UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

OntologyManagementTool – UMA FERRAMENTA PARA GERENCIAMENTO DE ONTOLOGIAS COMO TEORIAS LÓGICAS

Dissertação de Mestrado

ÂNGELA MARIA ALVES PINHEIRO

Fortaleza-CE, 2013

ÂNGELA MARIA ALVES PINHEIRO

OntologyManagementTool – UMA FERRAMENTA PARA GERENCIAMENTO DE ONTOLOGIAS COMO TEORIAS LÓGICAS

Dissertação apresentada à Coordenação do Programa de Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação.

Área de concentração: Banco de Dados

Orientador: Prof. Dr. José Antônio Fernandes Macêdo

Fortaleza-CE, 2013

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Ciências e Tecnologia

P718o Pinheiro, Ângela Maria Alves.
OntologyManagement Tool : uma ferramenta para gerenciamento de ontologias como teorias lógicas.. / Ângela Maria Alves Pinheiro. – 2013.
76f. : il. , enc. ; 30 cm.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Departamento de Computação, Programa de Pós Graduação em Ciência da Computação, Fortaleza, 2013.
Área de Concentração: Bancos de Dados.
Orientação: Prof. Dr. José Antônio Fernandes Macêdo.
.

1.Teoria dos grafos. 2. Banco de dados distribuído. 3. Algebra. I. Título.

CDD 005

ÂNGELA MARIA ALVES PINHEIRO

OntologyManagementTool – UMA FERRAMENTA PARA GERENCIAMENTO DE ONTOLOGIAS COMO TEORIAS LÓGICAS

Dissertação apresentada à Coordenação do Programa de Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação.

Composição da Banca Examinadora:

_____

Prof. Dr. José Antônio Fernandes de Macêdo (DC/UFC)

Orientador

_____

Prof. Dr. Marco Antonio Casanova (PUC/Rio)

_____

Prof. Dr João Fernando Lima Alcântara (DC/UFC)

A Deus, aos amigos e a minha família.

AGRADECIMENTOS

A Deus, por tudo e principalmente por ter guiado o meu caminho com fé, esperança e determinação.

A minha família, pelo suporte sempre que precisei seja celebrando as minhas conquistas ou acreditando em mim apesar das derrotas. Em especial, a minha irmã Angellyne que sempre tinha uma palavra de apoio e incentivo para os momentos desanimadores ou uma festa para celebrar as alegrias da vida.

Ao meu orientador José Antônio Fernandes de Macêdo pela paciência, ensinamentos, valiosas discussões e por contribuir para o meu amadurecimento profissional. Sou muito grata pela oportunidade de concluir esse desafio.

Ao professor Marco Antônio Casanova, pelas ideias, atenção, críticas e valiosas discussões que viabilizaram e contribuíram de forma imensurável para o desenvolvimento desse trabalho.

Ao professor João Fernando Lima Alcântara, pela participação em minha banca de mestrado.

A todos os professores do Departamento de Computação e, principalmente, membros do grupo ARIDA (Advanced Research In DAtabase).

À professora Bernadette Farias Lóscio, pela paciência, compreensão e os ensinamentos.

Aos meus queridos amigos, alguns destes me ajudaram com relação às disciplinas, com a amizade, ideias ou também conselhos, pode-se citar: Aragão, Clayton, Eveline, Fabiana, Fernanda Lígia, Jackson, Juliana e Luiz. Em especial, a Juliana que se mostrou uma amiga prestativa em todas as ocasiões.

Aos funcionários do Departamento de Computação, em especial, ao Orley que sempre esteve de forma prestativa cuidando dos assuntos burocráticos.

À FUNCAP (Fundação Cearense de Apoio a Pesquisa), pelo apoio financeiro durante todo o mestrado.

A Universidade Federal do Ceará pela oportunidade do grande crescimento profissional.

A todos que se alegraram comigo pela conquista deste objetivo.

RESUMO

Diversos projetos nacionais e internacionais, como o *dados.gov.br* e o *Linking Open Data*, foram desenvolvidos com a finalidade de fomentar a criação da Web de dados, que surge como uma nova abordagem para efetivamente publicar, recuperar e descrever dados distribuídos na Web. Diante desse cenário, tais projetos enfrentam o desafio de criar e manter os dados estruturados segundo os princípios do *Linked Data*, descritos no modelo de dados RDF e representados por ontologias. Esse desafio envolve outras tarefas complexas, tais como: reusar o vocabulário das ontologias largamente utilizadas na elaboração de novas ontologias (com a finalidade de promover a interoperabilidade e a integração entre as aplicações) e permitir a detecção de inconsistências entre os termos de uma determinada ontologia.

Com o objetivo de propor uma solução para esse desafio, o problema de gerenciamento de ontologias foi abordado nesta dissertação. Na literatura, existe uma grande variedade de trabalhos disponíveis com diferentes enfoques e processos que propõem o gerenciamento de ontologias. Entretanto, poucos trabalhos preocupam-se em auxiliar o especialista do domínio na elaboração de uma ontologia que representa um entendimento correto sobre a semântica das ontologias envolvidas, visto que, para isso faz-se necessário considerar as restrições lógicas das ontologias originais e propagá-las para a nova ontologia. Além disso, foi percebido que, nos trabalhos anteriores, há necessidade de se utilizar várias ferramentas durante o processo de gerenciamento de ontologias, o que aumenta o esforço manual a ser despendido pelo especialista do domínio na elaboração de novas ontologias. Sendo assim, a fim de oferecer algumas funcionalidades diferenciadas e de modo integrado às funcionalidades usuais de gerenciamento de ontologias, foi desenvolvido um protótipo, denominado ***OntologyManagementTool***.

O protótipo desenvolvido considera as ontologias não apenas como vocabulários, mas como teorias lógicas, isto é, leva em conta também os seus conjuntos de restrições. Cada ontologia manipulada é primeiramente normalizada para atender ao formalismo da *Lógica Descritiva*, com um número específico de restrições. Posteriormente, essa ontologia é transformada em um grafo de restrições, e assim, é possível gerenciá-la a partir de um conjunto de operações algébricas sobre o grafo. Destacam-se as seguintes operações: **união**, **interseção**, **diferença** e **projeção.** Após a execução de cada uma dessas operações, é possível obter uma nova ontologia, bem como, o mapeamento entre as ontologias envolvidas.

O trabalho proposto teve a sua aplicabilidade comprovada a partir de experimentos executados em ontologias descrevendo fontes de dados reais. A análise da complexidade dos algoritmos deste trabalho mostrou que a complexidade para gerar o grafo de restrições é linear em relação ao número de restrições das ontologias; já a complexidade do processamento das operações algébricas (interseção, diferença e projeção) é quadrática em relação ao número de vértices do grafo de restrições, sendo importante evidenciar que o fator determinante para obtenção dessa complexidade é o procedimento escolhido para lidar com as restrições de inclusão, denominado fecho transitivo.

Palavras-chaves: *Linked Data*, Grafo, Álgebra.

Lista de Tabelas

Lista de Ilustrações

Sumário

CAPÍTULO 1

1.1. INTRODUÇÃO

A Web é atualmente um enorme espaço global de documentos e dados interligados e distribuídos em diversas fontes de dados autônomas e heterogêneas, que se expande a cada dia, dificultando cada vez mais a recuperação de informações de modo integrado. Neste contexto, surgiu uma nova abordagem para efetivamente publicar, recuperar e descrever dados distribuídos na Web, denominada Web de dados, capaz de viabilizar a integração de dados na Web.

É importante evidenciar que a Web de dados possibilita a redução da complexidade de integração de dados devido às ligações estabelecidas entre os conjuntos de dados, bem como, o reuso de informações entre as fontes de dados.

A Web de dados fundamenta-se nos princípios dos *Linked Data*, conceito estruturado pelo pesquisador Tim Berners-Lee. *Linked Data* pode ser definido como um conjunto de melhores práticas para a publicação e a conexão de dados estruturados na Web baseado em tecnologias de Web Semântica. Essas tecnologias têm um papel importante na viabilidade da construção da Web de dados, destacam-se: **RDF** (*Resource Description Framework*) (Klyne,G. *et al.*,2010), um modelo de dados simples e extensível que descreve os dados e seus relacionamentos; **URI** (*Unified Resource Identification*) (Berners-Lee T. *et al.*, 2005), um mecanismo de nomeação global dos dados; e **OWL** (*Web Ontology Language*) (*McGuinness,D.L.,Harmelen,F.*,2010), um modelo computacional que possibilita maior descrição dos dados, ou seja, descreve relação entre classes de dados (ex. disjunção), cardinalidade (ex. "pelo menos um"), características das propriedades (ex. simetria), além de possibilitar o uso de mecanismos de inferência.

Um dos principais desafios que envolvem a área de *Linked Data* consiste na criação e manutenção dos dados por diversos usuários. Esse desafio acarreta outros problemas, tais como: possibilitar a criação de dados a partir do reuso dos vocabulários das fontes de dados existentes e permitir que as atualizações dos dados sejam realizadas e validadas com a detecção de inconsistências. Portanto, um mecanismo para gerenciamento da Web de dados se mostra relevante, já que a Web de dados é formada por dados provenientes dos mais diversos domínios, causando problemas quanto à confiabilidade do resultado disponibilizado. Além disso, a Web de dados é dinâmica e, assim, deve permitir que vários usuários realizem constantes atualizações e identifiquem possíveis inconsistências derivadas dessas atualizações.

1.2. MOTIVAÇÃO

Devido ao grande volume de informações disponíveis na Web e à busca pela facilidade de acesso integrado a dados distribuídos em diferentes fontes de dados foi imprescindível que ocorresse uma evolução da Web. Esta nova Web, denominada Web de dados, está se tornando um enorme espaço global de documentos e dados interligados, sendo fundamentada por um conjunto de melhores práticas para a publicação e a conexão de dados estruturados na Web, conhecido como *Linked Data*.

Diversos projetos nacionais e internacionais foram desenvolvidos com a finalidade de fomentar a criação da Web de dados. Como exemplo, destacam-se o projeto nacional *dados.gov.br*[1] e o projeto internacional *Linking Open Data (*LOD*)*[2].

O projeto nacional *dados.gov.br* visa a centralização de acesso e a disponibilização do maior

[1] www.data.gov.br

[2] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

número possível de dados e informações públicas do governo federal, sendo uma fonte de subsídio para pesquisadores, empresas, comunidade de Tecnologia da Informação (TI), gestores públicos e a sociedade em geral. Em maio de 2011, foi lançado um portal que funciona como um grande catálogo, facilitando a busca e o uso dos dados publicados por diversos órgãos do governo, sendo considerado um enorme banco de dados do governo federal composto por dados de saúde suplementar, sistema de transporte, segurança pública, indicadores de educação, gastos governamentais, processo eleitoral, etc.

Está prevista a expansão desse portal para publicação e acesso aos dados das esferas estaduais e municipais. Além disso, esse projeto pretende seguir as normas estabelecidas pela Infraestrutura Nacional de Dados Abertos (INDA)[3], que consiste em um conjunto de padrões, tecnologias, procedimentos e mecanismos de controle necessários para atender às condições de disseminação e compartilhamento de dados e informações públicas, considerando os princípios do *Linked Data*.

O projeto internacional *Linking Open Data* (LOD), iniciado em 2007, consiste em um esforço comunitário, mantido pelo W3C para identificar fontes de dados publicadas sob licenças abertas, convertê-las para RDF usando os princípios de *Linked Data* e publicá-las na Web. Até setembro de 2011, este projeto havia publicado 295 conjuntos de dados compostos por mais de 31 bilhões de triplas RDF e 500 milhões de links RDF, englobando os mais variados domínios como: informações geográficas, censo, pessoas, empresas, comunidades *online*, publicações científicas, filmes, músicas, livros, além de outros (Bizer, C. *et al*.,2011).

A Figura 1.1 mostra um diagrama de nuvem com as fontes de dados publicadas pelo projeto LOD[4] e as interligações entre elas em setembro de 2011. Cada nó no diagrama representa um conjunto distinto de dados publicados como *Linked Data*. O tamanho de cada nó corresponde ao número de triplas de cada fonte de dados, sendo classificado como a seguir: 1) **muito grande** - possui mais de 1 bilhão de triplas; 2) **grande** - possui um intervalo entre 1 bilhão e 10 milhões de triplas; 3) **médio** - possui um intervalo entre 10 milhões e 500 mil triplas; 4) **pequeno** - possui um intervalo entre 500 mil e 10 mil triplas; 5) **muito pequeno** - possui menos de 10 mil triplas.

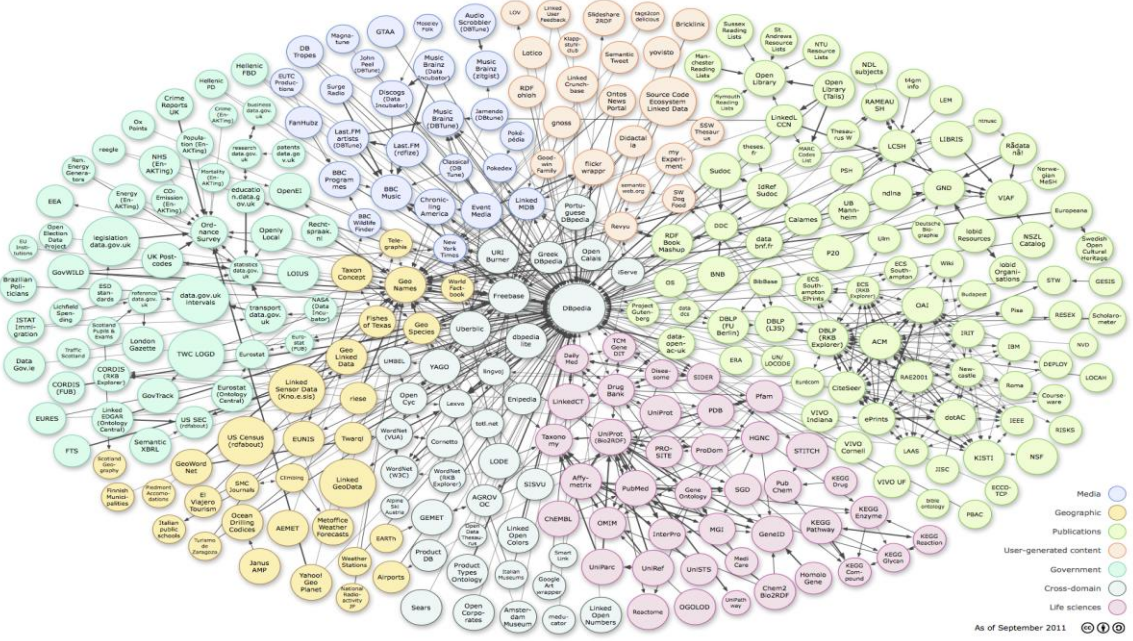

Figura 1.1. Diagrama de nuvem Linking Open Data

[3] http://wiki.gtinda.ibge.gov.br/

[4] http://lod-cloud.net/

Os arcos indicam a existência de pelo menos 50 links entre duas fontes. A origem de um arco indica a fonte que possui o link e a fonte referenciada é a fonte para a qual o arco está apontando. Arcos bidirecionais representam fontes que se referenciam mutuamente. A espessura da seta corresponde ao número de links.

Os conjuntos de dados são destacados com cores correspondentes a sete domínios, descritos a seguir: Mídia, Geográfico, Governo, Publicações, Domínios inter-relacionados, Ciências da vida e Conteúdos gerados por usuário. A Tabela 1 lista alguns dos conjuntos de dados disponíveis como parte da nuvem do *Linking Open Data*.

Tabela 1. Conjuntos de dados da nuvem Linking Open Data.

Conjunto de dados	Descrição	Quantidade de triplas RDF (aproximadamente)	Alguns conjuntos de dados conectados
Dbpedia	Informações a partir da Wikipédia	1 bilhão	Geonames, US Census, Freebase
Geonames	Dados geográficos	153 milhões	DBpedia, Jamendo, perfil do FOAF
Perfis do FOAF	Informações sobre pessoas	400 mil	SIOC, Flickr Exporter, Geonames
MusicBrainz	Infomações sobre música a partir da aplicação MusicBrainz	36 milhões	DBpedia, BBC Music, Disgos

Um caso típico de um grande conjunto de dados é a DBpedia que, essencialmente, faz com que o conteúdo da Wikipedia esteja disponível em RDF. A importância da DBpedia é que não só inclui dados da Wikipedia, mas também incorpora links para *outros* conjuntos de dados na Web, como por exemplo, para Geonames. Ao fornecer esses links adicionais (em termos de triplas RDF), as aplicações podem explorar o conhecimento extra (e possivelmente mais preciso) de outros conjuntos de dados e em virtude da integração de fatos de diversos conjuntos de dados.

A seguir, algumas aplicações que utilizam os princípios do *Linked Data* foram destacadas:

Portal SIOP[5] – é uma aplicação desenvolvida pela Secretaria de Orçamento Federal –SOF que utiliza os princípios do *Linked Data* para publicação de dados da Lei Orçamentária Anual (LOA) 2012, em prol da transparência pública. Os dados do orçamento em formato aberto, juntamente com o Manual de Referência, possibilitam à sociedade a elaboração de aplicativos para extrair, consultar, cruzar ou compartilhar, livremente, informações orçamentárias.

BBC Music[6] – é uma aplicação que provê informações relacionadas à música e ao entretenimento. Ela provê um único URI para entidades tais como artistas, trilha sonora, concertos,

[5] https://www.siop.planejamento.gov.br/siop/

[6] http://www.bbc.co.uk/music

etc. Essa aplicação utiliza dados provenientes de *MusicBrainz* e *Wikipedia* (*DBpedia*). Portanto, além de oferecer a funcionalidade de publicação dos dados, essa ferramenta possibilita a utilização de dados a partir de outras fontes. Por exemplo, essa aplicação pode obter a informação bibliográfica relacionada a um determinado artista a partir da *Wikipedia* e informações sobre novas músicas lançadas por um determinado artista a partir do *MusicBrainz.*

IBM Watson[7] – é um sistema de questionamento e resposta, capaz de processar, compreender e responder a consultas em linguagem natural. IBM Watson utiliza alguns pedaços de informação disponível como uma parte do *DBpedia*, com a finalidade de melhorar em 5% a precisão das respostas geradas pela ferramenta.

Faviki[8] – é uma ferramenta que permite ao usuário usar conceitos da *Wikipedia* como *tags* para descrever as entidades de interesse do usuário.

Portal British Museum[9] – provê informações sobre vários artefatos disponíveis em vários museus britânicos e informações relacionadas a eles. O portal obtém os dados a partir de diferentes conjuntos de dados LOD, tais como *DBpedia*, e fornece mais informações sobre os seus artefatos do que normalmente estaria disponível.

A Web de Dados cria inúmeras oportunidades para o desenvolvimento de novos tipos de aplicações e ferramentas. Muito esforço tem sido despendido pela comunidade para o desenvolvimento de navegadores, mecanismos de busca e outras ferramentas específicas. Além disso, muitas ferramentas para publicação de dados seguindo os princípios de *Linked Data* foram desenvolvidas e disponibilizadas, motivando a publicação de dados de forma aberta.

Diante desse cenário, este trabalho desenvolve uma ferramenta a fim de auxiliar o especialista do domínio no processo de gerenciamento de ontologias. Muitas ferramentas e técnicas têm sido desenvolvidas com o propósito de auxiliar o processo de elaboração de uma ontologia a partir de duas outras ontologias, utilizando técnicas de alinhamento de ontologias ou heurísticas. Entretanto, tais trabalhos não se preocuparam em auxiliar o especialista do domínio na elaboração de uma nova ontologia, que represente um entendimento correto sobre a semântica das fontes de dados envolvidas, visto que, para isso faz-se necessário considerar as restrições lógicas das ontologias originais e propagar essas restrições à nova ontologia. Além disso, as ferramentas que realizam o processo de mapeamento entre bases de dados relacionais e RDF, expresso a partir de uma linguagem de mapeamento, tal como, R2RML (Das, S. *et al.*, 2012), preocupam-se apenas em reutilizar vocabulários a partir das fontes de dados relacionais, como descrito em (Sahoo, S.S. *et al.*, 2009), sem considerar o reuso dessas restrições.

A estratégia proposta neste trabalho para gerenciamento de ontologias considera as ontologias como teorias lógicas compostas de vocabulários e restrições. Assim, foi definido um mecanismo que consiste em um conjunto de operações algébricas (projeção, união, interseção e diferença) sobre uma ou mais ontologias. Este mecanismo utiliza o conceito de grafo de restrições para a manipulação das ontologias, bem como, procedimentos de decisão baseados em regras lógicas e teoria dos grafos. A estratégia proposta neste trabalho para gerenciamento de ontologias considera as ontologias como teorias lógicas compostas de vocabulários e restrições. Assim, foi definido um mecanismo que consiste em um conjunto de operações algébricas (projeção, união, interseção e diferença) sobre uma ou mais ontologias. Este mecanismo utiliza o conceito de grafo de restrições para a manipulação das ontologias, bem como, procedimentos de decisão baseados em regras

[7] http://www.watson.ibm.com/index.shtml

[8] http://www.faviki.com/pages/welcome/

[9] http://www.britishmuseum.org/

lógicas e teoria dos grafos.

1.3. PROBLEMA

Nesta seção serão definidos os conceitos referentes ao problema de gerenciamento de ontologias visando à integração e à interoperabilidade entre aplicações que utilizam as tecnologias do *Linked Data* para publicação de dados.

Em particular, no contexto da Web de dados, o principal obstáculo à integração e à interoperabilidade entre as fontes de dados surge, em geral, a partir do fato dessas fontes serem descritas por ontologias criadas de forma independente e sem a preocupação de reutilizar as terminologias das ontologias existentes e consolidadas.

Outro aspecto importante a ser considerado como empecilho à integração consiste na existência dos conflitos semânticos que ocorrem quando um mesmo termo representa diferentes conceitos (Noy, N.F.,2004), ou seja, o símbolo que representa o termo é idêntico, mas representa dois conceitos com significado distintos. Por exemplo, o termo manga em fontes de dados do domínio hortifrutigranjeiro é diferente do mesmo termo manga em fontes de dados do domínio de estilismo e moda, visto que, esses domínios são disjuntos.

A fim de tratar os problemas descritos acima, este trabalho adota uma estratégia baseada em operações algébricas (união, diferença, interseção e projeção) envolvendo uma ou duas ontologias com a finalidade de viabilizar a integração e a interoperabilidade das fontes que seguem os princípios do *Linked Data*. Essas operações proporcionam a elaboração de novas ontologias e dos mapeamentos de equivalência entre as ontologias envolvidas, bem como, o gerenciamento de múltiplas ontologias considerando a semântica das ontologias envolvidas.

1.4. CONTRIBUIÇÕES

A principal contribuição deste trabalho é a implementação de uma ferramenta para auxiliar o especialista do domínio no gerenciamento de ontologias e a demonstração da sua aplicabilidade através de experimentos com ontologias derivadas de fontes de dados reais. Essa ferramenta denominada ***OntologyManagementTool*** permite realizar as seguintes operações algébricas utilizando duas ontologias: **união**, **interseção** e **diferença**. Além disso, ela permite obter um fragmento de uma determinada ontologia através da operação de **projeção**. O diferencial dessas operações algébricas consiste em considerar as restrições das ontologias utilizadas como operando. Portanto, a ontologia resultante dessas operações pode apresentar os conflitos semânticos, bem como, os mapeamentos entre os termos do vocabulário das ontologias envolvidas nas operações.

Uma particularidade sobre essa ontologia resultante das operações algébricas consiste na utilização de métodos baseados em "*bootstrapping"* (Jain, P. *et al.*, 2010) para identificar e considerar as restrições nessas operações algébricas, ou seja, consiste em métodos capazes de utilizar a informação contida nas próprias ontologias a fim de inferir novas informações.

Além disso, essa ferramenta oferece a possibilidade de detecção de inconsistências envolvendo termos do vocabulário de uma determinada ontologia, através da execução do procedimento de satisfatibilidade estrita. E ainda, permite que as inconsistências encontradas em uma determinada ontologia possam ser visualizadas de modo ilustrativo a partir do grafo de restrições.

Por fim, essa ferramenta possibilita ao especialista do domínio armazenar a ontologia resultante em um repositório de ontologias.

Durante os estudos e desenvolvimento do trabalho proposto foi obtido como resultado a

seguinte publicação:

- Casanova, M.A., Macêdo, J.A.F, Sacramento, E.R., Pinheiro, A.M.A, Vidal, V.M.P., Breitman, K.K. e Furtado, A.L.. Operations over Lightweight Ontologies. OTM 2012, Part II, LNCS 7566, pp. 646–663, 2012.

1.5. ORGANIZAÇÃO DO TRABALHO

Esse trabalho está dividido em sete capítulos, incluindo a introdução:

- O capítulo 2 aborda a fundamentação teórica, onde se encontram os principais conhecimentos e os fundamentos necessários para o desenvolvimento deste trabalho e inclui uma visão geral sobre a Web de dados e conceitos básicos sobre Ontologias.

- O capítulo 3 descreve as funcionalidades e algumas ferramentas relacionadas ao gerenciamento de ontologias, bem como, a análise comparativa entre o trabalho proposto e outros trabalhos encontrados na literatura.

- O capítulo 4 apresenta uma estratégia para a definição do grafo de restrições e suas principais características. Além disso, apresenta o algoritmo de satisfatibilidade estrita, bem como, exemplos relacionados tanto a definição do grafo de restrições quanto à execução do algoritmo de satisfatibilidade estrita.

- O capítulo 5 define as operações algébricas (união, interseção, diferença e projeção) sobre as ontologias, mostra os algoritmos desenvolvidos para a implementação dessas operações e, por fim, demonstra a aplicabilidade da execução de cada uma dessas operações utilizando ontologias que descrevem fontes de dados reais.

- O capítulo 6 apresenta o protótipo desenvolvido, denominado de ***OntologyManagementTool***, mostra uma visão geral das suas funcionalidades e apresenta as conclusões sobre os experimentos realizados, bem como, a ontologia resultante das operações utilizadas para a realização dos experimentos.

- O capítulo 7 expõe a conclusão do trabalho, apresentando contribuições, as dificuldades encontradas, além das perspectivas de trabalhos futuros e melhorias.

CAPÍTULO 2

Fundamentação Teórica

Neste capítulo serão abordados os conceitos fundamentais que serviram de alicerce para o desenvolvimento do trabalho proposto. A seção 2.1 apresenta as principais tecnologias de *Linked Data*. A seção 2.2 expõe os esquemas conceituais em Lógica Descritiva, uma breve descrição sobre a família de linguagens atributivas e os esquemas conceituais elementares com hierarquias de relacionamentos binários. Por fim, a seção 2.3 ilustra essas definições com exemplo de ontologias do domínio de Música utilizadas pela aplicação *MusicBrainz*.

2.1. TECNOLOGIAS DE LINKED DATA

Linked Data é um conjunto de melhores práticas para publicação e consumo de dados estruturados na Web, permitindo estabelecer ligações entre itens de diferentes conjuntos de dados para formar um único espaço de dados global (Heath, T. e Bizer, C., 2011).

O objetivo do *Linked Data* é permitir que as pessoas compartilhem dados estruturados na Web, tão facilmente quanto elas atualmente compartilham os seus documentos. Os dados publicados na Web de acordo com essas melhores práticas podem ser processados por máquinas e estar ligados a outras fontes de dados.

De acordo com (Jacobs e Walsh, 2004), o documento Web é construído sobre um pequeno conjunto de padrões simples: *Uniform Resource Identifiers* (URIs), como o mecanismo de identificação único; *Hypertext Transfer Protocol* (HTTP), como o mecanismo de acesso universal e; *Hypertext Markup Language* (HTML), como o formato de conteúdo largamente usado. Além disso, a Web é construída sobre a ideia de hiperlinks configuráveis entre documentos Web que devem residir em diferentes servidores Web.

As melhores práticas relacionadas à área de *Linked Data* foram inicialmente propostas por (Berners-Lee, 2007) e ficaram conhecidas como os princípios de *Linked Data* que são enumerados a seguir:

1. Usar URIs como nomes para coisas.
2. Usar URIs HTTP para que as pessoas possam procurar esses nomes.
3. Quando alguém procurar um URI, prover informação útil usando os padrões (RDF, SPARQL).
4. Incluir declarações RDF que ligam os URIs a outros, de forma a permitir a descoberta de coisas relacionadas.

Esses princípios fornecem a base para a publicação e interligação de dados estruturados na Web.

Os padrões abertos adotados em *Linked Data* são amplamente difundidos e serão detalhados, a seguir:

URI – *Uniform Resource Identifier*

Um Identificador Uniforme de Recursos (URI –*Uniform Resource Identifier*) é uma sequência compacta de caracteres que identifica um recurso físico ou abstrato (Berners-Lee T. *et al*., 2005).

Assim, um URI pode ser usado para identificar coisas na Web, mas como a arquitetura da Web é baseada em HTTP, geralmente, URIs identificam páginas Web ao invés de identificar coisas. Com a finalidade de diferenciar as URIs que representam páginas da Web das URIs que representam coisas presentes na Web, foram definidas duas formas de URI: *hash* URI (Berners-Lee,

T., 1994) e *slash* URI (Berners-Lee, T., 2007).

Geralmente, o *hash* URI é usado para identificar coisas, sendo composto por: (documento) # (termo introduzido no documento), por exemplo, http://www.w3.org/1999/02/22-rdf-syntax-ns#type. Por outro lado, o *slash* URI define um URL conforme a seguir: http://purl.org/dc/elements/1.1/title.

Ontologia

Gruber (Gruber *et al*,1993) formula uma definição de ontologias amplamente adotada na literatura de Web Semântica: "Uma ontologia é uma especificação explícita e formal de uma conceitualização compartilhada".

De acordo com (Breitman *et al*, 2006), na definição de Gruber, a conceitualização expressa um modelo abstrato; o termo explícito significa que os elementos devem ser claramente definidos e por fim, formal indica que a especificação deve ser processável por máquina.

Portanto, é possível resumir que, na visão de Gruber, uma ontologia é a representação do conhecimento do domínio, onde um conjunto de objetos e seus relacionamentos são descritos por vocabulários.

As ontologias podem ser representadas textualmente por várias linguagens baseadas em XML, como RDF (*Resource Description Framework*) (Manola, F. e Miller, E., 2004), RDF(S) (*RDF Schema*) (Manola, F. e Miller, E., 2004) e OWL (*Web Ontology Language*) (Antoniou, G. e van Harmelen, F., 2003).

Linguagem RDF – *Resource Description Framework*

A linguagem RDF é o *framework* básico da Web Semântica, que descreve um modelo de dados em que as ontologias são baseadas.

Segundo (Manola, F. e Miller, E., 2004), RDF é uma linguagem para representar informações sobre recursos da Web. Particularmente, representa metadados sobre esses recursos, tais como título, autor e data de modificação de uma página da Web, ou seja, a representação em RDF se baseia na associação entre descrições e recursos por meios de declarações que estabelecem um valor para uma propriedade associada a um recurso. Neste caso, o conceito "recurso da Web" é bem amplo, podendo ser usado para representar informações sobre as coisas que podem ser identificadas na Web, mesmo quando elas não podem ser diretamente acessadas.

RDF é baseado na ideia de que coisas são identificadas usando identificadores Web, denominados URIs (*Uniform Resource Identifiers*) e recursos de descrição em termos de propriedades simples e valores de propriedade.

Um recurso pode ser qualquer coisa que alguém deseja descrever. Deste modo, a forma de declaração fundamental para descrever usando essa linguagem está na forma de uma tripla: Sujeito – Predicado – Objeto, onde Sujeito refere-se a um recurso (denotado por um URI); Predicado representa uma propriedade do recurso e Objeto indica o valor dessa propriedade, que pode ser um literal (representado por um inteiro ou uma string, por exemplo), ou ainda outro recurso. Por exemplo, temos a seguinte declaração:

http://www.inf.puc-rio.br/ professor Marco A. Casanova

Assim é possível escrever uma tripla RDF com os seguintes valores:

- Sujeito: URL http://www.inf.puc-rio.br/

- Predicado: “professor”

- Objeto: “Marco A. Casanova”

Para uma padronização de uso do RDF, foram criados os esquemas RDF (RDF *Schema*), que fornecem tipos básicos para a criação de esquemas. As primitivas usadas para modelar esquemas RDF incluem construtores para classes, subclasses, propriedades e sub propriedades.

Vocabulários padronizados em RDF

Os vocabulários fornecem termos específicos do domínio para descrever classes de coisas do mundo, e como elas se relacionam umas com as outras. Dependendo do poder de expressividade, os vocabulários podem ser classificados desde taxonomias até como ontologias (McGuinness, D.L., 2002).

De acordo com (Heath, T. e Bizer, C., 2011), é considerado uma boa prática reusar termos bem difundidos de vocabulário RDF sempre que possível. O reuso de termos existentes é altamente desejado, pois maximiza a interoperabilidade entre as aplicações.

A seguinte lista apresenta um conjunto de ontologias dos mais diversos domínios largamente usado pela comunidade. Para assegurar a interoperabilidade entre sistemas que adotam os princípios do “*Linked Data*”, é recomendado que esses vocabulários sejam reusados sempre que possível.

- O vocabulário **Dublin Core Metadata Initiative** (DCMI)[10] visa descrever atributos de metadados gerais, tais como: criador, data, assunto, descrição e outros.
- O vocabulário **Friend-of-Friend** (FOAF)[11] define termos para descrever pessoas, suas atividades e seus relacionamentos com outras pessoas, objetos e páginas da Web (Brickley,D., Miller, L., 2010).
- O vocabulário de **Descrição de Projetos** (DOAP)[12] define termos para descrever projetos de *software*, particularmente, aqueles que são *Open Source*.
- **Ontologia de Música**[13] define termos para descrever vários aspectos relacionados à música, como artistas, álbuns, faixas musicais e arranjos.

Linguagem OWL

A linguagem OWL, recomendada pela W3C, foi projetada com a finalidade de adicionar um conjunto de vocabulários que representa um formalismo lógico não suportado pelo XML, RDF e RDF *Schema*. OWL é especificada em RDF e possui, adicionalmente, construtores específicos para termos básicos de uma ontologia (classes, instâncias, propriedades e restrições de cardinalidade), bem como, construtores para a formalização de relacionamentos mais complexos, como por exemplo, equivalência, união e interseção.

[10] https://github.com/edumbill/doap/

[11] http://www.foaf-project.org/

[12] http://semanticweb.org/wiki/DOAP

[13] http://musicontology.com/

As ontologias podem ser classificadas em três classes de sublinguagem (Antoniou, G. e van Harmelen, F. , 2003), de acordo com a expressividade do raciocínio:

- OWL – *Lite* suporta a criação de hierarquias de classificação e algumas restrições simples. Permite suporte a restrições de cardinalidade com valores de 0 ou 1. O objetivo desta sub linguagem é oferecer suporte à migração de taxonomias para o formato de ontologias.

- OWL DL (DL é acrônimo para Lógica Descritiva) suporta todos os construtores oferecidos pela linguagem OWL, mas oferece a máxima expressividade, a completude computacional e a decidibilidade da Lógica Descritiva. Essa sub linguagem possui construtores mais complexos que OWL *Lite*, possibilitando a modelagem de classes por meio dos operadores de união, interseção e complemento, além de representar restrições de disjunções entre classes.

- OWL *Full* permite a máxima expressividade e a liberdade sintática do RDF com nenhuma garantia computacional. Por exemplo, em OWL *Full* uma classe pode ser tratada simultaneamente com uma coleção de indivíduos e como um indivíduo. Portanto, é pouco provável que qualquer *software* será capaz de apoiar o raciocínio completo para todos os recursos do OWL *Full*.

A escolha do formalismo para representar a ontologia é um desafio crucial para que as aplicações de *Linked Data* sejam bem sucedidas nos seus propósitos. Esse formalismo deve utilizar uma família de esquema suficientemente expressiva para definir esquemas do mundo real e ainda ser computacionalmente tratável.

Neste trabalho, foi adotada uma família de esquemas denominado *Extralite* com hierarquias de relacionamentos binários, visto que, esses esquemas são suficientemente expressivos para capturar os construtores das principais linguagens de modelagem de dados, como OWL, UML e ER. Além disso, possibilitam a utilização de procedimentos de decisão que exploram a estrutura dos conjuntos de restrições (Borgida, A. e Brachman, R.J., 2003) (Casanova *et al.*, 2011).

A seguir, será apresentado o formalismo adotado neste trabalho.

2.2. ESQUEMAS CONCEITUAIS EM LÓGICA DESCRITIVA

As ontologias podem ser consideradas como esquemas conceituais, sendo usadas para minimizar o problema da heterogeneidade semântica dos dados encontrados na Web. Assim, uma ontologia pode ser utilizada, dentre outros propósitos, para o suporte à interoperabilidade de fontes de dados distribuídas e heterogêneas.

A família de esquemas denominado *Extralite* com hierarquias de relacionamentos binários pode ser comparada à família DL-Lite (Artale, A. et al, 2009), considerada como um subconjunto $DL\text{-}Lite_{core}^{\mathcal{HN}}$ com disjunções de papel.

Utilizando os termos da linguagem OWL para definir essas famílias, consideramos que elas suportam classes, propriedades de tipo de dado (*datatype properties*) e propriedades de tipo de objeto (*object properties*); restrições de cardinalidade mínima (*minCardinalities*) e de cardinalidade máxima (*maxCardinalities*); restrições de propriedade funcional inversa (*InverseFunctionalProperties*), que capturam chaves simples; e restrições de subconjunto de classes e classes disjuntas. Em particular, um esquema *Extralite* com hierarquias de relacionamentos binários também suporta restrições de subconjunto e de disjunção definidas por propriedades de tipo de dados e de tipo de objeto (formalizadas como papéis atômicos em Lógica Descritiva).

A seguir, vamos formalizar os termos apresentados anteriormente.

Uma família de linguagem atributiva (Baader, F.; Nutt, W., 2003) é definida como a seguir.

Uma *linguagem* $\mathcal{L}$ é caracterizada por um *alfabeto* $\mathcal{A}$, consistindo de um conjunto de *conceitos atômicos*; um conjunto de *papéis atômicos*; o *conceito universal* e o *conceito vazio*

denotados por $\top$ e $\bot$, respectivamente; por fim, o *papel universal* e o *papel vazio*, também denotados por $\top$ e $\bot$, respectivamente.

O conjunto de descrições de papel de $\mathcal{L}$ é indutivamente definido como a seguir:

- Um papel atômico e os papéis universal e vazio são descrições de papel.
- Se *p* é uma descrição de papel, então as seguintes expressões são descrições de papel:
  - ✓ p^- (a inversa de p)
  - ✓ $\neg p$ (a negação de p)

O conjunto de descrições de conceito de $\mathcal{L}$ é indutivamente definido como a seguir:

- Um conceito atômico, o conceito universal e o vazio são descrições de conceito.
- Se *e* é uma descrição de conceito, *p* é uma descrição de papel, e *n* é um inteiro positivo, então as seguintes expressões são descrições de conceito:

  $\neg e$ (negação)

  $\exists p$ (quantificação existencial restrita)

  $(\leq n\ p)$ (restrição de cardinalidade máxima)

  $(\geq n\ p)$ (restrição de cardinalidade mínima)

Uma *interpretação* **s** para os símbolos do alfabeto $\mathcal{A}$ consiste de um conjunto não vazio $\Delta^{\boldsymbol{s}}$, o domínio de **s**, cujos elementos são denominados *indivíduos*, e uma *função de interpretação* também denominada **s**, onde:

- $\boldsymbol{s}(\top) = \Delta^{\boldsymbol{s}}$, se $\top$ denota o conceito universal
- $\boldsymbol{s}(\top) = \Delta^s \times \Delta^s$, se $\top$ denota o papel universal
- $\boldsymbol{s}(\bot) = \varnothing$, se $\bot$ denota um conceito vazio ou papel vazio
- $\boldsymbol{s}(A) \subseteq \Delta^{\boldsymbol{s}}$, para cada conceito atômico *A* de $\mathcal{A}$
- $\boldsymbol{s}(P) \subseteq \Delta^s \times \Delta^s$ para cada papel atômico *P* de $\mathcal{A}$

A função **s** é estendida para descrições de conceitos e papeis de $\mathcal{L}$ como a seguir (onde *e* é uma descrição de conceito e *p* é uma descrição de papel):

- $\boldsymbol{s}(p^-) = \boldsymbol{s}(p)^-$ (o inverso de *s(p)*)
- $\boldsymbol{s}(\neg p) = \Delta^s \times \Delta^s - \boldsymbol{s}(p)$ (o complemento de *s(p)* com relação a $\Delta^s \times \Delta^s$)
- $\boldsymbol{s}(\neg e) = \Delta^{\boldsymbol{s}} - \boldsymbol{s}(e)$ (o complemento de *s(e)* com relação a Δ^s)
- $\boldsymbol{s}(\exists p) = \{I \in \Delta^{\boldsymbol{s}} / (\exists J \in \Delta^{\boldsymbol{s}})((I,J) \in \boldsymbol{s}(p)\}$ (o conjunto de indivíduos tal que *s(p)* refere-se a algum indivíduo)
- $\boldsymbol{s}(\geq n\ p) = \{I \in \Delta^{\boldsymbol{s}} / |\{J \in \Delta^{\boldsymbol{s}} / (I,J) \in \boldsymbol{s}(p)\}| \geq n\}$ (o conjunto de indivíduos tal que *s(p)* refere-se a no mínimo n indivíduos distintos)
- $\boldsymbol{s}(\leq n\ p) = \{I \in \Delta^{\boldsymbol{s}} / |\{J \in \Delta^{\boldsymbol{s}} / (I,J) \in \boldsymbol{s}(p)\}| \leq n\}$ (o conjunto de indivíduos tal que *s(p)* relaciona-se a no máximo n indivíduos distintos)

Uma *fórmula* de $\mathcal{L}$ é uma expressão da forma $\boldsymbol{u} \sqsubseteq \boldsymbol{v}$, denominada inclusão, ou da forma $\boldsymbol{u} \mid \boldsymbol{v}$, denominada disjunção, ou da forma $\boldsymbol{u} \equiv \boldsymbol{v}$, denominada equivalência, onde ambas *u* e *v* são

descrições de conceitos ou ambas são descrições de papel de $\mathcal{L}$. Outras denominações para as fórmulas de inclusão são definidas a seguir: $\boldsymbol{u \sqsubseteq v}$ é uma *inclusão de conceito*, se e somente se, ambas *u* e *v* são descrições de conceitos, e $\boldsymbol{u \sqsubseteq v}$ é *inclusão de papel*, se e somente se, ambas *u* e *v* são descrições de papel. A fórmula de disjunção é logicamente equivalente a uma inclusão da forma $u \sqsubseteq \neg v$, que pode ser representada por $u \mid v$.

Uma interpretação de **s** para $\mathcal{L}$ satisfaz $u \sqsubseteq v$, se e somente se, $\boldsymbol{s(u) \subseteq s(v)}$; **s** satisfaz $u \mid v$, se e somente se, $\boldsymbol{s(u) \cap s(v)} = \varnothing$, ou seja, são conjuntos disjuntos.

2.3. ESQUEMAS CONCEITUAIS EXTRALITE COM HIERARQUIAS DE RELACIONAMENTOS BINÁRIOS

Neste trabalho foi adotado o esquema conceitual *Extralite* com hierarquia de relacionamento binário definido em (Casanova *et al.*, 2011). Esse esquema foi definido a partir do esquema conceitual *Extralite* que corresponde parcialmente a OWL *Lite* (Harmelen F. v., McGuinness D. L., 2004). Os esquemas *Extralite* suportam classes, propriedades, e admitem restrições de domínio e imagem, restrições de subconjunto e disjunção, restrições de cardinalidade mínima e restrições de cardinalidade máxima, todas com o significado usual.

Formalmente, um esquema conceitual *Extralite* é um par $S = (\mathcal{A}, \Sigma)$, tal que:

- $\mathcal{A}$ é um alfabeto, chamado de vocabulário de *S*, cujos conceitos e papéis atômicos são chamados de classes e propriedades de *S*, respectivamente.
- Σ é um conjunto de fórmulas, chamado de restrições de *S*, que necessariamente possuem uma das formas a seguir, onde C e D são classes e P é uma propriedade de *S*:
  - ✓ Restrição de domínio:

    $\exists P \sqsubseteq D$, a propriedade P possui domínio D

  - ✓ Restrição de imagem:

    $\exists P^{-} \sqsubseteq R$, a propriedade P possui imagem R

  - ✓ Restrição de cardinalidade mínima

    $C \sqsubseteq (\geq k\ P)$ ou $C \sqsubseteq (\geq k\ P^{-})$, a propriedade *P*, ou sua inversa P^{-} mapeia cada indivíduo em C para no mínimo k indivíduos distintos.

  - ✓ Restrição de cardinalidade máxima

    $C \sqsubseteq (\leq k\ P)$ ou $C \sqsubseteq (\leq k\ P^{-})$, a propriedade P, ou sua inversa P^{-} mapeia cada indivíduo em C para no máximo k indivíduos distintos.

  - ✓ Restrição de subconjunto

    $C \sqsubseteq D$, a classe *C* é uma subclasse de *D*.

  - ✓ Restrição de disjunção

    $C \mid D$, a classe *C* é disjunta de *D*.

✓ Restrição de subconjunto (hierarquia) de propriedades

$P \sqsubseteq Q$, a propriedade *P* é uma sub propriedade de *Q*.

✓ Restrição de disjunção de propriedades

$P \mid Q$, a propriedade *P* é disjunta de *Q*.

✓ Também são admitidas restrições de uma das formas a seguir:

- $C \sqsubseteq \bot$, a classe *C* é vazia.
- $P \sqsubseteq \bot$, a propriedade *P* é vazia.
- $\exists P \sqsubseteq \bot$ e $\exists P^{-} \sqsubseteq R$, a propriedade P é sempre vazia, ou seja, P possui domínio vazio ou possui imagem vazia.
- $(\geq m\ p) \sqsubseteq \bot$, $(\geq m\ p) \sqsubseteq \neg C$, $(\geq m\ p) \sqsubseteq (\geq n\ q)$, $(\geq m\ p) \sqsubseteq \neg(\geq n\ q)$, onde C é um conceito atômico, p e q são ambos papéis atômicos ou ambos são inversas de papéis atômicos, e m e n são inteiros positivos.

Diz-se que uma interpretação ***s*** para $\mathcal{A}$ é um estado consistente de *S*, se e somente se, **s** satisfaz todas as restrições em Σ.

Considerando as definições apresentadas, tomamos a liberdade de utilizar ao longo do texto, a seguintes terminologias: classe, propriedade e vocabulário serão sinônimos de conceito atômico, papel atômico e alfabeto, respectivamente.

É importante evidenciar que a formalização aqui apresentada não faz distinção entre propriedade do tipo de objeto (*object property*) e do tipo de dado (*datatype property*), definidas na terminologia OWL. A distinção ficará visível apenas nos exemplos, onde a imagem de uma propriedade do tipo de objeto será uma classe definida no esquema, enquanto que a imagem de uma propriedade do tipo de dado será um tipo XML *Schema*, ou seja, um conjunto de valor de tipo de dado ou literal. Uma vez que a noção de domínio de uma interpretação não separa indivíduos que denotam elementos de classe dos indivíduos que correspondem a valores de tipo de dados, o desenvolvimento formal não tem como capturar esta distinção.

2.4. NORMALIZAÇÃO DOS ESQUEMAS CONCEITUAIS

As ontologias elaboradas para representar domínios têm sido amplamente reutilizadas e podem constituir um relevante papel de fornecedoras de conhecimento para a inferência dinâmica de informações. Por outro lado, muitas vezes, essas ontologias são construídas sem a preocupação de utilizar um formalismo que seja computacionalmente tratável.

Desse modo, é imprescindível normalizar as restrições dessas ontologias, de modo a permitir que elas sejam manipuladas conforme o formalismo adotado neste trabalho, baseado em conceitos da família DL-*Lite*. Segundo (Casanova *et al.*, 2011), as restrições definidas para um esquema conceitual *Extralite* com hierarquia de relacionamentos binários são manipuladas em tempo polinomial. Além disso, destaca que essa complexidade polinomial, também se aplica para um esquema conceitual *Extralite* sem hierarquia de relacionamentos binários.

A obtenção dessa complexidade só foi possível devido à particularidade do tratamento das restrições presentes na família de DL-Lite. Desse modo, destaca-se que a restrição de cardinalidade

máxima é tratada como a forma negada da restrição de cardinalidade mínima e as restrições de inclusão e disjunção de conceitos, e relacionamentos binários são formuladas de tal modo que, as descrições negadas ocorrem apenas no lado direito das inclusões (Casanova *et al.*, 2011).

Apesar de essas particularidades restringirem a expressividade, elas permitem o uso de uma nova abordagem para serviços dedutivos que podem ser executados em tempo polinomial, sem destruir a estrutura das restrições de um determinado esquema.

Portanto, as restrições descritas a seguir **não** são consideradas neste formalismo:

- $\neg C \sqsubseteq D$ (conceito atômico negado no lado esquerdo da inclusão de conceito)
- $\neg C \mid D$ (conceito atômico negado usado na disjunção de conceito)
- $\neg C \sqsubseteq (\leq k\ P)$ (conceito atômico negado no lado esquerdo da inclusão de conceito)
- $\neg P \sqsubseteq Q$ (papel atômico negado no lado esquerdo da inclusão de papel)
- $P^{-} \sqsubseteq Q$ (papel atômico inverso usado no lado esquerdo da inclusão de papel)
- $C \sqsubseteq (\leq k \neg P)$ (papel atômico negado na restrição de cardinalidade máxima)

A Tabela 2 apresenta os tipos de restrições, bem como, a relação entre a formalização das restrições e a sua normalização. Nessa tabela, as restrições da coluna "**Normalizada**" são denominadas de forma normal das restrições descritas na coluna "**Formalização**".

Tabela 2. Conjunto de Restrições *Extralite*.			
Tipo de Restrição	**Formalização**	**Normalizada**	**Significado**
Restrição de Domínio	$\exists P \sqsubseteq D$	$(\geq 1\ P) \sqsubseteq D$	Propriedade *P* tem classe *D* como domínio, ou seja, se *(a,b)* é um par em *P*,então *a* é um indivíduo em *D*.
Restrição de Imagem	$\exists P^{-} \sqsubseteq R$	$(\geq 1\ P^{-}) \sqsubseteq R$	Propriedade *P* tem classe *R* como imagem, ou seja, se *(a,b)* é um par em *P*,então *a* é um indivíduo em *R*.
Restrição de cardinalidade mínima	$C \sqsubseteq (\geq k\ P)$ ou $C \sqsubseteq (\geq k\ P^{-})$		Propriedade *P* ou sua inversa P^{-} mapeia cada indivíduo da classe *C* em no mínimo *k* indivíduos distintos.
Restrição de cardinalidade máxima	$C \sqsubseteq (\leq k\ P)$ ou $C \sqsubseteq (\leq k\ P^{-})$	$C \sqsubseteq \neg(\geq k+1\ P)$ ou $C \sqsubseteq \neg(\geq k+1\ P^{-})$	Propriedade *P* ou sua inversa P^{-} mapeia cada indivíduo da classe *C* em no máximo *k* indivíduos distintos.
Restrição de Subconjunto (Restrição de subclasses)	$E \sqsubseteq F$		Cada indivíduo em *E* está também em *F*, ou seja, classe *E* é um subconjunto da classe *F*.
Restrição de Disjunção	$E \mid F$	$E \sqsubseteq \neg F$	Nenhum indivíduo está em ambas as classes *E* e *F*, ou seja, as classes *E* e *F* são disjuntas.
Restrição de hierarquia de propriedades (Restrição de subpropriedades)	$P \sqsubseteq Q$		Propriedade Q possui o domínio *D* e a imagem R, ou seja, se *(a,b)* é um par em *P*, então *a* é um indivíduo em D e *b* é um indivíduo de R. Cada indivíduo do domínio de Q está em D e cada indivíduo da imagem de Q está em R.
Restrição de disjunção de propriedades	$P \mid Q$	$P \sqsubseteq \neg Q$	Propriedade *P* possui o domínio *D* e a imagem R, ou seja, se *(a,b)* é um par em *P*,então *a* é um indivíduo em *D* e b é um indivíduo de R. Nenhum indivíduo de D e R é um par em Q.

É importante observar que a restrição formalizada e sua normalização são tautologicamente equivalentes. A forma normal de uma restrição evita o uso de quantificador existencial e restrições de cardinalidade máxima; descrições negadas ocorrem apenas no lado direito da forma normal e relacionamento binário inverso não ocorre em restrições de subconjunto de relacionamentos binários e nem em restrições de disjunção de relacionamentos binários (Casanova *et al.*, 2011) .

2.5. EXEMPLO

Para exemplificar os conceitos definidos, este trabalho utiliza um trecho da ontologia de música utilizada pela aplicação *MusicBrainz*. Esta ontologia provê conceitos e propriedades para descrever artistas, álbuns, trilhas sonoras, etc.

A Figura 2.1 mostra o diagrama ER que representa um trecho do esquema *MusicBrainz*. A

Figura 2.2 apresenta a formalização das restrições, já normalizadas. A primeira coluna mostra as restrições de domínio e imagem; a segunda coluna descreve as restrições de cardinalidade; e por fim, a terceira coluna apresenta as restrições de subconjunto e de disjunção.

A primeira coluna da Figura 2.2 indica que:

- *shortName* é um papel atômico modelando um atributo de *Artist* com a imagem *String*.
- *hasTrack* é um papel atômico modelando um relacionamento binário a partir de *Artist* para *Track.*
- *releaseStatus* é um papel atômico modelando um relacionamento binário a partir de *Album* para *Type*.

A segunda coluna da Figura 2.2 apresenta as restrições de cardinalidade da ontologia *MusicBrainz*:

- *shortName* tem restrições de cardinalidade máxima e mínima, ambas com o valor 1 no domínio de *Artist*.
- *hasTrack* tem restrições de cardinalidade máxima e mínima, ambas com o valor 1 no domínio de *Artist*.
- *hasTrack*$^-$ tem restrições de cardinalidade máxima e mínima não limitada, ambas com o valor 0 no domínio de *Track*, que não precisa ser explicitamente declarada.
- *releaseStatus* tem restrições de cardinalidade máxima e mínima, ambas com o valor 1 no domínio de *Album*.
- *releaseStatus*$^-$ tem restrições de cardinalidade máxima e mínima não limitada, ambas com o valor 0 no domínio de *Type*, que não precisa ser explicitamente declarada.

A terceira coluna da Figura 2.2 apresenta as restrições de subconjuntos normalizadas:

- *Artist* e *Album* são disjuntos, portanto *Artist* é subconjunto de não *Album*.
- *Type* e *Track* são disjuntos, portanto *Track* é subconjunto de não *Type*.

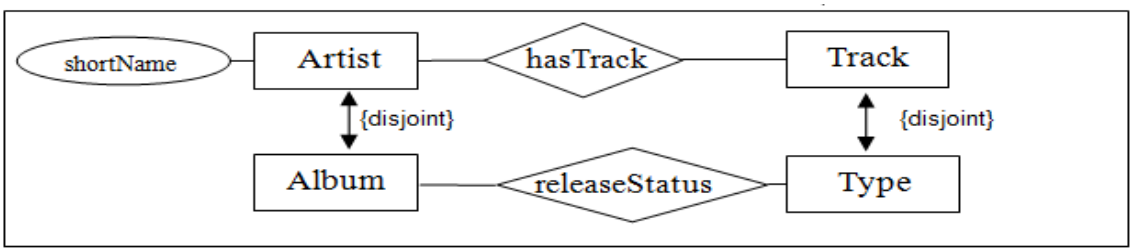

Figura 2.1. Diagrama ER da ontologia MusicBrainz (sem cardinalidades).

CAPÍTULO 3

∃hasTrack ⊑ Artist ∃hasTrack$^-$ ⊑ Track ∃releaseStatus ⊑ Album ∃releaseStatus$^-$ ⊑ Type ∃shortName ⊑ Artist ∃shortName$^-$ ⊑ String	Artist⊑(≤1hasTrack) Artist ⊑(≥1hasTrack) Album⊑(≤1releaseStatus) Album⊑(≥1releaseStatus) Artist ⊑(≤1shortName) Artist ⊑(≥1shortName)	Artist ⊑ ¬Album Track ⊑ ¬Type

Figura 2.2. Definição formal das restrições normalizadas da ontologia MusicBrainz.

Trabalhos Relacionados

3.1. INTRODUÇÃO

O gerenciamento de ontologias é imprescindível para o desenvolvimento de aplicações que utilizam as tecnologias de *Linked Data*. Ele pode ser definido como um conjunto de métodos e técnicas necessários para manipular eficientemente múltiplas ontologias, derivadas de fontes de dados heterogêneas e definidas para atender às mais diversificadas finalidades (Peter Haase *et al*, 2003).

Apesar dos diversos trabalhos existentes na área de gerenciamento de ontologias, neste capítulo, serão abordados trabalhos relacionados às funcionalidades de reuso, versionamento e evolução de ontologias, bem como, integração de dados envolvendo ontologias, uma vez que essas áreas são as principais beneficiadas com o desenvolvimento de processos e ferramentas para o gerenciamento de ontologias.

Este capítulo está organizado da seguinte forma. A seção 3.2 apresenta os trabalhos relacionados com o reuso de ontologias. Em seguida, a seção 3.3 expõe os trabalhos e abordagens desenvolvidas sobre versionamento de ontologias. A seção 3.4 aborda um processo para lidar com a evolução de ontologias. A seção 3.5 apresenta os trabalhos relacionados com integração de dados envolvendo ontologias e apresenta um resumo comparativo entre o mecanismo proposto e os trabalhos elencados ao longo da seção. Por fim, a seção 3.6 expõe as conclusões e um resumo comparativo entre o mecanismo proposto e os trabalhos elencados ao longo do capítulo.

3.2. REUSO DE ONTOLOGIAS

O reuso de ontologias pode ser definido como o processo em que uma ontologia de referência, ou seja, uma ontologia largamente utilizada, consolidada e testada pelo especialista do domínio, é usada como entrada para geração de uma nova ontologia (Bizer, C. *et al.*, 2009).

O processo de reuso de ontologias pode adotar duas abordagens: (1) *reuso sintático*, que consiste em copiar um conjunto de termos do vocabulário da ontologia existente em outra ontologia; e (2) *reuso semântico*, que se preocupa em aplicar as restrições das ontologias existentes aos termos da nova ontologia.

A abordagem do reuso sintático é tratada de forma simples por diversas ferramentas de gerenciamento de ontologias como, por exemplo, Protégé (Gennari J. H. *et al.*, 2003). Tais ferramentas apenas importam as ontologias OWL, utilizando a declaração <owl:import> da linguagem OWL. Todavia, essa abordagem oferece diversas desvantagens:

(1) Não oferece ao especialista do domínio um mecanismo de reuso que utilize apenas parte da ontologia original;

(2) Ao importar toda a ontologia, a inconsistência pode ser facilmente alcançada pela importação dos conflitos axiomáticos obtidos a partir da união dessas ontologias;

(3) Caso seja necessário importar apenas um subconjunto dos axiomas, o especialista do domínio deve executar o processo oneroso de identificar cuidadosamente qual o conjunto de axiomas codifica o conhecimento que ele quer importar. Por exemplo, suponha que uma ontologia contenha as restrições $A \sqsubseteq B$ e $B \sqsubseteq C$; caso o especialista do domínio queria importar somente $A \sqsubseteq C$ sem considerar B, é necessário que os axiomas da ontologia sejam analisados detalhadamente, o que nem sempre é um processo trivial, pois as ontologias podem ter muitos conceitos e restrições.

Em particular, o reuso semântico atende a algumas desvantagens do reuso sintático elencados acima, tais como, identificação dos conflitos axiomáticos a partir da união dessas ontologias, bem como, a análise das restrições inclusão da ontologia original e a sua aplicação na nova ontologia.

O reuso é extremamente recomendado no processo de desenvolvimento de novas ontologias, visto que, oferece diversas vantagens, tais como: reduz a duplicação de esforços; compartilha a semântica comum, devido ao uso de terminologias derivadas das ontologias de referência e reduz os custos e a complexidade de criar uma nova ontologia, em detrimento da criação a partir do "zero".

Além disso, o reuso contribui para a melhoria do desempenho de processamento em aplicações específicas, que necessitam de apenas parte das ontologias, mas que, por falta de alternativas disponíveis, os desenvolvedores utilizam-nas em sua totalidade. Frequentemente, o tamanho e a complexidade das ontologias disponíveis excedem as necessidades das aplicações – em geral, as aplicações necessitam apenas de parte do domínio descrito por uma ontologia. Apesar disso, com a ausência de opção, essas aplicações utilizam ontologias completas, criando penalidades computacionais como problemas de desempenho, espaço e tempo de processamento, principalmente no uso de mecanismos de inferência.

Finalmente, o reuso estimula a integração de dados e a interoperabilidade entre aplicações que utilizam as mesmas ontologias. As melhores práticas para modelagem de aplicações baseadas na Web Semântica recomendam a utilização do reuso de ontologias.

Na literatura, existem diversas ferramentas que tratam o problema de reuso através da extração de uma porção da ontologia para a criação de uma nova ontologia.

A abordagem (Volz R. *et al*., 2003) trata essa extração de modo similar à elaboração de uma visão em banco de dados, sendo extraída a partir da execução de uma consulta.

Por outro lado, outras abordagens consideram que a extração de uma porção da ontologia é conceitualmente diferente da definição de uma visão em banco dados, visto que, a visão em ontologia pode incluir outros tipos de elementos (classes, relacionamentos e restrições) não sendo possível defini-los em uma consulta.

Considerando essa última abordagem, em (Noy, N. F. e Musen, M., 2004) é apresentado o conceito de *View Traversal* que define uma visão onde o usuário especifica o conceito central ou conceitos do seu interesse, os relacionamentos utilizados para encontrar outros conceitos a serem incluídos na visão, bem como, o número do nível em profundidade da travessia. A partir dessa abordagem foi desenvolvido um *plug-in* na ferramenta Protégé.

Neste trabalho, o mecanismo adotado para lidar com o reuso de ontologia consiste em extrair porções da ontologia a partir da execução de operação de gerenciamento conhecida como **projeção,** de modo a obter um subconjunto da ontologia a partir da seleção de termos do vocabulário das ontologias (classes e propriedades).

3.3. VERSIONAMENTO DE ONTOLOGIAS

Muitas vezes, as ontologias são desenvolvidas de forma colaborativa por várias pessoas ou são acessadas de modo distribuído. Neste cenário, provavelmente, as aplicações ou os especialistas do domínio necessitam lidar com as diferentes versões de uma determinada ontologia ao longo do tempo.

Para tanto, é definido o controle de versionamento de ontologias, com a responsabilidade de gerenciar as atualizações das ontologias e seus efeitos através da criação e manutenção de suas diferentes versões (Klein, M., 2002). Às vezes, versões novas e antigas são armazenadas, mas nenhum mecanismo é provido para destacar as diferenças entre estas versões.

A detecção de diferenças, na maioria dos casos, compara duas versões de uma mesma ontologia, identificando as alterações existentes entre elas. Diferenças simples são aquelas que não afetam a estrutura da ontologia, como a alteração de nomes de classes e propriedades. Diferenças complexas incluem atualização na hierarquia de classes ou na semântica dos conceitos (mudanças nas restrições). Existem várias ferramentas que tratam da detecção de atualização em ontologias de

forma automática ou semiautomática, ou seja, com ou sem a intervenção do usuário.

Destaca-se a ferramenta PromptDiff (Noy N.F. *et al*, 2004) que identifica alterações estruturais na ontologia e permite aos usuários aceitar ou rejeitar cada alteração. PromptDiff auxilia o controle de versão, auxiliando o especialista do domínio no processo de desenvolvimento colaborativo das ontologias. Outra ferramenta denominada OntoDiff (Tury, M. e Bieliková. M., 2006) detecta automaticamente modificações entre a estrutura e o conteúdo de duas versões de uma ontologia, identificando termos adicionados, removidos e modificados.

O mecanismo desenvolvido neste trabalho para tratar o problema de versionamento de ontologias consiste em disponibilizar a operação de diferença entre duas ontologias. Além disso, não tratamos apenas das diferenças simples entre os vocabulários das ontologias envolvidas, mas também das diferenças complexas envolvendo as suas restrições.

3.4. EVOLUÇÃO DE ONTOLOGIAS

As ontologias devem ser atualizadas, mas acima de tudo devem preservar a sua consistência ao longo das atualizações. Portanto, a evolução de uma ontologia ocorre quando um sistema de gerenciamento permite a sua atualização, bem como, a preservação da sua consistência.

A evolução da ontologia é definida por (Maedche *et al.*, 2003) como "um processo de adaptação da ontologia em relação às crescentes mudanças do seu correspondente domínio, de forma a manter sua própria a consistência, bem como a consistência dos artefatos dependentes dessa ontologia". O processo de evolução foi definido, por (Stojanovic, L., 2004), em seis etapas: (1) **captura das atualizações**, (2) **representação das atualizações**, (3) **semântica das atualizações**, (4) **implementação das atualizações**, (5) **propagação das atualizações** e (6) **validação das atualizações**. Detalharemos, a seguir, cada uma dessas etapas:

A etapa de **captura das atualizações** consiste na obtenção da mudança da ontologia de duas formas: (i) *top-down* – a partir da explícita solicitação e execução do especialista do domínio, (ii) *bottom-up* – a partir de métodos de descoberta de mudanças que podem ser classificados em três padrões: guiados pela estrutura, guiados por dados e guiados por uso.

Na etapa de **representação das atualizações**, as atualizações são representadas sobre diversos níveis de granularidade, como: (i) atualizações elementares, por exemplo, remover um atributo e (ii) atualizações complexas, por exemplo, mover um conceito para uma classe generalizada. Portanto, com a finalidade de resolver as atualizações é necessário identificá-las e representá-las em um formato adequado.

A etapa de **semântica das atualizações** considera o efeito da atualização na própria ontologia, e verifica a sua consistência depois da atualização ser aplicada. É importante destacar que o significado da consistência depende fortemente do modelo de ontologia adotado, como, por exemplo: frame, lógica descritiva DL (*Description Logic*).

A etapa de **propagação das atualizações** considera que as ontologias são frequentemente reusadas e estendidas. Portanto, a atualização em uma ontologia poderá também corromper as suas dependências, ou seja, todos os artefatos baseados nesta ontologia atualizada. Assim, o objetivo dessa fase é assegurar a consistência de artefatos dependentes depois da atualização da ontologia ser executada.

A etapa de **implementação das atualizações** considera que, antes da aplicação da atualização, todas as implicações são apresentadas ao usuário, cabendo a ele a decisão de aceitá-la ou descartá-la. Caso o usuário concorde com a execução da atualização, então todas as atividades relacionadas à atualização serão executadas.

Finalmente, a etapa de **validação das atualizações** permite ao usuário validar a atualização executada e reverter os seus efeitos quando necessário.

Neste trabalho, considerando o processo de evolução de ontologias descrito acima, apenas a etapa de **semântica das atualizações** foi escolhida para ser desenvolvida. Isso ocorreu devido à limitação de tempo para lidar e propor soluções para tais problemas.

É importante destacar que o mecanismo de gerenciamento desenvolvido permite ao especialista do domínio verificar se a ontologia atualizada apresenta algum conceito inconsistente com o formalismo adotado.

3.5. INTEGRAÇÃO DE DADOS ENVOLVENDO ONTOLOGIAS

Os sistemas de integração de dados fornecem o acesso integrado a diferentes fontes de dados heterogêneas e distribuídas (Langegger, 2010). A principal vantagem desses sistemas consiste em permitir ao usuário a obtenção de uma visão completa e consistente de todos os dados existentes sem a necessidade de acessar as fontes separadamente.

Os sistemas de integração de dados seguem, normalmente, duas abordagens clássicas, segundo (Lenzerini, 2002): virtual ou materializada.

A abordagem materializada extrai os dados de múltiplas fontes e os importa em repositórios de dados chamados *datawarehouses*. Nessa abordagem, as consultas são realizadas sobre uma base de dados materializada e, dessa forma, apresentam melhor desempenho em relação à abordagem virtual. Contudo, considerando que os repositórios de dados são dinâmicos e autônomos não é trivial e factível manter um *datawarehouse* sempre atualizado. Assim, são necessários vários algoritmos a fim de extrair e constantemente atualizar as informações relevantes das bases distribuídas. Portanto, a desvantagem dessa abordagem é a necessidade de manter o repositório de dados sempre atualizado.

Na abordagem virtual, os dados são recuperados diretamente das fontes quando o sistema de integração precisa responder a uma consulta, ou seja, sistemas enviam consultas diretamente às fontes de dados e, em seguida, os resultados individuais obtidos são integrados compondo a resposta final da consulta submetida pelo usuário ou pela aplicação. A principal vantagem dessa abordagem consiste em garantir que os dados acessados estão sempre atualizados, entretanto, sabe-se que os custos de processamento das consultas e de acesso às fontes de dados são altos e devem ser considerados como fatores críticos.

É importante evidenciar que uma aplicação de integração deve escolher a abordagem mais adequada a sua arquitetura e suas características. A maioria das pesquisas envolvendo sistemas de integração de dados adota a abordagem virtual, com o objetivo de proporcionar o compartilhamento integrado das informações presentes em múltiplas fontes de dados. Nessa abordagem, cada fonte de dados independente e autônoma é representada por sua própria ontologia. Desse modo, temos várias ontologias integradas a fim de formar a ontologia global que, como em Banco de Dados, será obtida a partir do mapeamento entre as várias ontologias. Essa abordagem de *integração de ontologias*, também conhecida como união de ontologias, identifica entidades idênticas entre as ontologias que descrevem as fontes de dados e, posteriormente, constrói uma nova ontologia que consiste na representação única, consistente e mínima das informações das fontes de dados originais. Neste contexto, faz-se necessária a utilização de técnicas de *matching de esquemas* (alinhamento de esquemas) na criação de um esquema integrado que represente várias fontes de dados.

As técnicas de *matching de esquemas* buscam a identificação das similaridades entre os esquemas através da análise e do reconhecimento de “casamento” das partes de um esquema com outras partes de um determinado esquema. Diversos estudos e ferramentas têm sido desenvolvidos sobre as técnicas de *matching* de esquemas, em (Rahm e Bernstein, 2001) é possível encontrar detalhes sobre essas técnicas. Segundo (Shvaiko, P. e Euzenat., J., 2004), existem duas principais classificações para essas técnicas:

Técnicas de *matching* baseadas em **elementos** dos esquemas executam o alinhamento dos

elementos analisando as entidades isoladamente, ignorando seus relacionamentos com outras entidades. Nessa categoria, as seguintes técnicas são apresentadas:

- **Técnicas baseadas em string**: utilizam a similaridade dos nomes e descrições dos elementos do esquema – quanto mais similares forem as strings, maior será a possibilidade que elas denotem os mesmos conceitos;
- **Técnicas baseadas em linguagem**: consideram strings como palavras em alguma linguagem natural e exploram as propriedades morfológicas dos termos, como a identificação das formas básicas das palavras e a eliminação de artigos, preposições e conjunções;
- **Técnicas baseadas em restrições**: analisam as restrições aplicadas às definições das entidades, como seus tipos de dados, cardinalidade dos atributos e suas instâncias. Assim, essas técnicas envolvem a comparação de vários atributos das entidades considerando os tipos de dados do seu valor e, também, se duas entidades possuem as mesmas instâncias, pois então existe uma grande chance delas denotarem conceitos similares.
- **Técnicas baseadas em recursos linguísticos**: usam dicionários do domínio a fim de combinar palavras baseado nos relacionamentos linguísticos entre elas, como por exemplo, sinônimos, hiponímios e hiperonímios.

Técnicas de *matching* baseadas na **estrutura** do esquema executam o mapeamento dos elementos analisando a estrutura das entidades. As técnicas dessa categoria são:

- **Técnicas baseadas em grafos**: os esquemas são considerados como uma estrutura de grafo e são identificadas estruturas similares entre os esquemas analisando suas partes equivalentes. O problema de *matching* em grafos é um problema combinatório que pode ser muito oneroso computacionalmente e, em geral, é resolvido por métodos aproximados (Rahm e Bernstein, 2001). As similaridades entre o(s) nó(s) de um grafo podem ser obtidas baseado na similaridade de seu(s) nó(s) filho(s) ou baseado em suas relações;
- **Técnicas baseadas em taxonomia**: são algoritmos de grafos que consideram apenas as relações de especialização. Os relacionamentos do tipo "é um" ligam conceitos que já são similares e, dessa forma, os termos que são vizinhos a esses relacionamentos também pode ter alguma similaridade;
- **Técnicas baseadas em modelos lógicos:** são algoritmos que consideram a interpretação semântica do modelo, sendo fundamentados em métodos dedutivos, por exemplo, técnicas de dedução baseada em satisfatibilidade proposicional e lógica descritiva. Essa abordagem propõe a decomposição do problema de *matching* dos grafos em problemas de *matching* em conjunto de nós. Na abordagem baseada em satisfatibilidade proposicional, os nós correspondentes e as suas possíveis relações são traduzidas em uma fórmula proposicional utilizando os seguintes símbolos lógicos: $\sqsubseteq$, $\equiv$. Contudo, a linguagem proposicional utilizada em técnicas de dedução baseada em satisfatibilidade proposicional é limitada em sua expressividade, pois trata apenas predicados unários. Para superar essa limitação utiliza-se a lógica descritiva. Na lógica descritiva, é possível tratar os relacionamentos binários, tais como propriedades e papéis, bem como, as relações de equivalência ($\equiv$) e as relações de subsunção ($\sqsubseteq$).

Várias ferramentas para *integração de esquemas* foram desenvolvidas utilizando algumas das técnicas de *matching* descritas anteriormente. Em particular, as ferramentas Chimaera (McGuinness, D. L. 2000), ODEMerge (Ramos., J. A., 2001) e Prompt (Noy, N. F. e Musen, M. A., 2000) realizam a integração (união) de ontologias. A ferramenta ODEMerge realiza os seus

procedimentos de modo automático e utiliza um simples dicionário para identificar conceitos sinônimos e hiperonímios. A maioria das ferramentas é integrada aos ambientes de gerenciamento de ontologias, tais como o Protégé e o Ontolingua. A desvantagem das ferramentas que realizam a integração de forma automática consiste na imprecisão dos mapeamentos e, algumas vezes, elas geram mapeamentos incorretos. Já as ferramentas que realizam a integração de forma interativa, também apresentam desvantagem, visto que, em geral, sobrecarregam o usuário com verificação de todos os mapeamentos encontrados.

Outro aspecto referente à *integração de esquemas* consiste na fusão de dados provenientes de diferentes fontes de dados em uma representação consistente. Desse modo, faz-se necessária a negociação de conflitos derivados dos diferentes modos de representação dos mesmos objetos do mundo real nas várias fontes de dados. O trabalho proposto por (Bleiholder e Naumann, 2006) descreve e classifica diferentes estratégias para negociação de conflitos, como pode ser visto a seguir:

- **Ignorar os conflitos:** consiste em descrever estratégias que não executam nenhuma decisão com relação a conflitos. Ao empregar tal estratégia, o sistema não precisa estar ciente dos conflitos nos dados, pois esta informação não é necessária ou usada. Essas estratégias são viáveis em qualquer situação de integração e são fáceis de implementar, possuindo dois representantes: **Passe adiante** e **Considere todas as possibilidades**. Na estratégia **Passe adiante**, todos os valores em conflito são apresentados para o usuário ou outra aplicação e, posteriormente, os possíveis conflitos entre os valores são tratados por esses. Por outro lado, a estratégia **Considerar todas as possibilidades** busca ser a mais completa possível, pois enumera todas as eventualidades e apresenta ao usuário todas as combinações possíveis de valores de atributos e ocasionalmente cria combinações que ainda não estão presentes nas fontes.
- **Evitar os conflitos**: essas estratégias não resolvem os conflitos, embora, lidem com dados inconsistentes.

A Tabela 3 apresenta uma análise comparativa das ferramentas de integração de ontologias apresentadas neste capítulo, bem como, as características da operação de integração (união) de ontologias desenvolvida nesta dissertação.

Tabela 3. Análise comparativa das ferramentas de integração de ontologias

Ferramenta	**Chimaera**	**PROMP**	**ODEMerge**	**Union**
Técnica de *matching* aplicada	Técnicas baseadas em taxonomia e em string	Técnicas baseadas em taxonomia, em string, restrições de tipo e grafos.	Técnicas baseadas em recursos linguísticos (sinônimos e hiperônimos).	Técnicas baseadas em string e em grafos
Linguagem de ontologias	Ontolingua, XOL	RDF(S)	RDF(S), DAML + OIL	OWL
Ambiente de Gerenciamento	Ontolingua	Protégé	WebODE	OntologyManagement
Nível de Automação	Semiautomático	Semiautomático	Automático	Automático
Tipo de elementos integrados	classes e propriedades	classes, propriedades e instâncias	classes e propriedades	classes, propriedades e restrições
Estratégia de negociação dos conflitos	Ignorar conflitos	Ignorar conflitos	Ignorar conflitos	Evitar conflitos

O mecanismo de gerenciamento proposto neste trabalho permite ao especialista do domínio não apenas uma ontologia que represente a união de ontologias. Mas permite obter também uma ontologia que represente a sobreposição (interseção) entre duas ontologias. Além de permitir a fusão de dados a partir da união de ontologias é importante que evidenciar que este mecanismo permite a descoberta de mapeamentos de equivalência entre os termos da ontologia.

3.6. CONSIDERAÇÕES FINAIS

Este capítulo apresentou os principais trabalhos relacionados ao tema de gerenciamento de ontologias. Apesar das diversas estratégias de gerenciamento de ontologias usarem diferentes enfoques e processos apresentados pela literatura, é possível constatar que nenhuma das abordagens oferece ao especialista do domínio uma ferramenta integrada para a elaboração de novas ontologias ou para a manutenção de múltiplas ontologias, de forma a proporcionar o reuso, o versionamento, a evolução, bem como, a integração de ontologias.

Além disso, nenhum trabalho apresentado teve a preocupação em auxiliar o especialista do

domínio na elaboração de uma ontologia que representa um entendimento correto sobre a semântica das ontologias envolvidas, visto que, para isso faz-se necessário considerar as restrições lógicas das ontologias originais e propagá-las para as novas ontologias.

Dessa forma, o trabalho proposto visa propor uma solução que atenda a algumas lacunas deixadas pelos trabalhos elencados neste capítulo.

CAPÍTULO 4

Grafo de Restrições

4.1. INTRODUÇÃO

A Web de dados tem crescido rapidamente e disponibilizado informações derivadas de várias fontes de dados. Quando ocorre a combinação de dados provenientes de fontes semanticamente heterogêneas, pode ocorrer conflito de dados, ou seja, a união de um conjunto de restrições de integridade mutuamente inconsistentes. Porém, lidar com essas inconsistências, é um grande desafio, visto que, dependendo do formalismo utilizado para representar as restrições não é possível desenvolver um algoritmo computacionalmente tratável que verifique a existência de inconsistências, pois esse problema é complexo (Artale, A. et al, 2009).

Entretanto, (Casanova *et al.*, 2011) delineou uma solução computacionalmente tratável para lidar com as restrições inerentes a um esquema conceitual *Extralite* com hierarquias de relacionamentos binários, descrito como uma ontologia usando a linguagem OWL, conforme detalhado no capítulo 2.

Essa estratégia consiste em definir um grafo de restrições que captura a estrutura de implicação lógica presente nas restrições de subconjunto. Além disso, (Casanova *et al.*, 2011) demonstrou que essa estratégia permite equilibrar a expressividade e decidibilidade, considerando uma família usual de restrições, e ainda conduz a um procedimento de verificação de inconsistências em tempo polinomial, em relação ao tamanho das restrições. Esse equilíbrio é obtido a partir da análise cuidadosa da forma como as restrições interagem.

Portanto, este capítulo descreve a estratégia de criação do grafo de restrições, que possibilita lidar com as restrições como fórmulas booleanas na forma normal conjuntiva com pelo menos dois literais por cláusulas. Além disso, permite a utilização de procedimentos dedutivos de Lógica Descritiva baseada em técnica de tableaux (Baader,F.;Nutt, W., 2003), de modo a viabilizar a verificação da satisfatibilidade do esquema *Extralite* e, portanto, possibilita a verificação de inconsistências em tais esquemas.

A seção 4.2 define a representação do grafo de restrições, bem como, o algoritmo de satisfatibilidade estrita em um grafo de restrições, que viabiliza a identificação de inconsistências no esquema conceitual *Extralite* com hierarquias de relacionamentos binários. A seção 4.3 exemplifica os conceitos definidos na seção 4.2 através da apresentação de três cenários. Por fim, a seção 4.4 apresenta as considerações finais.

4.2. REPRESENTAÇÃO DO GRAFO DE RESTRIÇÕES

Seja Σ um conjunto de restrições *Extralite* normalizadas, e Ω um conjunto finito de expressões de restrições *Extralite*, ou seja, expressões que podem ocorrer no lado direito ou esquerdo de uma restrição normalizada. Observe que as regras que definem a normalização de uma restrição foram detalhadas na seção 2.5.

O alfabeto é definido como um conjunto finito de conceitos e propriedades atômicos que ocorrem em Σ e Ω.

Considere que o ***complemento*** de uma descrição não negada $\boldsymbol{c}$ é $\neg\boldsymbol{c}$, e vice-versa. Além disso, considere que $\boldsymbol{e}$ seja uma descrição de conceito e o conjunto de todos os pares de indivíduos; ou que $\boldsymbol{e}$ seja uma descrição de papel.

Além disso, define-se que Σ implica logicamente em $\boldsymbol{e} \sqsubseteq \bot$, se e somente se, qualquer modelo de Σ resultar no conjunto vazio para a descrição $\boldsymbol{e}$, e que Σ implica logicamente em $\top \sqsubseteq \boldsymbol{e}$, se e

somente se, qualquer modelo de Σ resultar no conjunto de todos os pares de indivíduos para ***e***.

A ***Proposição 1*** declara as propriedades das descrições que serão usadas para construir o grafo de restrições, conforme a seguir.

Proposição 1*:* Considere que ***e***, ***f*** e ***g*** são descrições de conceitos ou de papéis, ***P*** e ***Q*** são papéis atômicos, e ***p*** pode ser ***P*** ou $\boldsymbol{P}^{-}$. Então, temos:

(i) $(\geq n\ p) \sqsubseteq (\geq m\ p)$ é uma tautologia, onde 0<m<n.

(ii) $e \sqsubseteq f$ é tautologicamente equivalente a $\bar{f} \sqsubseteq \bar{e}$.

(iii) Se Σ implica logicamente em $e \sqsubseteq f$ e $f \sqsubseteq g$, então $e \sqsubseteq g$.

(iv) Se Σ implica logicamente em $P \sqsubseteq Q$, então Σ implica logicamente em $(\geq k\ P) \sqsubseteq (\geq k\ Q)$ e $(\geq k\ P^{-}) \sqsubseteq (\geq k\ Q^{-})$.

(v) Se Σ implica logicamente em $(\geq 1\ P) \sqsubseteq \neg(\geq 1\ Q)$ ou $(\geq 1\ P^{-}) \sqsubseteq \neg(\geq 1\ Q^{-})$, então Σ implica logicamente em $P \sqsubseteq \neg Q$.

(vi) Se Σ implica logicamente em $e \sqsubseteq f$ e $e \sqsubseteq \neg f$, então Σ implica logicamente $e \sqsubseteq \bot$.

(vii) Se Σ implica logicamente em $(\geq 1\ P) \sqsubseteq \bot$ ou $(\geq 1\ P^{-}) \sqsubseteq \bot$, então Σ implica logicamente em $P \sqsubseteq \bot$.

(viii) Se Σ implica logicamente em $P \sqsubseteq \bot$, então Σ implica logicamente em $(\geq m\ P) \sqsubseteq \bot$, $(\geq m\ P^{-}) \sqsubseteq \bot$, $\top \sqsubseteq (\leq n\ P)$ e $\top \sqsubseteq (\leq n\ P^{-})$, onde $m > 0$ e $n \geq 0$.

A partir da **Proposição 1**, (Casanova *et al.*, 2011) definiu várias regras para a construção do grafo de restrições.

Este grafo possui nó(s) que são rotulados com expressões ou conjuntos de expressões. O grafo de restrições é usado na elaboração de procedimentos eficientes para testar se Σ é estritamente satisfatível, bem como, para manipular as restrições de Σ.

O grafo de restrições é construído a partir da ***Definição 1*** e da ***Definição 2*** apresentadas a seguir. A ***Definição 1*** descreve quatro estágios de elaboração do grafo de restrições considerando as restrições do esquema *Extralite*, conforme elencado a seguir. Já a ***Definição 2*** representa os nó(s) a serem colapsados quando ocorre transitividade das restrições de subconjunto de modo a formar um ciclo no grafo.

Por outro lado, a ***Definição 3*** e a ***Definição 4*** são utilizadas para definir se um determinado nó do grafo pode ser considerado como um $\bot$-nó.

A Tabela 5 descreve a relação entre a ***Proposição 1*** e as definições utilizadas para construir o grafo de restrições.

Tabela 4. Descreve a relação entre a Proposição 1 e as Definições do grafo de restrições

Proposição 1(i)	Definição 1 – Estágio 3(ii)
Proposição 1(ii)	Definição 1 – Estágio 3(iii)
Proposição 1(iii)	Definição 2
Proposição 1(iv)	Definição 1 – Estágio 2
Proposição 1(v)	Definição 1 – Estágio 4
Proposição 1(vi)	Definição 3 – casos (i) e (ii-a)
Proposição 1(vii)	Definição 3 – caso (ii-c)
Proposição 1(viii)	Definição 3 – caso (ii-b)

A seguir, o processo de criação do grafo de restrições é detalhado.

Definição 1: O grafo rotulado $g(\Sigma,\Omega) = (\gamma,\delta,\kappa)$ que captura Σ e Ω, onde cada nó é rotulado com uma expressão κ, γ contém as descrições de conceitos e expressões das restrições e δ possui restrições de inclusão, conforme definido a seguir:

Estágio 1: Inicializar $g(\Sigma,\Omega)$ com os seguintes nó(s) e arcos:

(i) Para cada conceito atômico C, $g(\Sigma,\Omega)$ tem exatamente um nó em γ rotulado com $\boldsymbol{C}$.

(ii) Para cada papel atômico P, $g(\Sigma,\Omega)$ tem exatamente um nó em γ rotulado com $(\geq 1\ P)$, e um nó em γ rotulado com $(\geq 1\ P^{-})$.

(iii) Para cada expressão $\boldsymbol{e}$ que ocorre no lado direito ou esquerdo de uma inclusão em Σ, ou que ocorre em Ω, com exceção daquelas que ocorrem em (i) ou (ii), $g(\Sigma,\Omega)$ tem exatamente um nó em γ rotulado com $\boldsymbol{e}$.

(iv) Para cada inclusão $\boldsymbol{e \sqsubseteq f}$ em Σ, $g(\Sigma,\Omega)$ tem um arco $\boldsymbol{(M,N)}$ em δ, onde $\boldsymbol{M}$ e $\boldsymbol{N}$ são nos nó(s) rotulados com $\boldsymbol{e}$ e $\boldsymbol{f}$, respectivamente.

Estágio 2:

Até que nenhum novo nó ou arco possa ser adicionado em δ,

Para cada inclusão de propriedade $\boldsymbol{P \sqsubseteq Q}$ em Σ,

Para cada nó $\boldsymbol{K}$ em γ,

(i) Se $\boldsymbol{K}$ é rotulado com $\boldsymbol{(\geq k\ P)}$, para algum $\boldsymbol{k>0}$, então adicione um nó $\boldsymbol{L}$ rotulado com $\boldsymbol{(\geq k\ Q)}$ em γ e um arco $\boldsymbol{(K,L)}$ em δ, se tal nó e arco não existir;

(ii) Se $\boldsymbol{K}$ é rotulado com $\boldsymbol{(\geq k\ P^{-})}$, para algum $\mathbf{k>0}$, então adicione um nó $\boldsymbol{L}$ rotulado com $\boldsymbol{(\geq k\ Q^{-})}$ em γ e um arco $\boldsymbol{(K,L)}$ em δ, se tal nó e arco não existir;

(iii) Se $\boldsymbol{K}$ é rotulado com $\boldsymbol{(\geq k\ Q)}$, para algum $\boldsymbol{k>0}$, então adicione um $\boldsymbol{L}$ rotulado com $\boldsymbol{(\geq k\ P)}$ em γ e um arco $\boldsymbol{(L,K)}$ em δ, se tal nó e arco não existir;

(iv) Se $\boldsymbol{K}$ é rotulado com $\boldsymbol{(\geq k\ Q^{-})}$, para algum $\boldsymbol{k>0}$, então adicione um $\boldsymbol{L}$ rotulado com $\boldsymbol{(\geq k\ P^{-})}$ em γ e um arco $\boldsymbol{(L,K)}$ em δ, se tal nó e arco não existir;

Estágio 3: Até que nenhum novo nó ou arco possa ser adicionado em g(Σ,Ω),

(i) Se g(Σ,Ω) tem um nó em γ rotulado com uma expressão ***e***, então adicione um nó em γ rotulado com $\neg$***e***, se tal nó não existir.

(ii) Se g(Σ,Ω) tem um nó ***M*** em γ rotulado com ***(≥m p)*** e um nó ***N*** em γ rotulado com ***(≥n p)***, onde ***p*** é ***P*** ou ***P***$^{-}$ e ***0<m<n***, então adicione um arco ***(N,M)*** em δ, se tal arco não existir.

(iii) Se g(Σ,Ω) tem um arco ***(M,N)*** em δ, então adicione um arco ***(¬ N, ¬ M)*** em δ, se tal arco não existir.

Estágio 4: Até que nenhum novo nó ou arco possa ser adicionado em g(Σ,Ω),

Para cada par de nó(s) ***M*** e ***N*** tal que ***M*** e ***N*** são rotulados com ***(≥1 P)*** em γ e ***¬(≥k Q)*** em γ , respectivamente, e existe um caminho a partir de ***M*** para ***N***,

Adicione arcos ***(K,L)*** em δ e ***(¬ K, ¬ L)*** em δ, onde ***K*** e ***L*** são nó(s) rotulados com ***P*** e ***¬Q***, respectivamente, se tal arco não existir.

Definição 2: O *grafo de restrições* que representa Σ e Ω é o grafo rotulado G(Σ,Ω), onde cada nó é rotulado com um conjunto de expressões, definidas a partir de g(Σ,Ω) através da junção de cada clique de g(Σ,Ω) em um simples nó rotulado com as expressões que previamente rotularam os nós no clique. Quando Ω é um conjunto vazio, simplesmente escrevemos G(Σ) e dizemos que G(Σ) é o grafo de restrições que representa Σ. □

Considere as seguintes convenções para o entendimento das definições de criação do grafo de restrições. Se um nó ***K*** é rotulado com uma expressão ***e***, então ***¬K*** expressa o nó rotulado com ***¬e***. Além disso, também se utiliza ***K→M*** para indicar que existe um caminho a partir de um nó ***K*** para um nó ***M***, e ***K↛M*** indica que nenhum caminho existe. Nas definições abaixo, é utilizado que ***e→f*** define que existe um caminho a partir de um nó rotulado com ***e*** para o nó rotulado com ***f***, e ***e ↛ f*** para indicar que não existe caminho.

Definição 3: Seja G(Σ,Ω) o grafo de restrições que representa Σ e Ω. Define-se que o nó ***K*** de G(Σ,Ω) é um ⊥-*nó com nível n*, para um inteiro não negativo n, se somente se, uma das seguintes condições é satisfeita:

(i) ***K*** é um ⊥-*nó com nível 0*, se e somente se, existe nó(s) ***M*** e ***N***, não necessariamente distinto a partir de ***K***, e uma expressão positiva ***h*** tal que ***M*** e ***N*** são respectivamente rotulados com ***h*** e ***¬h***, e ***K→M*** e ***K→N***.

(ii) ***K*** é um ⊥-*nó com nível n+1* se e somente se

(a) Existe um ⊥-*nó* ***M*** do nível n, distinto a partir de ***K***, tal que ***K→M***, e ***M*** é o ⊥-*nó* com o menor nível tal que ***K→M***, ou

(b) ***K*** é rotulado com uma restrição de *minCardinality* da forma ***(≥k P)*** ou da forma ***(≥k P***$^{-}$***)*** e existe um ⊥-*nó* ***M*** do nível n, tal que ***M*** é rotulado com ***P***, ou

(c) ***K*** é rotulado com um papel atômico ***P*** e existe um ⊥-*nó* ***M*** do nível n, tal que ***M*** é rotulado com uma restrição de *minCardinality* da forma ***(≥1 P)*** ou da forma ***(≥k P***$^{-}$***)***. □

Definição 4: Um nó ***K*** é um ⊥-*nó* de G(Σ,Ω), se e somente se, **K** é um ⊥-*nó* com nível n, para algum inteiro não negativo n. Um nó ***K*** é um ⊤-nó de G(Σ,Ω), se e somente se, ***¬*****K** é um ⊥-*nó*. □

A partir das proposições e definições descritas anteriormente é possível definir o **Teorema 1** que serve como fundamento para o problema de verificação das inconsistências em esquemas *Extralite*.

Teorema 1: Considere que Σ seja um conjunto de restrições normalizadas. Considere que ***G(Σ)*** seja o grafo que representa Σ. Então, Σ é estritamente satisfatível, se e somente se, ***G(Σ)*** não tem nenhum **⊥-nó** rotulado com um conceito atômico ou com um papel atômico.

Baseado no Teorema1, a Figura 3.1 apresenta procedimento simples para testar a satisfatibilidade estrita, que tem a complexidade de tempo polinomial sobre o tamanho de Σ:

SAT (Σ)

Entrada: um conjunto de restrições Σ.

Saída: "SIM - Σ é estritamente satisfatível "
"NÃO - Σ não é estritamente satisfatível"

Início

1. **Normalizar** as restrições em Σ, criando um conjunto Σ'.
2. **Construir** o grafo de restrições G(Σ') que representa Σ'.
3. **Se** G(Σ') não tem ⊥-*nó* para um conceito atômico ou um papel atômico **então**
4. **retorne** "SIM - Σ é estritamente satisfatível "
5. **Senão**
6. **retorne** "NÃO - Σ não é estritamente satisfatível "

Fim

Figura 3.1. Algoritmo de Satisfatibilidade Estrita (Casanova *et al.*, 2011)

4.3. EXEMPLO

Esta seção apresenta exemplos aplicados a cenários que envolvem ontologias reais e que mostram a aplicabilidade das definições apresentadas na seção anterior.

Considere que uma instituição de pesquisa, denominada *Research Institution*, queira construir uma ontologia que represente o domínio de Pesquisas Acadêmicas e, para tanto uma equipe de especialistas do domínio define o vocabulário dessa ontologia, bem como as suas restrições reusando os vocabulários da ontologia FOAF (*Friends of a Friend*) (Brickley,D., Miller, L., 2010) e *XML Schema* (Thompson *et al*, 2004**)**. Os prefixos "foaf:" e "xsd:" referenciam, respectivamente, esses vocabulários. Os vocabulários da *Research Institution* utilizam o prefixo "ri:".

Neste contexto, três cenários diferentes foram modelados por especialistas do domínio que compõe essa equipe. As Figura 3.1, Figura 3.4 e Figura 3.7 apresentam os esquemas *Research Institution 1, Research Institution 2* e *Research Institution 3,* respectivamente. Os três esquemas possuem os mesmos vocabulários, mas se diferenciam em relação as suas restrições. A partir desses esquemas foram gerados os grafos de restrição correspondentes a cada esquema, conforme a Figura 3.3, Figura 3.6 e Figura 3.9.

Cenário 1: Um dos especialistas do domínio definiu o esquema *Research Institution 1* conforme a Figura 3.1. A Figura 3.2 formaliza as restrições: a primeira coluna apresenta as restrições de domínio e imagem; a segunda coluna descreve as restrições de cardinalidade; e a terceira coluna apresenta as restrições de subconjunto e de disjunção.

A primeira coluna da Figura 3.2 indica, por exemplo, que:

- *foaf:status* é um papel atômico modelando um atributo de *foaf:Agent* com a imagem *String*.
- *foaf:homepage* é um papel atômico modelando um relacionamento binário a partir de *foaf:Agent* para *foaf:Document*.
- *ri:homepageProject* é um papel atômico modelando um relacionamento binário a partir de *ri:ResearchInstitution* para *ri:Project*.

A segunda coluna da Figura 3.2 apresenta as seguintes restrições de cardinalidade:

- *foaf:status* tem restrições de cardinalidade máxima e mínima, ambas com o valor 1 no domínio de *foaf:Agent*.
- *foaf:status* $^{-}$ tem restrições de cardinalidade máxima e mínima não limitada, ambas com o valor 0 no domínio de *foaf:Agent*, que não precisa ser explicitamente declarada.
- *foaf:homepage* tem restrições de cardinalidade máxima e mínima, ambas com o valor 1 no domínio de *foaf:Agent*.
- *foaf:homepage* $^{-}$ tem restrições de cardinalidade máxima e mínima não limitada, ambas com o valor 0 no domínio de *foaf:Document*, que não precisa ser explicitamente declarada.
- *ri:homepageProject* tem restrições de cardinalidade máxima e mínima, ambas com o valor 1 no domínio de *ri:ResearchInstitution*.
- *ri:homepageProject* $^{-}$ tem restrições de cardinalidade máxima e mínima não limitada, ambas com o valor 0 no domínio de *ri:Project*, que não precisa ser explicitamente declarada.

A terceira coluna da Figura 3.2 apresenta as seguintes restrições normalizadas de subclasses e subpropriedades:

- *foaf:Agent* e *ri:ResearchInstitution*;
- *foaf:Document* e *ri:Project*;
- *foaf:homepage* e *ri:homepageProject;*

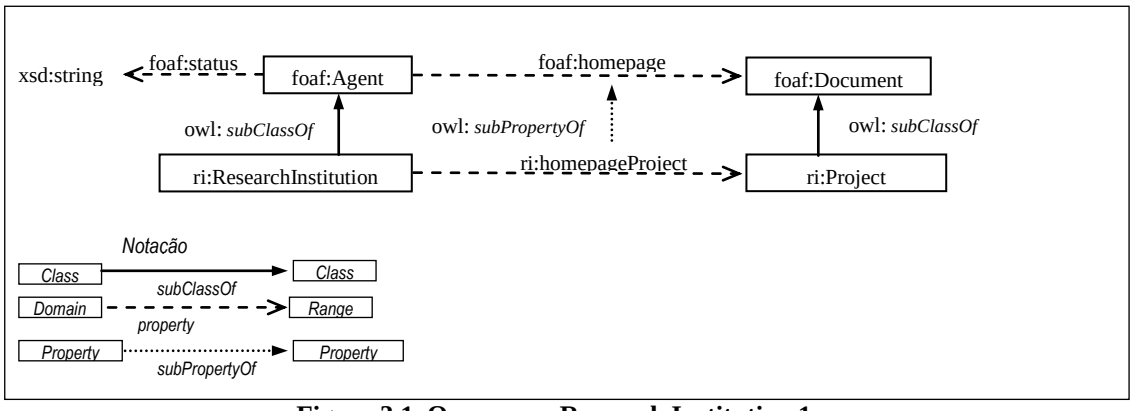

Figura 3.1. O esquema Research Institution 1.

Restrições de domínio e de imagem	Restrições de cardinalidade	Restrições de subclasse e sub propriedade
∃foaf:homepage ⊑ foaf:Agent ∃foaf:homepage – ⊑ foaf:Document ∃ri:homepageProject ⊑ ri:Project ∃ri:homepageProject ⊑ ri:ResearchInstitution ∃foaf:status – ⊑ String ∃status ⊑ foaf:Agent	foaf:Agent ⊑(≤1foaf:homepage) foaf:Agent ⊑(≥1foaf:homepage) ri:ResearchInstitution⊑(≥1 foaf:homepage) ri:ResearchInstitution⊑(≤1 foaf:homepage) foaf:Agent ⊑(≤1foaf:status) foaf:Agent ⊑(≥1foaf:status)	ri:ResearchInstitution⊑foaf:Agent ri:Project ⊑ foaf:Document foaf:homepageProject⊑ri:homepage

Figura 3.2. Definição formal das restrições normalizadas do esquema Research Institution 1 .

A Figura 3.3 mostra G(Σ), o grafo que representa o conjunto de restrições Σ, usando as restrições normalizadas. Todos os arcos representam os estágios descritos na seção 4.1.

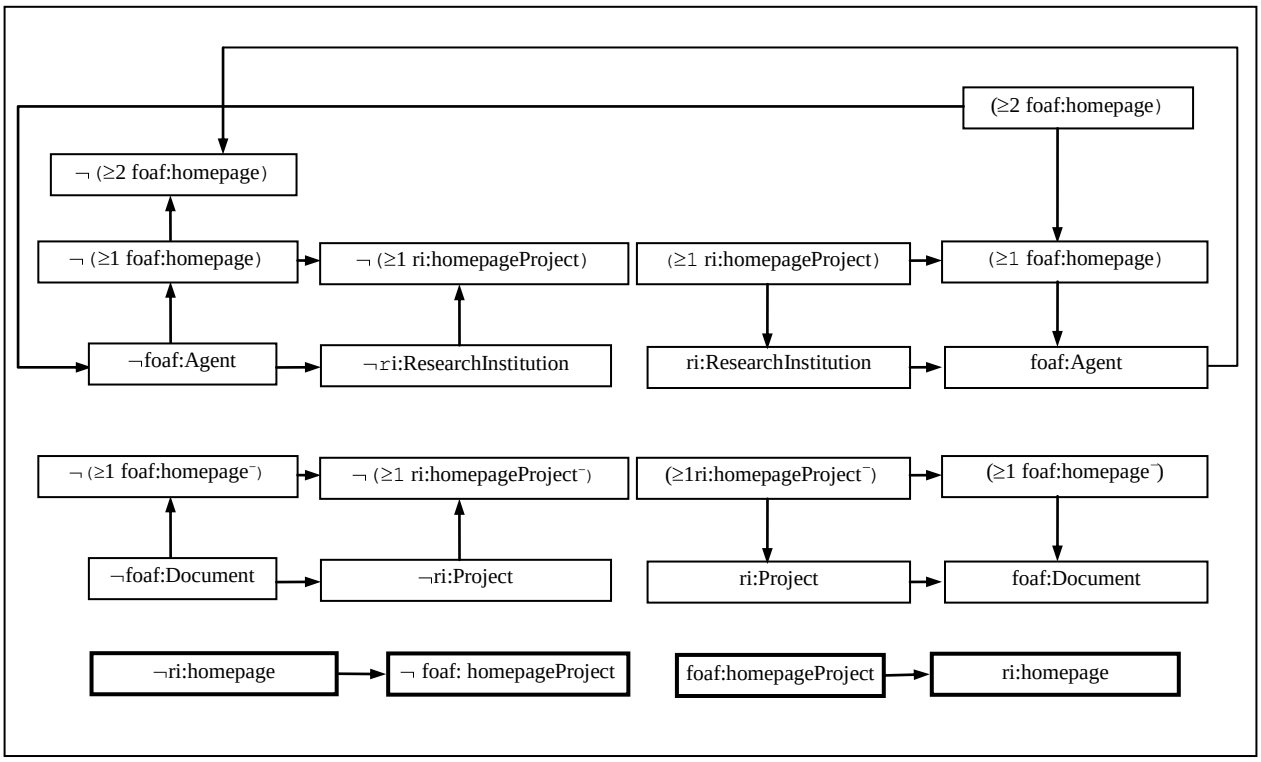

Figura 3.3. O grafo representando as restrições do esquema *Research Institution* 1.

Ao utilizar o grafo de restrições descrito acima para verificar a satisfatibilidade estrita do esquema *Research Institution 1*, verifica-se que G(Σ) não tem nó rotulado com ⊥-nó em um conceito atômico ou papel atômico, portanto Σ **é estritamente satisfatível** pelo **Teorema 1**.

Cenário 2: Um determinado especialista do domínio elaborou outro esquema conforme a Figura 3.4, modelando o problema sob outro ponto de vista. A Figura 3. 5 formaliza as restrições seguindo a mesma organização adotada no Cenário 1.

A primeira coluna da Figura 3. 5 indica as mesmas restrições de cardinalidade, de domínio e de imagem apresentada na Figura 3.2. Do mesmo modo, as restrições de cardinalidade da Figura 3.5 são as mesmas da Figura 3.2. Destaca-se a terceira coluna da Figura 3. 5 que apresenta as

restrições de disjunção dos seguintes termos:

- *foaf:Agent* e *ri:ResearchInstitution.*
- *foaf:Document* e *ri:Project*

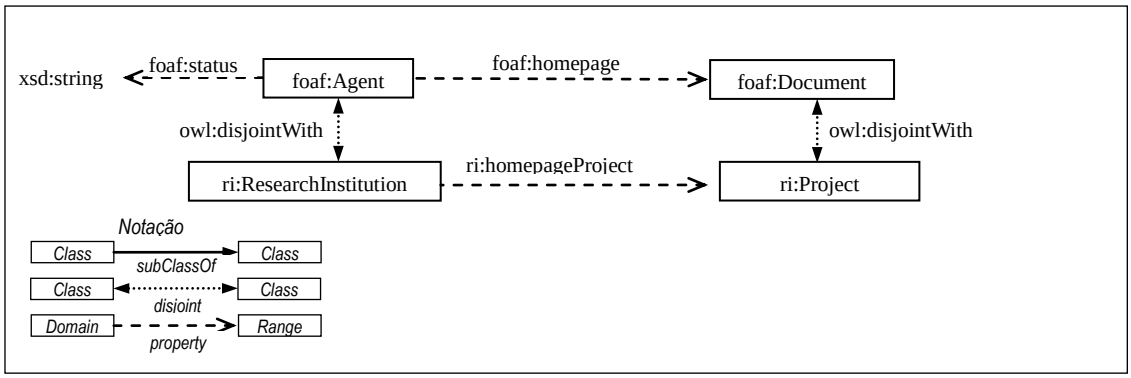

Figura 3.4. O esquema *Research Intitution* 2.

Restrições de domínio e de imagem	Restrições de cardinalidade	Restrições de disjunção
∃foaf:homepage ⊑ foaf:Agent ∃foaf:homepage – ⊑ foaf:Document ∃ri:homepageProject – ⊑ ri:Project ∃ri:homepageProject ⊑ ri:ResearchInstitution ∃foaf:status – ⊑ String ∃status ⊑ foaf:Agent	foaf:Agent ⊑(≤1foaf:homepage) foaf:Agent ⊑(≥1foaf:homepage) ri:ResearchInstitution⊑(≥1 foaf:homepage) ri:ResearchInstitution⊑(≤1 foaf:homepage) foaf:Agent ⊑(≤1foaf:status) foaf:Agent ⊑(≥1foaf:status)	ri:ResearchInstitution ⊑ ¬ foaf:Agent ri:Project ⊑ ¬ foaf:Document

Figura 3. 5. Definição formal das restrições normalizadas do esquema *Research Institution* 2

A Figura 3.6 mostra G(Σ), o grafo que representa o conjunto de restrições Σ, usando as restrições normalizadas.

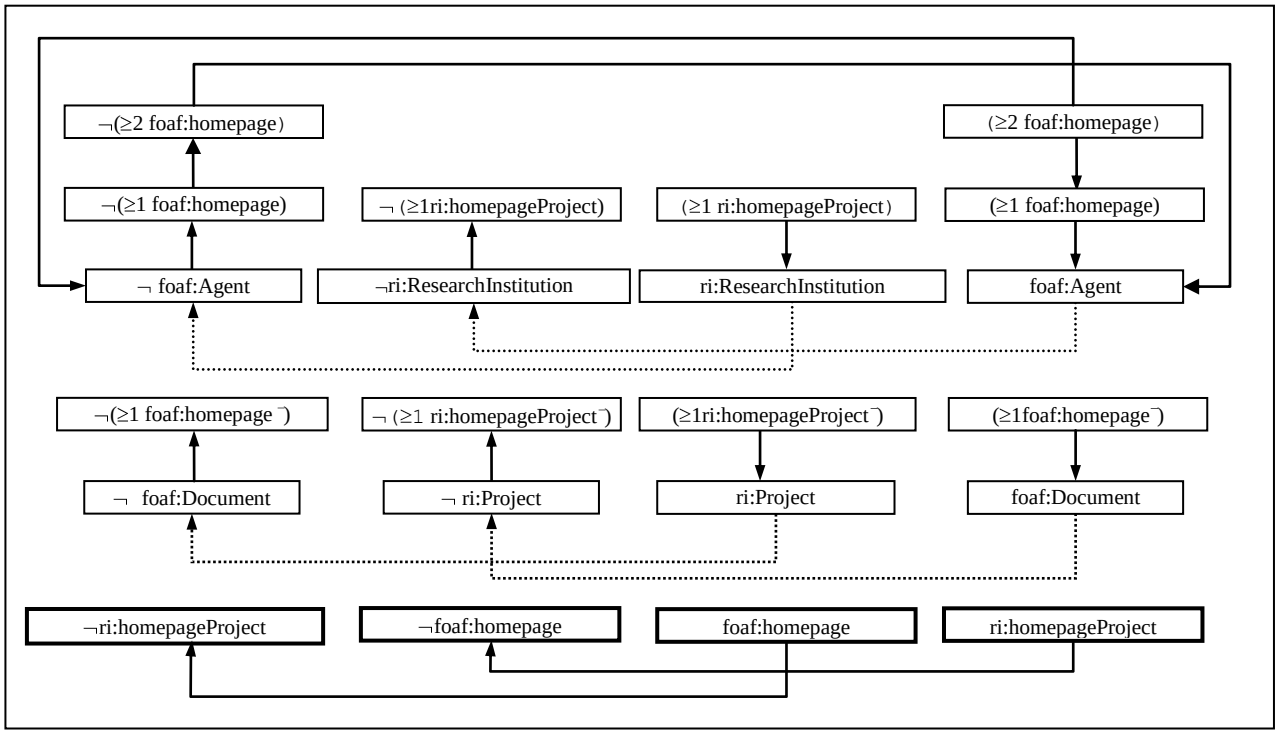

Figura 3.6. O grafo representando o esquema *Research Institution* 2 .

É importante destacar que G(Σ) não tem nó rotulado com ⊥-nó em conceito atômico ou papel atômico, isto é, Σ **é estritamente satisfatível** pelo **Teorema 1**. Contudo, observe que o nó (≥2 *foaf:homepage*) é um ⊥-nó de G(Σ). □

Cenário 3: Um dos especialistas do domínio definiu o esquema Research Institution 3 conforme a Figura 3.7. A Figura 3.8 formaliza as suas restrições: a primeira coluna apresenta as restrições de domínio e imagem; a segunda coluna descreve as restrições de cardinalidade; e a terceira coluna apresenta as restrições de subconjunto e de disjunção.

A primeira coluna da Figura 3.8 indica as mesmas restrições de domínio e imagem da Figura 3.2. Destaca-se a terceira coluna da Figura 3.8 que possui as restrições de disjunção dos seguintes termos:

- *foaf:Agent* e *ri:ResearchInstitution.*
- *foaf:Document* e *ri:Project*

E, a restrição de subconjunto dos seguintes termos:

- *foaf:homepage e ri:homepageProject.*

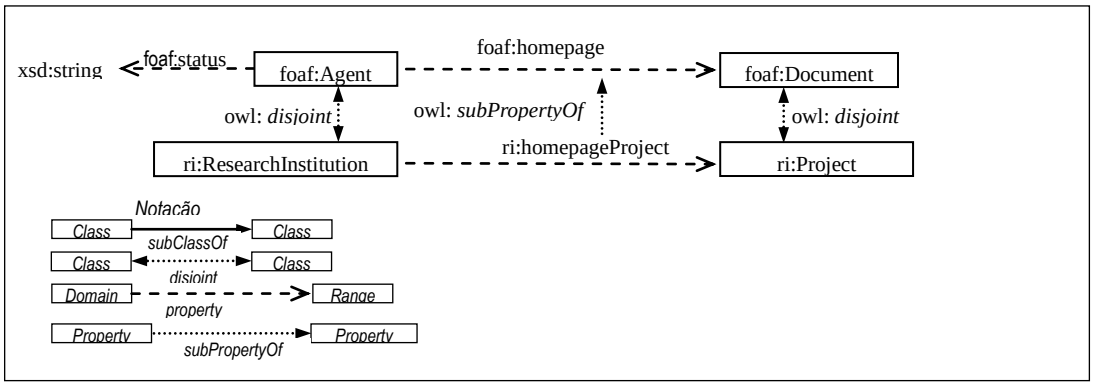

Figura 3.7. O esquema Research Institution 3.

Restrições de domínio e de imagem	Restrições de cardinalidade	Restrições de subconjunto e de disjunção
∃foaf:homepage ⊑ foaf:Agent ∃foaf:homepage – ⊑ foaf:Document ∃ri:homepageProject – ⊑ ri:Project ∃ri:homepageProject ⊑ ri:ResearchInstitution ∃foaf:status – ⊑ String ∃status ⊑ foaf:Agent	foaf:Agent ⊑(≤1foaf:homepage) foaf:Agent ⊑(≥1foaf:homepage) ri:ResearchInstitution⊑(≥1 foaf:homepage) ri:ResearchInstitution⊑(≤1 foaf:homepage) foaf:Agent ⊑(≤1foaf:status) foaf:Agent ⊑(≥1foaf:status)	ri:ResearchInstitution ⊑ ¬ foaf:Agent ri:Project ⊑ ¬ foaf:Document ri:homepageProject ⊑ foaf:homepage

Figura 3.8. Definição formal das restrições normalizadas do esquema *Research Institution* 3.

A Figura 3.9 mostra G(Σ), o grafo que representa Σ, usando as restrições normalizadas. Os arcos tracejados em vermelho destacam os caminhos que correspondem às condições do Estágio 4 da Definição 1.

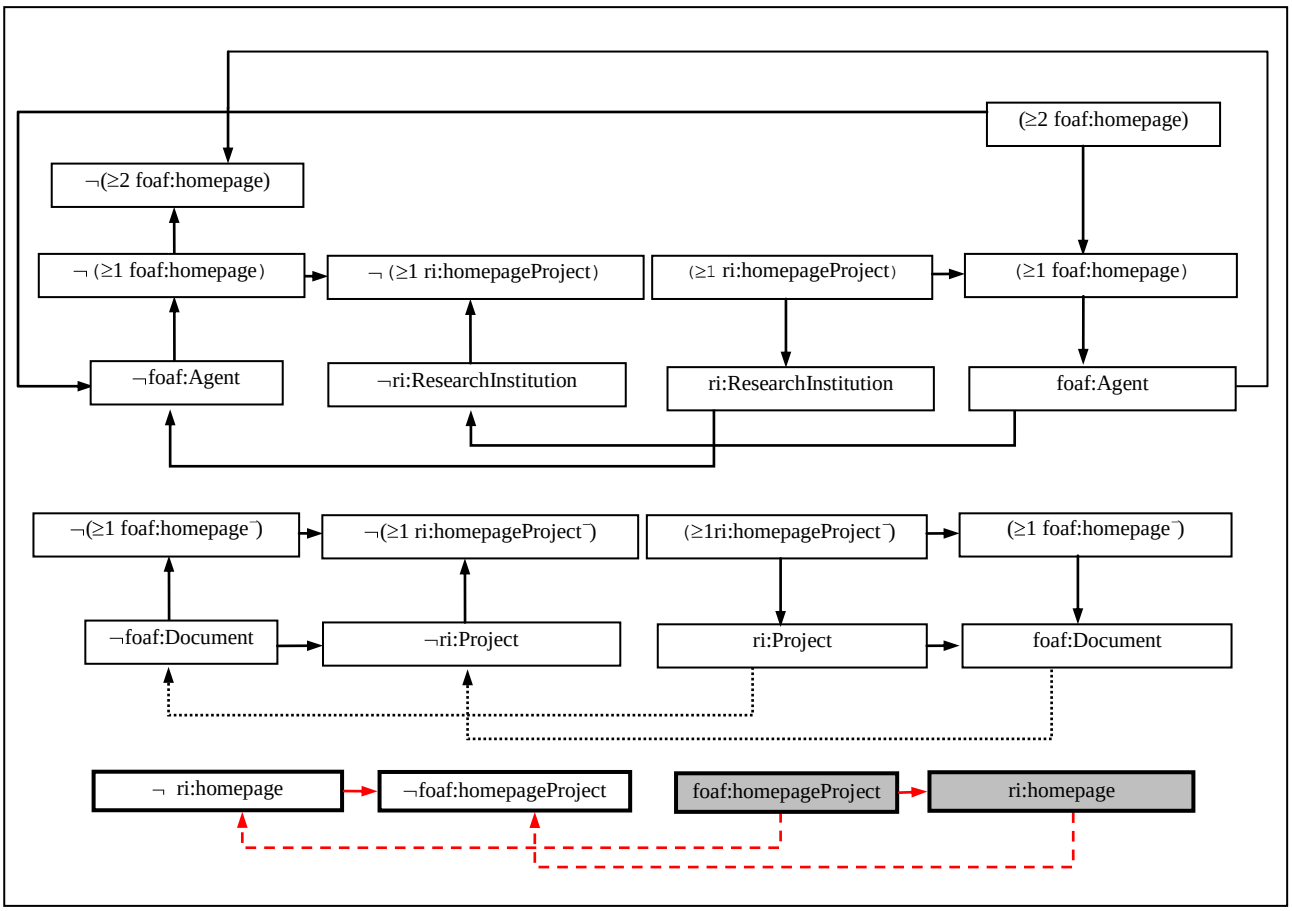

Figura 3.9. O grafo de restrições representando o esquema *Research Institution* 3.

É possível perceber que G(Σ) tem nó rotulado com ⊥-nó em um papel atômico, portanto Σ **não é estritamente satisfatível** pelo **Teorema 1**, visto que o nó rotulado com o conceito **foaf:homepage** e **foaf:homepageProject** é um ⊥-nó de G(Σ). □

Na verdade, qualquer interpretação s que satisfaz Σ é tal que s(foaf:homepage) ⊆ s(¬foaf:homepage) é satisfeita, o que implica que s (foaf:homepage) = ∅. □

Outra forma para explicar a não satisfatibilidade estrita de Σ será a partir das seguintes fórmulas:

(1) foaf: Agent | ri: ResearchInstitution (foaf: Agent e ri: ResearchInstitution são disjuntos)

(2) ∃foaf:homepage ⊑ foaf:Agent (A descrição do domínio de foaf:homepage é subconjunto de foaf:Agent)

(3)∃foaf:homepageProject ⊑ ri:ResearchInstitution(A descrição do domínio de foaf:homepageProject é subconjunto de ri: ResearchInstitution)

(4)foaf:homepage ⊑ foaf:homepageProject (foaf:homepage é subconjunto de foaf:homepageProject)

Então,

(5) ∃foaf:homepageProject | ∃foaf:homepage por (1), (2) e (3).

(6) foaf:homepageProject | foaf:homepage

Por isso, qualquer modelo s de Σ é tal que s (foaf:homepage) = ∅ considerando as fórmulas (4) e (6). Então, Σ não tem nenhum modelo estrito.

4.4. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada uma estratégia que transforma as ontologias em grafo de restrições e, posteriormente, utiliza esse grafo para tratar o problema de verificação da satisfatibilidade estrita dos termos de uma determina ontologia. Esta estratégia é utilizada também para manipular as ontologias envolvidas nas operações algébricas apresentadas no próximo capítulo.

CAPÍTULO 5

Operações sobre ontologias *Extralite*

5.1. INTRODUÇÃO

As aplicações, que utilizam os princípios do *Linked Data* para publicar e interligar dados estruturados na Web, enfrentam o desafio de descrever esses dados de modo a possibilitar a sua integração no espaço global de dados, conforme delineado em (Heath; Bizer, 2011).

Diante desse desafio, o especialista do domínio deve seguir as recomendações das melhores práticas de publicação de dados. Uma dessas recomendações consiste no reuso dos termos de vocabulários das ontologias largamente usadas. Além disso, este trabalho defende que o especialista do domínio deve fazer uma análise mais aprofundada das ontologias existentes, antes de reusar seus vocabulários na elaboração de novas ontologias e, portanto, recomenda a análise das restrições das ontologias reutilizadas.

Neste capítulo, será apresentado um conjunto de operações sobre ontologias, baseado em uma álgebra, que possui a finalidade de auxiliar a elaboração de novas ontologias considerando as restrições das ontologias envolvidas. Tais operações vão além da ideia de *namespaces*, pois, levam em consideração as restrições dessas ontologias e as tratam como teorias, ou seja, como um conjunto de restrições sobre um dado vocabulário.

Desse modo, a cada seção deste capítulo serão apresentados os algoritmos das operações de **união**, **interseção**, **diferença** e **projeção** sobre as ontologias construídas a partir do formalismo da Lógica Descritiva *Lite* (DL-*Lite*), conforme detalhado no capítulo 2. Devido a esse formalismo, no decorrer desse trabalho as ontologias serão denominadas ontologias *Extralite*.

Destaca-se o seguinte cenário de aplicação dessa estratégia de operação sobre ontologias *Extralite*. Considere uma fonte de dados **S** cujos dados são publicados de acordo com uma ontologia $\boldsymbol{O_0}$ elaborada a partir de ontologias conhecidas, tais como $\boldsymbol{O_1 ... O_n}$. Assumindo que as restrições de $\boldsymbol{O_0}$ são derivadas das outras *n* ontologias conforme descrito, e que os dados publicados por **S** são consistentes com as restrições de tais ontologias, então, em geral, qualquer aplicação que acessar **S** e que processar dados consistentes com $\boldsymbol{O_1 ... O_n}$, também poderá ser capaz de processar dados publicados por **S**, visto que, a aplicação requer dados consistentes com as restrições derivadas a partir daquelas ontologias $\boldsymbol{O_1 ... O_n}$ que contêm classes e propriedades do vocabulário de $\boldsymbol{O_0}$.

Este capítulo está definido da seguinte forma. A seção 5.1 apresenta as principais definições das operações (união, interseção, diferença e projeção) realizadas sobre ontologias *Extralite*. Além disso, apresenta o algoritmo responsável por gerar as restrições e o algoritmo responsável por lidar com as restrições de implicação lógica denominado fecho transitivo. A seção 5.2 apresenta a operação de união entre ontologias *Extralite* e exemplifica essa operação usando as ontologias *Research Institution 1* e *Research Institution 2*, detalhadas no capítulo 4. A seção 5.3 apresenta a operação de interseção sobre ontologias *Extralite* e um exemplo da execução dessa operação utilizando como operando a Ontologia da DBLP[14] e a Ontologia da plataforma *Lattes*[15]. A seção 5.4 apresenta a operação de diferença sobre ontologias *Extralite* e o exemplo da realização dessa operação em duas versões da ontologia FOAF[16]. Finalmente, a seção 5.5 expõe a operação de

[14] http://www.informatik.uni-trier.de/~ley/db/

[15] http://lattes.cnpq.br/

[16] http://xmlns.com/foaf/spec/

projeção, bem como, o exemplo da aplicação dessa operação sobre a ontologia DBpedia[17].

5.2. DEFINIÇÃO DAS OPERAÇÕES EM ONTOLOGIAS *EXTRALITE*

Uma ontologia é definida como um par $O = (V, \Sigma)$, onde V é um vocabulário finito, cujos conceitos e propriedades atômicos são denominados classes e propriedades, respectivamente, e Σ é um conjunto de inclusões finitas de V, denominado restrições de O. Em particular, uma ontologia *Extralite* é uma ontologia cujas restrições são inclusões de $DL\text{-}Lite_{core}^{\mathcal{HN}}$, conforme detalhado na Tabela 2 do Capítulo 2.

Além disso, considere que a teoria de Σ em V, denominada $\tau[\Sigma]$, é o conjunto de todas as inclusões de V que são consequência lógica de Σ.

A **Definição 5,** a seguir**,** apresenta as operações sobre ontologias *Extralite*. É importante evidenciar que a nova ontologia obtida a partir da execução dessas operações apresenta um conjunto de restrições que considera a semântica das restrições das ontologias envolvidas, bem como, o mapeamento entre tais ontologias.

Definição 5: Sejam $O_1 = (V_1, \Sigma_1)$ e $O_2 = (V_2, \Sigma_2)$ duas ontologias *Extralite*, W um subconjunto de V_1 e Φ é um conjunto de restrições sobre V_1.

(i) A *projeção* de $O_1 = (V_1, \Sigma_1)$ sobre W, representada por $\pi[W](O_1)$, retorna a ontologia $O_P = (V_P, \Sigma_P)$, onde $V_P = W$ e Σ_P é um conjunto de restrições em $\tau[\Sigma_1]$ que usa somente os símbolos de W.

(ii) A *união* de $O_1 = (V_1, \Sigma_1)$ e $O_2 = (V_2, \Sigma_2)$, representada por $O_1 \cup O_2$, retorna a ontologia $O_U = (V_U, \Sigma_U)$, onde $V_U = V_1 \cup V_2$ e $\Sigma_U = \Sigma_1 \cup \Sigma_2$.

(iii) A *interseção* de $O_1 = (V_1, \Sigma_1)$ e $O_2 = (V_2, \Sigma_2)$, representada por $O_1 \cap O_2$, retorna a ontologia $O_I = (V_I, \Sigma_I)$, onde $V_I = V_1 \cap V_2$ e $\tau[\Sigma_1] \cap \tau[\Sigma_2]$.

(iv) A *diferença* de $O_1 = (V_1, \Sigma_1)$ e $O_2 = (V_2, \Sigma_2)$, representada por $O_1 - O_2$, retorna a ontologia $O_D = (V_D, \Sigma_D)$, onde $V_D = V_1$ e $\Sigma_D = \tau[\Sigma_1] - \tau[\Sigma_2]$.

Considerando a **Definição 5**, obtemos os algoritmos utilizados para executar cada operação. Esses algoritmos seguem o mesmo padrão. Primeiramente, eles executam o algoritmo de fecho transitivo sobre os grafos de restrições das ontologias *Extralite* envolvidas em cada operação (exceto para a operação de União, que usa o grafo de restrições diretamente). Logo em seguida, cada algoritmo lida de forma particular com a sua respectiva operação, com a finalidade de gerar o grafo de restrições da ontologia resultante. Por fim, esses algoritmos executam o procedimento de gerar as restrições da nova ontologia resultante da operação.

A finalidade do algoritmo de fecho transitivo é gerar todas as arestas alcançáveis a partir de um determinado nó do grafo. Esta foi a principal motivação para a utilização deste algoritmo, pois ele permite obter a representação de todas as restrições de inclusão de uma determinada ontologia representada por um grafo de restrições. Por exemplo, considere duas restrições $A \sqsubseteq B$ e $B \sqsubseteq C$. É possível perceber que existe a restrição implícita $A \sqsubseteq C$ a partir dessas duas restrições. A forma adotada neste trabalho para obter essa informação implícita consiste em transformar as restrições de inclusão em arestas de um determinado grafo e utilizar o algoritmo do fecho transitivo para obter todos os caminhos entre dois nós.

A seguir, a definição do algoritmo de fecho transitivo é apresentada.

Definição 6: Seja G=(V, E) um grafo direcionado, onde V é o conjunto de vértices e E é o conjunto

[17] http://pt.dbpedia.org/

de arestas. O fecho transitivo de G é o grafo =), tal que, para todos os pares de vértices (v_i, v_j) em V, existe uma aresta de v_i para v_j em , se e somente se, v_j pode ser alcançável a partir de v_i em E.

A Figura 5.1 é utilizada para ilustrar o problema de fecho transitivo. Temos o grafo G com oito nós e suas arestas correspondentes e, além disso, temos o grafo com arestas pontilhadas, representando o fecho transitivo do grafo *G*.

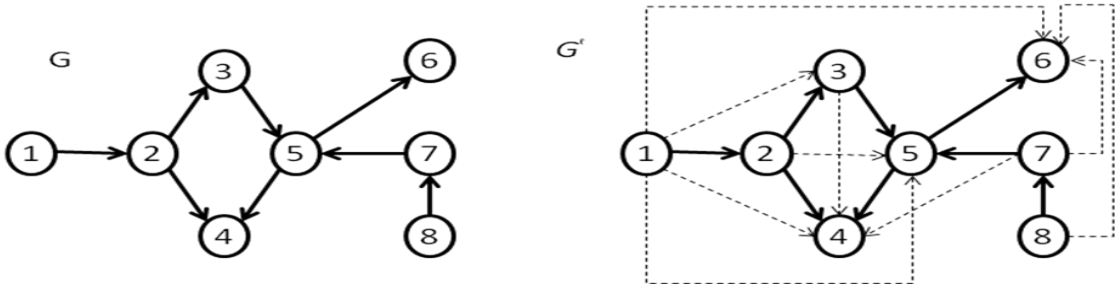

Figura 5. 1 do grafo G

Diversos algoritmos sequenciais e paralelos foram propostos para gerar o fecho transitivo (Roy, R., 1959) (Warshall, S., 1964), (Goodrich, M.T. e Tamassia., R., 2001) e (Koubková, A. e Koubek, V.,2002). Neste contexto, apresentamos duas abordagens para esses algoritmos, considerando as suas respectivas complexidades computacionais:

(1) Um algoritmo sequencial que utiliza uma matriz de adjacências para armazenar gradativamente o grafo que representa o fecho transitivo. Este algoritmo possui complexidade $O(n^3)$ e foi apresentado por Warshall (Warshall, S., 1964);

(2) Um algoritmo de busca em profundidade ou em largura é utilizado para encontrar todos os vértices v_j alcançáveis a partir do vértice v_i e gerar a aresta e_{ij}. Posteriormente, ao encontrar todos os vértices que um determinado vértice pode alcançar, o fecho transitivo é gerado. Ambas as buscas podem ser realizadas com complexidade de tempo $O(n + m)$, onde n representa a quantidade de vértices e m representa a quantidade de arestas do grafo. Consequentemente, a complexidade para obter o fecho transitivo, que utiliza uma busca em profundidade ou em largura para cada vértice do grafo, é $O(n(n + m))$ (Goodrich, M.T. e Tamassia., R., 2001).

Verificando a complexidade dos algoritmos para obtenção do fecho transitivo de um grafo elencadas anteriormente, é possível concluir que a abordagem mais adequada a ser adotada é a do algoritmo com complexidade de tempo quadrática. A Figura 5.2 descreve os passos do algoritmo para gerar o fecho transitivo de um grafo *G* conforme essa abordagem. Esse algoritmo de fecho transitivo verifica quais vértices um determinado vértice alcança. Utilizando uma busca em profundidade, encontramos todos os vértices alcançáveis a partir de v_i; através dessa busca é construída a árvore $\mathbf{T}_i$, e por fim, todos os vértices v_j presentes em $\mathbf{T}_i$ são alcançáveis a partir de v_i. Constrói-se $\mathbf{T}_i$, para todo $v_i \in V$ e, por fim, adiciona a aresta e_{ij} ao fecho transitivo final E .

Fecho Transitivo (G, G')

Entrada: um grafo G = (*V*, *E*), onde *V* e *E* descrevem os vértices e as arestas respectivamente;

Saída: um grafo G' = (*V*, *E*'), onde *V* e *E*' descrevem os vértices e as arestas do fecho transitivo, respectivamente;

início

1. **para cada** nó $v_i \in V$ **faça**
2. Compute $\mathbf{T}_i$
3. **para** cada nó $v_j \in \mathbf{T}_i$ **faça**
4. **crie** a aresta $e_{i,j}$ em *E*'
5. **retorne** *G*'
6. **fim**

Figura 5.2. Algoritmo do Fecho Transitivo.

Com a finalidade de simular o algoritmo do fecho transitivo, considere o grafo G ilustrado na Figura 5.3(a). Para cada vértice desse grafo é gerada uma árvore **T** que representa os vértices alcançáveis a partir desse vértice, conforme a iteração do laço na linha 1. Em particular, a Figura 5.33(b) refere-se a $\mathbf{T}_i$, árvore construída a partir do vértice v_i, onde i = 1. As arestas destacadas em vermelho representam a árvore do vértice 1. Finalmente, após a execução de todos os passos do algoritmo para cada nó do grafo, é possível obter o fecho transitivo de G, conforme ilustrado na Figura 5.33(c) a partir das arestas pontilhadas.

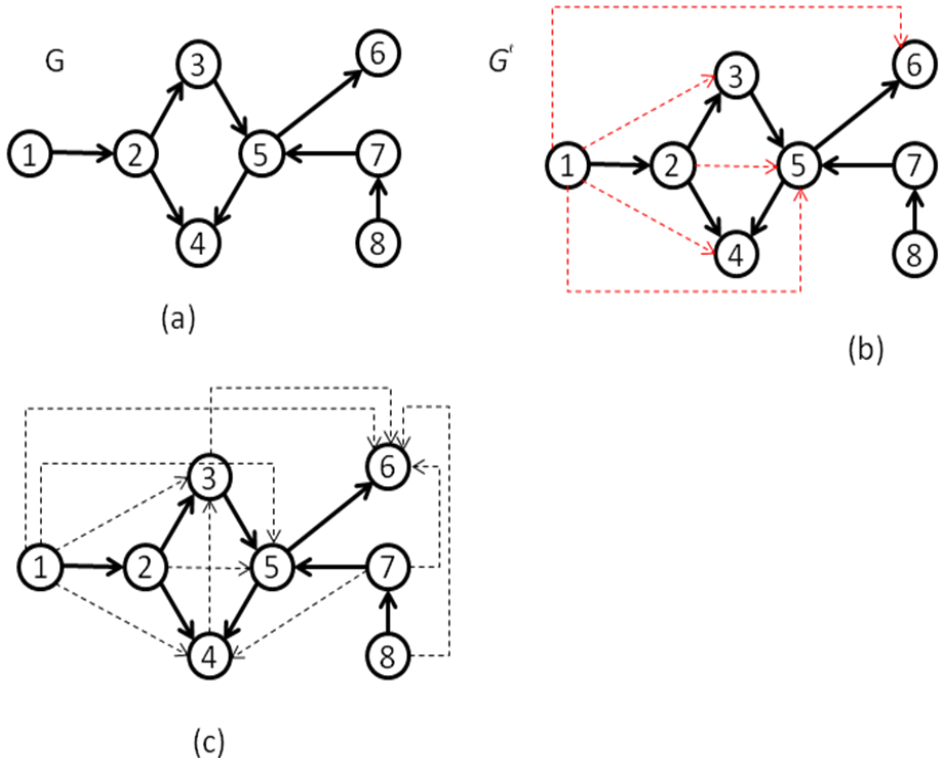

Figura 5.3. Execução do algoritmo Fecho Transitivo

A principal aplicabilidade do algoritmo de fecho transitivo é a detecção da propriedade de transitividade entre restrições de inclusão, sendo imprescindível para a realização correta das operações algébricas de interseção, diferença e projeção, detalhadas nas próximas seções. Por

exemplo, suponha que uma ontologia O_1 contém restrições Σ_1, tais como: A ⊑ B e B ⊑ C e outra ontologia O_2 contém a restrição Σ_2 , da seguinte forma: A ⊑ C. Em seguida, essas restrições são transformadas em arestas nos grafos de restrições correspondentes a cada ontologia. Caso seja aplicada a operação algébrica de diferença entre os grafos de restrições correspondentes às ontologias O_1 e O_2 , então serão obtidas as seguintes restrições resultantes: A ⊑ B e B ⊑ C. Essa operação apresenta apenas as restrições que estão em O_1 e não estão em O_2. É possível perceber que a restrição A ⊑ C não pertence à diferença entre O_1 e O_2, visto que, essa restrição é comum as duas ontologias podendo ser derivada de A ⊑ B e B ⊑ C. Desse modo, é importante evidenciar que a obtenção correta das restrições resultantes está estritamente relacionada ao algoritmo de fecho transitivo.

Além de utilizar o algoritmo de fecho transitivo, as operações sobre as ontologias utilizam um algoritmo para gerar as restrições resultantes da interação entre ontologias que serviram como operando dessas operações. Assim, ao finalizar cada algoritmo de operação sobre as ontologias, é executado o algoritmo **Gerador de Restrições**, conforme especificado na Figura 5.4.

O algoritmo **Gerador de Restrições** analisa o grafo de restrições *H* resultante da execução das operações sobre as ontologias e apresenta três possíveis estágios para geração de novas restrições e novos mapeamentos sendo detalhado a seguir:

(1) para cada nó-⊥ *M* de *H*, é gerada a restrição da forma $e \sqsubseteq \bot$, onde *e* rotula *M*, visto que, todas as expressões que rotulam um nó-⊥ denotam um conjunto vazio;

(2) para cada nó *M* de *H* que não é um nó-⊥ e nem é um nó-⊤, são geradas restrições da forma $e \equiv f$, onde *e* e *f* rotulam *M*, já que todas as expressões que rotulam o mesmo nó são equivalentes;

(3) Para cada arco *(M, N)* de *H*, tal que, *M* e *N* não são nem nó-⊥ e nem um nó-⊤, é gerada uma restrição da forma $e \sqsubseteq f$, onde *e* rotula *M* e *f* rotula *N*, uma vez que um arco denota uma restrição de inclusão. É importante evidenciar que as restrições de inclusão não são criadas para todos os pares de expressões a partir de *M* e *N*, respectivamente, considerando a perspectiva do passo (2).

A seguir, a Figura 5.4 apresenta o algoritmo **Gerador de Restrições**.

Gerador de Restrições (H; Σ)

Entrada: um grafo de restrições H
Saída: um conjunto de restrições Σ
início
1. **Inicialize** $\Sigma = \varnothing$;
2. **para cada** nó M de H **tal que** M é um nó-$\bot$ de H **faça**
3. **para cada** descrição de conceito e **tal que** e rotula M **faça**
4. **adicione** $e \sqsubseteq \bot$ para Σ;
5. **remova** todos os nó-$\bot$ e nó-$\top$ a partir de H;
6. **para cada** nó M de H **faça**
7. **para cada** par de descrições de conceito e e f **tal que** e e f rotulam M **faça**
8. **adicione** $e \equiv f$ em Σ;
9. **para cada** arco (M,N) de H **faça**
10. **selecione** expressões e e f **tal que**
11. e e f rotulam os nós M e N, respectivamente, e
12. $e \sqsubseteq f$ é uma restrição permitida (considere a Seção 2.5), e caso
13. $\bar{f} \sqsubseteq \bar{e}$ não esteja em Σ /* para evitar restrições redundantes*/
14. **adicione** $e \sqsubseteq f$ a Σ
15. **retorne** Σ
16. **Fim**

Figura 5.4. Algoritmo Gerador de Restrições.

Portanto, o algoritmo **Gerador de Restrições** permite a obtenção de possíveis novos mapeamentos e restrições derivados da execução das operações sobre as ontologias. Para simular a execução desse algoritmo e explicar a relevância de cada um dos três estágios descritos, considere o exemplo a seguir. Suponha que o algoritmo é a sua aplicação após a execução da operação de união entre as ontologias *ResearchInstituion 1* e *ResearchInstituion 2*, detalhadas no capítulo 4. Os vocabulários dessas ontologias são referenciados a partir dos seguintes prefixos "*ri1:*" e "*ri2:*", respectivamente. A seguir, serão apresentados os vocabulários das referidas ontologias e algumas das suas restrições normalizadas. Apenas algumas restrições de inclusão foram selecionadas para facilitar a explicação do referido algoritmo.

Formalmente, a ontologia *ResearchInstituion 1* = ($V_{ResearchInstituion1}$, $\Sigma_{ResearchInstituion1}$) , onde:

$V_{ResearchInstituion1}$ = {*foaf:Agent, foaf:Document, foaf:homepage, ri1:ResearchInstituion, ri1:Project, ri1:homepageProject*}

$\Sigma_{ResearchInstituion1}$ = {*foaf:Agent* $\sqsubseteq$ *ri1:ResearchInstituion, foaf:Document* $\sqsubseteq$ *ri1:Project, foaf:homepage* $\sqsubseteq$ *ri1:homepageProject*}

Por outro lado, formalmente, a ontologia *ResearchInstituion 2* = ($V_{ResearchInstituion2}$, $\Sigma_{ResearchInstituion2}$) , onde:

$V_{ResearchInstituion2}$ = { *foaf:Agent, foaf:Document, foaf:homepage, ri2:ResearchInstituion, ri2:Project, ri2:homepageProject*}

$\Sigma_{ResearchInstituion2}$ = { *ri2:ResearchInstituion* $\sqsubseteq \neg$ *foaf:Agent, ri2:Project* $\sqsubseteq \neg$ *foaf:Document* }

Considere que a Figura 5.5 ilustra o grafo de restrições *H* obtido pela execução da operação de união entre as ontologias *ResearchInstituion 1* e *ResearchInstituion 2* no momento anterior à invocação do algoritmo **Gerador de Restrições.** Assuma esse grafo como peça chave para o entendimento de cada estágio do referido algoritmo. É importante evidenciar que esse algoritmo é utilizado por todas as operações algébricas tratadas neste trabalho.

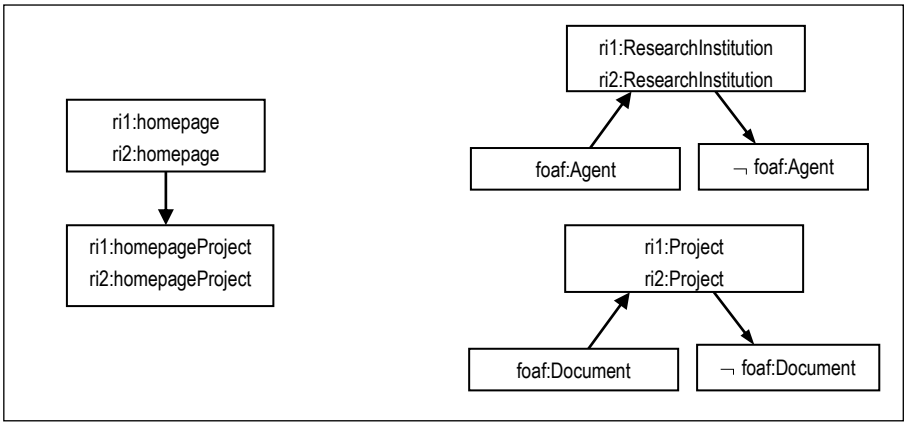

Figura 5.5. Exemplo do grafo H antes de invocar o algoritmo Gerador de Restrições

O estágio 1 verifica a ocorrência de nó-⊥ no grafo H. Na Figura 5.5, é possível verificar uma contradição ligando os conceitos *ri1:ResearchInstituion* e *ri1:Project*, portanto, os nós rotulados com esses conceitos são nó-⊥. Então, as seguintes restrições resultantes são obtidas, como a seguir: *ri1:ResearchInstituion* ⊑ ⊥ e *ri1:Project* ⊑ ⊥.

O estágio 2 busca descrever a equivalência entre as descrições de nós colapsados. Ao analisar o grafo H, conforme a Figura 5.5, é constatado que *ri1:homepage* e *ri2:homepage* descrevem um único nó e, portanto, são equivalentes, o mesmo ocorre para as descrições *ri1:homepageProject* e *ri2:homepageProject* de um determinado nó. Então, as seguintes restrições resultantes são obtidas, como a seguir: *ri1:homepage* ≡ *ri2:homepage* e *ri1:homepageProject* ≡ *ri2:homepageProject*.

O estágio 3 verifica se existe um arco no grafo H. Caso seja constatado que esse arco não representa uma restrição de inclusão redundante, então esta restrição é obtida com as expressões rotuladas pelos nós que compõem esse arco. A Figura 5.5 ilustra que o grafo H possui um arco ligando *ri1:homepage* e *ri2:homepageProject* e, portanto é gerada a restrição de inclusão *ri1:homepage* ⊑ *ri2:homepageProject*.

Outro aspecto importante relacionado com as operações binárias (união e interseção) abordadas nesse trabalho consiste na forma de lidar com os URIs que identificam os vocabulários das ontologias usadas como operandos de tais operações, pois, o mesmo termo de um vocabulário pode ser identificado por URI diferentes. Por exemplo, o mesmo termo *Article* na Ontologia da DBLP e na Ontologia da plataforma *Lattes* possui diferentes URI, tais como: http://www.semanticweb.org/ontologies/2009/10/academic-sbc.owl e http://sw.deri.org/~aharth/2004/07/dblp/dblp.owl, respectivamente.

Neste caso, este trabalho assume que tais termos podem ser considerados como o mesmo objeto do mundo real, conquanto, é possível verificar que eles são idênticos e, além disso, obter os seus mapeamentos de equivalência. Destaca-se que na ontologia resultante das operações de união e interseção, tais termos são fundidos em um único termo cuja URI é derivada de uma das ontologias conforme a opção do usuário ou aleatoriamente a partir de uma dessas URIs.

5.3. UNIÃO DE ONTOLOGIAS *EXTRALITE*

As aplicações que adotam os princípios do *Linked Data* perseguem o desafio de disponibilizar aos seus usuários informações integradas a partir de múltiplas fontes de dados. Para tanto, faz-se necessária a elaboração de uma ontologia integrada a partir da união de múltiplas ontologias.

Geralmente, o processo de integração de várias ontologias consiste na união de duas versões de ontologias sem o conhecimento se existiu uma ontologia comum que as originou (Hepp M. *et al*., 2008). Além disso, a união pode ocorrer entre dados de domínios antagônicos e, portanto, faz-se necessária a detecção de inconsistências entre esses termos idênticos, mas derivados de contextos divergentes e, portanto, geradores de conflitos de dados.

O processo de união pode ser realizado de diversos modos, considerando, por exemplo, somente a estrutura das ontologias envolvidas ou considerando-as como teorias em Lógica Descritiva (Baader et al., 2003). Em particular, neste trabalho, foi adotada essa última abordagem.

Outro aspecto considerado é o mapeamento de equivalências entre os vocabulários das ontologias envolvidas na operação de união, que consiste em fundir duas ontologias em uma única simples ontologia.

O processo de união de ontologias adotado neste trabalho é obtido a partir do operador algébrico de união entre duas ontologias, resultando em uma nova ontologia $O_3=(V_3,\Sigma_3)$, onde Σ_3 é o conjunto de restrições e V_3 representa o vocabulário de O_3. As duas ontologias O_1 e O_2 são normalizadas conforme detalhado na Tabela 1 do Capítulo 2. E, posteriormente, são gerados os grafos $G(\Sigma_1)= (\gamma_1,\delta_1,\kappa_1)$ e $G(\Sigma_2) = (\gamma_2,\delta_2,\kappa_2)$ a partir das restrições dessas ontologias, onde γ possui os nós que rotulam as descrições de conceitos e expressões das restrições dessas ontologias normalizadas; δ contém os arcos que representam as restrições de inclusão e, por fim, κ possui um conjunto de descrições de conceitos de um determinado nó, como especificado na seção 4.1.

A Figura 5.6 apresenta o algoritmo que executa a operação **união** de Ontologias *Extralite*.

União (V_1, Σ_1 , V_2, Σ_2 ; Σ_3)

Entrada: Duas ontologias *Extralite* $O_1=(V_1, \Sigma_1)$ e $O_2=(V_2, \Sigma_2)$;
Saída: Um conjunto de restrições Σ_3 ;
início
1. Construa $G(\Sigma_1) = (\gamma_1,\delta_1,\kappa_1)$ e $G(\Sigma_2)=(\gamma_2,\delta_2,\kappa_2)$; /* construir grafos para Σ_1 e Σ_2;
2. Inicialize $H =(\gamma,\delta,\kappa)$, onde $\gamma =\gamma_1\cup\gamma_2$, $\delta=\delta_1\cup\delta_2$ e $\kappa=\kappa_1\cup\kappa_2$;
3. **para** cada par de nós *M* e *N* de *H* **tal que**
4. existe a interseção dos conjuntos de expressões descritos em *M* e *N* **faça**
5. modifique H para fundir M e N em um nó único K, e
6. descreva K com todas expressões que rotulam M ou N;
7. **Gerador de Restrições**(H, Σ_3);
8. **retorne** Σ_3

fim

Figura 5.6. Algoritmo de União de Ontologias *Extralite*.

O algoritmo da operação de União, primeiramente, considera que $\gamma_1\cap\gamma_2=\varnothing$ e inicializa o grafo $H =(\gamma,\delta,\kappa)$, onde $\gamma =\gamma_1\cup\gamma_2$, $\delta=\delta_1\cup\delta_2$ e $\kappa=\kappa_1\cup\kappa_2$. Além disso, combinam-se dois nós *M* e *N* de *H* em um simples nó K, se e somente se, $\kappa(M)\cap\kappa(N)\neq\varnothing$ (ou seja, os conjuntos de rótulos de *M* e *N* têm uma expressão em comum); além disso, $\kappa(K)=\kappa(M)\cup\kappa(N)$ (o novo nó *K* é rotulado com todas expressões originalmente rotuladas como M ou N). Finalmente, o grafo H é passado como parâmetro para o algoritmo **Gerador de Restrições** com a finalidade de obter a nova ontologia $O_3=(V_1\cup V_2,\Sigma_3)$.

Para simular o algoritmo descrito anteriormente, considere a ontologia *ri:ResearchInstituion1* e a ontologia *ri:ResearchInstituion2*, detalhadas na seção 4.2. A Tabela 6 expõe algumas restrições de inclusão dessas duas ontologias que servem de operandos para a operação de união. Apenas algumas restrições foram consideradas com a finalidade de reduzir o tamanho do exemplo. A primeira coluna apresenta os termos do lado esquerdo da inclusão, *por outro lado, a segunda* coluna

apresenta os termos do lado direito da inclusão. Por exemplo, a linha 1 da Tabela 6 (a) indica que G(Σ_1) tem um arco a partir do nó rotulado com *ResearchInstitution* para o nó *Agent*. Por outro lado, as linhas 2 e 3 da Tabela 6 (b) representam o arco do nó *Agent* para o nó ¬ *ResearchInstitution* e o arco do nó *Agent* para o nó *¬(>=2 homepageProject)*.

Neste exemplo específico, destacam-se algumas linhas da Tabela 6(c), que descrevem as restrições de inclusão resultantes da operação de união:

- Linha 1 de G(Σ_3) representa um ⊥-nó obtido após a execução do algoritmo **Gerador de Restrições** que considera as restrições a partir da união das restrições *ResearchInstitution* ⊑ *Agent* e *ResearchInstitution* ⊑ *¬Agent*, logo temos: *ResearchInstitution* ⊑ ⊥
- Linhas 2 e 3 de G(Σ_3) representam a união dos nós correspondentes às restrições de expressões que aparecem em G(Σ_1) e G(Σ_2) , logo temos como resultado as seguintes restrições:*Agent* ⊑ ¬ *ResearchInstitution* e *Agent* ⊑ *¬(>=2 homepage)*.
- As demais linhas mostram a união das restrições que possuem termos em comum. Destaca-se a linha 7, que possui um ⊥-nó rotulado com *(>=2 homepage)* resultando na seguinte restrição: *(>=2 homepage)* ⊑ ⊥.

Tabela 5. União de trechos das Ontologias *Research Institution* 1 e *Research Institution* 2.

	(a) G(Σ_1)		(b) G(Σ_2)		(c) G(Σ_3)	
1	ResearchInstitution	Agent	ResearchInstitution	¬Agent	ResearchInstitution	⊥
2	Agent	¬ (>=2 homepage)	Agent	¬ResearchInstitution	Agent	¬ResearchInstitution
3			Agent	¬ (>=2 homepage)	Agent	¬ (>=2 homepage)
5	(>=1 homepageProject)	Project	(>=1 homepageProject)	Project	(>=1 homepageProject)	Project
6	(>=1 homepageProject)	ResearchInstitution	(>=1 homepageProject)	ResearchInstitution	(>=1 homepageProject)	ResearchInstitution
7	(>=2 homepage)		(>=homepage)	⊥	(>=2 homepage)	⊥

5.4. INTERSEÇÃO DE ONTOLOGIAS *EXTRALITE*

A Web de dados tem crescido exponencialmente, várias aplicações de diversos domínios têm aplicado os princípios do *Linked Data* para publicação de seus dados de modo estruturado e, dessa forma, esses dados são descritos no modelo de dados RDF e representados por ontologias. Na Web de dados, geralmente, as aplicações de um mesmo domínio compartilham dados, no entanto, muitas informações sobrepostas podem ser encontradas. Por exemplo, duas fontes de dados do domínio de Pesquisas Acadêmicas são descritas por ontologias que usam diferentes identificadores URIs para o mesmo objeto do mundo real (o objeto *Article* tem um determinado identificador na ontologia da DBLP[18], que é diferente do identificador da Ontologia da plataforma Lattes[19]).

Desse modo, é possível perceber que a falta de conhecimento sobre essas informações

[18] http://www.informatik.uni-trier.de/~ley/db/

[19] http://lattes.cnpq.br/

sobrepostas inviabiliza o desenvolvimento de mecanismos de integração de dados e de interoperabilidade entre tais fontes de dados. Visando auxiliar a descoberta de informações sobrepostas entre as ontologias ou a elaboração de novas ontologias, considerando os vocabulários e restrições já publicados na Web de dados, faz-se necessário o desenvolvimento de um método para comparar os vocabulários e as restrições de duas ontologias quaisquer de modo a obter o que existe em comum entre elas, ou seja, um método que gere a interseção entre as duas ontologias.

Esta seção expõe o operador algébrico de interseção de ontologias cujo objetivo é apresentar um conjunto de vocabulários (V_3) e de restrições (Σ_3) que representa a semântica da interseção entre duas ontologias *Extralite* $O_1=(V_1,\Sigma_1)$ e $O_2=(V_2,\Sigma_2)$. A partir desse conjunto de vocabulários (V_3) e de restrições (Σ_3), uma nova ontologia pode ser gerada.
Primeiramente, esse algoritmo identifica as expressões idênticas entre duas ontologias, sem considerar os seus identificadores (URIs) e, assim, cria e inicializa o grafo $H=(\gamma,\delta,\kappa)$, onde $\gamma=\gamma_1\cap\gamma_2$, $\delta=\varnothing$ e $\kappa=\kappa_1\cap\kappa_2$. Destaca-se que γ representa nós que rotulam a descrição de conceitos e expressões das restrições, δ possui arcos que representam as restrições de inclusão e κ descreve um conjunto de descrições de conceitos de um determinado nó. Em seguida, gera os fechos transitivos dos grafos de restrições Σ_1 e Σ_2, $G^*(\Sigma_1)=(\gamma^*_1, \delta^*_1, \kappa^*_1)$ e $G^*(\Sigma_2)=(\gamma^*_2, \delta^*_2, \kappa^*_2)$, respectivamente. As arestas do grafo H são construídas a partir das arestas dos fechos $G^*(\Sigma_1)$ e $G^*(\Sigma_2)$. Portanto, se existem as seguintes arestas em $G^*(\Sigma_1)$ e $G^*(\Sigma_2)$: (M_1,N_1) do $G^*(\Sigma_1)$ e (M_2,N_2) do $G^*(\Sigma_2)$, bem como, as expressões **e** e **f,** tal que, M_1 e M_2 são ambos rotulados com **e**, assim como, N_1 e N_2 são ambos rotulados com **f**, então, crie em H um arco $(M_1,N_1)\in\gamma$. Além disso, as expressões únicas dos fechos transitivos $\kappa(M_1)=\kappa^*_1(M_1)\cap\kappa^*_2(M_2)$ e $\kappa(N_1)=\kappa^*_1(N_1)\cap\kappa^*_2(N_2)$ são concatenadas em uma única expressão. Finalmente, o grafo H é passado como parâmetro para o algoritmo **Gerador de Restrições** com a finalidade de obter a nova ontologia $O_3=(V_1\cap V_2,\Sigma_3)$.

É importante evidenciar que ao gerar a nova ontologia, é necessária a intervenção do usuário para auxiliar na escolha do URI que identificará os conceitos idênticos das duas ontologias. Caso não seja escolhido o URI, então um dos URIs das ontologias envolvidas será escolhido de forma aleatória.

A Figura 5.7 apresenta o algoritmo i**nterseção** de Ontologias *Extralite*.

Interseção (V_1, Σ_1 , V_2, Σ_2 ; Σ_3)

Entrada: Duas ontologias *Extralite* $O_1=(V_1, \Sigma_1)$ e $O_2=(V_2, \Sigma_2)$;
Saída: Um conjunto de restrições Σ_3 ;
início
1. **Construa** $G(\Sigma_1) = (\gamma_1,\delta_1,\kappa_1)$ e $G(\Sigma_2)=(\gamma_2,\delta_2,\kappa_2)$; /* construir grafos para Σ_1 e Σ_2; */
2. **Construa** $G^*(\Sigma_1)=(\gamma^*_1,\delta^*_1,\kappa^*_1)$ e $G^*(\Sigma_2)=(\gamma^*_2,\delta^*_2,\kappa^*_2)$; /* fecho transitivo */
3. **Inicialize** $H=(\gamma,\delta,\kappa)$, onde $\gamma=\gamma_1\cap\gamma_2$, $\delta=\varnothing$ e $\kappa=\kappa_1\cap\kappa_2$;
4. **para** cada par de arcos (M_1,N_1) de $G^*(\Sigma_1)$ e (M_2,N_2) de $G^*(\Sigma_2)$ **tal que**
6. M_1 e M_2 são ambos rotulados com **e** e
7. N_1 e N_2 são ambos rotulados com **f** faça
8. adicione um arco (M_1, N_1) para H com
9. $\kappa(M_1)=\kappa^*_1(M_1)\cap\kappa^*_2(M_2)$ e
10. $\kappa(N_1)=\kappa^*_1(N_1)\cap\kappa^*_2(N_2)$;
11. **Gerador de Restrições**(H , Σ_3);
12. **retorne** Σ_3

Fim

Figura 5.7. Algoritmo Interseção de Ontologias *Extralite*.

Com a finalidade de apresentar a aplicabilidade da operação de interseção entre ontologias que descrevem fontes de dados de um mesmo domínio, foram escolhidas duas fontes de dados do domínio de Pesquisa Científica: DBLP e Plataforma *Lattes*.

DBLP *(Digital Bibliographic and Logic Programming*) armazena um grande número de descrições bibliográficas de diversas revistas da ciência da computação, tais como, revistas de conferências como VLDB *(Very Large Data Bases*), IEEE *(Institute of Electrical and Eletronics Engineers*) e ACM *(Association Computing Machinery*). Além disso, a base de dados do DBLP registra mais de meio milhão de artigos e várias centenas de *"links*" para as *"homepages*" de cientistas da computação.

A Plataforma *Lattes* é uma plataforma que integra as bases de dados de currículos, grupos de pesquisa e instituições das áreas de Ciência e Tecnologia, atuando no Brasil, em um único sistema de informações. Foi desenvolvida com a finalidade de facilitar as ações de planejamento, gestão e operacionalização do fomento à pesquisa.

A Tabela 7 apresenta algumas das restrições de inclusão do grafo de restrições G(Σ_1) derivadas da ontologia da DBLP (Tabela 7(a)) e outras restrições de inclusão do grafo de restrições G(Σ_2) derivadas da ontologia da plataforma *Lattes* (Tabela 7(b)).

A primeira coluna da Tabela 7 apresenta o termo do lado esquerdo da inclusão, por outro lado, a segunda coluna apresenta o termo do lado direito da inclusão. Por exemplo, a linha 2 da Tabela 7(a) indica que G(Σ_1) tem um arco a partir do nó rotulado com *Conference* para o nó *Event* (*Conference* $\sqsubseteq$ *Event*). A Tabela 7 (c) mostra as restrições resultantes da operação de interseção entre as referidas ontologias. Perceba que apenas uma das colunas está preenchida e, portanto, entende-se que neste caso a linha da tabela não apresenta uma inclusão, mas apenas uma expressão do grafo. Portanto, é possível verificar que o resultado da interseção a partir do algoritmo apresentado na Figura 5.7 releva apenas a interseção dos termos idênticos dos vocabulários dessas duas ontologias, tais como: *Article* e *Proceedings*, isso se deve a execução da linha 3 desse algoritmo. É possível verificar a partir das restrições de inclusão da Tabela 7 (a) e Tabela 7 (b) que não existem arcos com expressões em comum entre tais restrições de inclusão e, portanto, nenhuma restrição de inclusão é apresentada na Tabela 7 (c).

Tabela 6. Interseção das Ontologias DBLP e *Lattes*.

	(a) G(Σ_1)		(b) G(Σ_2)		(c) G(Σ_3)	
1	Article	Publication	Article	Document	Article	
2	Conference	Event	Book	Document	Proceedings	
3	ConferencePaper	Article	Collection	Document		
5	Continent	Place	Phdthesis	Document		
6	Proceedings	Publication	Proceedings	Document		
7	Professor	Person	Series	Document		

5.5. DIFERENÇA DE ONTOLOGIAS *EXTRALITE*

Inevitavelmente as ontologias evoluirão ou serão atualizadas ao longo do tempo, seja devido a mudanças no domínio que elas representam ou porque foram construídas de forma colaborativa e, portanto, precisam ser atualizadas para representar um entendimento comum aos diversos usuários.

A detecção de diferenças, na maioria dos casos, compara duas versões de uma mesma

ontologia, identificando as alterações existentes entre elas. As diferenças podem ser classificadas como simples ou complexas, segundo (M. Klein, 2004). As diferenças simples são aquelas que não afetam a estrutura da ontologia, pois consideram apenas a alteração de nomes de classes e propriedades ou tipos de dados. Por outro lado, diferenças complexas incluem modificações na hierarquia de classes ou em outras restrições, como disjunção e cardinalidade.

O processo de verificação da diferença entre duas ontologias é imprescindível para lidar com o gerenciamento de múltiplas ontologias. Em particular, o procedimento adotado neste trabalho é bastante abrangente e não se preocupa em representar apenas as alterações entre duas versões de uma mesma ontologia, mas procura elencar os vocabulários e as restrições que não são comuns entre duas ontologias quaisquer e independentemente do domínio que elas pertençam. Outra particularidade desse método consiste em detectar as diferenças considerando tanto os vocabulários quanto as restrições.

Esta seção apresenta o operador algébrico de diferença entre ontologias *Extralite* cujo objetivo é apresentar um conjunto de vocabulários (V_3) e de restrições (Σ_3) que representa a semântica da diferença entre ontologias *Extralite* $O_1=(V_1,\Sigma_1)$ e $O_2=(V_2,\Sigma_2)$, tal que $O_3=(V_1-V_2,\Sigma_3)$, ou seja, a ontologia resultante apresenta todos os termos do vocabulário e as restrições que estão na ontologia O_1 , mas não podem ser encontradas em O_2.

O algoritmo **Diferença**, primeiramente, gera os fechos transitivos $G^*(\Sigma_1)=(\gamma^*_1,\delta^*_1,\kappa^*_1)$ e $G^*(\Sigma_2)=(\gamma^*_2,\delta^*_2,\kappa^*_2)$, onde γ representa nós que rotulam a descrição de conceitos e expressões das restrições, δ possui arcos que representam as restrições de inclusão e κ descreve um conjunto de descrições de conceitos de um determinado nó. Em seguida, inicializa o grafo $H=(\gamma,\delta,\kappa)$, tal que, $\gamma=\gamma_1-\gamma_2$ e $\kappa=\kappa_1-\kappa_2$. Posteriormente, a partir dos fechos transitivos constrói o grafo H, tal que, existe um arco $(M_1,N_1)\in\gamma$, se e somente se, para cada par de expressões ***e*** e ***f*** , tal que, M_1 é rotulado com ***e*** ;e N_1 é rotulado com ***f***, não existe arco (M_2,N_2) de $G^*(\Sigma_2)$, tal que, M_2 é também rotulado com ***e*** ;e N_2 é também rotulado com ***f***. $\kappa(M_1)=\kappa^*1(M_1)$ e $\kappa(N_1)=\kappa^*1(N_1)$. Finalmente, o grafo *H* é passado como parâmetro para o algoritmo **Gerador de Restrições** com a finalidade de gerar um conjunto de restrições Σ_3, tal que $O_3=(V_1-V_2,\Sigma_3)$ é equivalente a diferença de O_1 e O_2.

Diferença(V1, Σ_1 , V2, Σ_2 ; Σ_3)

Entrada: Duas ontologias *Extralite* O1=(V1, Σ_1) e O2=(V2, Σ_2);
Saída: Um conjunto de restrições Σ_3 ;
início

1. **Construa** $G(\Sigma_1) = (\gamma_1,\delta_1,\kappa_1)$ e $G(\Sigma_2)=(\gamma_2,\delta_2,\kappa_2)$; /* construir grafos para Σ_1 e Σ_2; */
2. C**onstrua** $G^*(\Sigma_1)=(\gamma^*_1,\delta^*_1,\kappa^*_1)$ e $G^*(\Sigma_2)=(\gamma^*_2,\delta^*_2,\kappa^*_2)$; /* fecho transitivo */
3. I**nicialize** $H=(\gamma,\delta,\kappa)$, onde $\gamma=\gamma_1-\gamma_2$, $\delta=\varnothing$ e $\kappa=\kappa_1-\kappa_2$;
4. **para** cada arco (M_1,N_1) de $G^*(\Sigma_1)$ **tal que**
5. qualquer par das expressões ***e*** e ***f*** , onde M_1 seja rotulado com ***e*** e N_1 é rotulado com ***f***, não existe um arco (M_2,N_2) de $G^*(\Sigma_2)$, onde M_2 é também rotulado com ***e*** e N_2 é também rotulado com **f faça**
6. $\kappa(M_1)=\kappa^*_1(M_1)$ e $\kappa(N_1)=\kappa^*_1(N_1)$;
7. **Gerador de Restrições**(H, Σ_3);
8. **retorne** Σ_3;
9. **fim**

Figura 5. 8. Algoritmo Diferença de Ontologias *Extralite*.

Com a finalidade de demonstrar a aplicabilidade dessa operação, considere o cenário em que o especialista do domínio, que adota os princípios d*o Linked Data*, tenha utilizado a primeira versão da ontologia FOAF lançada no dia 01 de janeiro de 2010. Posteriormente, uma nova versão da ontologia FOAF foi lançada no dia 09 de agosto de 2010. Então, se faz necessário verificar o que

foi alterado entre as versões dessas ontologias a partir da aplicação da operação de diferença entre elas.

Diante desse cenário, observe o exemplo da aplicação do algoritmo de diferença de ontologias *Extralite*. A Tabela 8 representa algumas das restrições de inclusão da ontologia FOAF lançada no dia 01 de janeiro de 2010 (Tabela 8 (a)), da ontologia FOAF lançada no dia 09 de agosto de 2010 (Tabela 8 (b)) e da ontologia resultante da diferença entre essas ontologias (Tabela 8 (c)). A Tabela 8 representa o grafo de restrições, onde as linhas com apenas uma coluna preenchida simbolizam uma classe ou propriedade atômica. Por outro lado, quando as duas colunas estiverem preenchidas simbolizam que existe uma restrição de inclusão, onde a primeira coluna apresenta o termo do lado esquerdo e a segunda coluna apresenta o termo do lado direito dessa restrição. Neste exemplo, ao aplicar o algoritmo de diferença (Figura 5.8), considerando algumas restrições conforme a Tabela 8 (a) e a Tabela 8 (b), é possível perceber que os termos do vocabulário e as restrição resultantes da diferença entre G(Σ_1) e G(Σ_2) são: *Concept*, *focus* e *Image* $\sqsubseteq$ *Document*, pois são termos do vocabulário e restrições exclusivos de G(Σ_1) , conforme a Tabela 8 (a).

Tabela 7. Diferença de duas versões da Ontologia FOAF.

	(a) G(Σ_1)		(b) G(Σ_2)		(c) G(Σ_3)	
1	Agent		Agent	~Document		
2	Concept		Document	~Agent	Concept	
3	Document	~Project	Document	~Project		
4	Document	~Organization	Document	~Person		
5			Document	~Organization		
6	Group	Agent	Group	Agent		
7	Image	Document	Image		Image	Document
8	focus				focus	

5.6. PROJEÇÃO SOBRE UMA ONTOLOGIA *EXTRALITE*

As boas práticas para modelagem de aplicações que seguem os princípios do *Linked Data* recomendam o reuso de ontologias conhecidas e consolidadas para a elaboração de novas ontologias. Contudo, construir uma nova ontologia utilizando ontologias largamente usadas e consolidadas não é uma tarefa trivial, visto que, para tanto o especialista do domínio deve entender sobre as ontologias existentes, pesquisar os termos do seu interesse nos vocabulários das extensas ontologias e, por fim, retirar apenas um subconjunto dessas ontologias. Desse modo, todo esse processo desencoraja o reuso manual de ontologias existentes e conduz o especialista a escolher outro processo, também, oneroso que consiste na elaboração de ontologias a partir “do nada”, ou em utilizar apenas a importação de *namespaces*.

Em resumo, o processo de reuso de ontologias envolve dois problemas: (1) seleção de um conjunto de termos a partir do vocabulário das ontologias; e (2) uso das restrições derivadas das ontologias existentes e a propagação dessas restrições ao conjunto de termos selecionados (Casanova *et al.*, 2012).

Visando lidar com o processo de reuso de ontologias, este trabalho apresenta o operador algébrico de projeção cuja finalidade consiste em permitir ao especialista do domínio extrair um fragmento da ontologia relacionada a um termo ou a um conjunto de termos selecionados e, além disso, analisar as restrições relacionadas aos termos selecionados, que se aplicam ao fragmento da ontologia.

As principais vantagens do operador de projeção consistem em automatizar a onerosa tarefa do especialista do domínio na formalização de novas ontologias e aumentar a qualidade das novas ontologias, pois, são reusados conceitos altamente consolidados. Além disso, quando as ontologias compartilham os seus termos através do reuso é possível obter mapeamentos mais simples entre as ontologias, o que facilita o processo de integração e manutenção de múltiplas ontologias, isto é, favorece o gerenciamento de ontologias.

A Figura 5.9 apresenta o algoritmo de Projeção sobre ontologias *Extralite*.

Projeção (V1, Σ_1 , W ; Σ_2)

Entrada: Um vocabulário V_1;
Um conjunto Σ_1 das restrições normalizadas sobre V_1;
Um subconjunto W de V_1;

Saída: Um conjunto de restrições Σ_2 ;
início

1. **Construa** G(Σ_1), o grafo de restrições para Σ_1;
2. **Construa** G*(Σ_1), o fecho transitivo de G(Σ_1);
3. **Construa** um grafo de restrições H modificando G*(Σ_1), como segue:
4. **Remova** todos rótulos (ou seja, expressões que rotulam nó(s)) que usa um conceito atômico ou um papel atômico que não estão em $\boldsymbol{W}$;
5. **Remova** todos os nós que não têm rótulos;
6. **Gerador de Restrições**(H, Σ_2);
7. **retorne** Σ_2;
8. **fim**

Figura 5.9. Algoritmo de Projeção sobre ontologias *Extralite*.

Considere que $O_1=(V_1,\Sigma_1)$ seja uma ontologia *Extralite* e $\boldsymbol{W}$ seja um subconjunto de V_1. O algoritmo de Projeção gera Σ_2,tal que, O_2=(W, Σ_2) é equivalente à projeção de $O_1=(V_1,\Sigma_1)$ sobre W. Primeiramente, esse algoritmo constrói o fecho transitivo G*(Σ_1) do grafo de restrições G(Σ_1). Então, ele usa G*(Σ_1) para criar um grafo H descartando todas as expressões que rotulam os nós de G*(Σ_1) e que não envolvem somente símbolos em W. Finalmente, o grafo H é passado como parâmetro para o algoritmo **Gerador de Restrições** com a finalidade de gerar um conjunto de restrições Σ_2 que é equivalente ao conjunto de restrições da ontologia projetada.

Com a finalidade de exemplificar a operação algébrica de projeção, considere o seguinte cenário: um especialista do domínio de Música utiliza dados da ontologia DBpedia (Figura 5.10) a fim de criar uma ontologia específica para a sua aplicação, denominada Ontologia de Música.

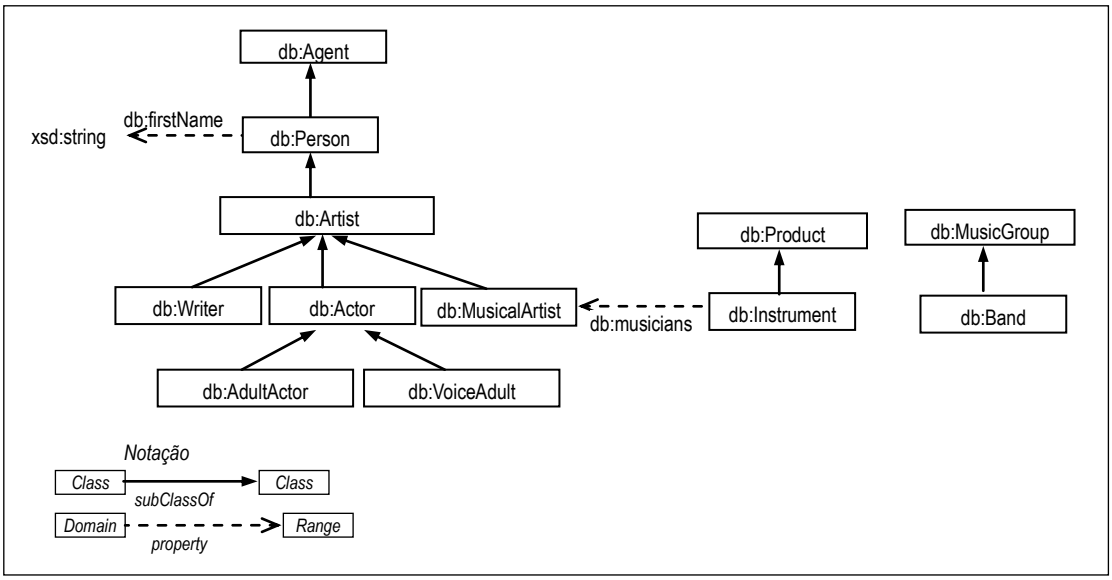

Figura 5.10. Trecho da Ontologia DBpedia

Posteriormente, a partir da ontologia do DBpedia, o especialista do domínio seleciona alguns conceitos relacionados ao domínio de Música de modo a elaborar uma nova ontologia específica. Essa ontologia $O_1 = (V_1, \Sigma_1)$ é obtida a partir da projeção dos termos de V_1 sobre a ontologia DBpedia, conforme a Figura 5.10.

V_1 = {*db:MusicGroup, db:Band, db:MusicalArtist, db:Instrument, db:Person, db:Artist, db:firstName, db:musicians, xsd:string*}

Σ_1 é o conjunto de restrições que captura a semântica dos termos em V_1 e contém as seguintes restrições:

Σ_1 = {*db:Artist* $\sqsubseteq$ *db:Person*, *db: MusicalArtist* $\sqsubseteq$ *db:Artist*, *db:Band* $\sqsubseteq$ *db:MusicGroup*, ($\geq$1 *db:firstName*) $\sqsubseteq$ *db:Person*), ($\geq$1 *db: firstName*$^-$) $\sqsubseteq$ *xsd:string* , ($\geq$1 *db:musicians* $^-$) $\sqsubseteq$ *db: MusicalArtist* , ($\geq$1 *db:musicians*) $\sqsubseteq$ *db:Instrument*, *db:banda* $\sqsubseteq$ *db:MusicGroup* }

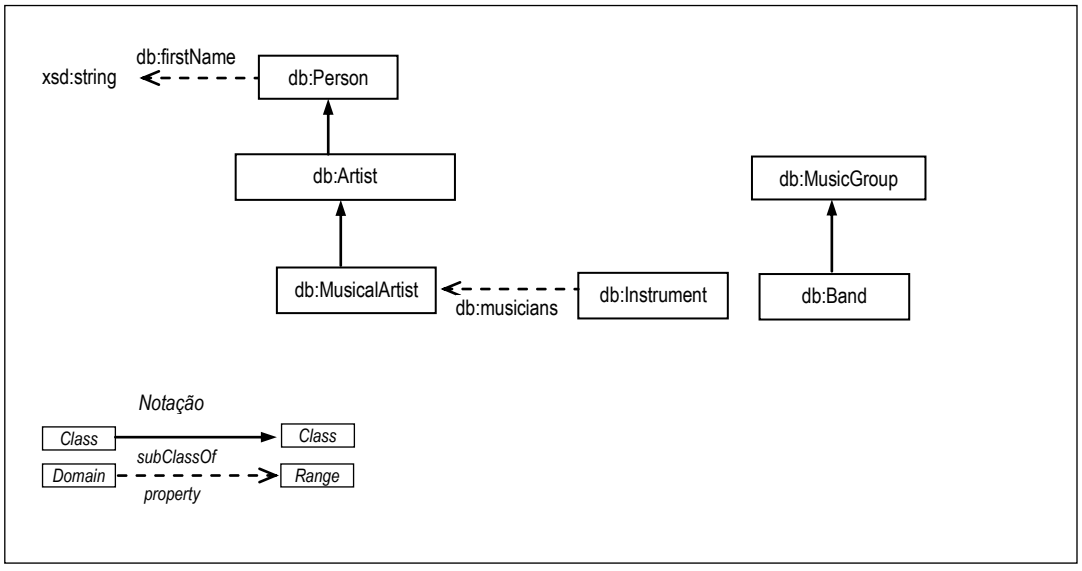

Figura 5. 11. Fragmento do DBpedia considerando o domínio de Música

5.7. CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os principais algoritmos utilizados para a implementação das operações de **união**, **interseção**, **diferença** e **projeção** sobre as ontologias construídas a partir do formalismo da Lógica Descritiva *Lite* (DL-*Lite*). A partir da execução essas operações é possível obter novas ontologias.

Essa operações buscam responder a alguns desafios relacionados ao problema de gerenciamento de múltiplas ontologias.

A **união de ontologias *Extralite*** possibilita a elaboração de uma nova ontologia, que representa a integração e a fusão entre termos de duas ontologias; além disso, gera os mapeamentos de equivalência entre termos de diferentes ontologias usados por fontes de dados para representar termos idênticos no mundo real e, portanto, contribui para tornar transparente para o especialista do domínio a integração entre duas fontes de dados que seguem os princípios do *Linked Data*.

A **interseção de ontologias *Extralite*** auxilia na descoberta de informações sobrepostas entre as ontologias ou na elaboração de novas ontologias considerando os vocabulários e restrições já publicados na Web de dados.

A **diferença de ontologias *Extralite*** possibilita identificar a evolução entre duas versões de uma mesma ontologia ou a incompatibilidade entre duas ontologias quaisquer.

A **projeção em uma ontologia *Extralite*** automatiza a onerosa tarefa do especialista do domínio na formalização de novas ontologias e aumenta a qualidade das novas ontologias, pois, são reusados conceitos altamente consolidados. Além disso, quando as ontologias compartilham os seus termos através do reuso é possível obter mapeamentos mais simples entre as ontologias e, portanto, tal operação facilita o processo de integração e manutenção de múltiplas ontologias favorecendo, portanto, ao gerenciamento de ontologias.

CAPÍTULO 6

OntologyManagementTool – Uma Ferramenta de Gerenciamento de Ontologias

6.1. INTRODUÇÃO

Os estudos obtidos na revisão bibliográfica, descritos no capítulo 3, mostraram que existe uma grande variedade de ferramentas disponíveis com diferentes enfoques e processos a fim de proporcionar o gerenciamento de ontologias. Entretanto, poucos trabalhos preocupam-se em auxiliar o especialista do domínio na elaboração de uma ontologia que represente um entendimento correto sobre a semântica das ontologias envolvidas, visto que, para isso faz-se necessário considerar as restrições lógicas das ontologias originais e propagar essas restrições para as novas ontologias.

Além disso, a necessidade de utilizar várias ferramentas durante o processo de gerenciamento de ontologias aumenta o trabalho manual que deve ser realizado pelo especialista de domínio. Sendo assim, na tentativa de oferecer algumas funcionalidades diferenciadas e de modo integrado às funcionalidades tradicionais de gerenciamento de ontologias, foi desenvolvido um protótipo, chamado ***OntologyManagementTool***.

É importante evidenciar que essa ferramenta ***OntologyManagementTool*** também facilita o processo de reuso das ontologias, proporcionando ao especialista de domínio um mecanismo automatizado para realizar esse gerenciamento de forma mais eficaz, considerando o reuso semântico.

Este capítulo está organizado da seguinte forma: a seção 6.1 apresenta a arquitetura da ferramenta ***OntologyManagementTool***. A seção 6.2 descreve um exemplo de uso da ferramenta ***OntologyManagementTool***. A seção 6.3 apresenta os experimentos e os arquivos gerados pela operação de Interseção de ontologias. Por fim, a seção 6.4 expõe as considerações finais.

6.2. FERRAMENTA *OntologyManagementTool*

Diante da necessidade de uma ferramenta que atendesse aos requisitos de gerenciamento de ontologias especificados nos capítulos anteriores, foi desenvolvido o protótipo ***OntologyManagementTool***.

A arquitetura desse protótipo é constituída por oito módulos, como pode ser visualizado na Figura 6.1. A seguir, é realizada uma explicação de cada módulo, bem como, a interação entre os mesmos a partir das setas enumeradas.

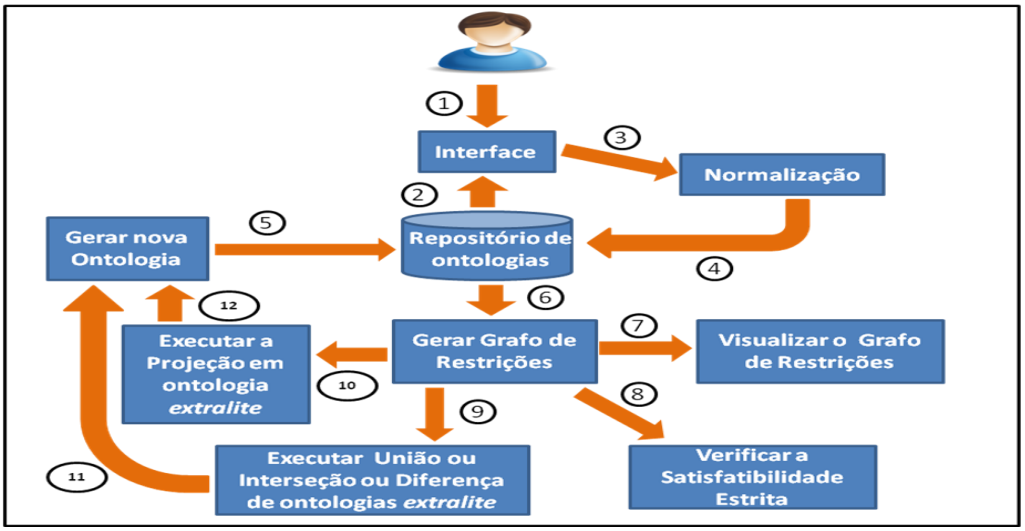

Figura 6.1. Arquitetura da ferramenta *OntologyManagementTool*

É importante destacar a participação imprescindível do **Repositório de ontologias** que possibilita o armazenamento e a recuperação das ontologias envolvidas.

- Módulo ***Interface***: é responsável por disponibilizar o acesso aos demais módulos da ferramenta através da utilização do *framework* Java *Swing*, em ①. Este módulo permite ao usuário selecionar uma ou mais ontologias a partir do repositório de ontologias, como representado em ②, e apresenta na tela as ontologias normalizadas em forma de tabela após a geração do grafo de restrições, conforme indicado na seta ⑥.
- Módulo ***Normalização***: o usuário seleciona uma ou mais ontologia e, posteriormente, essas ontologias são normalizadas conforme as regras definidas na seção 2.5 do Capítulo 2. Por fim, uma nova ontologia normalizada é gerada e armazenada no repositório de ontologias, conforme ④.
- Módulo ***Gerar Grafo de Restrições***: é responsável por utilizar uma ontologia normalizada para a geração de seu respectivo grafo de restrições, seguindo as especificações do capítulo 4, conforme ⑥.
- Módulo ***Verificar a Satisfatibilidade Estrita***: permite ao usuário verificar se existem termos inconsistentes em uma ontologia; ou seja, verifica se existe a satisfatibilidade estrita em uma determinada ontologia, no passo indicado em ⑧, após a obtenção do grafo de restrições dessa ontologia, conforme o algoritmo descrito na seção 4.2 do capítulo 4.
- Módulo ***Executar União ou Interseção ou Diferença de ontologias Extralite***: é responsável por utilizar os grafos de restrições a partir de duas ontologias selecionadas e da opção do usuário por uma dessas operações para aplicação de um dos algoritmos descritos no capítulo 5, conforme indicado em ⑨.
- Módulo ***Executar a Projeção em ontologia Extralite:*** permite ao usuário obter um fragmento de uma ontologia, bem como, as restrições relacionadas aos termos do vocabulário selecionados pelo usuário, conforme indicado em ⑩. Para isso, neste módulo aplica-se o algoritmo de projeção descrito na seção 5.4 do capítulo 5.
- Módulo ***Gerar nova ontologia***: é responsável por gerar uma nova ontologia a partir da execução de um dos seguintes módulos: ***Executar União ou Interseção ou Diferença de ontologias Extralite*** e ***Executar a Projeção em ontologia Extralite***, segundo a indicação em ⑪ e ⑫. Outro aspecto importante deste módulo consiste em tratar a resolução dos identificadores dos termos idênticos. Isso ocorre em algumas operações binárias (união e

interseção), quando existem diferentes URIs que identificam o mesmo objeto do mundo real, e além disso, há mapeamentos de equivalência entre os termos idênticos dos vocabulários da ontologias envolvidas. Neste caso, o módulo oferece ao usuário a escolha de qual ontologia será usada como referência para determinar o URI dos termos idênticos e, portanto, resolve o impasse sobre qual será a URI do termo unificado (fundido). A nova ontologia resultante desse módulo é armazenada no repositório de ontologias, conforme a indicação em ⑤. O acesso, a navegação ao conteúdo das ontologias e a criação de novas ontologias são obtidos pelo framework OWL API[20]. Essa biblioteca foi adotada para geração de ontologias, pois, facilita o desenvolvimento de uma nova ontologia ao tratá-la como um modelo orientado a objetos, permitindo a recuperação dos vocabulários e restrições da ontologia a partir de serviços previamente definidos. Desta forma, uma ontologia OWL pode ser manipulada não só como um conjunto de triplas RDF (Sujeito–Predicado–Objeto), mas também como um conjunto de classes, instâncias e propriedades.

- Módulo ***Visualizar o Grafo de Restrições***: apresenta o conjunto de restrições de uma determinada ontologia na forma de um grafo. Para isso, utiliza a API JUNG[21] (*Java Universal Network*). Essa API possui um conjunto de classes que permite a modelagem, análise e visualização de dados representáveis como grafo.

6.3. EXEMPLO DE USO DA FERRAMENTA *OntologyManagementTool*

O protótipo construído apresenta a tela inicial conforme a Figura 6.2. A seguir, esta seção apresenta cada funcionalidade oferecida ao especialista do domínio e as figuras das telas correspondentes através de um passo-a-passo.

Inicialmente, o especialista do domínio deve manipular as ontologias armazenadas localmente em um repositório específico da ferramenta. O caminho desse repositório pode ser visualizado pela opção *File → Repository Ontology*.

Uma atividade necessária para acessar e executar os módulos descritos na seção 6.1 consiste na seleção de uma ou duas ontologias a partir do repositório. Essa seleção pode ocorrer a partir das seguintes opções da tela inicial da Figura 6.2: "*Load Ontology 1*", "*Load Ontology 2*", "*Load Ontology Projection*". Além disso, essa seleção pode ocorrer de forma implícita quando selecionadas as funcionalidades correspondentes ao módulo "*Ontology Satisfability"* ou "*Visualization of Constraints Graph*".

[20] http://owlapi.sourceforge.net/documentation.html

[21] http://jung.sourceforge.net/

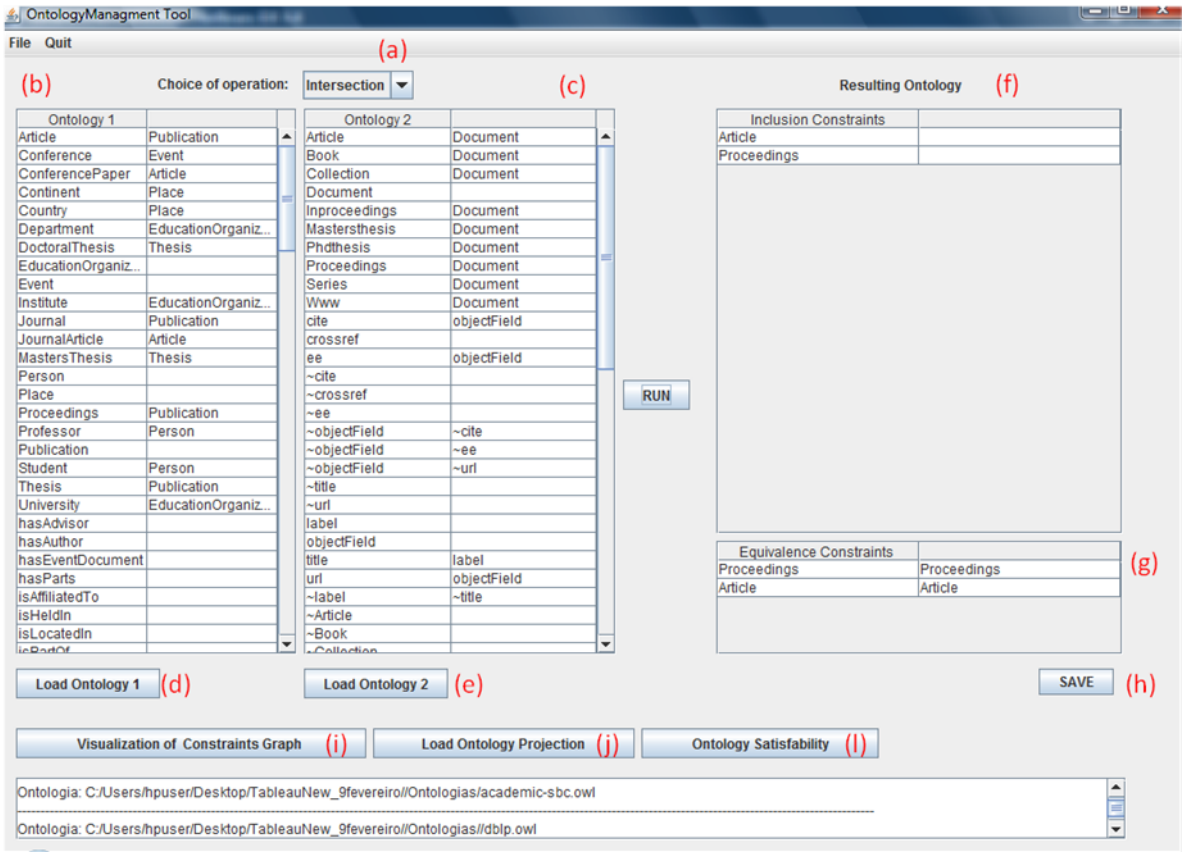

Figura 6.2. Ferramenta *OntologyManagementTool*

Portanto, todas as funcionalidades de gerenciamento de ontologias iniciam com a seleção das ontologias listadas no repositório, conforme a Figura 6.3.

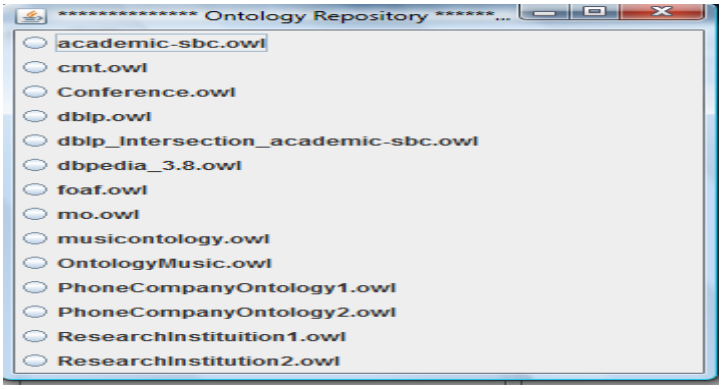

Figura 6.3. Listas de Ontologias disponíveis no repositório

A seguir, a Figura 6.4 retrata o fluxo de atividades relacionadas à execução das operações binárias (União, Interseção e Diferença), ou seja, detalha o módulo ***Executar União ou Interseção ou Diferença de ontologias Extralite***. Este fluxo aplica os algoritmos descritos no Capítulo 5.

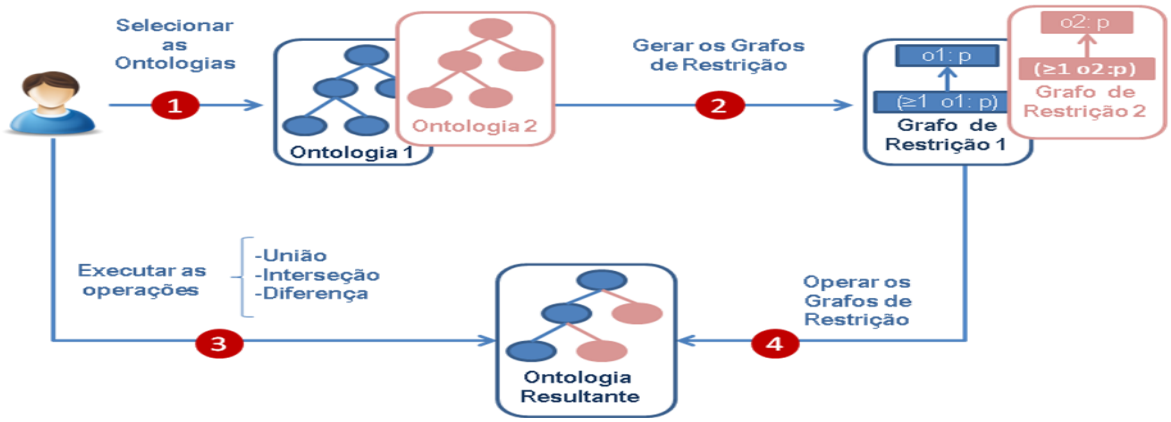

Figura 6.4. Fluxo de execução das operações: União, Interseção ou Diferença

Visando associar a descrição do fluxo da Figura 6.4 com cada componente de interface da ferramenta, cada passo numerado nessa figura foi mapeado a um item que descreve a tela da ferramenta, conforme visto na Figura 6.2.

No passo 1, o especialista do domínio escolhe duas ontologias para uma das operações binárias, a partir das ontologias listadas no repositório da ferramenta. Portanto, uma das ontologias é selecionada a partir da funcionalidade "*Load Ontology* 1", na Figura 6.2 (d), e a outra ontologia a partir da funcionalidade "*Load Ontology* 2", na Figura 6.2 (e).

No passo 2, as ontologias selecionadas são normalizadas, conforme a seção 2.5 do Capítulo 2, e armazenadas no repositório da ferramenta. Em seguida, esses arquivos são utilizados para gerar os grafos de restrições dessas ontologias, conforme explicado no Capítulo 4. Por fim, é possível visualizar esses grafos a partir da interface da ferramenta através de tabelas com duas colunas (A ontologia 1 na Figura 6.2 (b) e a ontologia 2 na Figura 6.2 (c)). Desse modo, cada linha dessas tabelas representa apenas uma expressão do grafo, quando apenas uma coluna da linha é preenchida, ou representa uma inclusão da forma $e \sqsubseteq f$, onde e e f são ambas descrições de expressões do grafo.

No passo 3, um dos operadores binários (União, Interseção ou Diferença) deve ser escolhido pelo especialista do domínio.

Por fim, no passo 4, a ferramenta executa a operação selecionada e gera o grafo de restrições resultante que possui restrições de equivalência e de inclusão, sendo apresentadas por duas tabelas intituladas "*Inclusion Constraints*" e "*Equivalence Constraints*", conforme a Figura 6.2 (f) e (g), respectivamente. A tabela de "*Equivalence Constraints*", semelhantemente à tabela "*Inclusion Constraints*", possui duas colunas, conforme verificado na Figura 6.2 (g), onde cada coluna representa uma expressão, representando uma equivalência da forma $\boldsymbol{e \equiv f}$, onde e e f são ambas expressões do grafo de restrições.

Finalmente, as ontologias resultantes das operações descritas anteriormente podem ser salvas a partir da funcionalidade "*Save*" na Figura 6.2 (h). O arquivo gerado nessa etapa é armazenado no repositório da ferramenta. A Figura 6.5 expõe uma ontologia resultante da operação de interseção das seguintes ontologias *Extralite*: Ontologia da DBLP e Ontologia da Plataforma *Lattes*. O nome do arquivo gerado é resultado da concatenação do nome das ontologias de entrada e a operação utilizada, por exemplo, a interseção das ontologias dblp.owl e academic-sbc.owl terá o arquivo resultante da união denominado de dblp_Intersection_academic-sbc.owl.

Ontologia dblp_Intersection_academic-sbc.owl

```
1:  <?xml version="1.0"?>
2:  <!DOCTYPE rdf:RDF [
      <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
      <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
      <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
      <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
      <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
      <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
      <!ENTITY dblp "http://sw.deri.org/~aharth/2004/07/dblp/dblp.owl#" >
      <!ENTITY academic-sbc "http://www.semanticweb.org/ontologies/2009/10/academic-sbc.owl#" >


3:  <rdf:RDF xmlns=" http://www.semanticweb.org/ontologies/2009/10/academic-sbc.owl #"
      xml:base=" http://www.semanticweb.org/ontologies/2009/10/academic-sbc.owl "
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
4:  <!-- http://www.semanticweb.org/ontologies/2009/10/academic-sbc.owl#Article -->

5:    <owl:Class rdf:about="#Article"/>

6:    <owl:Class rdf:about="#Proceedings"/>

7:    <owl:Class rdf:about="&academic-sbc;Article">
        <owl:equivalentClass rdf:about="& dblp :Article"/>
      </owl:Class>

8:    <owl:Class rdf:about="&academic-sbc;Proceedings">
        <owl:equivalentClass rdf:about="& dblp :#Proceedings"/>
      </owl:Class>

9:  </rdf:RDF>
```

Figura 6.5.Ontologia resultante da operação de interseção das ontologias dblp.owl e academic-sbc.owl

Outro módulo, também, descrito a seguir em forma de passo-a-passo é o ***Executar a Projeção em ontologia Extralite***. A explicação desse módulo segue a mesma organização da descrição do módulo anterior. Assim, primeiramente, é apresentado o seu fluxo de execução, conforme a Figura 6.6. Em seguida, cada componente de interface da ferramenta é mapeado às funcionalidades desse módulo.

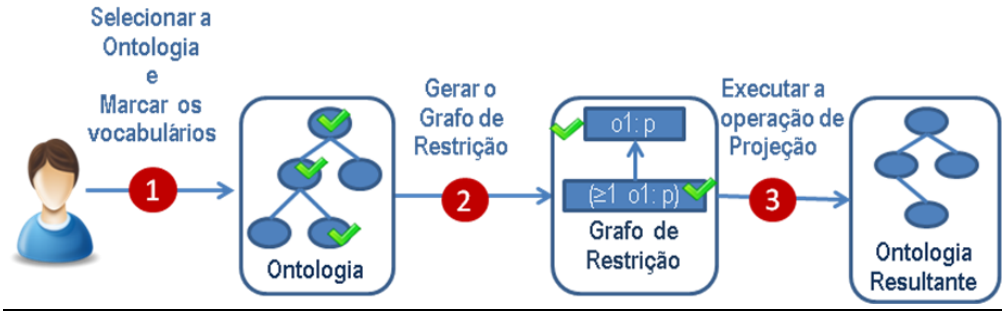

Figura 6.6. Fluxo de execução da operação de Projeção

No passo 1, o especialista do domínio seleciona uma ontologia específica a partir da funcionalidade "*Load Ontology Projection*", conforme a Figura6.7(a). Na tela do protótipo, é carregado o grafo de restrições correspondente à ontologia selecionada em forma de tabela, conforme a Figura 6.7 (b), e essa tabela é nomeada de "*Ontology* 1". Cada linha dessa tabela pode representar tanto uma expressão desse grafo quanto suas restrições de inclusão. O primeiro caso ocorre quando apenas uma coluna da tabela é preenchida, já o segundo caso ocorre quando as duas colunas são preenchidas e, assim, a primeira coluna representa o lado esquerdo da inclusão e a segunda coluna representa o lado direito, por exemplo, a restrição de inclusão $e \sqsubseteq f$, *e* preenche a coluna da esquerda e *f* preenche a coluna da direita. Além disso, o especialista do domínio pode selecionar os termos do vocabulário desta ontologia a partir da tabela denominada "*Vocabulary*", conforme a Figura 6.7 (c). É importante evidenciar que a escolha desses termos é relevante para a obtenção do fragmento da ontologia selecionada a partir da aplicação do algoritmo descrito na seção 5.6 do Capítulo 5.

No passo 2, o grafo de restrições gerado é recortado de modo a excluir todas as restrições que não foram selecionadas pelo especialista do domínio, ou seja, o grafo só contém expressões relacionadas aos termos do vocabulário selecionados pelo usuário. A ontologia resultante é apresentada em 6.7 (d).

Por fim, no passo 3, o grafo de restrições gerado pela execução da operação de Projeção pode ser convertido em uma nova ontologia descrita em OWL através da funcionalidade "*Save*", segundo a Figura 6.7 (e).

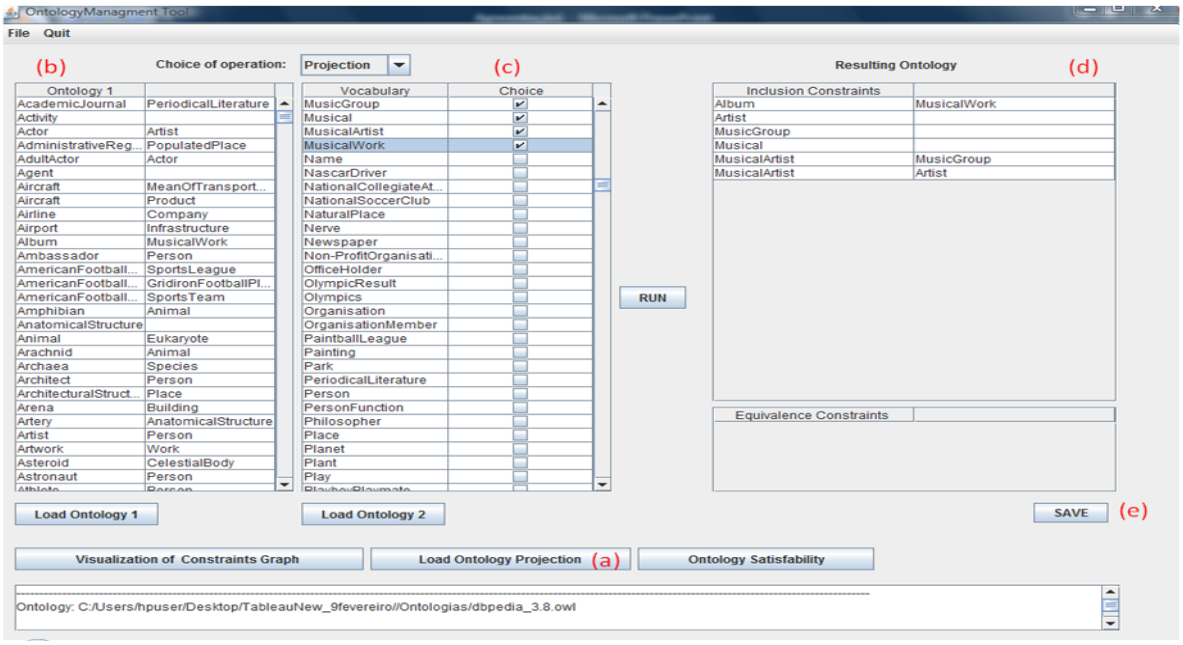

Figura 6.7. A ferramenta *OntologyManagementTool* na execução da operação de Projeção

O Módulo ***Visualizar Grafo de Restrições*** permite ao usuário visualizar o grafo de restrições da ontologia selecionada. Inicialmente, o especialista do domínio seleciona a ontologia a partir da opção "*Visualization of Constraints Graph*", conforme a Figura 6.2 (i). Em seguida, é apresentada uma tela com o grafo ilustrado conforme a Figura 6.8. Cada nó do grafo desenhado em azul representa um nó estritamente satisfatível. Por outro lado, caso o nó esteja desenhado em vermelho significa que ele representa um ⊥-nó ou um⊤-nó. Além disso, é possível visualizar o caminho no grafo que acarretou na definição do ⊥-nó e ⊤-nó, sendo esse caminho destacado em azul.

Para exemplificar esse módulo foi selecionada uma ontologia denominada *PhoneCompanyOntology* 1, cujas expressões (>= 2 *placedBy Call*) e ~(>= 2 *placedBy Call*) representam um ⊤-nó e um ⊥-nó, respectivamente.

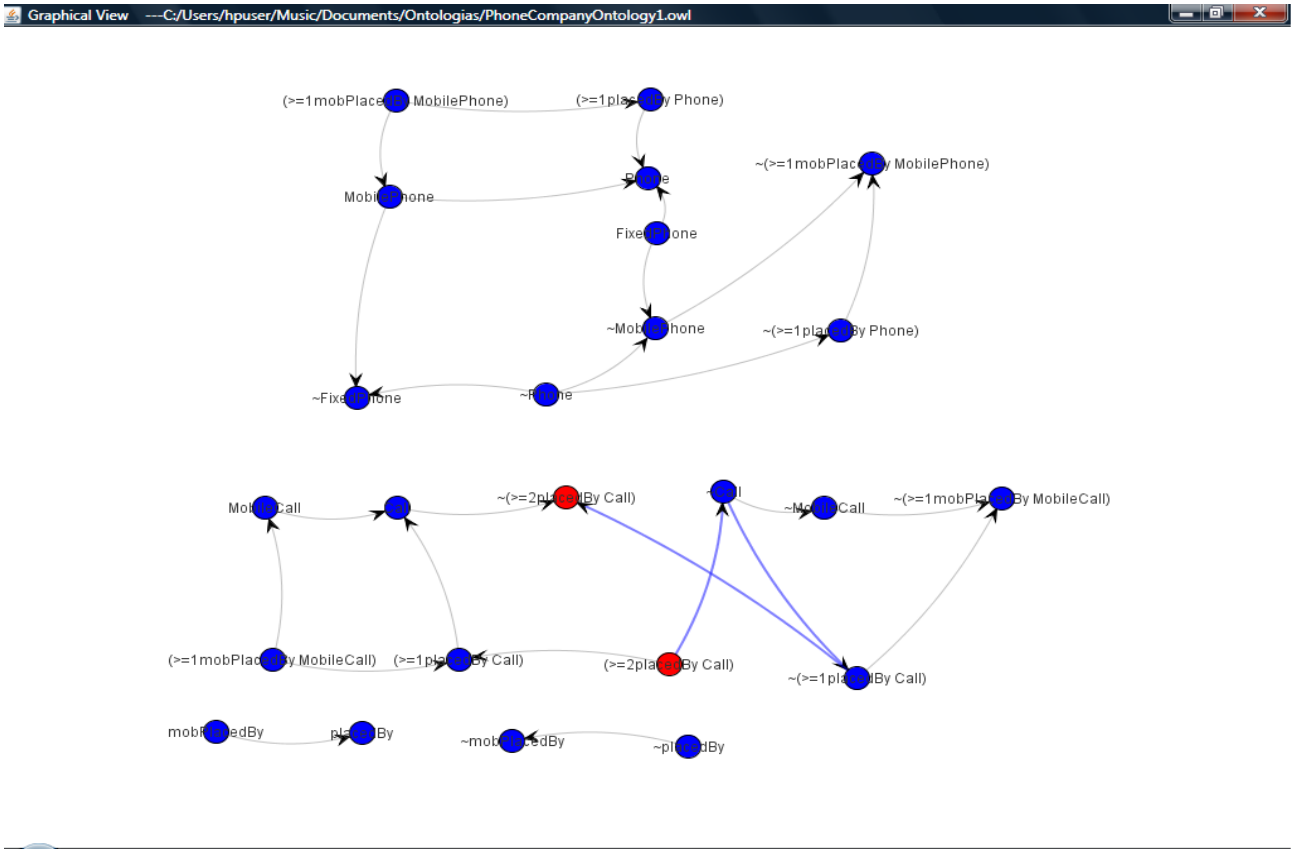

Figura 6.8. Visualização do grafo de restrições

Por fim, o módulo ***Verificar a Satisfatibilidade Estrita*** é apresentado. Inicialmente, uma única ontologia é selecionada, e posteriormente, é executado o algoritmo responsável por apresentar a interpretação das expressões em uma determinada ontologia, conforme o algoritmo de satisfatibilidade estrita, descrito no Teorema 1 do Capítulo 4. Em particular, uma ontologia é definida como estritamente satisfatível quando nenhum dos seus conceitos atômicos (classes e propriedades) possuem uma interpretação vazia, ou seja, não existe nenhuma contradição entre esses conceitos atômicos. Na ferramenta ***OntologyManagementTool***, este módulo pode ser acessado a partir da funcionalidade "*Ontology Satisfatibility*", conforme a Figura 6.2 (l).

É importante destacar que as ontologias resultantes da ferramenta implementada neste trabalho podem ser manipuladas utilizando a ferramenta Protégé (J. Gennari *et al*., 2003), que possui um *plug-in* que permite a manipulação de ontologias na linguagem OWL, e uma interface gráfica para criação de classes, propriedades e restrições.

6.4 EXPERIMENTOS

Com a finalidade de realizar os experimentos foram selecionadas ontologias de diversos domínios, tais como, Ontologia da DBLP[22], Ontologia da plataforma *Lattes*[23], Ontologia da DBpedia[24], Ontologia FOAF[25] (lançada em 01 de janeiro de 2010) e Ontologia FOAF[26] (lançada em 9 de agosto de 2010), Ontologia *Research Institution* 1, Ontologia *Research Institution* 2 e Ontologia *Research Institution* 3.

A eficiência da ferramenta proposta pode ser constatada a partir da análise da complexidade dos principais algoritmos. Essa análise mostrou que a complexidade do processamento das operações algébricas (interseção, diferença e projeção) é quadrática em relação ao número de vértices do grafo de restrições, sendo importante evidenciar que o fator determinante para obtenção dessa complexidade é o procedimento escolhido para lidar com as restrições de inclusão, denominado fecho transitivo. Alguns experimentos foram propostos, visando demonstrar a aplicabilidade da ferramenta. Os detalhes de cada um dos experimentos podem ser verificados, a seguir:

A operação de interseção foi testada utilizando ontologias de domínio sobreposto e, portanto, foram utilizadas a Ontologia da DBLP e a Ontologia da plataforma Lattes.

A operação de união foi testada utilizando Ontologia *Research Institution* 1 e Ontologia *Research Institution* 2, bem como, Ontologia da DBLP e Ontologia da plataforma Lattes.

A operação de diferença foi testada utilizando as ontologias Ontologia FOAF, lançada em 01 de janeiro de 2010 e Ontologia FOAF, lançada em 9 de agosto de 2010.

A operação de projeção foi testada usando a Ontologia da plataforma *Lattes* e a Ontologia da DBpedia. Em particular, a geração do grafo de restrições para ontologia da DBpedia requer um tempo de espera superior a todas as outras ontologias.Isso ocorre porque a Ontologia da DBpedia é muito extensa e seu grafo de restrição possui mais de 5000 nós. Desta forma, a principal dificuldade dos testes foi verificar os resultados esperados em cada passo dos algoritmos sem grafos muito extensos.

A principal dificuldade dos testes foi verificar os resultados esperados em cada passo dos algoritmos em grafos muito extensos.

A Figura 6.9 retrata as classes da Ontologia da DBLP (dblp.owl) e da Ontologia da plataforma Lattes (academic-sbc.owl) através da ferramenta **Protégé**.

[22] http://www.informatik.uni-trier.de/~ley/db/

[23] http://lattes.cnpq.br/

[24] http://dbpedia.org/Downloads38#h224-1

[25] http://xmlns.com/foaf/spec/20100101.html

[26] http://xmlns.com/foaf/spec/20100809.html

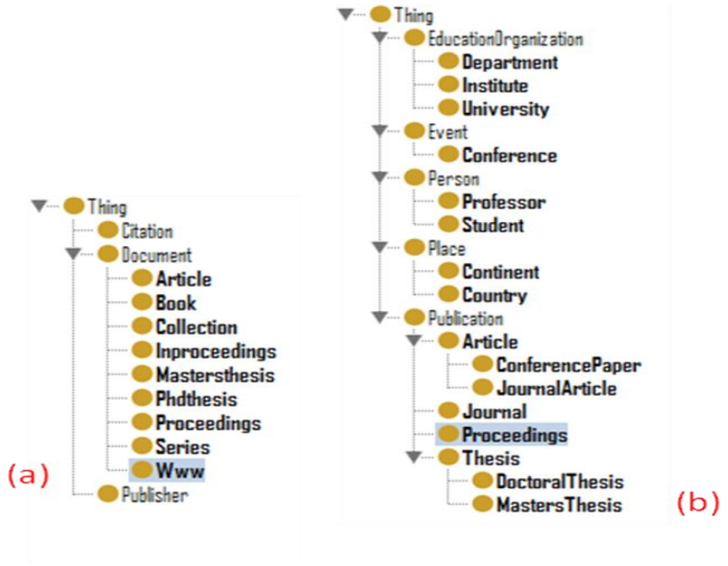

Figura 6.9. As ontologias dblp.owl (a) e academic-sbc.owl (b)

Ao executar a operação de interseção entre as ontologias dblp e academic-sbc, é possível obter a ontologia resultante descrita na Figura 6.10.

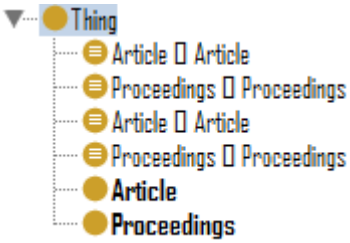

Figura 6.10. A ontologia de interseção entre dblp.owl (a) e academic-sbc.owl (b)

6.5. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a especificação e a implementação de uma ferramenta para gerenciamento de ontologias a fim de auxiliar o especialista do domínio no processo de gerenciamento de ontologia. A ferramenta proposta suporta acesso às ontologias a partir de um repositório local e oferece um conjunto de operações algébricas (união, interseção, diferença e projeção) que tratam as ontologias como teorias lógicas.

A ferramenta ***OntologyManagmentTool*** é a chave fundamental para processo de gerenciamento de ontologias delineado neste trabalho que consiste em um tópico imprescindível para a implantação das tecnologias de *Linked Data* nos projetos de Web de dados. Essa ferramenta utiliza os algoritmos definidos nos Capítulos 4 e 5. Inicialmente, na seção 6.2 foi apresentada uma visão geral da ferramenta ***OntologyManagmentTool*** e informações sobre a sua implementação. Em seguida, a seção 6.3 expõe um exemplo do uso da ferramenta. Posteriormente, a seção 6.4 descreve os experimentos que comprovam a sua viabilidade e aplicabilidade em ontologias que descrevem fontes de dados reais.

CAPÍTULO 7

Conclusões

7.1 CONTRIBUIÇÕES

Nesta dissertação foi apresentado um estudo sobre alguns processos e ferramentas relacionados ao gerenciamento de múltiplas ontologias, tais como: o reuso, o versionamento, a integração e a evolução de ontologias. Existe uma grande variedade de trabalhos disponível com diferentes enfoques que propõe o gerenciamento de ontologias. Entretanto, poucos trabalhos preocupam-se em auxiliar o especialista do domínio na elaboração de uma ontologia que representa um entendimento correto sobre a semântica das ontologias envolvidas, visto que, para isso faz-se necessário considerar as restrições lógicas das ontologias originais e propagá-las para a nova ontologia. Além disso, foi percebido que, nos trabalhos anteriores, há necessidade de se utilizar várias ferramentas durante o processo de gerenciamento de ontologias, o que aumenta o esforço manual a ser despendido pelo especialista do domínio na elaboração de novas ontologias. É importante destacar que algumas ferramentas apresentadas por trabalhos referentes ao tema de gerenciamento de ontologias não foram atualizadas e, portanto, não seguem os padrões atuais das linguagens de descrição das ontologias, como OWL, RDFS e RDF.

Em particular, este trabalho, primeiramente, considera a ontologia segundo um formalismo específico da Lógica Descritiva, denominada de ontologia *Extralite*, conforme descrito no Capítulo 2. Em seguida, no Capítulo 3, foram apresentados e discutidos os principais trabalhos relacionados com as funcionalidades de gerenciamento de ontologias tratadas nesta dissertação. As principais contribuições desse trabalho consistem na implementação dos algoritmos descritos nos Capítulos 4 e 5 que lidam com as ontologias em forma de grafo de restrições; tratam a satisfatibilidade estrita a partir de um determinado grafo de restrições com a finalidade de verificar a existência de contradição entre os termos do vocabulário das ontologias representadas a partir desse grafo; por fim, oferecem a possibilidade de elaborar uma nova ontologia a partir de um conjunto de operações algébricas (projeção, união, interseção, diferença) sobre uma ou duas ontologias.

Para validar os algoritmos descritos e demonstrar aplicabilidade desse trabalho foi desenvolvido uma ferramenta denominada de ***OntologyManagementTool***, conforme descrito no Capítulo 6. Além disso, neste capítulo foram apresentados alguns experimentos com ontologias que descrevem fontes de dados reais.

7.2. LIMITAÇÕES

O desenvolvimento desse trabalho apresentou diversas dificuldades. Uma das mais relevantes está relacionada à implementação da ferramenta ***OntologyManagementTool***, como o aprendizado do *framework* OWL API responsável pela manipulação das ontologias existentes e pela geração de novas ontologias. Um aspecto marcante relacionado a esse framework consiste na ausência de serviços para a manipulação do complemento de relacionamento binário, e assim, foi necessário considerar esse complemento como relacionamento binário cujo nome é prefixado com um til (~) e, portanto, para obter uma restrição de complemento do relacionamento binário em uma determinada ontologia é preciso considerar essa particularidade e utilizar um serviço disponível nesse *framework* para recuperar um relacionamento binário e, em seguida, verificar se o seu nome é prefixado com um til (~).

Houve, além disso, dificuldade relacionada à manipulação dos URIs, considerando os dois possíveis padrões: *hash* URI e *slash* URI. Isso ocorre principalmente em ocasiões em que é necessário obter a descrição dos componentes das ontologias desvinculada do seu URI ou a

recuperação do URI de um determinado vocabulário da ontologia. Por exemplo, na ontologia FOAF, caso seja requerido os conceitos *OnlineGamingAccount* e *Person* desvinculados das URIs, como definida a seguir: http://xmlns.com/foaf/0.1/OnlineGamingAccount e http://www.w3.org/2000/10/swap/pim/contact#Person, respectivamente, faz-se necessário desenvolver *parses*, um com a finalidade de tratar o *hash* URI e outro para lidar com o *slash* URI.

Outra dificuldade encontrada está relacionada com a validação das operações propostas. O processo de validação das operações em ontologias muito extensas com a DBpedia é muito demorado , pois, o grafo de restrição dessa ontologia contém mais de 5.000 nós.

7.3. TRABALHOS FUTUROS

Este trabalho permite diversas extensões relacionadas tanto a pesquisa quanto a implementação, algumas das quais listadas a seguir. Além disso, a ferramenta apresentada requer algumas melhorias.

A ferramenta ***OntologyManagementTool*** poderia provê uma interface mais intuitiva e amigável aos usuários finais. Neste caso, ao invés de apresentar as restrições de inclusão em forma de tabelas seria apresentada uma estrutura em hierarquia, assim, seria mais intuitivo ao usuário o entendimento sobre a relação de hierarquia entre os termos dessa ontologia. Outra melhoria prevista para esta ferramenta consiste em elaborar uma estrutura para o repositório de ontologias de modo que viabilizasse a busca das ontologias a partir de filtros de busca.

Além disso, outros módulos podem ser desenvolvidos, como listados a seguir:

- Módulo de otimização: consiste em eliminar os fechos transitivos redundantes através da utilização de algoritmos que gerem o grafo de equivalência mínima, MEG (*Minimum Equivalent Grafo*) (Hsu, H.T, 1975). Portanto, espera-se que melhore o desempenho das execuções das operações sobre as ontologias, tanto na diminuição do tempo de resposta quanto em relação ao consumo de memória.

- Módulo de remoção de inconsistências: consiste em auxiliar o usuário no processo de remover os termos que não são estritamente satisfatíveis.

- Integração dessa ferramenta ao editor de ontologias Protégé que possibilita a obtenção das vantagens já disponíveis no Protégé, tais como, visualização e manipulação da ontologia, bem como, a utilização de motores de inferência.

REFERÊNCIAS BIBLIOGRÁFICAS

(Antoniou, G. e van Harmelen, F., 2003). Antoniou, G. e van Harmelen, F.. Web Ontology Language: OWL. In S. Staab and R. Studer, editors, Handbook on Ontologies in Information Systems, pages 76–92.(2003)

(Artale, A. et al, 2009) Artale, A.; Calvanese, D.; Kontchakov, R.; Zakharyaschev, M. "The DL-Lite family and relations". J. of Artificial Intelligence Research 36, pp. 1–69,(2009).

(Baader, F.; Nutt, W., 2003) Baader, F.; Nutt, W.. "Basic Description Logics". In: Baader, F.; Calvanese, D.; McGuiness, D.L.; Nardi, D.; Patel-Schneider, P.F. (Eds) The Description Logic Handbook: Theory, Implementation and Applications. Cambridge U. Press, Cambridge, UK. (2003).

(Berners-Lee T. *et al.*, 2005) Berners-Lee, T.; Fielding, R.; Masinter, L. RFC 3986 – Uniform Resource Identifier (URI): Generic Syntax. http://tools.ietf.org/html/rfc3986 (2005).

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

(Berners-Lee, T., 2007) Berners-Lee, T..Semantic Web URIs. Realizado o acesso em fevereiro de 2013, a partir de http://dig.csail.mit.edu/2007/Talks/0108-swuri-tbl/#(1) (2007).

(Berners-Lee, T., 2007) Berners-Lee, T. Linked Data: Design issues. Realizado o acesso em fevereiro de 2013, a partir de http://www.w3.org/DesignIssues/LinkedData.html. (2007)

(Bizer, C. *et al.*, 2009) Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. Int. Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)

(Bizer, C. *et al.*, 2011) Bizer, C.; Jentzsch, A.; Cyganiak, R. State of the LOD Cloud.http://www4.wiwiss.fu-berlin.de/lodcloud/state/. (2011)

(Bleiholder e Naumann, 2006) Bleiholder, J. e Naumann, F..Conflict handling strategies in an integrated information system. In Proceedings of the IJCAI Workshop on Information on the Web (IIWeb), 2006.

(Borgida, A. e Brachman, R.J., 2003) Borgida, A. e Brachman, R.J. "Conceptual modeling with Description Logics". In: Baader, F.; Calvanese, D.; McGuiness, D.L.; Nardi, D.; Patel-Schneider, P.F. (Eds) The Description Logic Handbook: Theory, Implementation and Applications. Cambridge U. Press, UK. (2003)

(Breitman *et al*, 2006) Breitman, K., M. A. Casanova e W. Truszkowski. Semantic web: Concepts, technologies and applications (monografia da NASA em sistemas e engenharia de software). Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 2006.

(Brickley,D., Miller, L., 2010) Brickley,D., Miller, L. Foaf vocabulary specification 0.98. http://xmlns.com/foaf/spec/. (2010)

(Casanova *et al.*, 2011) Casanova, M.A., Breitman, K. K., Furtado, A. L., Vidal, V. M. P., Macêdo, J.A.F.: The Role of Constraints in Linked Data. Proc. 10th International Conference on Ontologies,

DataBases, and Applications of Semantics (ODBASE 2011). Lecture Notes in Computer Science 7045. Springer, Heidelberg, pp. 781-799. (2011)

(Casanova *et al.*, 2012) Casanova, M.A., Macêdo, J.A.F, Sacramento, E.R., Pinheiro, A.M.A, Vidal, V.M.P., Breitman, K.K. e Furtado, A.L.. Operations over Lightweight Ontologies. OTM 2012, Part II, LNCS 7566, pp. 646–663. (2012)

(Das, S. *et al.*, 2012) Das, S., Sundara, S., Cyganiak, R. R2rml: RDB to RDF mapping language. W3C RDB2RDF working group. http://www.w3.org/TR/r2rml/ (2012).

(Gennari J. H. *et al*., 2003) Gennari, J. H., Musen, M. A., Fergerson, R., Grosso, W. E., Crubzy, M., Eriksson, H., Noy, N. F. e Tu, S.W.. The Evolution of Protege: An Environment for Knowledge-Based Systems Development. International Journal of Human-Computer Studies, 58(1):89–123, 2003.

(Goodrich, M.T. e Tamassia., R. 2001) Goodrich, M.T. e Tamassia, R.. Algorithm Design. IE-Wiley. (2001)

(Gruber, T.R. *et al*., 1993) Gruber, T.R. *et al*. A translation approach to portable ontology specifications. Knowledge acquisition, 5: 199-199 (1993).

(Heath, T. e Bizer, C., 2011) Heath, T. e Bizer, C.. Linked Data: Evolving the Web into a Global Data Space. 1st. ed.[S.l.]: Morgan & Claypool, 136 p. ISBN 9781608454303. (2011)

(Hepp M. *et al*., 2008) Ontology Management: Semantic Web, Semantic Web Services, and Business Applications. Semantic Web And Beyond Computing for Human Experience. ISBN 978-0-387-69900-4. (2008)

(Hsu, H.T, 1975) An Algorithm for Finding a Minimal Equivalent Graph of a Digraph. Journal of the Association for Computing Machinery 22(1), 11–16 (1975) .

(Jacobs e Walsh, 2004) Jacobs e Walsh. Architecture of the world wide web. Obtido em março de 2013. http://www.w3.org/TR/webarch/ (2004).

(Jain, P. *et al.*, 2010) Jain, P., Hitzler, P., Sheth, A. P. Flexible Bootstrapping-Based Ontology Alignment In: P. Shvaiko,P., Euzenat, J., Giunchiglia, F., Stuckenschmidt, H., Mao, M., I. Cruz (eds.), Ontology Matching, OM-2010. Proceedings of the 5th International Workshop on Ontology Matching, at ISWC2010, Shanghai, China, November 2010, pp. 136-137.

(Klein, M., 2002) Klein, M. et al. “Ontology versioning and change detection on the web”. In Proceedings of EKAW’02, 197–212, Siguenza, Spain, (2002).

(Klyne,G. *et al.*,2010) Klyne,G.,Carroll,J.J,McBride,B. Resource Description Framework (RDF): Conceitos e sintaxe abstrata. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ (2010).

(Koubková, A. e Koubek, V.,2002). Koubková, A. e Koubek, V.. Algorithms for transitive closure. Information processing letters, 81(6):289–296. (2002)

(Langegger, A.A., 2010) Langegger, A. A Flexible Architecture for Virtual Information Integration based on Semantic Web Concepts. Tese (Doutorado) — J. Kepler University Linz, 2010.

(Lenzerini, 2002) Lenzerini, M. *Data Integration: A Theoretical Perspective*. In Proceedings of ACM Symposium on Principles of Database Systems. (2002).

(Manola, F. e Miller, E., 2004) Manola, F. e Miller, E.. Resource Description Framework (RDF) Model and Syntax Specification, http://www.w3.org/TR/rdf-primer/. (2004)

(Maedche *et al.*, 2003) Maedche et al., "Managing multiple and distributed ontologies on the Semantic WEB", VLBD Journal 12, Springer, Location, pp. 286-302. (2003)

(McGuinness, D. L. *et al* , 2000) D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. The Chimaera Ontology Environment. In Proc. of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence, pages 1123–1124, 2000.

(McGuinness, D.L., 2002) McGuinness, D.L. .Ontologies come of age. Spinning the semantic web: bringing the World Wide Web to its full potential, page 171-192. (2002)

(McGuinness,D.L., Harmelen, F., 2010)McGuinness,D.L., Harmelen, F. OWL Web Ontology Language Overview (OWL). http://www.w3.org/TR/2004/REC-owl-features-20040210/ (2010).

(M. Klein et al., 2002) M. Klein et al. "Ontology versioning and change detection on the web". In Proceedings of EKAW'02, 197–212, Siguenza, Spain. (2002)

(Noy, N. S., 2004) Noy, N.S., Semantic Integration: A Survey Of Ontology-Based Approaches, ACM SIGMOD Record, vol. 33, nº4, p.65-70, 2004

(Noy, N. F. e Musen, M. A., 2000) Noy, N. F. e Musen, M. A. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In Seventeenth International Joint Conference on Artificial Intelligence AAAI/IAAI, pages 450–455. (2000)

(Noy N.F. *et al*, 2004) N. F. Noy, S. Kunnatur, M. Klein, and M. A. Musen. Tracking changes during ontology evolution. In Sheila A. Mcilraith, Dimitris Plexousakis, e Frank van Harmelen. Third International Semantic Web Conference, pages 259–273, Hiroshima, Japan. (2004)

(Noy, N. F. e Musen, M., 2004) Noy, N. F. e Musen, M. A. Specifying Ontology Views by Traversal. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, International Semantic Web Conference, volume 3298 of Lecture Notes in Computer Science, pages 713–725.(2004)

(Peter Haase *et al*, 2003) Peter Haase, York Sure e Denny Vrandei (Institute AIFB, University of Karlsruhe) Ontology Management and Evolution – Survey, Methods and Prototypes (2003).

(Rahm, E. e Bernstein,P.A. 2001) E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Mat- ching. VLDB Journal: Very Large Data Bases, 10(4):334–350.(2001)

(Ramos., J. A., 2001) Ramos, J. A..Mezcla automática de ontologías y catálogos electrónicos. Final

YearProject. Facultad de Inform´atica de la Universidad Polit´ecnica de Madrid. Spain, 2001.

(Roy,R., 1959) R. Roy. Transitivité et connexité. C.R. Acad. Sci. Paris, 249:216–218.(1959)

(Sahoo, S.S. *et al.*, 2009) Sahoo, S.S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr, T., Auer, S., Sequeda J., Ezzat, A. A survey of current approaches for mapping of relational databases to rdf. W3C RDB2RDF Incubator Group Report (2009).

(Shvaiko, P. e Euzenat., J., 2004) P. Shvaiko and J. Euzenat. A Survey of Schema-based Matching Approaches. Tech- nical Report DIT-04-087, University of Trento.(2004)

(Stojanovic, L., 2004) Stojanovic, L. Methods and Tools for Ontology Evolution, Ph.D. Thesis, University of Karlsruhe, Germany. (2004).

(Thompson, H. S. *et al***.,** 2004)Thompson, H. S. *et al***.** XML Schema Part 1: Structures Second Edition, W3C Recommendation. Disponível em: http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/. (2004)

(Tury, M. e Bieliková. M., 2006) Tury, M. e Bieliková, M.. An Approach to Detection Ontology Changes. In ICWE ’06: Workshop proceedings of the sixth international conference on Web engineering, page 14, New York, NY, USA. ACM Press. (2006)

(Volz R., 2003) Volz, R., Oberle, D. e Studer, R.. Implementing Views for Light-Weight Web Ontologies. In Proc. of Int. Database Engineering and Application Symposium - IDEAS, pages 160–169, Hong Kong, China. IEEE Computer Society. (2003)

(Warshall, S., 1964) Warshall, S.. A theorem on boolean matrices. Journal of the ACM , 9(1):11–12. (1962)