantindo a reprodução da execução de experimentos em projetos de bioinformática.

A restrição mais importante para manter o grafo de proveniência estruturado diz respeito à obrigatoriedade de que cada elemento do grafo tenha um identificador único, inclusive o próprio grafo. Isso facilita a criação das relações utilizando-se os identificadores dos elementos pretendidos.

Outra restrição estrutural diz respeito à criação de relações válidas no grafo de proveniência. Uma relação somente pode ser criada a partir de dois nós existentes e válidos. Por nó existente entende-se um nó já criado e inserido no experimento, e por válido entende-se um nó do tipo esperado para aquela relação. Por exemplo, para se criar uma relação *WasGeneratedBy* precisam existir um nó do tipo atividade e outro do tipo coleção ou entidade e, devido a relação *WasGeneratedBy* representar uma aresta direcionada, sua origem deve ser, obrigatoriamente, uma coleção ou uma entidade e seu destino deve ser, obrigatoriamente, uma atividade.

As relações são as principais responsáveis pela estrutura do grafo de proveniência. Assim, definiu-se uma lista de restrições que devem ser observadas em cada tipo de relação:

- Não podem existir duas relações iguais. Por iguais entende-se duas relações de mesmo tipo, mesma origem e mesmo destino;
- Somente será admitida uma relação *WasGeneratedBy* ligada a cada entidade ou coleção contidas no grafo;
- Em uma relação *MemberOf* os elementos origem e destino devem ser diferentes;
- Em uma relação *WasDerivedFrom* os elementos origem e destino devem ser diferentes.

4.4.2 Restrições Temporais

As restrições temporais têm como objetivo evitar a criação de grafos que representem sequências de processos impossíveis de terem ocorrido em um projeto de bioinformática. A Figura 4.6, por exemplo, demonstra um grafo circular onde a Entidade 1 é usada por uma Atividade 1, que por sua vez gerou a Entidade 2, usada pela Atividade 2 para geração da Entidade 1 inicial. Essa situação é possível conforme o seguinte exemplo: a Atividade 1 inicia no instante t e começa a gerar a Entidade 2 no instante $t+1$; a Atividade 2,

por sua vez inicia no instante t+2 e começa a utilizar a Entidade 2 no instante t+3 (a Entidade 2 começou a existir no instante t+1); a geração da Entidade 1 começa em t+4 e em t+5 começa a ser utilizada pela Atividade 1; ou seja, a Atividade 1 foi iniciada em t, mas somente começou a usar a Entidade 1 em t+5, após o início de sua existência em t+4. Apesar de ser possível em algumas situações específicas, este grafo não pode ocorrer em projetos de bioinformática visto que, cada processo deve ser completamente encerrado para a geração de um arquivo.

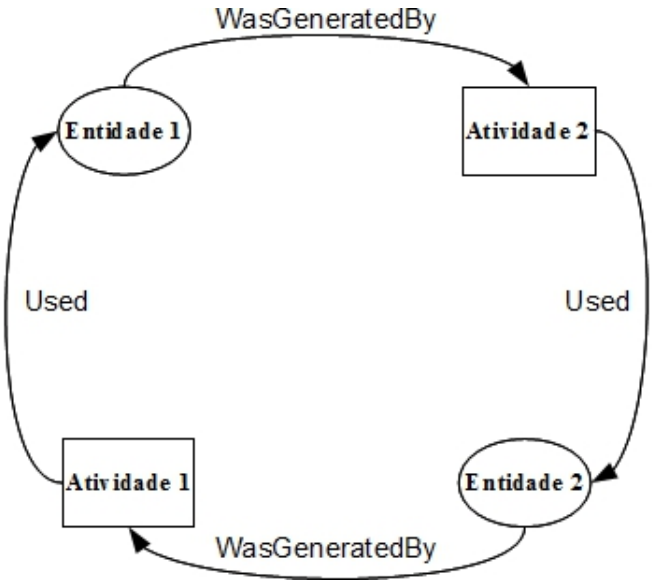


Figura 4.6: Exemplo de um grafo com inconsistência temporal.

A fim de evitar esse tipo de ocorrência, são definidas as restrições temporais. Os termos hora início e hora fim estão se referindo à informação que representa a data e hora do acontecimento. Assim, para fins práticos de validação, o termo hora será considerado como sendo dia, mês, ano, hora, minuto e segundo do momento informado. O termo data inicial será considerado como dia, mês e ano, e para fins de comparação com a hora será considerado a 00:00 deste dia. O termo data final será considerado as 23:59 desta data.

Ainda, para simplificar a criação do grafo de proveniência e ajustar a aplicação para projetos de bioinformática, definiu-se que apenas três elementos terão característica temporal, são eles: o projeto, a conta (que representa o próprio grafo) e a atividade.

Uma atividade representa a execução de algum processo que tem um hora de início e uma hora de fim (hora em que foi completamente concluída). Esta característica é transmitida, quando necessário, aos demais elementos da seguinte forma:

- *WasGeneratedBy*: As entidades e coleções terão como hora de início de sua existência a hora em que a atividade que as gerou for completamente concluída;
- *Used*: As entidades e coleções terão como hora de início de seu uso a hora de início da atividade que as usou;
- *WasAssociatedWith*: A hora de início e de fim da ação de um agente sobre uma atividade é igual a hora inicial e final desta atividade;

- *MemberOf*: Uma coleção ou entidade, que pertence a uma coleção, herdam todas as características temporais desta coleção.

Dadas estas características, as seguintes restrições são adotadas:

- *Atividade*: A hora inicial de uma atividade deve ser maior que a sua hora final;
- *WasGeneratedBy*: Uma coleção ou entidade somente pode ser gerada por uma atividade cuja hora fim seja menor que todas as horas de início do uso desta coleção ou entidade;
- *Used*: Da mesma forma, uma atividade somente poderá usar uma coleção ou uma entidade se a hora inicial dessa atividade for maior que a hora de início da existência desta coleção ou entidade;
- *WasDerivedFrom*: Uma coleção ou entidade somente pode ser derivada de outra coleção ou entidade original se a hora do início do uso da original for menor que a hora da geração da derivada.

Por outro lado, os grafos de proveniência criados pelo usuário podem representar apenas parte da proveniência de um determinado elemento. Desta forma, admite-se que os grafos de proveniência podem ser construídos com coleções ou entidades sem a indicação das atividades que as geraram ou as usaram. Da mesma forma, admite-se também a existência de grafos de proveniência com atividades sem a indicação de uso ou geração de nenhuma coleção ou entidade. Isso é exemplificado na Figura 4.7 onde dois diferentes grafos são apresentados exemplificando diferentes elementos em suas extremidades.

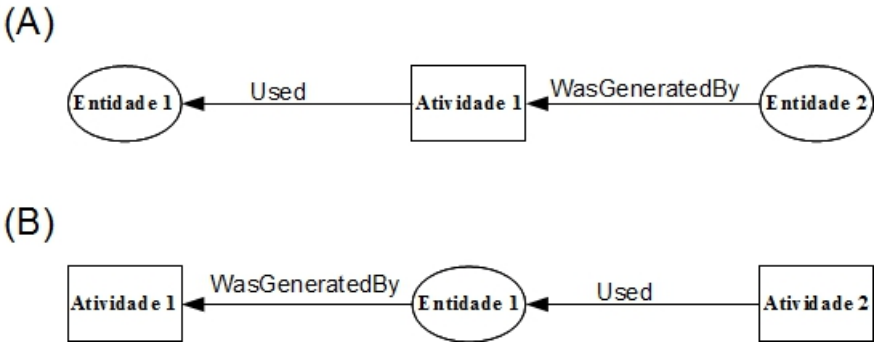


Figura 4.7: Exemplo de grafos com diferentes formações.

A partir dessa afirmação, pode-se concluir que as restrições temporais sobre as relações *Used*, *WasGeneratedBy* e *WasDerivedFrom* somente poderão ser avaliadas quando os elementos possíveis de validação forem localizados. Isso significa que as restrições somente podem ser observadas a partir do momento em que existam ao menos uma coleção ou uma entidade sendo usada por uma atividade e sendo gerada por outra atividade. Portanto, este é um conjunto mínimo para a verificação das restrições temporais. Na Figura 4.8 tem-se um exemplo de grafo que possui um conjunto mínimo de elementos para a verificação destas três restrições temporais, uma vez que estão presentes as características temporais (hora inicial e final de cada atividade) necessárias para a validação.

Além do elemento atividade, mais dois elementos possuem características temporais, a conta e o projeto. Diferente do elemento atividade, esses elementos trabalham somente

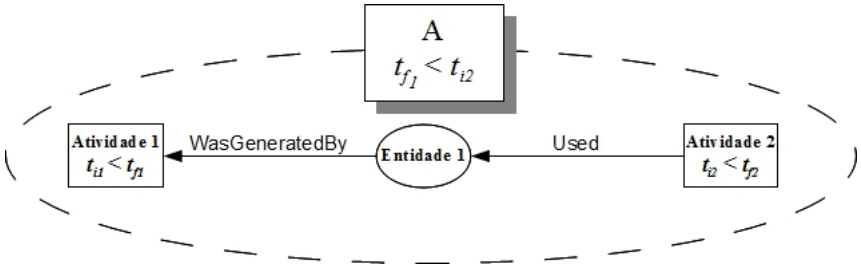


Figura 4.8: Exemplo de conjunto mínimo para validação temporal das arestas.

com a data de início e data de fim de cada um deles. Dessa forma mais quatro restrições são definidas:

- *Conta*: A data de início de uma conta deve ser menor ou igual a sua data final;
- *Conta - Atividade*: Uma conta somente poderá conter atividades que sejam executadas entre a data de início e data de fim da execução desta conta;
- *Projeto*: A data de início de um projeto deve ser menor ou igual a sua data final;
- *Projeto - Conta*: Um projeto somente poderá conter contas que representam experimentos executados entre a data de início e data de fim deste projeto.

4.4.3 Restrições Funcionais

O último tipo de restrição diz respeito a funcionalidade do experimento. As coleções apontam para conjuntos de dados que, normalmente, são arquivos de computador contendo os dados usados e gerados durante a execução do experimento. Essa restrição diz respeito ao correto apontamento dessas coleções. Assim, coleções somente podem apontar para conjuntos de dados válidos, ou seja, se apontar para um arquivo, sua existência deve ser verificada.

4.5 Inferências

As inferências permitem que, a partir da verificação de algumas regras, possa-se chegar a uma determinada conclusão. Em nossa aplicação algumas inferências foram determinadas a partir da observância das restrições descritas anteriormente.

4.5.1 Inferências Estruturais

A primeira inferência que pode ser feita é relacionada a validade estrutural de uma conta. Esta validade pode ser verificada a partir das restrições estruturais da conta. Uma vez que todas as restrições estruturais sejam verificadas e garantidas pode-se inferir que a conta é válida estruturalmente. Caso contrário a conta será inferida como inválida estruturalmente.

Da mesma forma, a validade estrutural de um projeto também pode ser inferida. Assim, se todos as contas pertencentes a um projeto forem válidas estruturalmente, então

o projeto também é válido estruturalmente, caso contrário, se pelo menos uma conta for inválida estruturalmente, o projeto também será considerado inválido estruturalmente.

4.5.2 Inferências Temporais

A primeira inferência temporal a ser analisada diz respeito ao caráter temporal da conta. Assim, as primeiras restrições a serem verificadas são as relacionadas às datas de início e fim da conta. Caso todas estas restrições sejam obedecidas, outras duas situações relacionadas às restrições temporais das atividades devem ser observadas.

A verificação das restrições temporais referentes às relações *Used*, *WasGeneratedBy* e *WasDerivedFrom* somente será possível se existir um conjunto mínimo de nós e arestas. Para as contas que não possuem um conjunto mínimo, esta restrição estará satisfeita.

Para as contas que possuam um ou mais conjuntos mínimos, todos devem ser verificados. Esta restrição somente será satisfeita se todos os conjuntos mínimos satisfizerem as restrições. Na Figura 4.9, tem-se um grafo com três conjuntos mínimos, circundados por elipses pontilhadas e nomeados A, B e C. Dado que, cada atividade tem a sua hora de início e fim (t_i e t_f), as suas restrições foram expressas em cada conjunto mínimo. Assim supondo-se que as restrições em A, B e C são satisfeitas, então tem-se que se $t_{f1} < t_{i2}$ e $t_{f2} < t_{i4}$ e $t_{f3} < t_{i4}$ e , em todo os casos, $t_{in} < t_{fn}$, assim conclui-se que $t_{f1} < t_{i4}$. Dessa forma se as restrições de todas os conjuntos mínimos forem satisfeitas, pode-se afirmar que o grafo, como um todo, satisfaz estas restrições temporais.

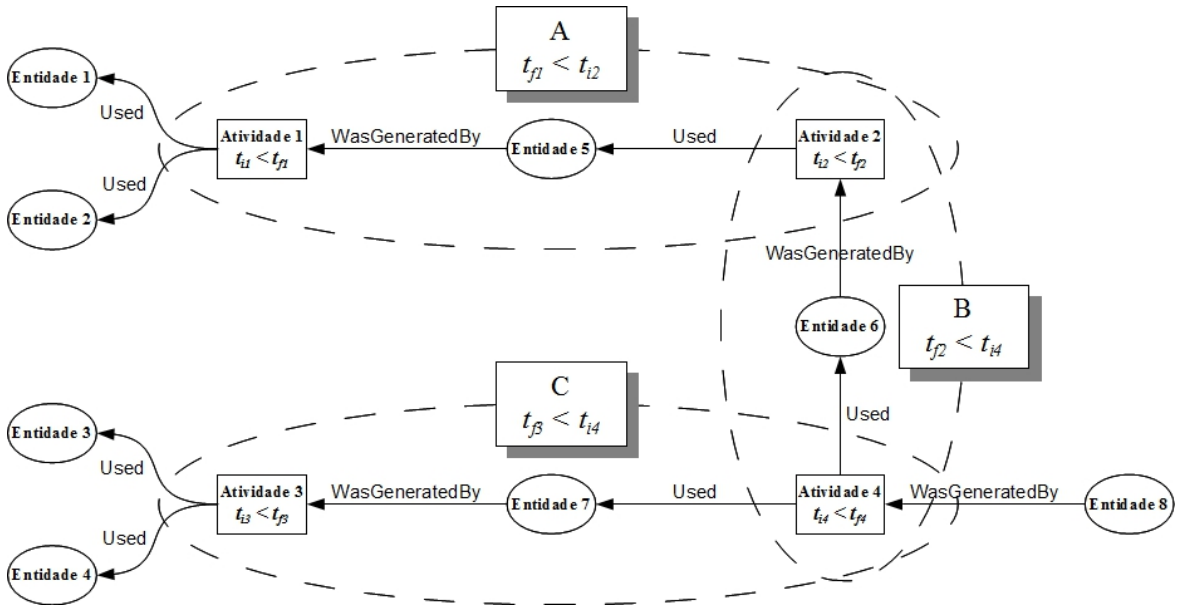


Figura 4.9: Exemplo de grafo com validação temporal.

A Figura 4.9 mostra ainda que as restrições temporais entre as Entidades 1 e 2 e a Atividade 1 não podem ser validadas, uma vez que o grafo não mostra quais atividades geraram estas entidades, não se tendo assim as suas horas de geração. O mesmo ocorre com as Entidades 3 e 4, usadas pela Atividade 3. Situação semelhante ocorre entre a Atividade 4 e a Entidade 8, porém, nesse caso a restrição não pode ser avaliada, pois o

grafo não mostra quais atividades usam essa entidade. Estas situações, por não formarem conjuntos mínimos, não são consideradas para estas restrições.

Assim, para que a conta seja inferida como válida temporalmente, devem ser satisfeitas tanto as restrições temporais de suas datas de início e fim, como também as restrições temporais das atividades e das arestas, caso contrário, é considerada inválida temporalmente.

Um projeto, por sua vez, pode ser inferido como válido temporalmente se as restrições em relação às suas datas de início e fim sejam satisfeitas e todos as contas pertencentes a este projeto sejam válidas temporalmente, caso contrário, o projeto é inferido como inválido temporalmente.

Por fim, uma última inferência que pode ser feita está relacionada à relação *WasDerivedFrom* e pode ser vista na Figura 4.10. Uma vez garantidas as restrições temporais baseadas nos conjuntos mínimos do grafo, pode-se inferir que: dadas duas entidades, E1 e E2, e uma atividade A1, se a entidade E2 foi gerada pela atividade A1, e esta usou a entidade E1, então pode-se inferir que a entidade E2 é derivada da entidade E1.

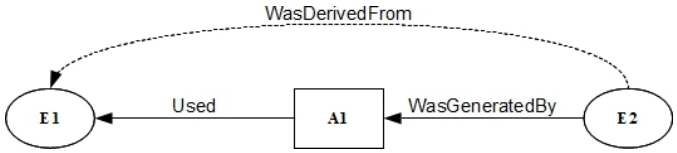


Figura 4.10: Aresta *WasDerivedFrom* entre as entidades E1 e E2 foi inferida a partir das relações *Used* e *WasGeneratedBy* com a atividade A1.

A Figura 4.11 demonstra as inferências feitas a partir da garantia das restrições temporais. As linhas cheias na figura representam a real dependência entre os dados gerados e utilizados. Essa inferência permite a construção de um grafo mais completo e confiável.

4.5.3 Inferências Funcionais

As inferências funcionais dizem respeito às restrições funcionais, assim pode-se inferir que um experimento é válido funcionalmente se todas as suas coleções apontam para conjuntos de dados válidos, caso contrário, será inferido como inválido funcionalmente.

Um experimento considerado inválido funcionalmente não deve ser tratado como um experimento que contenha erros, apenas significa que pelo menos uma de suas coleções aponta para algum conjunto de dados que não está disponível. Como já descrito anteriormente, o usuário pode optar por manter ou não alguns arquivos utilizados no experimento e, no caso de ter excluído algum arquivo, isso tornará o experimento inválido funcionalmente.

Apesar de não ser considerado um erro, a avaliação da validade funcional é importante, pois diz ao usuário que este experimento não pode ser reexecutado pois faltam elementos necessários para sua completa reexecução. Além disso, o usuário também terá restrições em relação as consultas que podem ser feitas, uma vez que o conteúdo das coleções está no Nível 2 do modelo e, conforme visto na Seção 4.2.2, estas informações são necessárias para fazer a ligação real entre as coleções e as atividades.

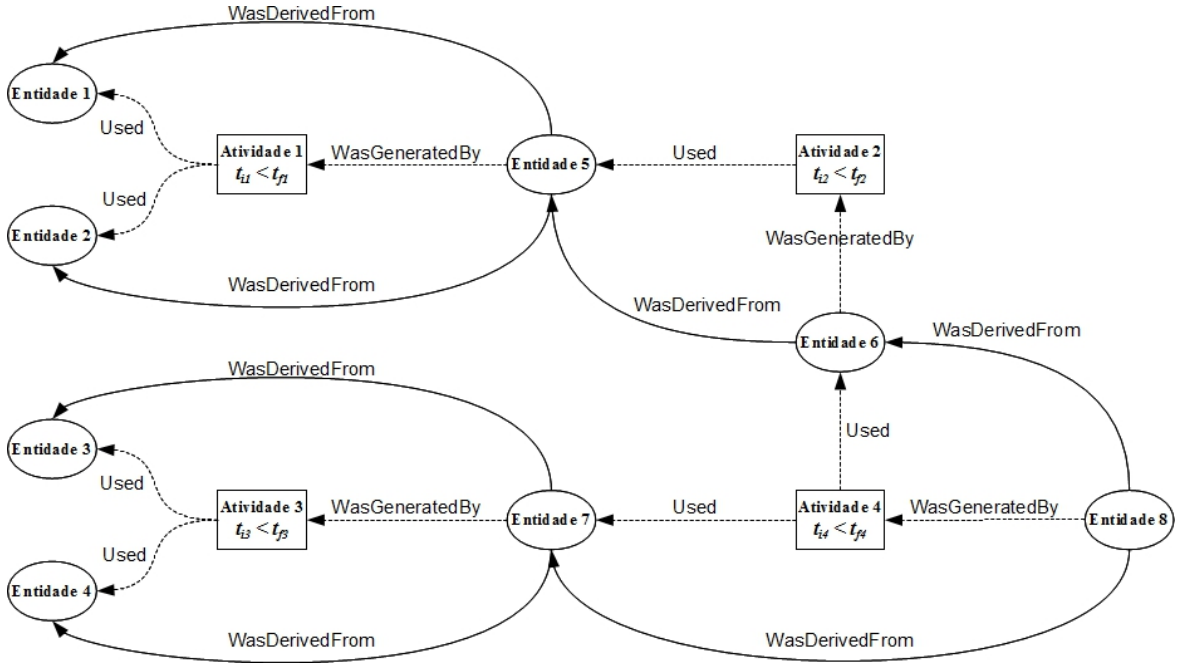


Figura 4.11: Aresta *WasDerivedFrom* foi inferida a partir da validação de restrições temporais.

Da mesma forma, um projeto será considerado válido funcionalmente se todos os experimentos dentro dele forem válidos funcionalmente. Se pelo menos um experimento for considerado inválido funcionalmente, então o projeto também o será.

Capítulo 5

Estudos de Caso em Projetos de Bioinformática

Este capítulo descreve o simulador *Provenance* e a sua aplicação em dois estudos de caso de projetos de bioinformática. O capítulo está dividido da seguinte forma. A Seção 5.1 descreve as principais características do simulador *Provenance*. Na seção 5.2 é demonstrado como o simulador garante as restrições a fim de permitir que as inferências sejam feitas. Por fim a Seção 5.3 demonstra a aplicação do modelo PROV-DM em dois estudos de caso com dados reais.

5.1 Simulador *Provenance*

O simulador *Provenance* foi implementado com o objetivo de demonstrar a validade da solução proposta para projetos de bioinformática. A construção deste simulador foi baseada nas seguintes premissas:

- Implementar as características do modelo de proveniência de dados PROV-DM para projetos de bioinformática;
- Verificar a validade dos grafos de proveniência;
- Criar uma interface simples e amigável para o usuário;
- Facilitar a adição de módulos externos para funções específicas;
- Facilitar o intercâmbio de informações entre diferentes computadores;
- Manter máxima independência da utilização de outros programas;
- Desenvolver uma solução multiplataforma.

Para a implementação do simulador, optou-se pela utilização da linguagem Java, devido a sua natureza multiplataforma, o que facilita a criação ou incorporação de módulos ou pacotes externos tais como BIOJAVA (HOLLAND et al., 2008), OPM Toolbox (OPM, 2011c), Prov Toolbox (MOREAU, 2011), entre outros. Assim, novas funcionalidades podem ser adicionadas ao simulador.

Também, visando a independência de ferramentas externas, optou-se pela gravação dos dados em arquivos XML, o que facilita a troca de informações pois podem ser facilmente copiados e distribuídos entre os usuários.

O simulador foi desenvolvido usando o programa NetBeans IDE 7.1 executando com Java 1.7 com compatibilidade com Java 1.6.

5.1.1 Armazenamento dos Dados

Os dados gravados pelo simulador podem ser divididos em três tipos diferentes: dados de configuração, dados relacionados ao projeto e os dados necessários para a construção dos grafos. Cada tipo é gravado em um arquivo XML diferente.

O primeiro arquivo XML gravado é o de configuração do simulador, e seu esquema pode ser visto na Figura 5.1. Esse arquivo é gravado automaticamente quando o usuário executa o simulador pela primeira vez. O arquivo terá sempre o nome *Provenance_Config.XML* e deve ficar armazenado na mesma pasta que o programa. Os campos podem ser alterados a qualquer momento pelo usuário. Vale destacar aqui o campo *folderProvenance*, no qual o usuário informa uma pasta padrão onde os projetos criados no simulador serão armazenados.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="provenance_config">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="version"                      type="xs:string" minOccurs="1"
                    maxOccurs="1"/>
        <xs:element name="folderProvenance"             type="xs:string" minOccurs="1"
                    maxOccurs="1"/>
        <xs:element name="nameGraphConstructor"         type="xs:string" minOccurs="1"
                    maxOccurs="1"/>
        <xs:element name="folderGraphConstructor"       type="xs:string" minOccurs="1"
                    maxOccurs="1"/>
        <xs:element name="commandLineGraphConstructor" type="xs:string" minOccurs="1"
                    maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figura 5.1: Esquema do arquivo XML de configuração do simulador *Provenance*.

O segundo arquivo corresponde aos dados do projeto do usuário, sendo assim, um arquivo XML é criado para cada projeto gravado. Para cada projeto, também é criada uma subpasta com o mesmo nome do arquivo XML do projeto, para o armazenamento de todos os seus dados e seus diferentes grafos. O esquema desse arquivo pode ser visto na Figura 5.2.

Por fim, o terceiro arquivo XML corresponde ao grafo de proveniência. Diversos grafos podem ser criados e anexados a um projeto, cada um correspondendo a um arquivo XML que fica armazenado na pasta do projeto.

O esquema XML utilizado no simulador é baseado no modelo OPM. Isso ocorre porque, para a construção do simulador *Provenance*, foi utilizado um conjunto de classes chamado OPM4J (OPM, 2011b), que é responsável pela criação dos grafos de proveniência. Apesar do pacote OPM4J ter sido desenvolvido baseado no modelo OPM, este é bastante semelhante ao modelo PROV-DM, sendo que, no esquema XML, as diferenças ficam apenas

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="project">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name"          type="xs:string" minOccurs="1"
                  maxOccurs="1"/>
      <xs:element name="description"   type="xs:string" minOccurs="1"
                  maxOccurs="1"/>
      <xs:element name="funding"       type="xs:string" minOccurs="0"
                  maxOccurs="unbounded"/>
      <xs:element name="participating" type="xs:string" minOccurs="0"
                  maxOccurs="unbounded"/>
      <xs:element name="coordinator"   type="xs:string" minOccurs="1"
                  maxOccurs="1"/>
      <xs:element name="start"         type="xs:date"   minOccurs="1"
                  maxOccurs="1"/>
      <xs:element name="end"           type="xs:date"   minOccurs="1"
                  maxOccurs="1"/>
      <xs:element name="notes"         type="xs:string" maxOccurs="1"
                  maxOccurs="1"/>
      <xs:element name="accounts"      type="xs:string" minOccurs="0"
                  maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Figura 5.2: Esquema do arquivo XML para os dados dos projetos.

nos nomes de alguns elementos. Na Tabela 5.1 tem-se a equivalência entre os nomes dos elementos dos modelos OPM e PROV-DM utilizados neste trabalho.

Tabela 5.1: Equiparação dos elementos dos modelos OPM e PROV-DM.

Tipo	**OPM**	**PROV-DM**
Grafo	*OPMGraph*	*Account*
	Account	—
Nós	*Agent*	*Agent*
	Artifact	*Collection*
	Process	*Activity*
Arestas	*Dependency*	*Relation*
	Used	*Used*
	WasControlledBy	*WasAssociatedWith*
	WasDerivedFrom	*WasDerivedFrom*
	WasGeneratedBy	*WasGeneratedBy*
Outros	*Annotation*	*Annotation*
	Role	*Role*

Como pode ser visto na Tabela 5.1, muitos dos elementos têm o mesmo nome nos dois modelos. Vale destacar que o elemento chamado *Account*, do modelo OPM, não possui correspondente no modelo PROV-DM e aparecerá vazio no arquivo XML do grafo de proveniência. O elemento *Account* do modelo PROV-DM corresponde ao elemento *OPMGraph* do modelo OPM, e ambos representam o conjunto de informações que formam o grafo de proveniência.

O esquema completo para o arquivo de *Account* (do modelo PROV-DM) foi definido em OPM (2010), que descreve o esquema XML para o modelo OPM. Assim, a *Account* é gravado como *OPMGraph* no arquivo XML.

A Figura 5.3 exemplifica a estrutura de configuração e pasta do simulador *Provenance*. No exemplo, o simulador está na pasta local do usuário (\usr) onde também se encontra o arquivo Provenance_Config.XML. Este, por sua vez, tem a informação do local onde os projetos devem ser armazenados (\usr\ProvenanceProjects) e onde foram gravados os arquivos de projeto RimXFigado.XML e CancerProstata.XML. Para cada projeto foi criada uma pasta específica (\usr\ProvenanceProjects\RimXFigado e \usr\ProvenanceProjects\CancerProstata) onde estão os arquivos de cada projeto. No caso do projeto RimXFigado.XML, existem dois grafos (ou *Accounts*) armazenados sendo Alinhamento.XML e ExpressaoDiferencial.XML, além de outros arquivos relacionados ao projeto.

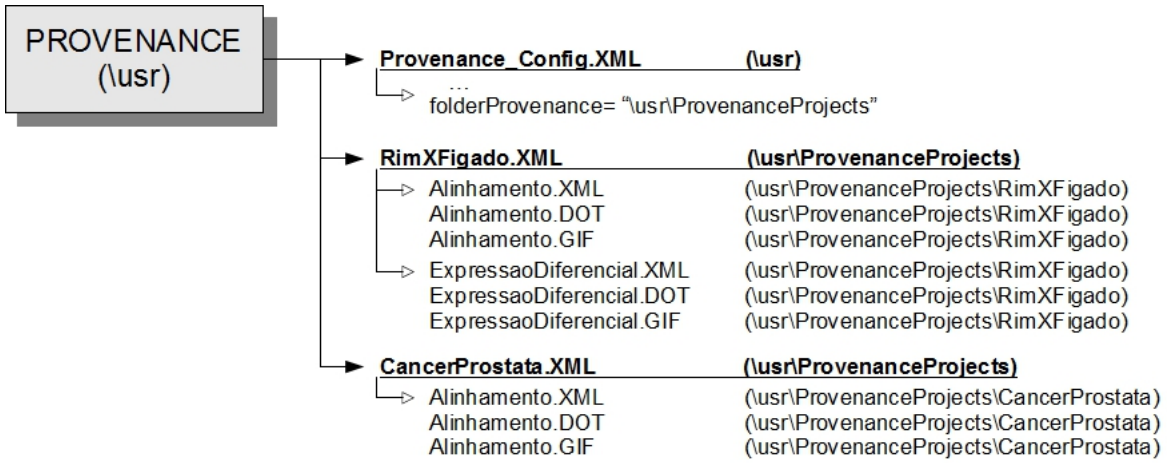


Figura 5.3: Exemplo da estrutura de diretórios e arquivos gravados pelo simulador *Provenance*.

5.1.2 Bibliotecas Externas

Para a construção do simulador foram utilizadas duas bibliotecas externas, a OPM4J (OPM, 2011b) e a GraphViz (ELLSON et al., 2003), descritos a seguir.

OPM4J fornece classes Java para a representação de grafos de proveniência no formato OPM. Esta biblioteca foi utilizada devido ao PROV-DM ser um modelo recente e ainda não possuir ferramentas deste tipo para utilização, além disso os modelos PROV-DM e OPM são muito semelhantes, o que permite uma fácil adaptação para a utilização das classes.

A segunda biblioteca externa é a GraphViz. O GraphViz, ou *Graph Visualization*, é um pacote que fornece diversas ferramentas para tratamento de grafos em diferentes formatos. Um dos recursos fornecidos pelo pacote processa arquivo no formato DOT (KOUTSOFIOS; NORTH, 1993), criando uma imagem do grafo, que pode ser gravado no formato GIF. DOT é uma linguagem textual que permite a especificação de diferentes tipos de diagramas. A Figura 5.4 montra o código que gerou o grafo desenhado. No código, as linhas 2 à 5 definem os quatro nós do grafo enquanto que as linhas 6 à 9

definem as arestas, a linha 10 determina que os nós *a* (*Reads1.fasta*) e *b* (*Reads2.fasta*) fiquem na mesma altura no grafo.

```
1    digraph graphname {
2       a [label="Reads1.fasta"];
3       b [label="Reads2.fasta"];
4       c [label="Filter" shape=box];
5       d [label="FilteredReads.fasta"];
6       a -> c -> d [color=gray];
7       b -> c [color=gray];
8       d -> a [style=dotted];
9       d -> b [style=dotted];
10      { rank=same; a b }
11   }
```

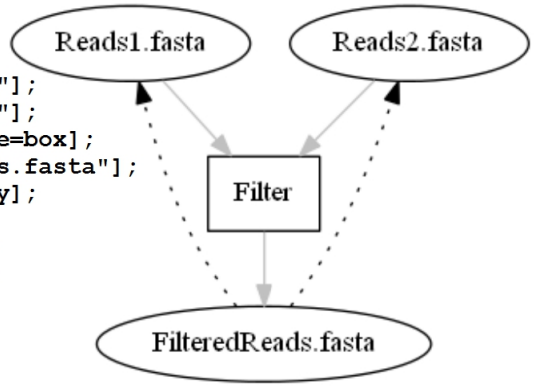


Figura 5.4: Exemplo de um grafo construído a partir da linguagem DOT.

A Figura 5.5 demonstra a integração das bibliotecas OPM4J e do GraphViz com o simulador *Provenance*. As classes da biblioteca OPM4J representam o grafo de proveniência gravando ou lendo seus dados em um arquivo XML. Quando é solicitada a criação do grafo de proveniência, o OPM4J interpreta os dados e grava um arquivo no formato DOT com as instruções para a construção do grafo. A partir desse momento, uma classe da biblioteca do GraphViz lê o arquivo DOT e aciona um programa externo para a criação do grafo em um arquivo no formato GIF (Imagem).

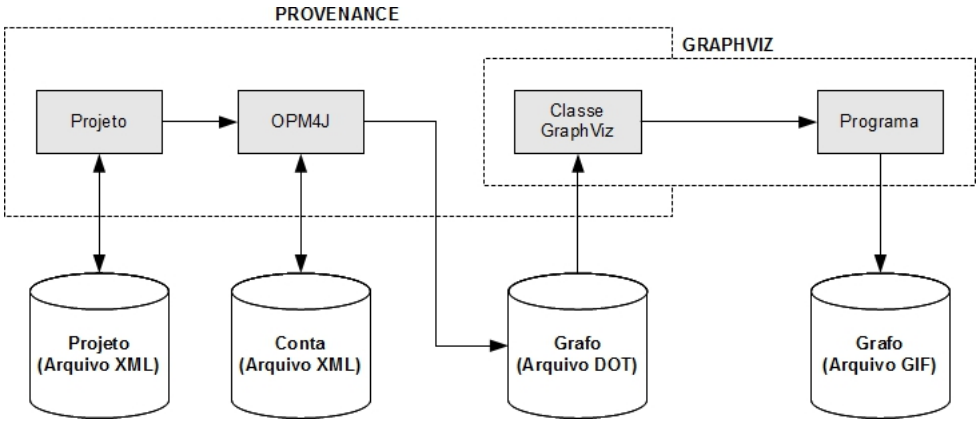


Figura 5.5: Uso das bibliotecas externas para a geração do grafo de proveniência.

Dessa forma, para que o simulador possa ser executado, o usuário precisa instalar em seu computador o JVM (*Java Virtual Machine*) versão 1.6 e o programa GraphViz versão 2.28, ambas disponíveis para diferentes plataformas incluindo Windows, Linux e Mac. Caso o usuário não instale ou não queira utilizar o programa GraphViz, poderá utilizar o simulador *Provenance* e gerar o arquivo no formato DOT da mesma forma. Neste caso, o arquivo DOT gerado pelo simulador pode ser utilizado em qualquer outro programa que interprete essa linguagem para a geração do grafo de proveniência.

5.1.3 Visão Geral de Um Grafo de Proveniência

Esta seção descreve, resumidamente, a tela principal do simulador para a construção de grafos de proveniência. Um guia detalhado sobre o uso do simulador pode ser encontrado no Anexo I deste trabalho.

Ao abrir um grafo (ou *Account)*, o usuário tem acesso à tela que permite criar cada nó do grafo juntamente com as relações existentes entre eles. A Figura 5.6 mostra a tela de um grafo previamente preenchida. Esta tela pode ser dividida em três partes: cadastro geral, relação dos nós e relação das arestas.

O cadastro geral do grafo fica na parte superior esquerda, onde pode ser vista a área com os dados principais do grafo, tais como nome, data da execução, local, entre outros.

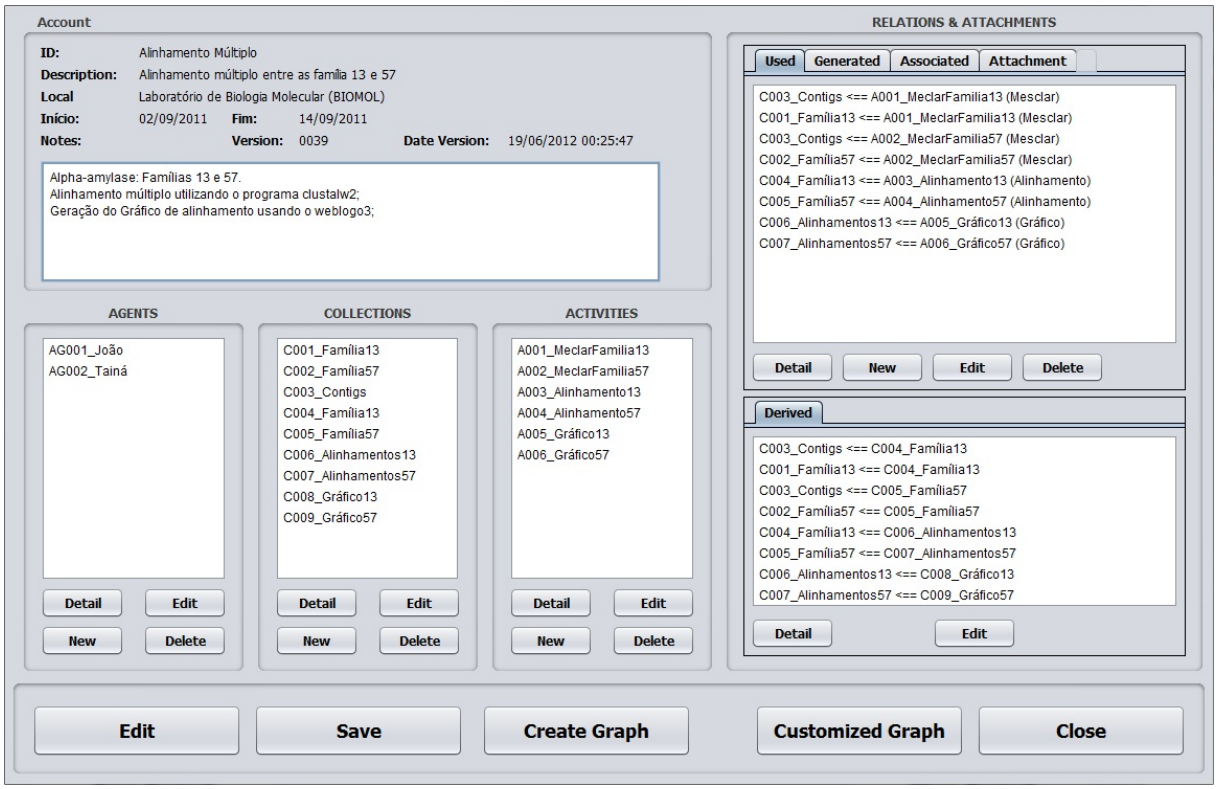


Figura 5.6: Exemplo de tela preenchida com os dados de um grafo.

Na parte inferior esquerda tem-se a relação dos nós, separada em três áreas: agente, coleção e atividade. Cada área exibe seus ítens e possui botões para incluir, alterar, exclui ou detalhar. Cada ítem corresponde a um nó do grafo.

A parte direita da tela é reservada para a relação das arestas. Cada aresta representa uma relação existente entre os nós, sendo que na parte superior, existem abas para escolher entre as três relações *Used*, *Generated* e *Associated*. A relação *Derived* está separada na parte inferior da tela porque não permite inclusão nem exclusão. Esta relação é criada ou excluída automaticamente, obedecendo as regras de inferência temporais, descritas no Seção 4.5.2. Na parte superior direita, juntamente com três primeiras relações, existe uma aba chamada *Attachment*. Esta aba permite que o usuário anexe informações adicionais ao grafo, tais como bibliografias, sites da internet, artigos, ou qualquer outro arquivo. Além disso, cada grafo personalizado criado é incluído como um anexo nesta aba.

A Figura 5.7 mostra um exemplo de tela de um agente. As telas de edição dos agentes, atividades e coleções são semelhantes e podem ser divididas em três partes. Na parte superior da tela são exibidos os campos para preenchimento. Na parte mediana são encontradas abas com as seguintes opções: *Groups*, *Notes*, *Graph Notes* e *Attachments*. A aba *Groups*, mostrada na Figura 5.7, permite ao usuário indicar a quais grupos o nó pertence. Dessa forma, o usuário poderá construir grafos personalizados, informando os grupos desejados. A segunda aba, *Notes*, permite que o usuário inclua qualquer anotação referente ao nó. A aba *Graph Notes* permite ao usuário incluir uma pequena observação que poderá ser impressa no grafo. Por fim, a aba *Attachment* fornece opções ao usuário para anexar diversos arquivos, sites da internet ou qualquer outra informação relevante para o nó. Na parte inferior da tela são exibidas diversas abas com as relações (arestas) das quais aquele nó participa, como origem ou destino.

5.2 Restrições e Inferências

A implementação das restrições estruturais e temporais trazem uma série de ligações entre si que devem ser explicadas. Assim, nesta seção é descrita a forma de implementação, no simulador *Provenance*, das restrições estruturais e temporais justificando assim, as inferências feitas.

Em relação à obrigatoriedade de um identificador único para cada elemento do modelo, as seguintes condições são garantidas pelo simulador *Provenance*:

- *Nós*: Não permite que o usuário crie dois nós (agentes, coleção ou atividade) com o mesmo identificador;
- *Arestas*: O identificador da aresta é formado pela união do identificador do nó de origem e do destino. Como o simulador não permite que o usuário crie duas arestas iguais (mesma origem e mesmo destino) esta unicidade está garantida;
- *Grafo*: O identificador do grafo (*Account*) é formado pelo local (pasta) e pelo nome do arquivo XML onde o grafo foi gravado. Assim o identificador é único para cada projeto.

Em relação à criação de relações válidas, as seguintes condições são garantidas pelo simulador:

- Antes da criação de uma relação qualquer, o simulador exige que dois nós sejam selecionados. Dessa forma, uma relação somente poderá ser criada se os dois nós já tiverem sido criados;
- O simulador somente criará uma nova relação se os dois nós escolhidos forem compatíveis com as características da relação que está sendo criada;
- Não permite que um nó seja excluído se existir uma relação em que este nó seja origem ou destino;
- Não permite a criação de uma nova relação se já existir outra com a mesma origem e destino da que está sendo criada;

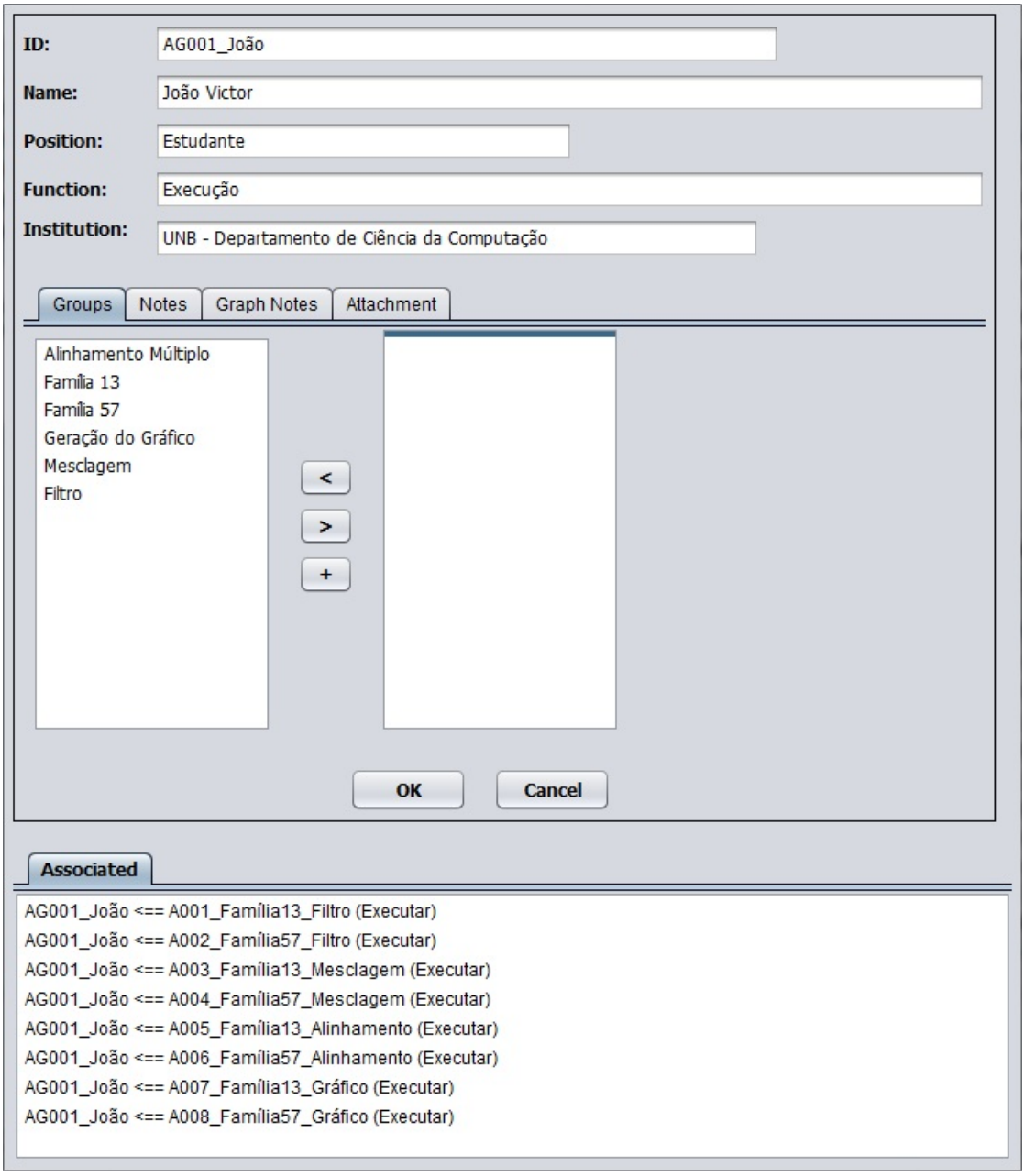


Figura 5.7: Tela de edição do agente.

- Uma aresta *WasGeneratedBy* somente poderá ser criada se não existir outra relação do mesmo tipo cuja origem seja a coleção selecionada. Assim, existirá no máximo uma ligação *WasGeneratedBy* para cada coleção.

As demais restrições estruturais são garantidas pelas restrições temporais.

As primeiras restrições temporais a serem tratadas dizem respeito a data de início e data de fim do projeto, do grafo e da atividade. A seguir são listadas algumas situações e as soluções adotadas no simulador em cada caso, a fim de garantir as restrições temporais:

- A criação ou alteração de uma atividade somente será aceita pelo simulador se:
  - A hora de início da atividade for menor que a hora de fim da atividade;

  - As horas de início e fim da atividade estiverem entre as 00:00 horas do dia de início do grafo e as 23:59 do dia de término do grafo.

- A criação ou alteração de um grafo somente será aceita pelo simulador se:
  - A data de início do grafo for menor ou igual a data de fim do grafo;
  - As horas de início e fim de todas as atividades contidas no grafo estiverem entre as 00:00 horas do dia de início do grafo e as 23:59 do dia de término do grafo;
  - As datas de início e fim do grafo estiverem entre as datas de início e fim do projeto.

- A inclusão de um grafo, feito em outro projeto (opção *"Attach" da tela do projeto*), somente será aceita pelo simulador se as datas de início e fim deste grafo estiverem entre a data de início e fim do projeto que está recebendo o grafo;

- A criação ou alteração de um projeto somente será aceita pelo simulador se:
  - A data de início do projeto for menor ou igual a data de fim do projeto;
  - As datas de início e fim de todos os grafos contidos no projeto estiverem entre as datas de início e fim do projeto.

As restrições temporais envolvidas com as relações são tratadas da seguinte forma:

- *Alteração de atividade*: A alteração de uma atividade somente será aceita após a validação de todas as arestas *WasGeneratedBy* e *Used* ligadas a essa atividade;
  - *WasGeneratedBy*: Estas arestas indicam quais coleções foram geradas por uma determinada atividade. Dessa forma, a hora de término da atividade deve ser menor que a hora de uso de todas as coleções dessas arestas;
  - *Used*: Estas arestas indicam quais coleções esta atividade usou. Dessa forma, a hora de início da atividade deve ser maior que a hora de geração de todas as atividades dessas arestas.

- *Inclusão de WasGeneratedBy*: Uma aresta *WasGeneratedBy* somente poderá ser criada se a hora de fim da atividade, destino da aresta, for menor que a menor hora de uso da coleção, origem da aresta;

- *Inclusão de Used*: Uma aresta *Used* somente poderá ser criada se a hora de início da atividade, origem da aresta, for maior que a hora de geração da coleção, origem da aresta.

Conforme descrito na Seção 4.4.2, para que as restrições das arestas *Used*, *WasGeneratedBy* e *WasDerivedFrom* sejam verificadas, é necessário que o grafo seja constituído de um conjunto mínimo de nós. Esse conjunto mínimo é mostrado na Figura 5.8, onde os termos t_{in} e t_{fn} representam as horas de início e fim das Atividades 1 e 2. Conforme descrito nas restrições temporais das Atividades, tem-se que $t_{i1} < t_{f1}$ e $t_{i2} < t_{f2}$. Por outro lado, o conjunto mínimo é necessário para a validação da restrição das arestas (A) onde $t_{f1} < t_{i2}$, ou seja, para que as três relações apresentadas na Figura 5.8 sejam criadas, a Atividade 2 somente pode ser iniciada após o termino da Atividade 1.

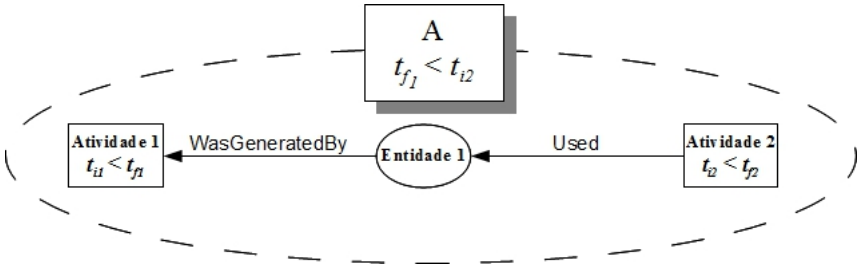


Figura 5.8: Exemplo de conjunto mínimo para validação temporal das arestas.

Assim, as restrições das arestas *Used*, *WasGeneratedBy* e *WasDerivedFrom* somente são verificadas para os grafos que possuem pelo menos um conjunto mínimo, para os demais, estas restrições serão consideradas satisfeitas.

Dadas as garantias das restrições estruturais e temporais descritas até esse ponto, pode-se fazer a primeira inferência estrutural do modelo. Essa inferência estrutural trata da criação da relação *WasDerivedFrom*. Assim, tem-se os procedimentos do simulador *Provenance* para cada uma das seguintes situações:

- *Inclusão de WasGeneratedBy*: Quando esta aresta é criada entre uma coleção C e uma atividade A, são criadas, automaticamente, arestas *WasDerivedFrom* para cada coleção usada pela atividade A. Em cada caso a origem será sempre a coleção C e os destinos serão cada uma das coleções usadas pela atividade A;
- *Exclusão de WasGeneratedBy*: Quando esta aresta é excluída, também são excluídas as arestas *WasDerivedFrom* relacionadas com ela;
- *Inclusão de Used*: Quando esta aresta é criada entre uma atividade A e uma coleção C, são criadas, automaticamente, arestas *WasDerivedFrom* para cada coleção gerada pela atividade A. Em cada caso o destino será a coleção C e as origens serão cada uma das coleções geradas pela atividade A;
- *Exclusão de Used*: Quando esta aresta é excluída, também são excluídas as arestas *WasDerivedFrom* relacionadas com ela.

Esta inferência, por outro lado, resolve as restrições estruturais e temporais restantes que são as seguintes:

- *Restrição estrutural de WasDerivedFrom*: Dois fatos devem ser levados em conta: (a) devido ao fato do simulador garantir as restrições temporais descritas até aqui, não é possível que um atividade use uma coleção que ela mesma gerou; (b) as relações *WasDerivedFrom* são geradas automaticamente a partir das relações *WasGeneratedBy* e *Used*; A partir desses fatos não é possível que uma aresta *WasDerivedFrom* tenha a mesma coleção como origem e destino;
- *Restrição temporal de WasDerivedFrom*: A criação automática desta aresta, juntamente com a garantia das restrições temporais nas relações *WasGeneratedBy* e *Used*, garante também esta restrição temporal.

Considerando-se que a relação *MemberOf* não está sendo utilizada no simulador, pode-se afirmar que todas as restrições estruturais definidas na Seção 4.4.1 foram garantidas

pelo simulador e que, tanto os grafos como os projetos criados a partir dele, podem ser inferidos como estruturalmente válidos.

Da mesma forma, todas as restrições temporais foram garantidas pelo simulador, assim pode-se inferir que, tanto os grafos como os projetos criados a partir dele, são temporalmente válidos.

5.3 Estudos de Casos

Nesta seção são apresentados dois experimentos de bioinformática cujos dados foram inseridos no simulador *Provenance*. Ao final de cada experimento é feita uma análise sobre o impacto no espaço de memória ocupado pelos dados de proveniência adicionados.

5.3.1 Estudo de Caso 1: Alpha-amylase

O primeiro estudo de caso apresentado nesse trabalho tratou de sequências de DNA da bactérias extremófila Alpha-amylase obtidas a partir do banco de dados UNIPROT (UNIPROT, 2012). Este projeto objetivou encontrar o gene que codifica uma determinada proteína, no genoma do bacilo, assim como também comparar a sequência encontrada com outras disponíveis. O experimento deste estudo de caso refere-se a comparação, do gene encontrado, com outras sequências disponíveis. Para tal foram realizados alinhamentos múltiplos para amostras das famílias 13 e 57 desta bactéria, e por fim foram gerados gráficos destes alinhamentos. Esse experimento foi realizado no Laboratório de Biologia Molecular (BIOMOL) do Departamento de Biologia Celular (CEL) da Universidade de Brasília (UnB) entre os dias 02/09/2011 e 14/09/2011.

A Figura 5.9 mostra o *workflow* dos processos executados nesse experimento. As sequências de DNA, obtidas do banco de dados do UNIPROT, foram filtradas e alinhadas com um arquivo contendo a sequência do gene. Por fim, o arquivo com os alinhamentos foi usado para gerar um gráfico. Esse processo foi repetido com as famílias 13 e 57 da bactéria Alpha-amylase.

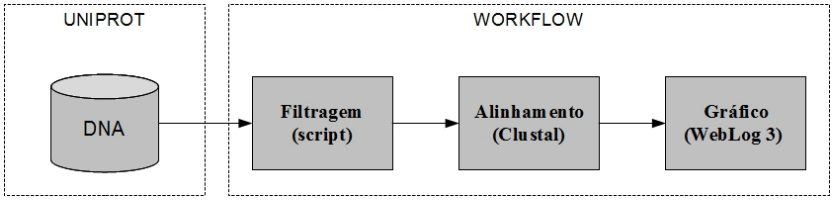


Figura 5.9: *Workflow* resumido do experimento com a bactéria Alpha-amylase indicando os programas utilizados.

A Figura 5.10 mostra o grafo de proveniência gerado a partir da inserção dos dados desse experimento no simulador. Conforme definido no modelo PROV-DM, as atividades são representadas por retângulos, enquanto que as coleções são representadas por elipses e os agentes por pentágonos com a base reta. O grafo exibido na Figura 5.10 apresenta 2 agentes, 11 coleções e 8 atividades que estão ligadas através de 40 arestas, formando o histórico do experimento, cada um descrito a seguir.

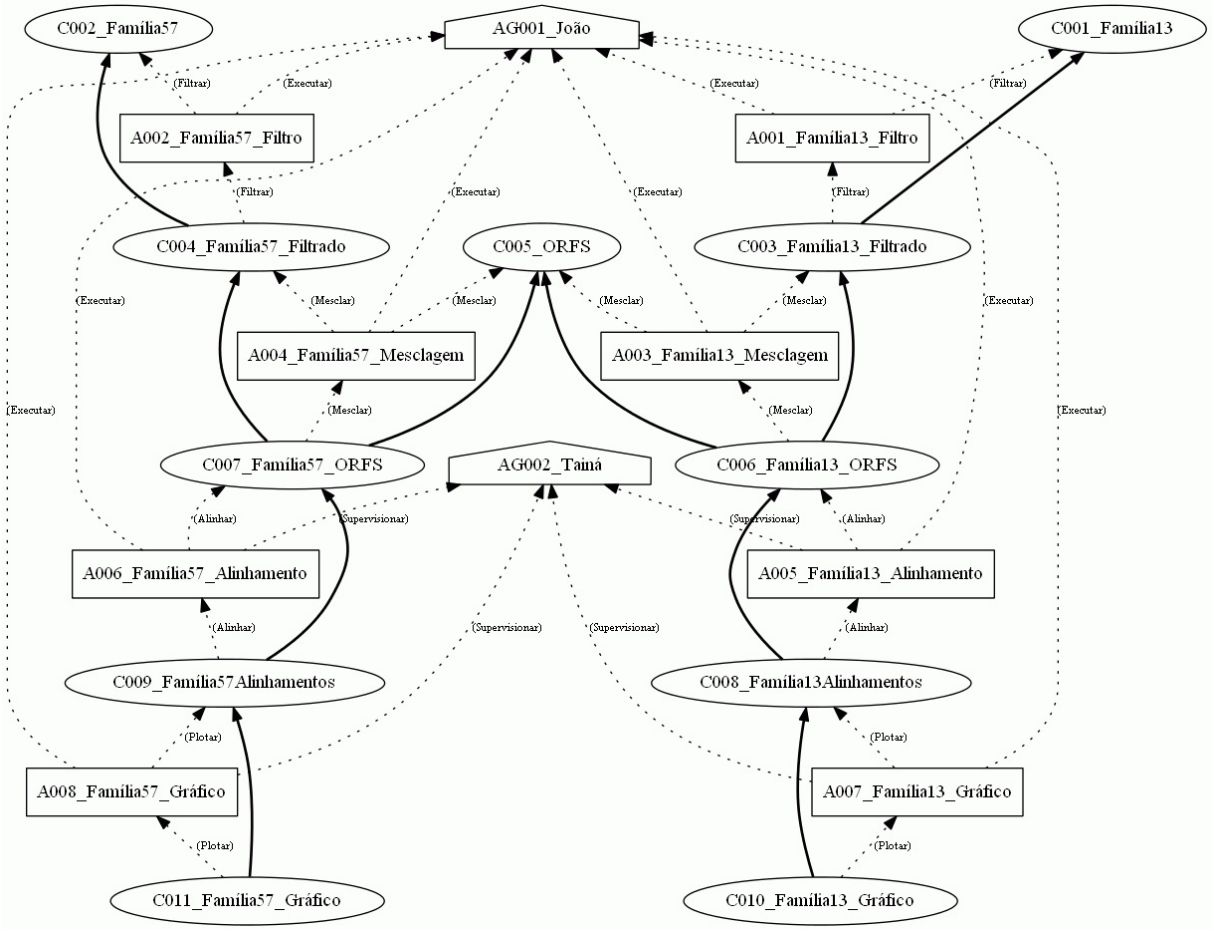


Figura 5.10: Grafo gerado a partir da entrada dos dados do experimento da bactéria Alpha-amylase.

Esse experimento foi executado por um estudante do Departamento de Ciência da Computação (CIC) da UnB que é representado pelo agente com identificador *AG001_João*, e foi validado por uma pesquisadora do Departamento de Biologia Celular, também da UnB, que é representada pelo agente com identificador *AG002_Tainá*.

Pelo grafo da Figura 5.10 o agente *AG001_João* está associado a todas as atividades executadas nesse experimento. Isso pode ser visto pelas arestas pontilhadas entre o agente e cada atividade no grafo. Essa aresta representa a relação *WasAssociatedWith* do modelo PROV-DM. A origem é sempre a atividade e o destino é o agente. Também pode-se ver que o papel desse agente em cada atividade foi o de execução, demonstrado pela regra *Executar* presente entre parênteses em cada uma das suas relações.

Também pode ser visto que o agente *C002_Tainá* agiu como validador em 4 atividades do experimento (*A005_Família13_Alinhamento*, *A006_Família57_Alinhamento*, *A007_Família13_Gráfico13* e *A008_Família57_Gráfico*). Isso é percebido pelas relações entre esse agente e suas atividades e pela regra presente em cada uma dessas relações.

As coleções *C001_Família13* e *C002_Família57* representam arquivos no formato FASTA com sequências de DNA das famílias 13 e 57, respectivamente, da bactéria Alpha-amylase obtidas a partir do banco de dados do UNIPROT. A coleção *C005_ORFS* também representa um arquivo FASTA no formato de proteínas, com os *contigs* do gene encontrado, e

que farão parte dos alinhados em cada família.

Inicialmente foi realizado um processo de filtro, representado pela atividade *A001_Família13_Filtro* que usou a coleção *C001_Família13* para gerar a coleção *C003_Família13_Filtrado*. A partir daí, a atividade *A003_Família13_Mesclagem* executou um processo de junção da coleção *C003_Família13_Filtrado* e *C005_ORFS* gerando a coleção *C006-_Família13_ORFS*. A seguir, a coleção *C006_Família13_ORFS* é usada pela atividade *A005_Família13_Alinhamento* a fim de fazer o alinhamento múltiplo das sequências dessa coleção. Esses alinhamentos são gravados em um arquivo representado pela coleção *C008_Família13_Alinhamentos*. E, por fim, a coleção *C010_Família13_Gráfico* é gerada pela atividade *A007_Família13_Gráfico* que usou os alinhamentos contidos na coleção *C008_Família13_ Alinhamentos*.

Pode-se perceber que a mesma sequência de processo ocorre com as coleções da família 57, gerando ao final a coleção *C011_Família57_Gráfico*.

No grafo da Figura 5.10, as arestas pontilhadas que possuem uma atividade como origem e uma coleção como destino representam a relação *Used*, e as arestas pontilhadas que possuem uma coleção como origem e uma atividade como destino representam as relações *WasGeneratedBy*.

Dessa forma, pode-se afirmar que o grafo da Figura 5.10 apresenta a proveniência de duas coleções (*C010_Família13_Gráfico* e *C011_Família57_Gráfico*) e as arestas cheias que começam nessas coleções e percorrem as demais, estão demonstrando as derivações até as coleções iniciais.

O resultado do experimento com os dados das Famílias 13 e 57 da bactéria extremófila Alpha-amylase, foi a geração de duas imagens no formato PNG, com os gráficos dos alinhamentos feitos. Como esses arquivos são importantes documentos do resultado deste experimento, eles foram anexados ao grafo.

A Figura 5.11 mostra, em detalhe, a parte da tela em que podem ser vistos os anexos do grafo (*Attachment*), onde diversas informações e arquivos podem ser anexados. Os dois primeiros anexos correspondem aos dois arquivos gerados no experimento, com os gráficos dos alinhamentos. O terceiro anexo corresponde ao site do banco de dados UNIPROT, de onde foram obtidas as sequências da bactéria. Os últimos três anexos são grafos personalizados criados com as opções oferecidas pelo simulador. O caractere "(*)" na frente de um anexo, indica que existe um arquivo ou um site anexado nele e, para abri-lo, basta selecionar o mesmo.

A Figura 5.12 mostra a tela gerada pelo simulador após o usuário ter solicitado a criação de um grafo personalizado. Pode-se ver nessa tela o nome do anexo e a localização do arquivo GIF com o grafo personalizado. Na caixa de texto no centro da tela, pode-se ver a descrição do grafo personalizado, indicando a data e hora da sua criação, dados do grafo original, opções escolhidas e os grupos selecionados.

Neste grafo foi selecionado somente o grupo *Família 57* e não foram selecionadas as opção de exibir as regras e as anotações do grafo. A Figura 5.13 mostra o primeiro grafo personalizado, onde somente os nós que pertencem ao grupo *Família 57* são exibidos, sem as regras e as anotações do grafo.

O segundo grafo foi criado incluindo o grupo *Alinhamento Múltiplo* e com a opção de apresentar as regras. Assim, a Figura 5.14 mostra o segundo grafo gerado onde somente os nós pertencentes ao grupo *Alinhamento Múltiplo* são exibidos. Como o usuário escolheu

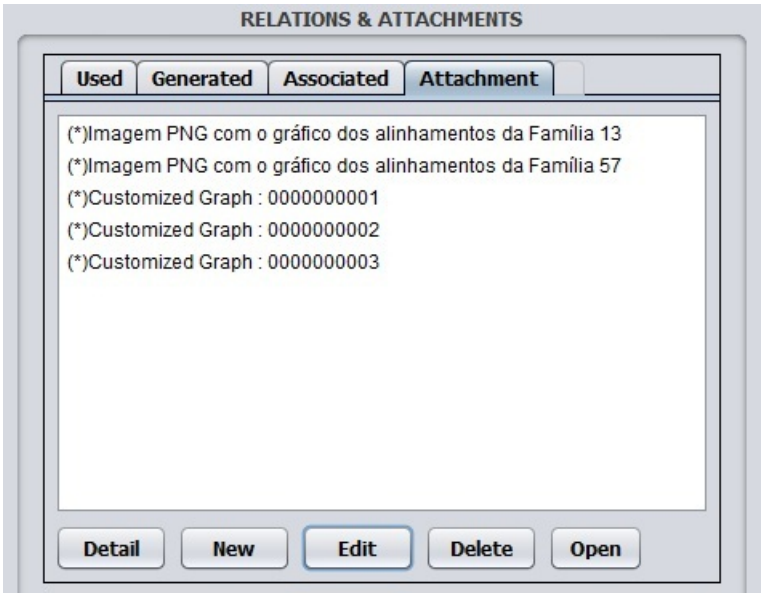


Figura 5.11: Detalhe da tela do grafo com os anexos.

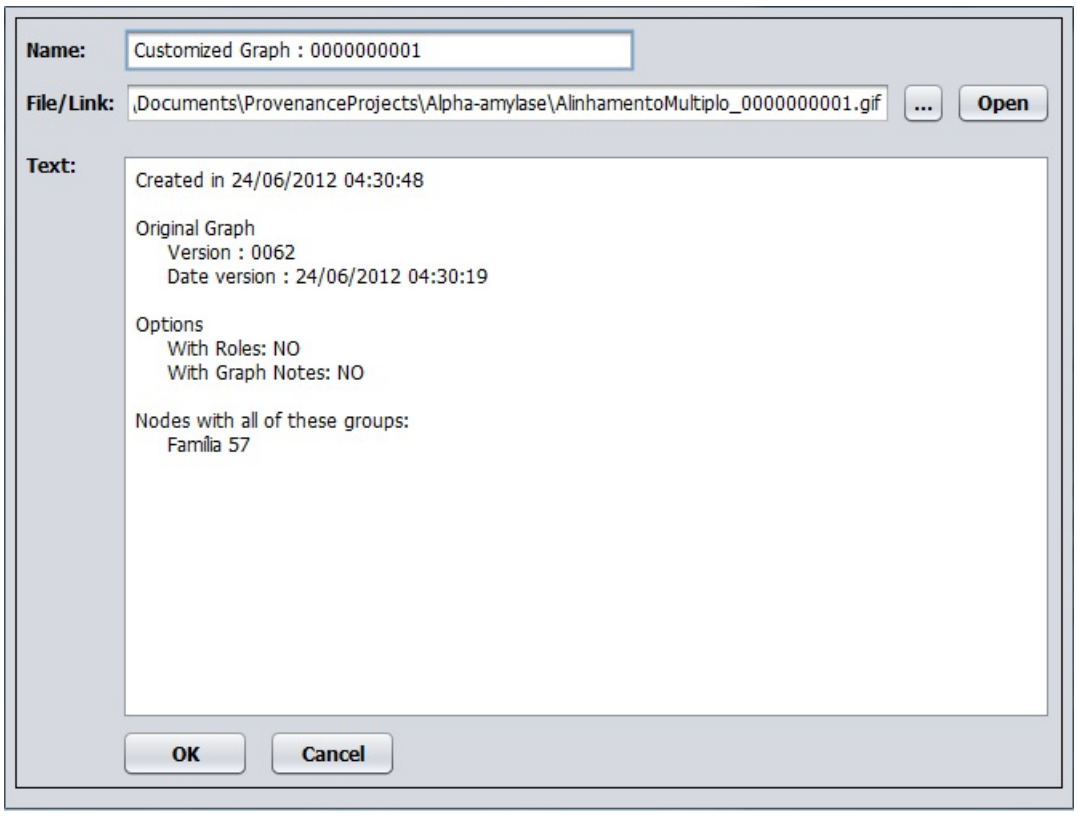


Figura 5.12: Tela do anexo com o primeiro grafo personalizado.

a opção de exibir as regras, estas são exibidas nas arestas *WasAssociatedWith*, *Used* e *WasGeneratedBy*.

Por fim, foi escolhida a opção de gerar um grafo com os nós que pertençam a pelo menos um dos grupos *Filtro* ou *Alinhamento Múltiplo*. Também foi escolhida a opção

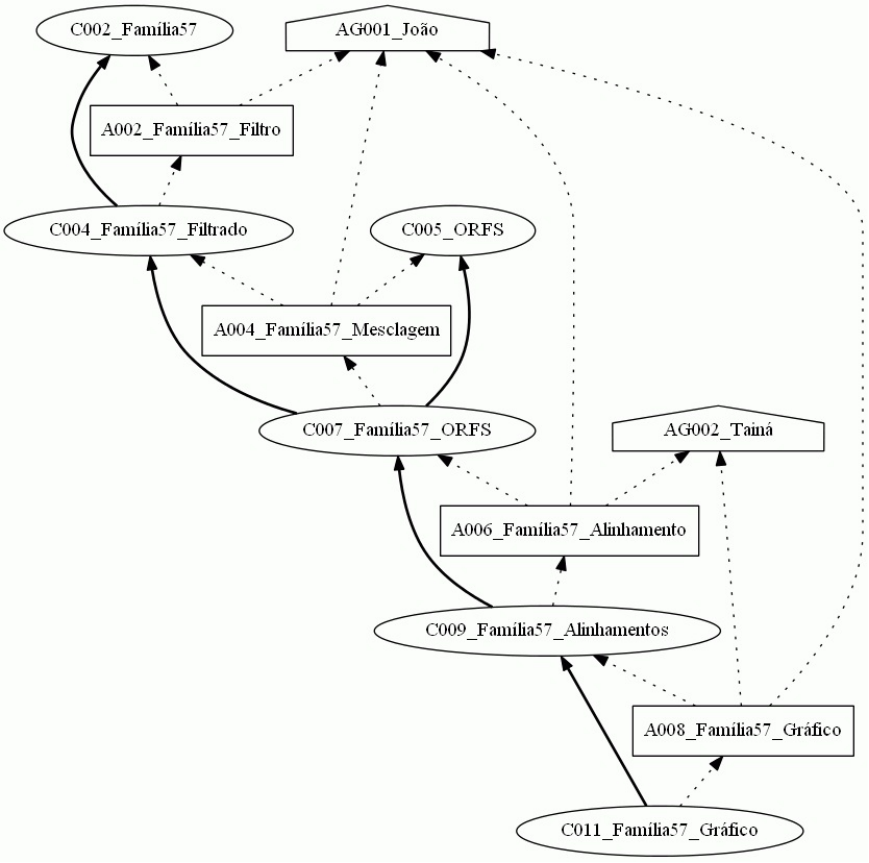


Figura 5.13: Primeiro grafo personalizado.

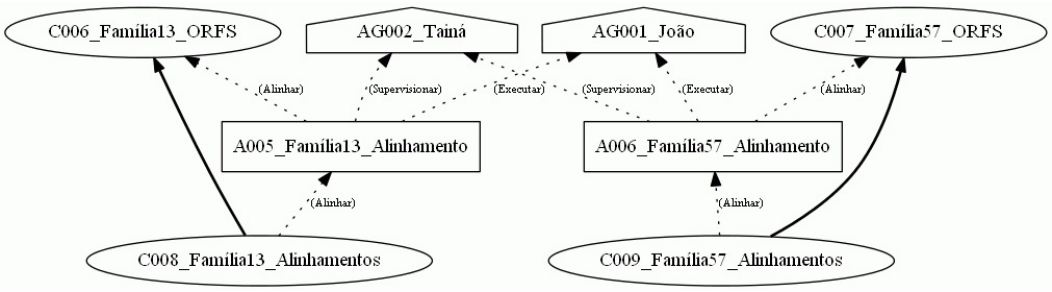


Figura 5.14: Segundo grafo personalizado.

de mostrar as anotações do grafo. A Figura 5.15 mostra o terceiro grafo com os nós pertencentes a pelo menos um dos grupos escolhidos. Também pode-se ver as anotações que aparecem em retângulos de cor cinza, que indicam a origem de algumas coleções e os programas usados para execução das atividades. As anotações podem ser incluídas pelo usuário em cada nó do grafo e não são obrigatórias, como pode ser visto no agente *AG001_João*, que não possui anotação, assim como outros nós do grafo.

Para este experimento, o simulador *Provenance* criou os seguintes arquivos: dois arquivos XML (um para o projeto e outro para o grafo), quatro arquivos GIF (um para o grafo principal e três para os grafos personalizados) e um arquivo DOT (usado para a construção do grafo principal). O total de espaço em disco ocupado por estes arquivos

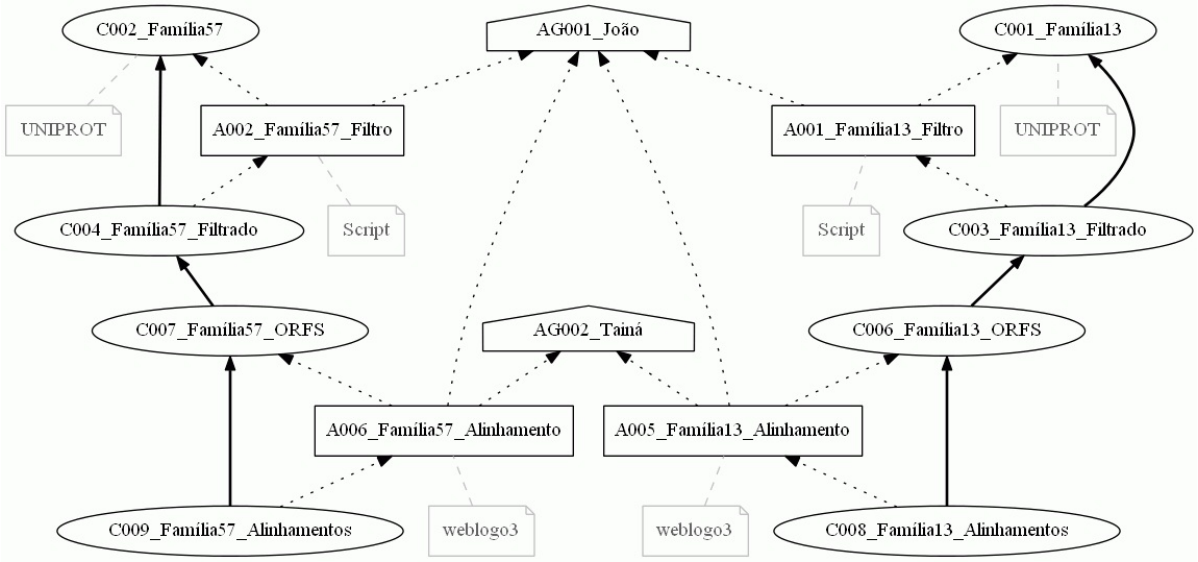


Figura 5.15: Terceiro grafo personalizado.

compreende menos de 180 KB (Kilobytes). Por outro lado, esse experimento usou e gerou em torno de 6,1 MB (megabytes) de dados. Assim, os dados adicionados correspondem a menos de 3% do total de dados do experimento. Vale ressaltar que este experimento utilizou arquivos relativamente pequenos, se comparados à maioria dos experimento de bioinformática.

5.3.2 Estudo de Caso 2: Expressão Diferencial Entre Células Humanas de Rim e Fígado

Este experimento é descrito por Marioni et al. (2008) e foi parcialmente refeito no Laboratório de Biologia Molecular (BIOMOL) do Departamento de Biologia Celular (CEL) da Universidade de Brasília (UnB). O experimento trata arquivos com sequências de RNA de amostras de rim e fígado de um mesmo indivíduo. Estas sequências são alinhadas com o genoma humano e depois são avaliados quais genes foram expressos em cada órgão (rim e fígado).

A Figura 5.16 mostra o *workflow* dos processos executados nesse experimento. As sequências de RNA foram obtidas a partir do banco de dados do NCBI e foram filtrados levando em conta a qualidade das sequências contidas em cada arquivo. Em seguida foram alinhadas com o genoma humano e os alinhamentos encontrados foram ordenados para tornar a análise estatística mais rápida. Por fim, foi realizada a análise de expressão diferencial entre os alinhamentos das células de rim e fígado. As três primeiras etapas foram feitas em paralelo com as amostras de rim e fígado e, na quarta etapa, os dados são unidos e analisados.

O grafo exibido na Figura 5.17 foi gerado a partir da inserção dos dados desse experimento no simulador *Provenance*. Conforme definido no modelo PROV-DM, as atividades são representadas por retângulos, enquanto que as coleções são representadas por elipses e os agentes por pentágonos com a base reta. O grafo mostrado na Figura 5.17 apresenta

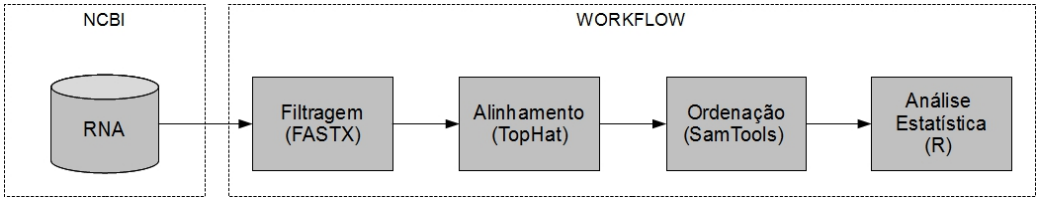


Figura 5.16: *Workflow* resumido do experimento com células de rim e fígado, com indicação dos programas utilizados.

1 agente, 14 coleções e 7 atividades que estão ligadas através de 42 arestas, formando o histórico do experimento, cada um descrito a seguir.

Como pode ser visto no grafo, apenas um agente (*AG001_Executor*) trabalhou nesse experimento, executando todas as atividades.

O experimento iniciou com três arquivos contendo sequências de RNA de células de rim, cada um sendo representado por uma coleção (*C001_Rim*, *C002_Rim* e *C003_Rim*). As coleções passaram por um filtro a fim de garantir uma qualidade mínima das sequências. A atividade *A001_Rim_Filtrado* representa tanto o filtro como a junção de todos os arquivos. Assim, a saída desta atividade é a coleção *C007_Rim_Filtrado* que contém as três coleções iniciais filtradas.

A Figura 5.18 mostra a tela com as informações da coleção *C002_Rim* onde pode ser visto o nome do arquivo e a sua localização, a quantidade de sequências no arquivo, o tipo de arquivo e as anotações sobre a origem do arquivo e suas especificações.

A Figura 5.19 mostra a tela com as informações da atividade *A003_Alinhamento_Rim*. Nesta tela podem ser vistas diversas informações sobre a atividade tais como linha de comando executada, hora de início e fim da execução, entre outras. Na aba *Groups* pode-se ver que essa atividade pertence aos grupos *Alinhamento* e *Rim*. Esse grupos são definidos pelo usuário e são usados para a geração de grafos personalizados.

O experimento continua com as coleções *C007_Rim_Filtrado* e *C009_HG* sendo usadas pela atividade *A003_Alinhamento_Rim* para fazer o alinhamento das sequências (*C007_Rim_Filtrado*) com o genoma de referência (*C009_HG*). Deste alinhamento foi gerada a coleção *C010_Rim_Alinhamento*.

A coleção *C010_Rim_Alinhamento*, por sua vez, é usada pela atividade *A005_Ordenação_Rim* que ordena os dados desta coleção. O resultado é a geração da coleção *C012_Rim_Ordenado*.

A mesma sequência de atividades é feita com as sequências de fígado, começando com as coleções *C004_Fígado*, *C005_Fígado* e *C006_Fígado*. Ao final é gerada a coleção *C013_Fígado_Ordenado*.

Por fim, a atividade *A007_Expressão_Diferencial* faz a análise estatística das coleções *C012_Rim_Ordenado* e *C013_Fígado_Ordenado*. O resultado dessa análise é representado pela coleção *C014_RimXFígado*.

No grafo da Figura 5.17, as arestas pontilhadas que possuem uma atividade como origem e uma coleção como destino representam a relação *Used*, e as arestas pontilhadas que possuem uma coleção como origem e uma atividade como destino representam as relações *WasGeneratedBy*. Dessa forma, esse grafo demonstra a proveniência da coleção *C014_RimXFígado* e as arestas representadas pelas linhas cheias no grafo representam as derivações feitas das coleções iniciais até a geração desta coleção.

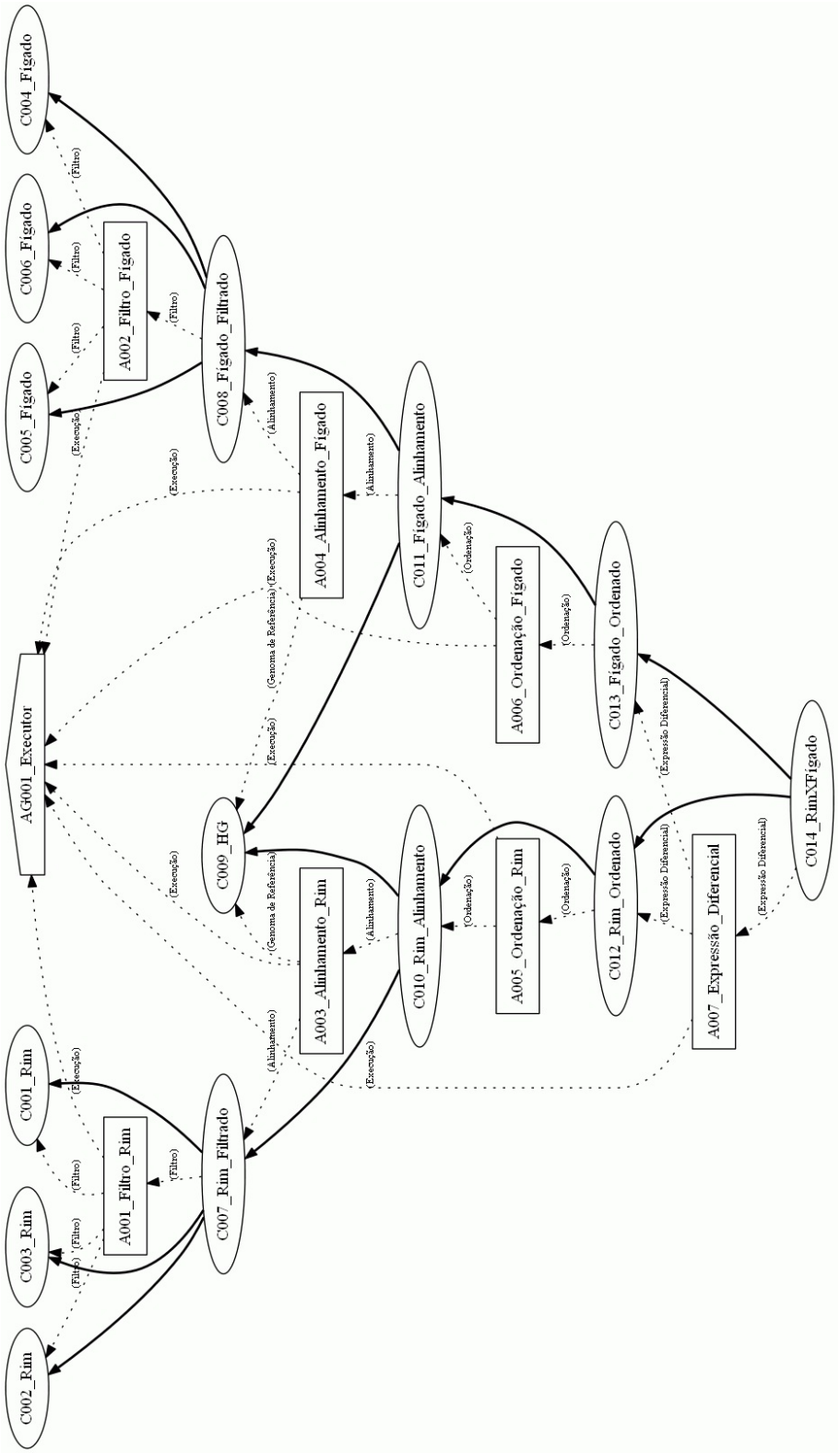


Figura 5.17: Grafo do experimento que trata da expressão diferencial entre células de rim e fígado.

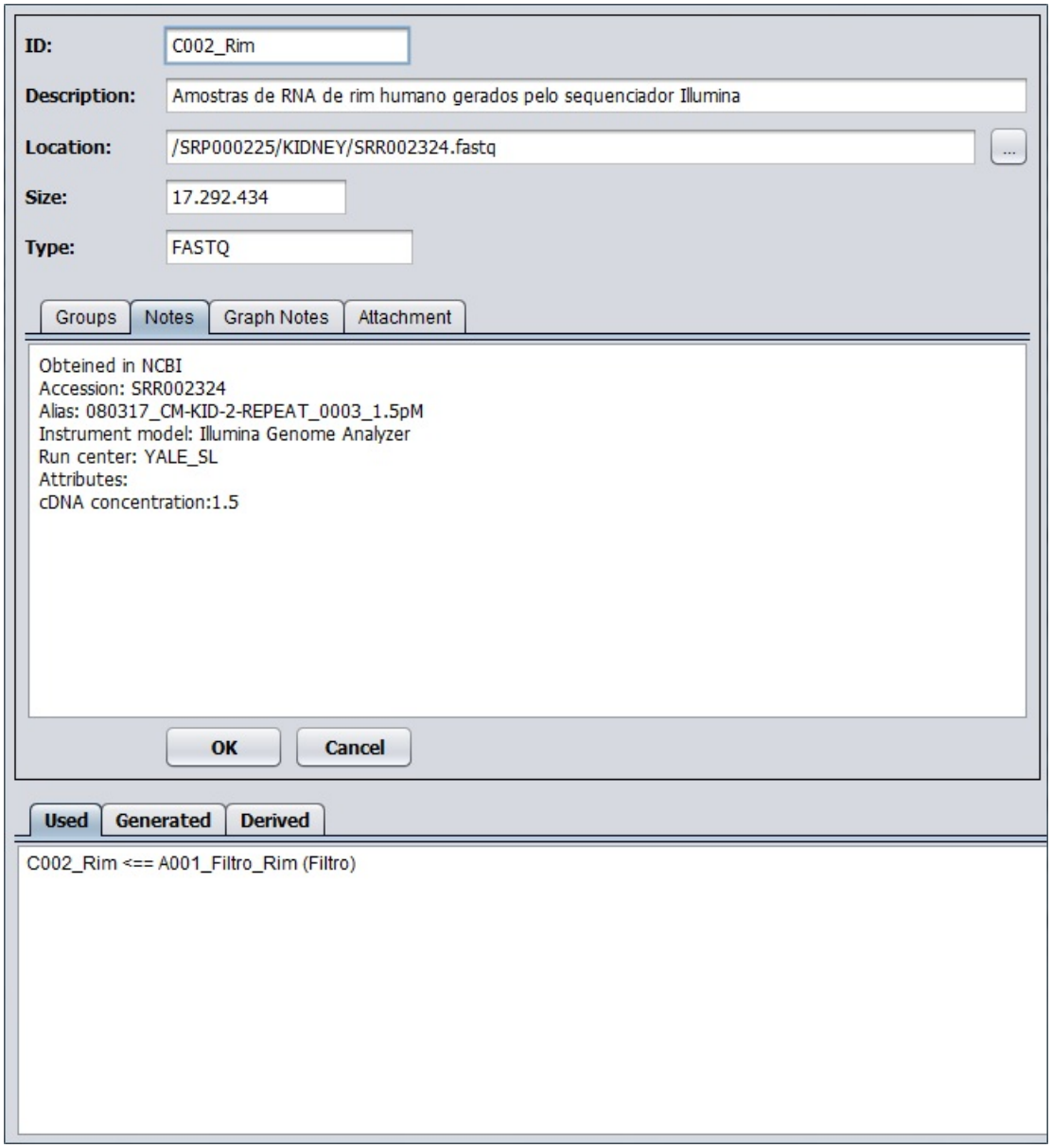


Figura 5.18: Tela da coleção *C002_Rim*.

A Figura 5.20 mostra em detalhe a parte da tela em que podem ser vistos os anexos (*Attachments*) do grafo, onde diversas informações e arquivos podem ser anexados. O primeiro anexo corresponde ao artigo, em formato PDF, com a descrição do experimento. O segundo anexo corresponde à página do banco de dados do NCBI, de onde foram obtidas as amostras para o experimento. Os outros três anexos são grafos personalizados criados com as opções oferecidas pelo simulador. O caractere "(*)" na frente de um anexo indica que existe um arquivo ou um site anexado nele e, para abrí-lo, basta selecionar o mesmo.

A Figura 5.21 mostra a tela gerada pelo simulador quando o usuário solicita a criação de um grafo personalizado. Pode-se ver nessa tela o nome do anexo e a localização do arquivo GIF com o grafo personalizado. Na caixa de texto no centro da tela podem ser vistos a descrição do grafo personalizado, indicando a data e hora da sua criação, dados do grafo original, opções escolhidas e os grupos selecionados.

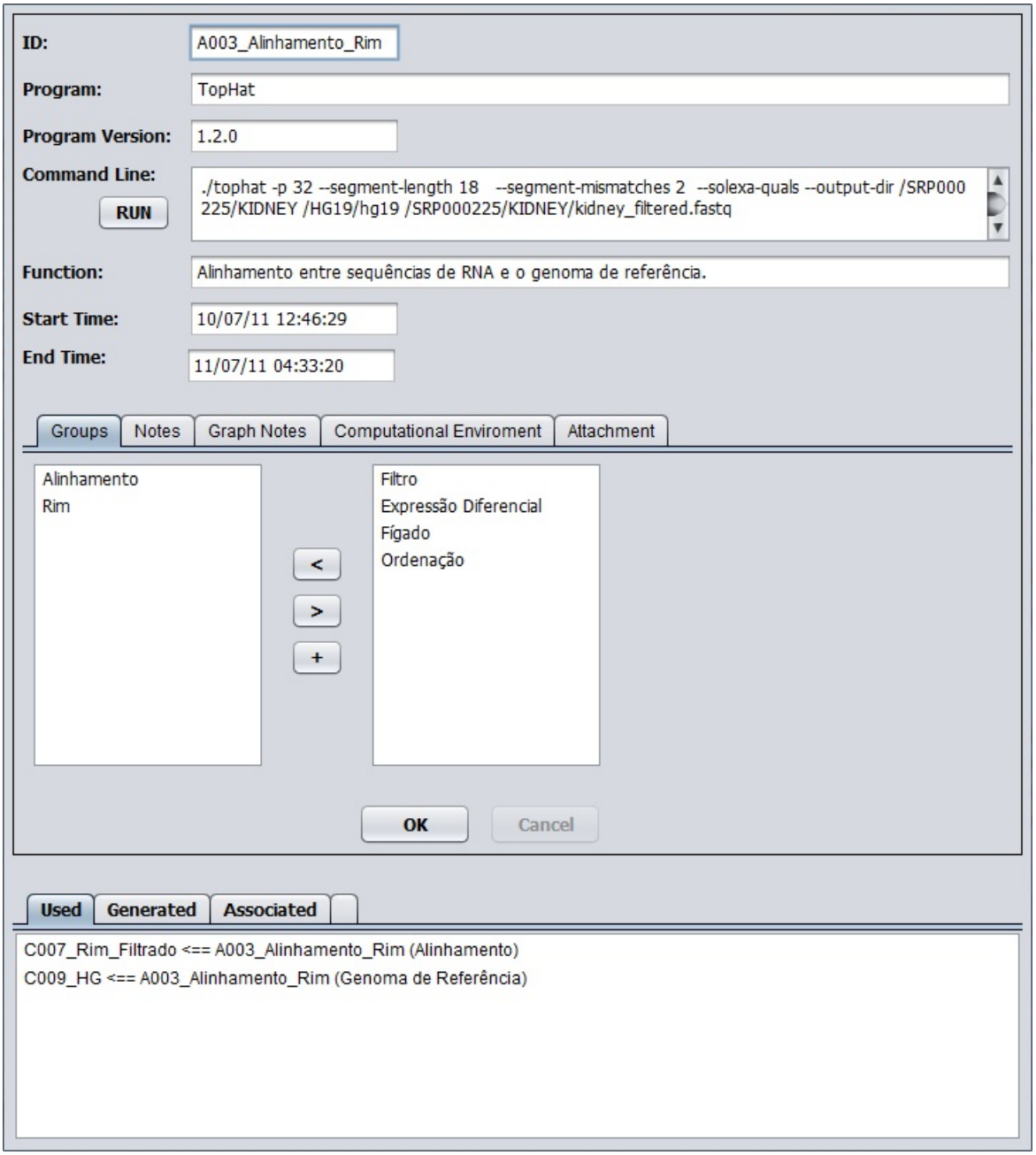


Figura 5.19: Tela da atividade *A003_Alinhamento_Rim*.

Neste grafo foi selecionado somente o grupo *Filtro* e não foram selecionadas as opções de exibir as regras e as anotações do grafo. A Figura 5.22 mostra o primeiro grafo personalizado onde aparecem os processos de filtragem de rim e fígado, sem as regras e as anotações do grafo.

O segundo grafo foi gerado com a opção de incluir os nós que pertençam a pelo menos um dos grupos *Filtro* ou *Alinhamento*. Também foi escolhida a opção de mostrar as anotações do grafo. Assim, a Figura 5.23 mostra o segundo grafo com os nós pertencentes a pelo menos um dos grupos escolhidos. Também pode-se ver as anotações que aparecem em retângulos de cor cinza, que indicam a data e hora em que cada atividade foi executada.

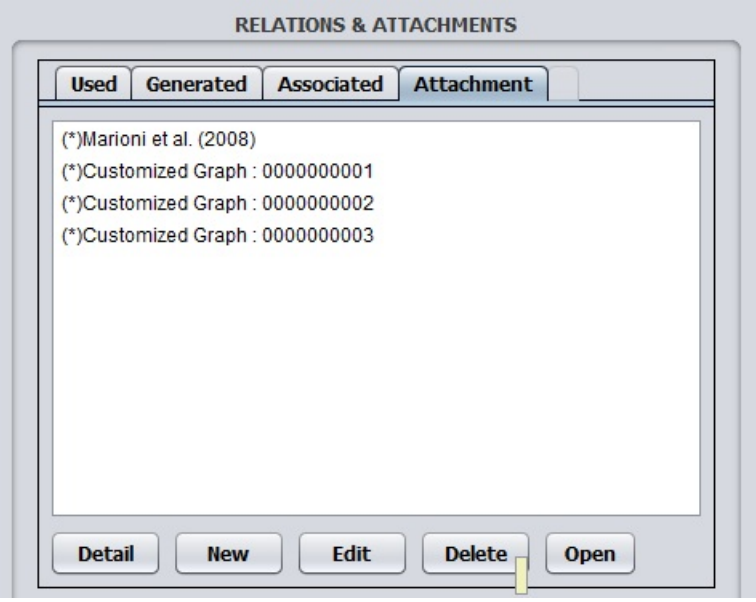


Figura 5.20: Tela dos anexos do estudo de caso 2.

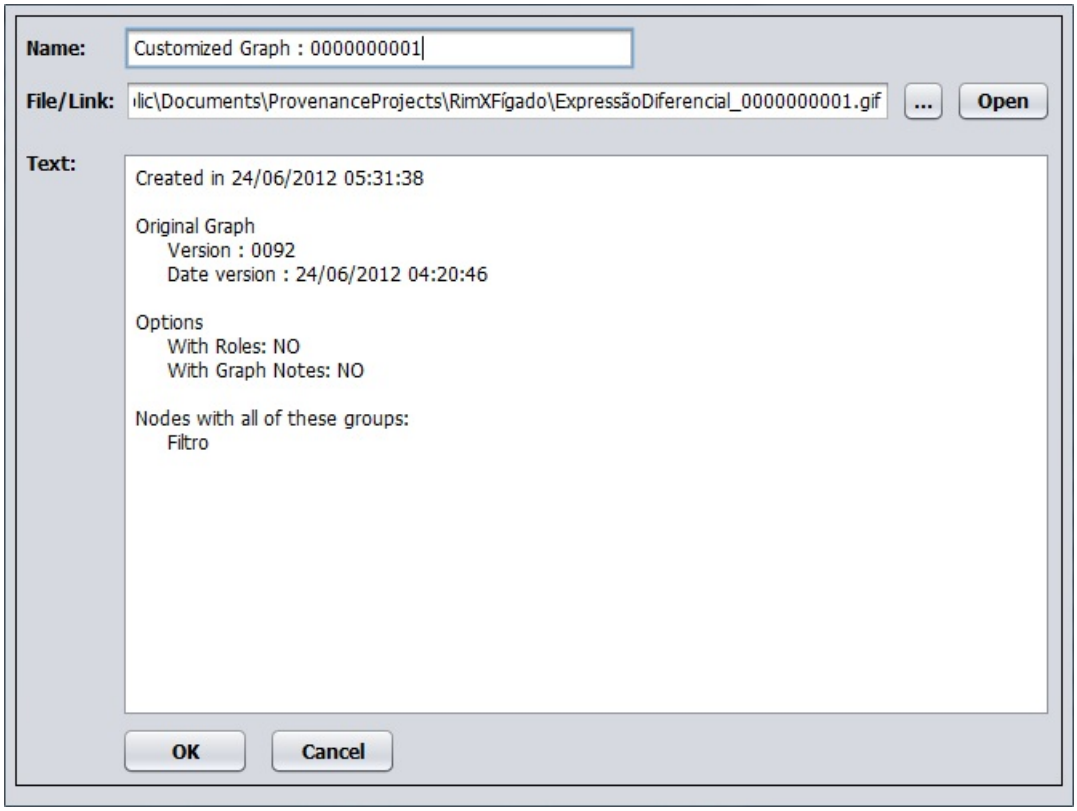


Figura 5.21: Tela do anexo com primeiro grafo personalizado.

As anotações podem ser incluídas pelo usuário em cada nó do grafo e não são obrigatórias, como pode ser visto no agente e nas coleções deste grafo, que não possuem anotação.

Por fim, foi escolhida a opção de gerar um grafo com os nós que pertençam aos grupos *Filtro* e *Rim*. Também foi escolhida a opção de mostrar tanto as regras quanto as anotações

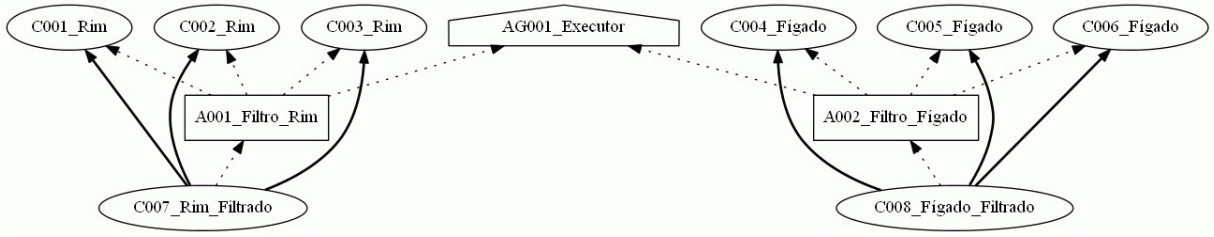


Figura 5.22: Primeiro grafo personalizado.

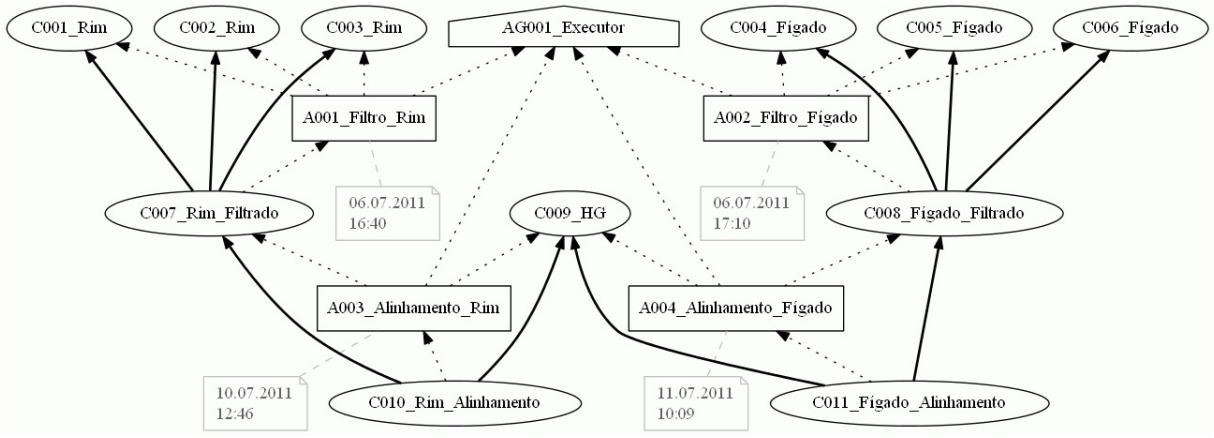


Figura 5.23: Segundo grafo personalizado.

do grafo.

A Figura 5.24 mostra o terceiro grafo com os nós pertencentes aos dois grupos escolhidos. Também pode-se ver as regras das arestas e as anotações. Neste grafo o usuário escolheu como anotação o tipo de arquivo e o tamanho de cada coleção e, no caso da atividade, está sendo exibido como anotação, o programa utilizado e a sua versão.

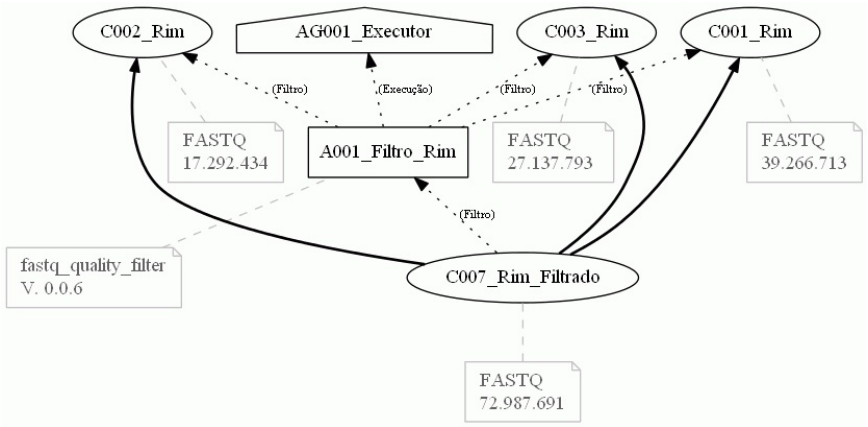


Figura 5.24: Terceiro grafo personalizado.

Os arquivos criados pelo simulador para este experimento correspondem a dois arquivos XML (um para o projeto e outro para o grafo), um arquivo DOT para a geração do grafo principal, e quatro arquivos GIF (um para o grafo principal e três grafos personalizados). Estes arquivos totalizam 205 KB (Kilobytes). Por outro lado, esse experimento

usou e gerou em torno de 128 GB (gigabytes) de dados. Assim, os dados adicionados correspondem a menos de 0.0001% do total de dados do experimento.

Capítulo 6

Conclusão

Neste trabalho foi apresentado um estudo com os principais modelos de proveniência de dados disponíveis na literatura. A partir da comparação feita entre eles, foi escolhido o modelo PROV-DM para representar a proveniência de dados, uma vez que ele possui todos os requisitos necessários para aplicação em projetos de bioinformática. Entre os principais requisitos estão: capacidade de representar a proveniência de uma peça de dado, descrevendo os processos e insumos utilizados em sua geração; uma representação gráfica adequada, com diferentes símbolos para cada elementos, e relações suficientes para demonstrar a proveniência de forma objetiva; símbolo para representar grandes conjuntos de dados. Além dos requisitos, outras características devem ser destacadas em relação ao modelo PROV-DM, como o extenso material disponível cobrindo diferentes aspectos da proveniência de dados e o fato de estar sendo desenvolvido pelo W3C, podendo vir a ser uma recomendação desta instituição. Essas duas características são importantes indicadores da capacidade de disseminação do modelo.

A aplicação do modelo PROV-DM para representar a proveniência de dados em projetos de bioinformática mostrou-se bastante simples e direta. Os componentes do modelo, tais como o agente, atividade e coleção, representam elementos presentes em grande parte dos experimentos executados em projetos de bioinformática. As relações, por sua vez, demonstram de forma objetiva as dependências entre cada elemento no grafo, e a utilização das regras e do tipo de derivação permitem maior grau de especificidade quando necessário.

A quantidade de espaço em disco utilizada pelos projetos de bioinformática foi um desafio para este trabalho. Assim, a divisão das informações de proveniência em dois níveis propicia diversas vantagens neste aspecto. O Nível 1 permite que poucos dados sejam adicionados a fim de demonstrar a proveniência. O Nível 2, por sua vez, propicia que os próprios arquivos originais do projeto sejam utilizados como insumo para as consultas de proveniência. O cruzamento das informações, nos dois níveis, fornece uma ferramenta capaz de percorrer os dados desde o objeto da proveniência até os arquivos inicialmente usados, descrevendo o processo de transformação ocorrido em cada atividade executada.

A representação gráfica da proveniência, na forma de um grafo direcionado, permite ao usuário rever a execução de seus experimentos de forma clara e objetiva. A utilização do modelo PROV-DM, uma provável recomendação pelo W3C, propicia que o usuário possa trocar informações com outros pesquisadores em um formato padronizado de representação.

Com o desenvolvimento do simulador *Provenance*, foi possível demonstrar a aplicação do modelo PROV-DM em projetos de bioinformática. A inclusão de dados adicionais ao projeto, no estudo de caso 1, ficou em menos de 3% do total dos dados originalmente usados no projeto, e no estudo de caso 2, bem menos de 1%. Isso mostra que o gerenciamento da proveniência de dados utilizando o PROV-DM, dividindo as informações em dois níveis e utilizando os arquivos originais do projeto, causa um impacto muito pequeno em relação ao espaço de memória em projetos de bioinformática.

A gravação dos dados em arquivos XML e a separação em pastas padronizadas, facilita a troca de informações, permitindo que o usuário compartilhe seus experimentos com outros pesquisadores. Além disso, o recurso que permite incluir anexos no experimento é uma importante ferramenta para que o usuário guarde dados importantes para a execução de seus experimentos, assim como os resultados obtidos.

A implementação das restrições no simulador permite que o usuário construa seus grafos com mais facilidade, focando nos detalhes do experimento e deixando a verificação das restrições de forma automática. Além disso, a criação automática da aresta *WasDerivedFrom* cria grafos mais precisos, permitindo ao usuário informar somente o uso e a geração das coleções.

Por fim, a criação de grafos personalizados fornece ao usuário uma ferramenta capaz de demonstrar diferentes visões de seus experimentos. A inclusão de anotações enriquece o grafo com informações importantes para o seu entendimento. Isso permite que o usuário demonstre seus experimentos de maneira mais clara, compartilhando seus resultados.

6.1 Trabalhos Futuros

Alguns aspectos ainda precisam ser implementados no simulador *Provenance* a fim de que a aplicação do modelo PROV-DM seja completamente realizada.

A implementação das restrições funcionais deve ser realizada, não somente garantindo a existência dos arquivos apontados pelas coleções, como também garantindo a sua autenticidade, seja armazenando informações desses arquivos para serem comparadas posteriormente ou utilizando algoritmos específicos de validação, como MD5 (RIVEST, 1992), por exemplo.

A implementação de classes para a leitura dos arquivos apontados pelas coleções, interpretando-os a fim de recuperar as entidades que as compõem. A existência de diferentes tipos de arquivos é um desafio para esse trabalho, e a implementação de uma classe genérica e configurável pelo usuário seria de grande utilidade. Essas classes, além de permitir a recuperação das entidades, também podem ser ferramentas interessantes para a análise dos arquivos a fim de recuperar, de forma automática, informações tais como o tamanho de cada coleção e dados estatísticos relevantes.

Outro ponto importante é a captura de informações de proveniência. Mecanismos de captura podem ser acoplados ao simulador *Provenance*, permitindo que informações da execução das atividades e seus resultados sejam recuperadas e armazenadas de forma automática ou semi-automática.

E, finalmente, o aprimoramento de algumas características do simulador *Provenance* podem ser sugeridas, tais como: implementação das restrições funcionais; permitir a criação de grafos de proveniência mais flexíveis, sem informações temporais, por exemplo; incluir novas opções para a geração de grafos personalizados como uma lista de grupos

excludentes, ou a inclusão, de forma automática, de dados dos nós na forma de anotações; gravar o grafo (*Account*) e todos os seus elementos com o formato XML padrão do modelo PROV-DM, além de exportá-lo utilizando a notação padrão definida pelo modelo.

6.2 Contribuições e Publicações

O principal objetivo deste trabalho é propiciar o gerenciamento de proveniência de dados para projetos de bioinformática assim, pode-se separar as contribuições feitas em relação à extensão do modelo PROV-DM e relacionadas ao desenvolvimento do simulador *Provenance*.

Primeiramente, pode-se citar a avaliação de diferentes modelos de proveniência de dados objetivando a sua aplicação em projetos de bioinformática. Os critérios avaliados permitem verificar diversos aspectos da aplicação e fornecem uma visão geral de

Também, a divisão dos dados em proveniência em dois níveis, permitindo que o grande volume de dados relacionado aos experimentos de bioinformática seja mantido no Nível 2 e corresponde aos arquivos originalmente tratados no experimento. O Nível 1, por suas vez, possui as informações adicionais de proveniência que constituem um esquema de ligação das atividades executadas e s arquivos usados e gerados. Conforme pode-se ver nos estudos de caso, essa adição de dados é relativamente pequena porém, suficiente para garantir que os experimentos sejam descritos com detalhes e o fluxo dos dados seja demonstrado.

Outra contribuição importante relacionada ao PROV-DM refere-se a adição da entidade Projeto. Essa entidade permite que os biólogos agrupem experimentos relacionados facilitando o seu gerenciamento.

Por fim, o simulador *Provenance* demonstrou a capacidade de fazer a ligação das informações de proveniência em dois níveis com um pequeno acréscimo de dados, além de propiciar a criação de grafos de proveniência com diferentes configurações.

A partir deste trabalho, foram elaborados 2 textos aceitos para as seguintes conferências:

- *A Provenance Model For Bioinformatics Workflow: A Case Study*: *Short paper* aceito para a conferência de Bioinformática e Biomedicina (BIBM) realizado em Atlanta, Estados Unidos em 2011 (BIBM/2011);
- *A Provenance Data Model to Manage RNA-SEQ Projects*: Artigo aceito para a 4ª Conferência em Bioinformática e Biologia Computacional (BICoB) realizada em Las Vegas, Estados Unidos em 2012.
- *Managing Data Provenance in Genome Project Workflow*: Artigo aceito para o *workshop Data-Mining of Next-Generation Sequencing* da Conferência de Bioinformática e Biomedicina (BIBM) realizado na Philadelphia, Estados Unidos em 2012 (BIBM/2012), além do convite para publicação deste artigo em uma edição especial da revista *BMC Bioinformatics*;

Referências Bibliográficas

ACAR, U. et al. A graph model of data and workflow provenance. In: *Proceedings of the 2nd conference on Theory and practice of provenance*. Berkeley, CA, USA: USENIX Association, 2010. (TAPP'10), p. 1–10.

ALDECO-PÉREZ, R.; MOREAU, L. *Provenance-based Auditing of Private Data Use*. BCS, 2008. 141–152 p. Disponível em: <http://eprints.ecs.soton.ac.uk/16580/>.

ALTSCHUL, S. F. et al. Basic local alignment search tool. *Journal of Molecular Biology*, n. 215, p. 403–410, 1990.

BENTLEY, D. R. Whole-genome re-sequencing. *Current opinion in genetics and development*, v. 16, n. 6, p. 545–552, dez. 2006. ISSN 0959-437X.

BOWERS, S. et al. *A model for user-oriented data provenance in pipelined scientific workflows*. Springer, 2006. 133–147 p. Disponível em: <http://www.ipaw.info/ipaw06/proceedings/CameraReady_s5_3.pdf>.

BRAUN, U. J. et al. Towards query interoperability : Passing plus. *TaPP*, p. 1–10, 2010.

BUNEMAN, P.; KHANNA, S.; TAN, W. chiew. Why and where: A characterization of data provenance. In: *In ICDT*. [S.l.]: Springer, 2001. p. 316–330.

BUNGE, M. *Treatise on Basic Philosophy: Vol. 3: Ontology I: The Furniture of the World. Reidel*. 1977.

CAO, B. et al. Provenance information model of karma version 3. *3rd international workshop on scientific workflows SWF*, IEEE, I, n. Vdl, p. 348–351, 2008.

CBRG. *Examples of common sequence file formats*. 2011. Disponível em: <http://www.compbio.ox.ac.uk/bioinformatics_faq/format_examples.shtml>.

CHAPMAN, A.; BLAUSTEIN, B.; ELSAESSER, C. Provenance-based belief. *on the Theory and Practice of*, p. 1–14, 2010.

CHENEY, J. Workshop on theory and practice of provenance event report. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 38, p. 57–60, 2009. ISSN 0163-5808.

COCK, P. J. A. et al. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, Oxford University Press, v. 38, n. 6, p. 1767–1771, abr. 2010. ISSN 1362-4962.

DAHM, R. Discovering DNA: Friedrich Miescher and the early years of nucleic acid research. *Human Genetics*, Springer Berlin / Heidelberg, v. 122, n. 6, p. 565–581, jan. 2008. ISSN 0340-6717.

DAVIDSON, S. B.; FREIRE, J. Provenance and scientific workflows: challenges and opportunities. In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2008. (SIGMOD '08), p. 1345–1350. ISBN 978-1-60558-102-6.

ELLSON, J. et al. Graphviz and dynagraph - static and dynamic graph drawing tools. In: *GRAPH DRAWING SOFTWARE*. [S.l.]: Springer-Verlag, 2003. p. 127–148.

FELSENSTEIN, J. *Phylip*. 2009. Disponível em: <http://evolution.genetics.washington.edu/phylip.html>.

FREIRE, J. et al. Provenance for computational tasks: A survey. *Computing in Science Engineering*, IEEE Computer Society, v. 10, n. 3, p. 11–21, 2008. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4488060>.

GLAVIC, B.; DITTRICH, K. *Data Provenance: A Categorization of Existing Approaches*. [S.l.]: Verlagshaus Mainz, Aachen, 2007. 227–241 p.

GOMES, L. S. A. *Proveniência para Workflows de Bioinformática*. Dissertação (Mestrado) — Pontificial Catholic University, Rio de Janeiro, Brazil, April 2011.

GROTH, P. et al. *Applying the Provenance Data Model to a Bioinformatics Case*. [S.l.]: IOS Press, 2008. 1–15 p.

HANNON, G. J. *FASTX-Toolkit*. 2012. Disponível em: <http://hannonlab.cshl.edu/fastx_toolkit/>.

HARTIG, O.; ZHAO, J. Using web data provenance for quality assessment. In: *SWPM*. [S.l.: s.n.], 2009. p. 1–6.

HARTIG, O.; ZHAO, J. Publishing and consuming provenance metadata on the web of linked data. *Provenance and Annotation of Data and Processes*, Springer Berlin Heidelberg, v. 6378, n. 24, p. 78–90, 2010.

HGSCI, Human Genome Sequencing Consortium International. Finishing the euchromatic sequence of the human genome. *Nature*, Nature Publishing Group, v. 431, n. 7011, p. 931–945, out. 2004. ISSN 0028-0836.

HIGGINS, D. G.; SHARP, P. M. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, v. 73, n. 1, p. 237–244, dez. 1988.

HOLLAND, R. C. G. et al. Biojava: an open-source framework for bioinformatics. *Bioinformatics*, v. 24, n. 18, p. 2096–2097, 2008.

JCBN. Abbreviations and symbols for the description of conformations of polynucleotide chains. *Proceedings of the 16th Jerusalem Symposium Nucleic acids, the vectors of life)*, Dordrecht, Holanda, p. 559–565, 1983.

JR, C. da S.; SASSON, S. *Biologia 1 - Reformulado*. 7. ed. São Paulo: Saraiva, 2002.

KEßLER, C.; TRAME, J.; KAUPPINEN, T. Tracking editing processes in volunteered geographic information: The case of openstreetmap. *Processes and Events in Spatio-Temporally Distributed Data (IOPE), workshop at Conference on Spatial Information Theory 2011 (COSIT'11)*, Belfast, Maine, USA, p. 1–7, 2011. Disponível em: <http://www.carsten.io/iope2011.pdf>.

KENT, J. J. BLAT - the BLAST-like alignment tool. *Genome Research*, Department of Biology and Center for Molecular Biology of RNA, University of California-Santa Cruz, Santa Cruz, CA 95064, USA. kent@biology.ucsc.edu, v. 12, n. 4, p. 656–664, abr. 2002. ISSN 1088-9051.

KOUTSOFIOS, E.; NORTH, S. C. *Drawing Graphs With dot*. 1993. Disponível em: <http://www.phil.uu.nl/ js/graphviz/dotguide.pdf>.

LANGMEAD, B. et al. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, v. 10, n. 3, p. 1–10, 2009. ISSN 1465-6906.

LEVENE, P. A.; LONDON, E. S. The structure of thymonucleic acid. *Journal of Biological Chemistry*, v. 83, n. 3, p. 793–802, 1929. Disponível em: <http://www.jbc.org/content/83/3/793.short>.

LI, H. et al. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, Oxford University Press, v. 25, n. 16, p. 2078–2079, ago. 2009. ISSN 1460-2059.

LIU, J.; RAM, S. Who does what: Collaboration patterns in the wikipedia and their impact on data quality. In: *Proceedings of nineteenth Annual Workshop on Information Technologies and Systems (WITS 2009)*. Phoenix, Arizona, USA: [s.n.], 2009. p. 1–8.

MARDIS, E. R. Next-generation dna sequencing methods. *Annu Rev Genomics Hum Genet*, v. 9, p. 387–402, 2008.

MARINHO, A. et al. A strategy for provenance gathering in distributed scientific workflows. In: *Proceedings of the 2009 Congress on Services - I*. Washington, DC, USA: IEEE Computer Society, 2009. p. 344–347. ISBN 978-0-7695-3708-5. Disponível em: <http://portal.acm.org/citation.cfm?id=1590963.1591573>.

MARINS, A. et al. Modeling provenance for semantic desktop applications. In: *Proceedings of the XXVII Congresso da SBC - XXXIV Seminário Integrado de Software e Hardware*. [S.l.: s.n.], 2007. p. 2100–2012.

MARIONI, J. C. et al. RNA-SEQ: An assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, v. 18, n. 9, p. 1509–1517, 2008.

MISSIER, P.; BELHAJJAME, K. *A PROV encoding for provenance analysis using deductive rules*. Newcastle upon Tyne, NE1 7RU, England., April 2012. 1–17 p.

MISSIER, P. et al. Janus: From workflows to semantic provenance and linked open data. *Procs IPAW 2010*, Springer, "I", n. i, p. 129–141, 2010.

MOREAU, L. *The PROV Toolbox*. 2011. Disponível em: <https://github.com/lucmoreau/ProvToolbox/wiki/ProvToolbox-Home>.

MOREAU, L. et al. *The Open Provenance Model - Core Specification (v1.1)*. 2010. 1–15 p. Disponível em: <http://openprovenance.org/>.

MOREAU, L. et al. Governance of the open provenance model. 2009. Disponível em: <http://twiki.ipaw.info/pub/OPM/WebHome/governance.pdf>.

MOREAU, L. et al. *The Open Provenance Model (OPM)*. 2010. Http://openprovenance.org/. Disponível em: <http://openprovenance.org/>.

MOREAU, L. et al. Special issue: The first provenance challenge. *Concurrency and Computation: Practice and Experience*, John Wiley and Sons, Ltd., v. 20, n. 5, p. 409–418, 2008. ISSN 1532-0634.

NCBI. *Standard Flowgram Format - SFF*. 2011. Disponível em: <http://www.ncbi.nlm.nih.gov/Traces/trace.cgi?cmd=show/&/#38;f=formats/&/#38;m=doc/&/#38;s=formats/#sff>.

OMITOLA, T.; GIBBINS, N.; SHADBOLT, N. Provenance in linked data integration. In: *Future Internet Assembly*. [s.n.], 2010. p. 1–8. Disponível em: <http://eprints.ecs.soton.ac.uk/21954/>.

OPM. *Open Provenance Model (OPM) XML Schema Specification*. 2010. Disponível em: <http://openprovenance.org/model/opmx>.

OPM. *Open Provenance Model - Wiki*. 2011. Disponível em: <http://twiki.ipaw.info/bin/view/Challenge/OPM>.

OPM. *Open Provenance Specifications*. 2011. Disponível em: <http://openprovenance.org/java/site/1_1_8/>.

OPM. *OPM Toolbox*. 2011. Disponível em: <http://openprovenance.org/java/site/1_1_8/toolbox/index.html>.

ORLANDI, F.; PASSANT, A.; CHAMPIN, P.-A. Semantic representation of provenance in wikipedia. In: *Second International Workshop on Role of Semantic Web in Provenance Management (SWPM 2010 - Workshop of ISWC 2010)*. [s.n.], 2010. p. 1–6. Disponível em: <http://liris.cnrs.fr/publis/?id=5017>.

PATNI, H. et al. Provenance aware linked sensor data. In: *2nd Workshop on Trust and Privacy on the Social and Semantic Web, Co-located with ESWC*. [S.l.: s.n.], 2010. p. 1–12.

PAULA, R. de et al. A provenance data model to manage RNA-SEQ projects. *Proceedings of 4th International Conference on Bioinformatics and Computational Biology 2012*, Las Vegas, NV, USA, p. 165–170, 2012.

Provenance Challenge Wiki. *Provenance Challenge Wiki*. 2012. Disponível em: <http://twiki.ipaw.info/bin/view/Challenge/>.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2009.

RAM, S.; LIU, J. Active conceptual modeling of learning. In: CHEN, P. P.; WONG, L. Y. (Ed.). *Active conceptual modeling of learning*. Berlin, Heidelberg: Springer-Verlag, 2007. cap. Understanding the semantics of data provenance to support active conceptual modeling, p. 17–29. ISBN 3-540-77502-1, 978-3-540-77502-7. Disponível em: <http://portal.acm.org/citation.cfm?id=1793834.1793838>.

RIVEST, R. *The MD5 Message-Digest Algorithm*. Fremont, CA, USA, abr. 1992. Disponível em: <http://www.rfc-editor.org/rfc/rfc1321.txt>.

ROBERTS, A. et al. Identification of novel transcripts in annotated genomes using RNA-Seq. *Bioinformatics*, Oxford University Press, v. 27, n. 17, p. 2325–2329, jun. 2011. ISSN 1460-2059.

ROTHBERG, J. M.; LEAMON, J. H. The development and impact of 454 sequencing. *Nature Biotechnology*, Nature Publishing Group, v. 26, n. 10, p. 1117–1124, out. 2008. ISSN 1087-0156.

SAHOO, S. S.; SHETH, A. Provenir ontology: Towards a framework for escience provenance management. *Microsoft eScience Workshop*, Microsoft Research, v. 1, 2009.

SAHOO, S. S. et al. Ontology-driven provenance management in escience: An application in parasite research. In: *Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part II*. Berlin, Heidelberg: Springer-Verlag, 2009. (OTM '09), p. 992–1009. ISBN 978-3-642-05150-0.

SANGER, F. et al. Nucleotide sequence of bacteriophage Phi X174 DNA. *Nature*, v. 265, n. 5596, p. 687–695, 1977.

SCHUSTER, S. C. Next-generation sequencing transforms today's biology. *Nature Methods*, Nature Publishing Group, v. 5, n. 1, p. 16–18, 2008. Disponível em: <http://www.biovip.org/UploadFiles/2008-1/116511530.pdf>.

SETUBAL, J. C.; MEIDANIS, J. *Introduction To Computational Molecular Biology*. 1. ed. [S.l.]: PWS Publishing Company, 1997.

TAN, W.-C. Research problems in data provenance. *IEEE Data Engineering Bulletin*, v. 27, p. 45–52, 2004.

TRAPNELL, C.; PACHTER, L.; SALZBERG, S. L. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, Oxford University Press, v. 25, n. 9, p. 1105–1111, maio 2009. ISSN 1460-2059.

UNIPROT. *Universal Protein Resource*. 2012. Disponível em: <http://www.uniprot.org>.

W3C. *Formal Semantics*. 2011. Disponível em: <http://www.w3.org/2011/prov/wiki/FormalSemanticsStrawman>.

W3C. *ProvXML*. 2011. Disponível em: <http://www.w3.org/2011/prov/wiki/ProvXML>.

W3C. *Constraints of the Provenance Data Model*. 2012. Disponível em: <www.w3.org/TR/prov-constraints/>.

W3C. *PROV-AQ: Provenance Access and Query*. 2012. Disponível em: <http://www.w3.org/TR/prov-o/>.

W3C. *PROV-DM: The PROV Data Model*. 2012. Disponível em: <www.w3.org/TR/prov-dm/>.

W3C. *PROV Model Primer*. 2012. Disponível em: <http://www.w3.org/TR/prov-primer/>.

W3C. *PROV-N: The Provenance Notation*. 2012. Disponível em: <http://www.w3.org/TR/prov-n/>.

W3C. *PROV-O: The PROV Ontology*. 2012. Disponível em: <http://www.w3.org/TR/prov-o/>.

WATSON, J. D.; CRICK, F. H. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature*, Springer, v. 171, n. 4356, p. 737–738, 1953. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/17804965>.

Anexo I

Usando o Simulador *Provenance*

Este anexo apresenta um guia rápido para a utilização do simulador *Provenance*.

A primeira vez que o simulador é executado em um computador, uma janela pede para o usuário preencher alguns campos referentes à sua configuração. Esta tela é mostrada Na Figura I.1, onde podem ser vistos os campos para a indicação do local onde são gravados os projetos e a configuração do programa para a criação do grafo.

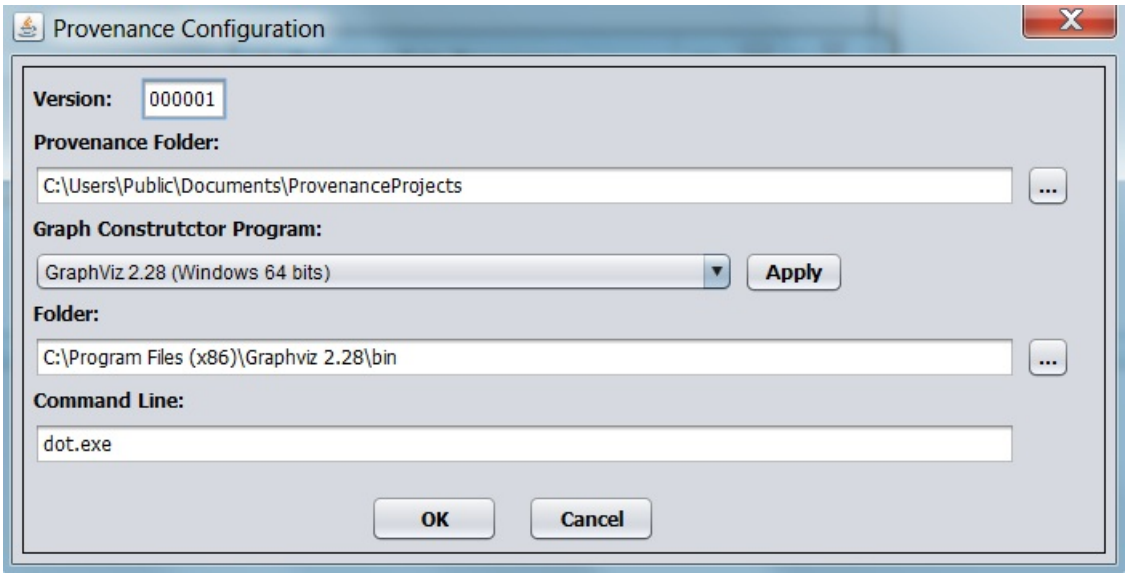


Figura I.1: Tela de configuração do simulador.

Após a configuração, o simulador exibe a tela mostrada na Figura I.2 onde o usuário pode escolher entre abrir um projeto gravado anteriormente, criar um novo projeto ou abrir a tela de configuração do simulador.

A Figura I.3 mostra a tela de edição de um projeto. Todos os campos são descritos na Tabela 4.1. Cada projeto pode conter diversos grafos, que são listados na lista *"Attached Accounts"*. Para ver qualquer grafo de proveniência basta o usuário selecionar o arquivo na lista e clicar no botão *"Open"*. O usuário também tem a opção de criar um novo grafo (*"Create"*), tirar um grafo do projeto (*"Detach"*) ou incluir uma grafo de outro projeto (*"Attach"*). Para incluir um grafo feito em outro computador basta o usuário copiar o arquivo XML para a pasta do projeto e incluí-lo usando a opção *"Attach"*.

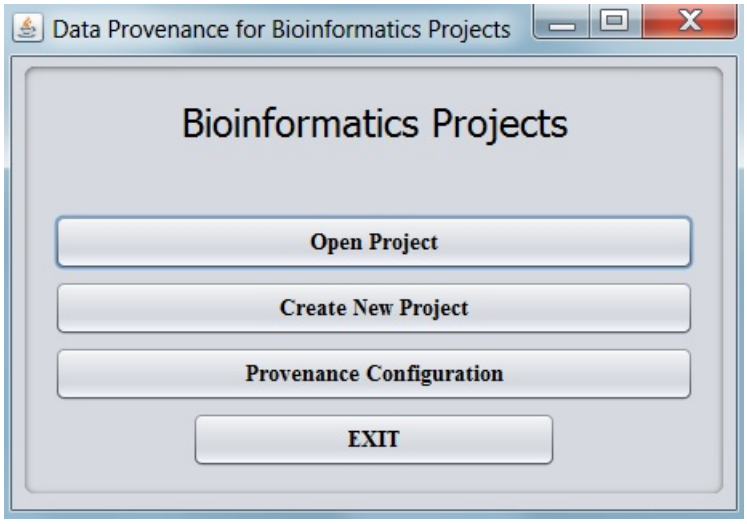


Figura I.2: Tela inicial do simulador.

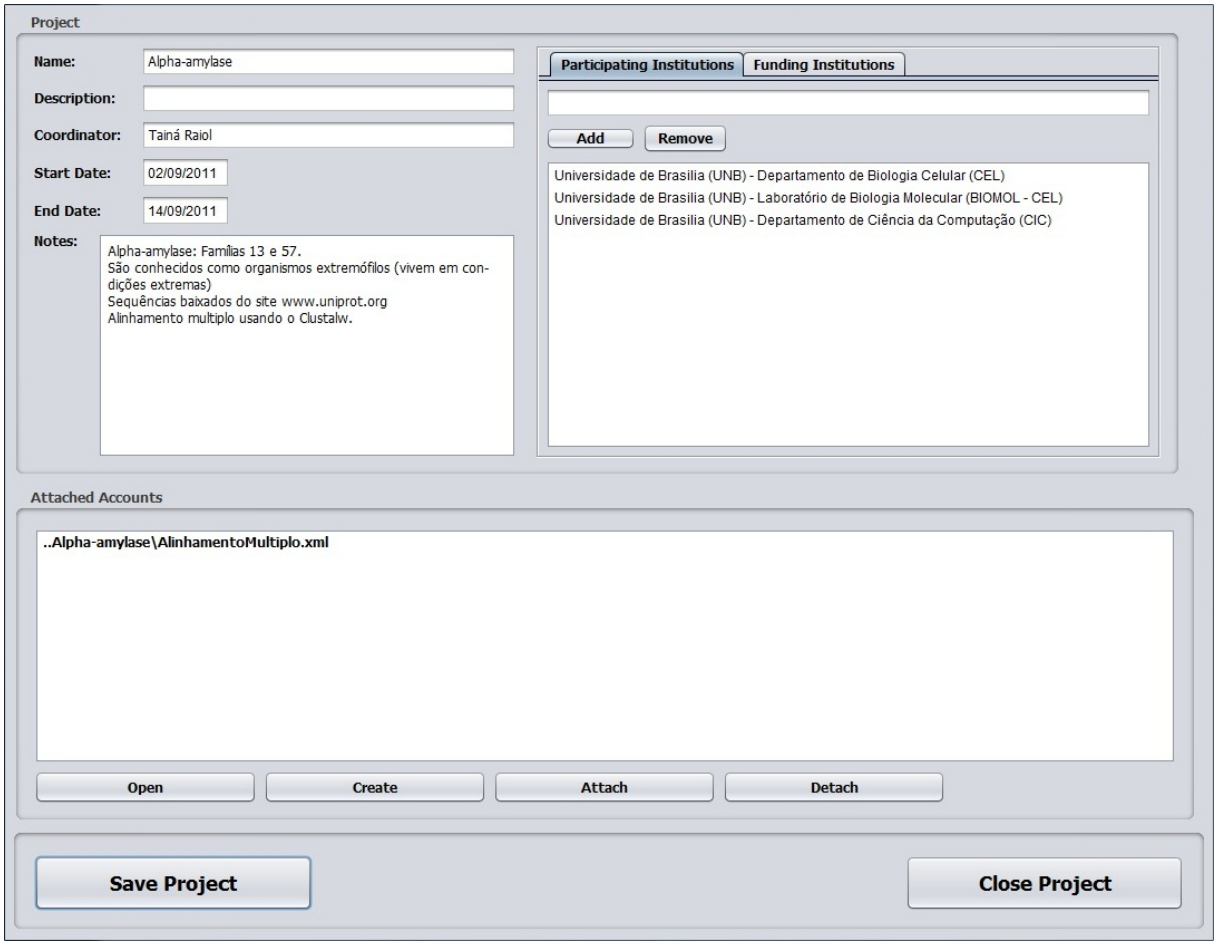


Figura I.3: Exemplo de tela preenchida com dados de um projeto.

Ao abrir um grafo (ou *Account)*, o usuário tem acesso à tela que permite criar cada nó do grafo juntamente com as relações existente entre eles. A Figura I.4 mostra a tela de um grafo previamente preenchida. Esta tela pode ser dividida em três partes: cadastro

geral, relação dos nós e relação das arestas.

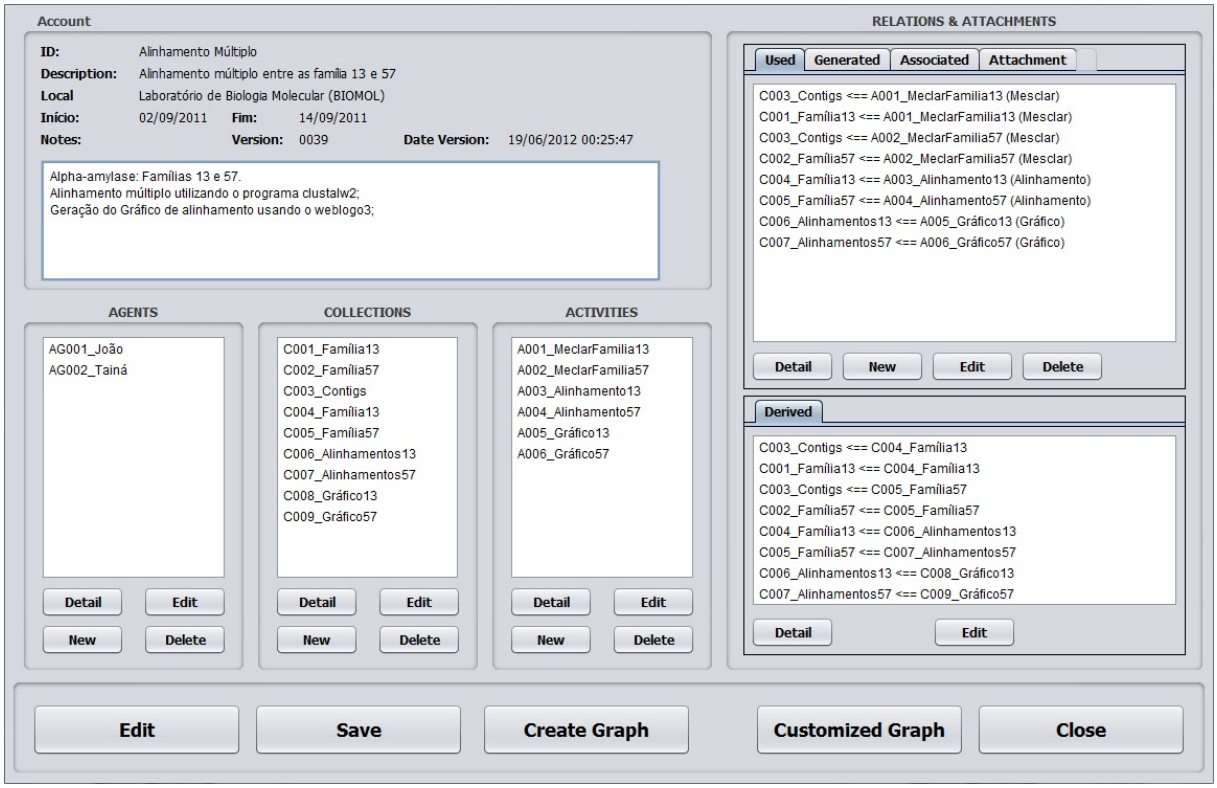


Figura I.4: Exemplo de tela preenchida com os dados de um grafo.

O cadastro geral do grafo fica na parte superior esquerda, onde pode ser visto a área com os dados principais do grafo, tais como nome, data da execução, local, entre outros.

Na parte inferior esquerda tem-se a relação dos nós, separada em três áreas: agente, coleção e atividade. Cada área exibe seus ítens e possui botões para incluir, alterar, excluir ou detalhar. Cada ítem corresponde a um nó do grafo.

A parte direita da tela de edição do grafo é reservada para a relação das arestas. Cada aresta representa uma relação existente entre os nós sendo que, na parte superior, existem abas para escolher entre as três relações *Used*, *Generated* e *Associated*. A inclusão de uma nova relação é feita escolhendo-se os dois elementos desejados e pressionando-se o botão *"New"* na aba que corresponde à relação desejada. A relação *Derived* está separada na parte inferior da tela porque não permite inclusão nem exclusão. Esta relação é criada ou excluída automaticamente, obedecendo as regras de inferência temporais descritas na Seção 4.5.2. Na parte superior esquerda, juntamente com as três primeiras relações, existe uma aba chamada *Attachment*, a qual permite que o usuário anexe informações adicionais ao grafo, tais como bibliografias, sites da internet, artigos, ou qualquer outro arquivo.

A Figura I.5 mostra um exemplo de tela de um agente. As telas de edição dos agentes, atividades e coleções são semelhantes e podem ser divididas em 3 partes. Na parte superior da tela de edição de nó são exibidos os campos principais para edição. Na parte mediana são encontradas abas com as seguintes opções: *Groups*, *Notes*, *Graph Notes* e *Attachments*. A aba *Groups*, mostrada na Figura I.5, permite ao usuário indicar a quais grupos o nó pertence. Dessa forma, o usuário poderá construir grafos personalizados somente com os

grupos desejados. A segunda aba, *Notes*, permite que o usuário inclua quaisquer anotações referente ao nó. A aba *Graph Notes* permite ao usuário incluir uma pequena observação que poderá ser impressa no grafo. Por fim, a aba *Attachment* fornece opções ao usuário para anexar diversos arquivos, sites da internet ou qualquer outra informação relevante para o nó. Na parte inferior da tela de edição de nós, são exibidas diversas abas com as relações (arestas) das quais aquele nó participa, como origem ou destino. As abas que serão exibidas depende do tipo do nó.

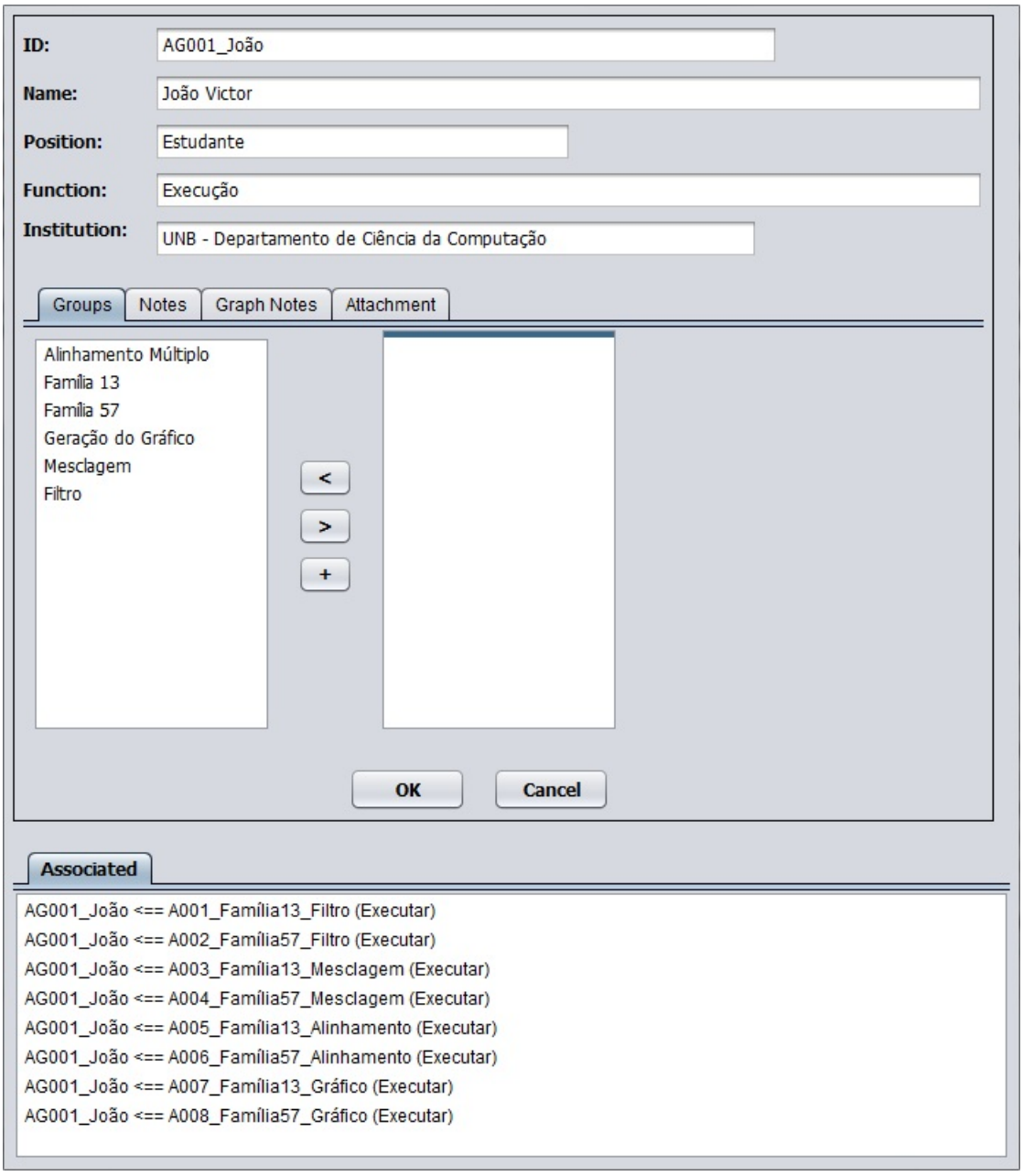


Figura I.5: Tela de edição do agente.

Na parte inferior da tela existem duas opções importantes para o uso do simulador, são elas *Create Graph* e *Customized Graph*. A primeira cria o grafo de proveniência a partir de todos os dados incluídos nesta tela. A segunda permite que o usuário escolha os grupos

que deseja incluir no grafo. Dessa forma somente os elementos participantes dos grupos escolhidos serão exibidos. Cada grafo personalizado criado pelo usuário é gravado como um anexo no grafo, permitindo que o usuário reveja seus grafos a qualquer momento. Além disso, em ambas as opções, o usuário pode escolher se os grafos devem conter as regras das relações e as anotações feitas em cada elemento. A imagem mostrada à esquerda na Figura I.6 demonstra a tela para a criação do grafo de proveniência completo e na direita a tela para a escolha dos grupos e opções para criação de um grafo personalizado.

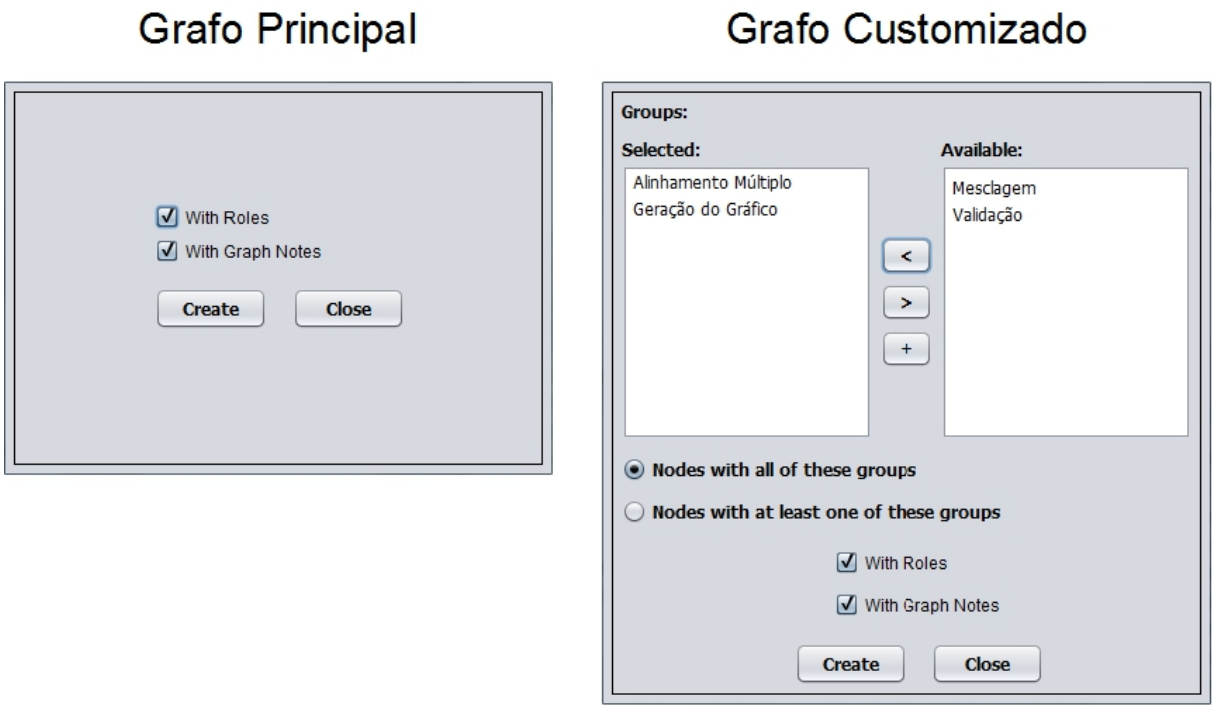


Figura I.6: Telas de criação do grafo principal e dos grafos personalizados.