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

PR-OWL 2 RL: um Formalismo para Tratamento de Incerteza na Web Semântica

Laécio Lima dos Santos

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientador
Prof. Dr. Li Weigang

Coorientador
Prof. Dr. Marcelo Ladeira

Brasília
2016

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

SSA237p

Santos, Laécio Lima dos
PR-OWL 2 RL: um formalismo para tratamento de incerteza na Web Semântica / Laécio Lima dos Santos; orientador Li Weigang; co-orientador Marcelo Ladeira. -- Brasília, 2016.
110 p.

Dissertação (Mestrado - Mestrado em Informática) - Universidade de Brasília, 2016.

1. Incerteza. 2. Raciocínio Probabilístico. 3. Web Semântica. 4. Ontologias. 5. Triplestores. I. Weigang, Li, orient. II. Ladeira, Marcelo, co orient. III. Título.

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

PR-OWL 2 RL: um Formalismo para Tratamento de Incerteza na Web Semântica

Laécio Lima dos Santos

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof. Dr. Li Weigang (Orientador)
CIC/UnB

Prof. Dr. Fabio Gagliardi Cozman
Poli/USP

Prof. Dr. Maurício Ayala Rincón
CIC/UnB

Prof. Dr. Rommel Novaes Carvalho
CIC/UnB

Prof.a Dr.a Célia Ghedini Ralha
Coordenadora do Programa de Pós-graduação em Informática

Brasília, 15 de julho de 2016

Laécio Lima dos Santos

PR – OWL 2 RL: Um Formalismo para tratamento de Incerteza na Web Semântica.

Tese aprovada como requisito parcial para obtenção do grau de **Mestre** no Curso de Pós-graduação em Informática da Universidade de Brasília, pela Comissão formada pelos professores:

Prof. Dr. Li Weigang, CIC/UnB / Orientador

Prof. Dr. Maurício Ayala Rincón, CIC/UnB

Prof. Dr. Fabio Gagliardi Cozman, Poli/USP

Vista e permitida a impressão.
Brasília, 15 de Julho de 2016.

Prof.ª Dr.ª Célia Ghedini Ralha
Programa de Pós-Graduação em Informática
Departamento de Ciência da Computação
Universidade de Brasília

Dedicatória

Dedico este trabalho aos meus pais, que sempre me incentivaram a dedicar-me com afinco aos estudos, e a minha esposa, que foi compreensiva e atenciosa ao me acompanhar nesta longa jornada.

Agradecimentos

Agradeço primeiramente aos meus orientadores, prof. Dr. Li Weigang e prof. Dr. Marcelo Ladeira, que me guiaram para que eu fizesse o melhor trabalho possível. Um agradecimento especial ao professor Marcelo Ladeira por ter me convidado para trabalhar com o UnBBayes e com redes bayesianas multi-entidades durante a minha graduação, há aproximadamente nove anos atrás. Fazer parte do Grupo de Inteligência Artificial da UnB foi sem dúvida uma experiência engrandecedora e me trouxe a oportunidade de trabalhar com pessoas com ideias grandiosas. Agradeço ao professor Li Weigang por ter me ajudado a fazer boas publicações e a buscar inovações em minha pesquisa. Sua orientação foi de suma importância para ajudar a me desenvolver como pesquisador.

Agradeço ao prof. Dr. Rommel Carvalho por ter me ajudado a delinear o tema da pesquisa e ter me dado dicas importantes sobre como melhorar a escalabilidade do UnBBayes, sugerindo a possibilidade de utilizar bases de dados noSQL. Agradeço ainda por sua paciência ao revisar o meu inglês em nossas publicações. Agradeço também ao MSc. Shou Matsumoto, que, mesmo estando bastante atarefado com o doutorado, sempre foi rápido e prestativo ao discutir minhas dúvidas e sugerir soluções e correções.

Agradeço ainda à prof(a) Dr(a) Kathryn B. Laskey e ao prof Dr Paulo Cesar G. Costa por terem me dado feedbacks importantes sobre a minha pesquisa e me orientado em que pontos a melhorar. Agradeço ao prof Dr Paulo Cesar G. Costa por ter apresentado o artigo aceito no Fusion.

Um agradecimento especial ao meu gerente no Banco do Brasil, Márcio Darci, que me incentivou a seguir com o mestrado, e, à medida do possível, sempre procurou me oferecer as condições necessárias para que eu pudesse atacar este objetivo ambicioso mesmo com a rotina corrida de uma diretoria de tecnologia.

Por fim, agradeço aos meus amigos e familiares que me incentivaram a concluir mais esta fase da minha carreira.

Resumo

A Web Semântica (WS) adiciona informações semânticas a Web tradicional, permitindo que os computadores entendam conteúdos antes acessíveis apenas aos humanos. A *Ontology Web Language* (OWL), linguagem padrão para criação de ontologias na WS, se baseia em lógica descritiva para permitir uma modelagem formal de um domínio de conhecimento. A OWL, no entanto, não possui suporte para tratamento de incerteza, presente em diversas situações, o que motivou o estudo de várias alternativas para tratar este problema. O *Probabilistic* OWL (PR-OWL) adiciona suporte à incerteza ao OWL utilizando *Multi-Entity Bayesian Networks* (MEBN), uma linguagem probabilística de primeira ordem. A inferência no MEBN ocorre através da geração de uma rede bayesiana específica de situação (SSBN). O PR-OWL 2 estende a linguagem original oferecendo uma maior integração com o OWL e permitindo a construção de ontologias que mesclam conhecimento determinístico e probabilístico.

PR-OWL não permite lidar com domínios que contenham bases assertivas muito grandes. Isto se deve a alta complexidade computacional da lógica descritiva na qual a OWL é baseada e ao fato de que as máquinas de inferência utilizadas nas implementações das versões do PR-OWL requerem que a base assertiva esteja carregada em memória. O presente trabalho propõe o PR-OWL 2 RL, uma versão escalável do PR-OWL baseada no profile OWL 2 RL e em *triplestores*. O OWL 2 RL permite raciocínio em tempo polinomial para as principais tarefas de inferência. *Triplestores* permitem armazenar triplas RDF (*Resource Description Framework*) em bancos de dados otimizados para trabalhar com grafos. Para permitir a geração de SSBN para bases contendo muitas evidências, este trabalho propõe um novo algoritmo, escalável ao instanciar nós de evidência apenas caso eles influenciem o nó objetivo. O plug-in PR-OWL 2 RL para o framework UnBBayes foi desenvolvido para permitir uma avaliação experimental dos algoritmos propostos. O estudo de caso abordado foi o de fraudes em licitações públicas.

Palavras-chave: Web Semântica, Ontologia, OWL, MEBN, Incerteza, PR-OWL, Triplestores, OWL 2 RL

Abstract

Semantic Web (SW) adds semantic information to the traditional Web, allowing computers to understand content before accessible only by human beings. The Web Ontology Language (OWL), main language for building ontologies in SW, allows a formal modeling of a knowledge domain based on description logics. OWL, however, does not support uncertainty. This restriction motivated the creation of several extensions of this language. Probabilistic OWL (PR-OWL) improves OWL with the ability to treat uncertainty using Multi-Entity Bayesian Networks (MEBN). MEBN is a first-order probabilistic logic. Its inference consists of generating a Situation Specific Bayesian Network (SSBN). PR-OWL 2 extends the PR-OWL offering a better integration with OWL and its underlying logic, allowing the creation of ontologies with deterministic and probabilistic parts.

PR-OWL, however, does not deal with very large assertive bases. This is due to the high computational complexity of the description logic of OWL. Another fact is that reasoners used in PR-OWL implementation require that the data be fully load into memory at the time of inference. To address this issue, this work proposes PR-OWL 2 RL, a scalable version of PR-OWL based on OWL 2 RL profile and on triplestores. OWL 2 RL allows reasoning in polynomial time for the main reasoning tasks. Triplestores can store RDF (Resource Description Framework) triples in databases optimized to work with graphs. To allow the generation of SSBNs for databases with large evidence base, this work proposes a new algorithm that is scalable because it instantiates an evidence node only if it influence a target node. A plug-in for the UnBBayes framework was developed to allow an empirical evaluation of the new algorithms proposed. A case study over frauds into procurements was carried on.

Keywords: Semantic Web, Ontology, OWL, MEBN, Uncertainty, PR-OWL, Triplestores, OWL 2 RL

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

BNF Backus-Naur Form.

CPT *Conditional Probabilistic Table*, ou tabela de probabilidade condicional.

DAG *Directed Acyclic Graph*, ou grafo acíclico direcionado.

DLP *Description Logic Programs*.

DSN *Domain Name System*.

GIA-UnB Grupo de Inteligência Artificial da Universidade de Brasília.

GUI *Graphical User Interface*, ou interface gráfica do usuário.

HTML *Hypertext Markup Language*.

HTTP *Hypertext Transfer Protocol*.

IRI *Internationalized Resource Identifiers*.

KIF *Knowledge Interchange Format*.

LPD *Local Probability Distribution*, ou distribuição de probabilidade local.

LUBM *Lehigh University Benchmark*.

MEBN *Multi-Entity Bayesian Networks*, ou redes bayesianas multi-entidade.

OWL *Web Ontology Language*.

PR-OWL *Probabilistic* OWL.

RDF *Resource Description Framework*.

RDFS *Resource Description Framework Schema.*

SPARQL acrônimo recursivo para SPARQL, *Protocol and RDF Query Language.*

SSBN *Situation-Specific Bayesian Network*, ou rede bayesiana de situação específica.

Turtle *Terse RDF Triple Language.*

UBF *UnBBayes Format.*

URI *Universal Resource Identifier.*

URL *Uniform Resource Locator.*

URW3-XG *Uncertainty Reasoning for the World Wide Web Incubator Group.*

W3C *World Wide Web Consortium.*

WWW *World Wide Web.*

XML *eXtensible Markup Language.*

Capítulo 1

Introdução

1.1 Contextualização

A Web Semântica estende a web tradicional adicionando informações semânticas, tornando possível que os computadores extraiam conteúdos antes só disponíveis aos humanos [3]. O significado da informação é tornado explícito através de ontologias, isto é, representações formais do conhecimento que modelam o domínio através de classes, atributos e relacionamentos. Neste intuito, a linguagem *Resource Description Framework* (RDF) [26] foi proposta para permitir a criação de ontologias através de triplas compostas por sujeito-predicado-objeto, onde cada elemento é descrito utilizando um identificador único universal (*Universal Resource Identifier*, URI). O RDF foi estendido com um vocabulário especial com semântica pré-definida, denominado RDF *Schema* (RDFS). O RDFS permite expressar hierarquia de classes, domínios e contra-domínios de propriedades, entre outras características necessárias para a montagem de ontologias leves. Afim de obter maior expressividade, permitindo a criação de ontologias complexas, foi criado a *Web Ontology Language* (OWL) [59], baseado em lógica descritiva. O uso da OWL é o atual padrão para criação de ontologias na Web Semântica.

A Web Semântica não possui um mecanismo padrão para representação de domínios que envolvam incerteza. A incerteza pode estar presente em ontologias de diversas formas: informações ambíguas, inconsistentes, incompletas, vagas, ignorância do modelador, entre outras. Há domínios que são estocásticos por natureza, como por exemplo previsão do tempo, mercado de ações e prevenção de fraudes. Diversas abordagens foram propostas para tratar a incerteza na Web Semântica, usando metodologias variadas, como redes bayesianas (Probabilistic-OWL [21], OntoBayes [84], BayesOWL [27]), lógica fuzzy (Fuzzy OWL [76]), extensões probabilísticas de lógicas descritivas ($\mathcal{P} - \mathcal{SHOQ}$(D) [35],

P-CLASSIC [48], CR$\mathcal{ALC}$ [25], estendendo respectivamente as lógicas $\mathcal{SHOQ}$, CLASSIC e $\mathcal{ALC}$) e funções de crença de Dempster-Shaffer (BeliefOWL [30]).

A *World Wide Web Consortium* (W3C), comunidade internacional responsável por propor padrões abertos que garantam o crescimento da Web [82], criou em 2007 o grupo de trabalho *Uncertainty Reasoning for the World Wide Web Incubator Group* (URW3-XG), com os objetivos de identificar e descrever as situações da *World Wide Web* (WWW) para as quais o raciocínio com incerteza pode aumentar significativamente o potencial de extração de informações úteis e estudar as metodologias aplicáveis a estas situações, buscando formas de representação padronizadas para estas [52]. O relatório final do grupo, publicado em 2008 [52], concluiu que nenhum dos formalismos analisados era adequado para tratar todos os diferentes casos de uso estudados, tornando-se necessário o estudo de novas propostas e extensões.

Um dos formalismos estudados pelo URW3-XG foi a linguagem *Probabilistic* OWL (PR-OWL). O PR-OWL, proposto por Costa [21], se baseia em redes bayesianas multi-entidades (*Multi-Entity Bayesian Networks*, MEBN) para acrescentar ao OWL o ferramental necessário para representação e inferência de ontologias probabilísticas [23].

O MEBN [51] adiciona às redes bayesianas a expressividade da lógica de primeira ordem, resolvendo as limitações desta quanto a incapacidade de representar domínios onde a quantidade de nós é desconhecida em momento de modelagem. Um modelo MEBN é baseado na integração de fragmentos parametrizados, denominados MEBN *Fragments* (MFrags). Uma MFrag representa uma distribuição de probabilidade condicional das instâncias de suas variáveis aleatórias residentes, dados os valores de seus pais no grafo [12]. Cada MFrag se relaciona com as outras por meio de variáveis aleatórias de entrada. As condições de validade de cada MFrag são determinadas por expressões lógicas, em lógica de primeira ordem, definidas por variáveis denominadas nós de contexto. O conjunto de MFrags formam uma MEBN *Theory* (MTheory), que definem restrições que garantem a existência de uma distribuição conjunta única de probabilidade. A inferência em MEBN se dá através da geração de redes bayesianas de situação específica (SSBN).

Em 2008, foi desenvolvida no framework UnBBayes [57] a primeira implementação mundial de MEBN e PR-OWL. O UnBBayes é um software de código aberto para raciocínio probabilístico desenvolvido pelo Grupo de Inteligência Artificial da Universidade de Brasília (GIA-UnB) que implementa diversos formalismos, utilizando uma arquitetura de plug-ins de fácil extensão. A implementação de PR-OWL/MEBN no UnBBayes permite criar modelos MEBN graficamente, salvá-los como uma ontologia PR-OWL e realizar inferências, montando SSBN [22]. Este desenvolvimento foi uma cooperação entre o GIA-UnB e a Universidade George Mason, com o objetivo de criar uma ferramenta para construção de ontologias probabilísticas que pudesse ser utilizada na avaliação do PR-OWL pelo

URW3-XG.

O PR-OWL 2 foi proposto por Carvalho [9] como uma extensão do PR-OWL resolvendo as limitações deste quanto à representação de ontologias híbridas, contendo partes probabilísticas e determinísticas. A linguagem possui construções que permitem que o modelador acrescente informações de incerteza a propriedades de uma ontologia determinística e aproxima a semântica do PR-OWL à semântica do OWL ao utilizar construções como Classes, Propriedades e *Datatypes* do OWL, ao invés de redefini-lás, como era feito no PR-OWL original. Em 2011 foi implementado no UnBBayes um plug-in para PR-OWL 2 [56], utilizando o *Protégé* como software de modelagem para a parte determinística da ontologia e seu raciocinador padrão, o HermiT [75], para obter as informações necessárias à montagem das SSBN.

1.2 Definição do Problema

O W3C ainda não definiu um formalismo padrão para representação de incerteza na Web Semântica, motivando a criação de novos formalismos e extensões.

A maioria das abordagens propostas não possuem implementações práticas que possam ser utilizadas efetivamente para representação e raciocínio, dificultando a validação pelos usuários quanto às necessidades de seus domínios. O PR-OWL e o PR-OWL 2 foram implementados no UnBBayes, porém estas implementações possuem limitações de escalabilidade e expressividade, descritas à seguir, que restringem o seu uso em aplicações de grande porte.

Em aplicações reais, a base de dados contendo as declarações assertivas pode ser bastante volumosa. A ontologia para análise de fraudes em licitações públicas apresentada em [17], por exemplo, possui uma base com milhões de declarações. Ontologias utilizadas em Linked Data [6] utilizam bases de dados para armazenar informações volumosas e interligadas sobre domínios de conhecimento como informações geográficas (GeoNames [1]), filmes (LDB [2]), ou mesmo informações enciclopédicas (DBPedia [3]). O PR-OWL 2, tentando aproximar a semântica do PR-OWL com a do OWL 2 DL, utiliza um raciocinador OWL 2 DL para realizar inferência com as expressões em lógica de primeira ordem da linguagem. Raciocinadores OWL 2 DL possuem complexidade temporal N2EXPTIME completo (ao menos duas vezes exponencial no tamanho da entrada), o que os torna inadequados para o trabalho com grandes bases de dados. Além disso, ambas as máquinas de inferência utilizadas para acessar a base de declarações assertivas nos plug-ins de PR-OWL

[1] http://www.geonames.org/ontology/
[2] http://linkedmdb.org/
[3] http://dbpedia.org/

1 e 2 do UnBBayes necessitam que os dados estejam carregados em memória, limitando o tamanho da base à capacidade de memória do recurso computacional utilizado.

Outra restrição na escalabilidade ocorre no algoritmo de geração de SSBN implementado no UnBBayes. A implementação se baseia no algoritmo *Bottom-Up* descrito em [51], que inicia a geração com os nós de questionamento do usuário (nós de *query*) e os nós de evidência contidos na base (nós de *finding*), e a partir destes vai iterativamente adicionando os nós pais até chegar a uma iteração onde não haja mais pais à serem adicionados. Uma fase de poda posterior retira da rede os nós que não influenciam probabilisticamente o nó de *query*, utilizando critérios bem conhecidos como o *d-separation*. Este algoritmo apresenta problemas em situações onde a base assertiva é volumosa, pois haverá uma grande quantidade de evidências na base, causando a geração de diversos nós que possivelmente serão retirados da rede na fase de poda.

Além disso, ao implementar o PR-OWL 2 foram feitas diversas restrições nos formatos das expressões lógicas válidas para os nós de contexto, reduzindo a expressividade original da linguagem. As restrições foram necessárias pela impossibilidade de utilizar um raciocinador de OWL 2 DL, baseado em lógica descritiva, para avaliar todas as fórmulas possíveis dos nós de contexto em lógica de primeira ordem. As lógicas descritivas são subconjuntos da lógica de primeira ordem, modeladas de forma a obter uma maior tratabilidade, fazendo para tal restrições na expressividade. Portanto, nem todas as expressões válidas na lógica de primeira ordem podem ser expressas em lógica descritiva. A implementação de PR-OWL 1 utiliza o raciocinador PowerLoom [19] para permitir a avaliação de fórmulas em lógica de primeira ordem, porém falha por não permitir a criação de ontologias híbridas, fundamentais para a integração de informações oferecidas por diversas fontes, um dos objetivos da Web Semântica.

As limitações das versões existentes do PR-OWL motivaram a proposição de uma extensão que o torne mais adequado ao trabalho com aplicações de grande porte.

1.3 Abordagem Proposta

A proposta deste trabalho é estender o PR-OWL 2 para a construção de ontologias no *profile* OWL 2 RL, permitindo que estas sejam armazenadas em *triplestores*, repositórios semânticos escaláveis, criando uma linguagem adequada para o trabalho com ontologias contendo grandes bases assertivas.

Diferente da abordagem Entidade-Relacionamento dos bancos de dados tradicionais, onde os dados são armazenados utilizando tabelas, *triplestores* armazenam os dados utilizando triplas RDF, organizando a informação em grafos, onde os recursos são identificados por URIs. *Triplestores* como GraphDB [5], Jena TDB [31], AllegroGraph [44] e Oracle

Spatial and Graph [49] permitem o armazenamento e consulta de bilhões de triplas RDF, tornando-os opções viáveis inclusive para o trabalho com Big Data. A maioria das *triplestores* permitem inferência em RDFS, utilizando regras dedutivas para implementar a semântica da linguagem. Algumas implementações de *triplestores* permitem o trabalho com versões restritivas do OWL, como por exemplo o profile OWL 2 RL, baseado em regras, e o profile OWL 2 QL, baseado em *queries* conjuntivas. Enquanto na primeira abordagem a *triplestore* utiliza materialização para expandir novas expressões geradas a partir das existentes na base por meio de um conjunto de regras que implementam o *profile*, a segunda abordagem reescreve as *queries* de forma que elas capturem como resultado também as expressões que são inferidas a partir do conteúdo da base aplicando a semântica do profile.

A proposta da nova linguagem, chamada de PR-OWL 2 RL consiste na definição da sintaxe da nova linguagem, adequada ao profile OWL 2 RL, na definição dos formatos válidos para os nós de contexto, avaliados utilizando expressões de consultas à *triplestore*, e da proposta de um novo algoritmo escalável para geração de SSBN. O novo algoritmo proposto, que pode ser utilizado também nas outras implementações de PR-OWL, se baseia no algoritmo Bayes-Ball [74], utilizado para identificação de nós d-separados a um conjunto de nós objetivos. Através deste novo algoritmo, a geração de SSBN é iniciada apenas dos nós de *query* e são gerados apenas nós d-conectados a este, seguindo um conjunto de regras definidas pelo algoritmo Bayes-Ball.

1.4 Objetivos

Este trabalho tem como objetivo geral contribuir para a proposição de um formalismo que possa ser utilizado no tratamento de incerteza na Web Semântica e que seja capaz de lidar com ontologias com grande volume de declarações assertivas. A incerteza é representada por meio da teoria de probabilidade para permitir construir e realizar inferências com base em ontologias contendo conhecimento probabilístico e determinístico, com milhões de triplas RDF.

Os objetivos específicos deste trabalho são:

- criar uma nova versão do PR-OWL que permita o uso da *profile* OWL 2 RL (que possui complexidade polinomial para os principais tipos de raciocínios) e bases de dados RDF (*triplestores*);

- propor uma gramática formal para produção das expressões em lógica de primeira ordem aceitas em nós de contexto de MFrag que possam ser avaliadas utilizando *triplestores*;
- criar um algoritmo escalável para geração de redes bayesianas de situação específica (SSBN) adequado a domínios contendo grande base assertiva;
- implementar o PR-OWL 2 RL e o novo algoritmo de geração de SSBN no UnBBayes e avaliar empiricamente a linguagem por meio de testes.

1.5 Metodologia

A pesquisa foi desenvolvida utilizando a seguinte metodologia:

- levantamento do estado da arte, analisando as alternativas existentes para representação de incerteza na Web Semântica;
- análise da escalabilidade e da expressividade do PR-OWL e do PR-OWL 2 e de suas implementações no UnBBayes;
- análise das alternativas para extensão do PR-OWL 2 para o trabalho com ontologias contendo bases assertivas grandes;
- análise dos sistemas gerenciadores de bases de dados RDF existentes;
- modelagem da linguagem PR-OWL 2 RL e dos formatos de expressões de lógica de primeira ordem aceitos;
- modelagem do novo algoritmo para geração de SSBN;
- codificação do PR-OWL 2 RL como um plug-in para o framework UnBBayes;
- análise das limitações do PR-OWL 2 RL, das características dos domínios que poderão ser modeladas e comparação com as outras versões existentes;
- avaliação empírica por meio de testes utilizando a ontologia Fraudes em Licitações Públicas, proposta em [16] [17].

1.6 Organização do Documento

A seguir é apresentada a estrutura desta dissertação. Os dois primeiros capítulos fazem o levantamento do estado da arte, onde o primeiro apresenta a Web Semântica e suas tecnologias e o segundo o PR-OWL como mecanismo para tratamento de incerteza. Os

capítulos seguintes apresentam a linguagem proposta, PR-OWL 2 RL, sua implementação no UnBBayes, os testes realizados para validar o formalismo e uma discussão dos resultados.

Segue descrição detalhada sobre cada capítulo:

- O Capítulo 2 apresenta os conceitos de Web Semântica necessários para a apresentação deste trabalho. São apresentadas as principais linguagens da Web Semântica: RDF, RDFS e OWL, além dos três profiles do OWL: OWL 2 RL, OWL 2 QL e OWL 2 EL. O capítulo descreve também *triplestores*.

- O Capítulo 3 apresenta a linguagem PR-OWL e sua extensão, o PR-OWL 2, além das implementações de ambas as linguagens feitas no UnBBayes. Para tal, antes são apresentados os conceitos de redes bayesianas e redes bayesianas multi-entidades. O capítulo apresenta também um levantamento dos outros formalismos existentes para tratamento de incerteza na Web Semântica.

- O Capítulo 4 apresenta a solução proposta: o PR-OWL 2 RL. Para tal, descreve de forma mais aprofundada os problemas com as abordagens atuais do PR-OWL, explicando como a abordagem proposta irá solucioná-los. A linguagem PR-OWL 2 RL é definida através da especificação de sua sintaxe e dos formatos de fórmulas em lógica de primeira ordem aceitos.

- O Capítulo 5 descreve os algoritmos de geração de SSBN previamente implementados no UnBBayes: o algoritmo *Bottom-Up* e o algoritmo proposto pelo GIA-UnB e discute as limitações de ambos, propondo um novo algoritmo baseado no algoritmo Bayes-Ball.

- O Capítulo 6 apresenta os testes realizados com a nova linguagem e com o novo algoritmo de geração de SSBN a fim de validar a solução. O capítulo também faz uma discussão das diferenças da linguagem com as versões do PR-OWL anteriores.

- O Capítulo 7 apresenta as considerações finais e os trabalhos futuros.

Capítulo 2

Web Semântica e Tecnologias Envolvidas

Este capítulo apresenta os conceitos de Web Semântica necessários ao entendimento deste trabalho. A Seção 2.1 apresenta a Web Semântica. A Seção 2.2 apresenta RDF e RDFS. A Seção 2.3 apresenta a linguagem OWL e seus profiles OWL 2 QL, EL e RL, além de fazer uma breve descrição sobre lógica descritiva, base da semântica direta do OWL. A Seção 2.4 apresenta detalhadamente o profile OWL 2 RL, baseado em regras. Finalmente, a Seção 2.5 apresenta *triplestores*.

2.1 Web e Web Semântica

A *World Wide Web* (WWW) surgiu em 1989 no CERN (Organização Européia para Pesquisas Nucleares, do francês *Conseil Européen pour la Recherche Nucléaire*) como uma forma de permitir aos diversos pesquisadores envolvidos no projeto trocarem documentos com informações diversas como relatórios, figuras, etc. Idealizada por Tim Berners-Lee, a WWW, também conhecida apenas como Web, é composta por páginas interligadas através de vínculos, os *hyperlinks*. Utilizando um navegador Web, o usuário pode visualizar estas páginas e "navegar" entre elas através dos *hyperlinks*, ficando invisível detalhes como o local de armazenamento de cada página. A Web rapidamente se popularizou, inclusive para meios não acadêmicos, tornando-se a tecnologia mais utilizada da Internet e revolucionando o acesso a informação.

O *World Wide Web Consortium*(W3C) foi criada em 1994 como uma organização voltada para o desenvolvimento da Web, a padronização de protocolos e para o incentivo à interoperabilidade entre os sites [77]. A principal linguagem utilizada para a criação de páginas na Web é o HTML (*Hypertext Markup Language*), que utiliza um conjunto de

tags para descrever o conteúdo dos documentos. As páginas são identificadas de forma universal através de URLs (*Uniform Resource Locator*), identificadores pelos quais os servidores DSN (*Domain Name System*) são capazes de recuperar o endereço IP do host onde se encontra armazenado a página e recuperar o seu conteúdo. O protocolo HTTP (*Hypertext Transfer Protocol*) é o mais utilizado para fazer a comunicação com o host. Além desses, há diversos outros padrões definidos pela W3C, como por exemplo o XML.

A Web Semântica, também idealizada por Tim Berners-Lee [3], possui como objetivo estender a Web tradicional explicitando informações sobre a semântica dos conteúdos de forma que as máquinas possam compreender informações antes disponíveis apenas para os humanos. O conhecimento é representado principalmente através de ontologias: especificações formais e explicitas de uma conceptualização compartilhada [39], onde conceptualização corresponde a uma visão abstrata e simplificada de aspectos do mundo que deseja-se representar para alguma proposta [39].

A W3C definiu diversos padrões para a Web Semântica, dos quais pode-se destacar o *Resource Description Framework* (RDF) e a *Ontology Web Language* (OWL). O RDF, linguagem fundamental da Web Semântica, permite representar o conhecimento de forma estruturada através de triplas compostas por sujeito, predicado e objeto, onde cada elemento é identificado por um identificador único universal (URI). O OWL permite a criação de ontologias complexas, baseando-se em lógicas descritivas para permitir a modelagem e inferência em domínios complexos.

Como parte da Web Semântica, Linked Data [6] consiste em disponibilizar os dados na Web de forma estruturada e interligada, de forma semelhante a feita com as páginas utilizando *hyperlinks*. Estas bases de dados formam uma imensa base de dados aberta, contribuindo para o que é chamado de Web of Data [68]. Os dados são organizados em grafos RDF e seguem alguns princípios para garantir a interoperabilidade: uso de URIs para nomear as entidades, retorno de dados em RDF sobre entidades quando sua URI é pesquisada e inclusão de links para outros documentos RDF. A linguagem de consulta SPARQL [81] é a principal forma de recuperar informações em bases de dados Linked Data.

2.2 RDF

O *Resource Description Framework* (RDF) é uma linguagem formal utilizada para a descrição de informações estruturadas [43], sendo o formato básico de representação e troca de informações na Web Semântica.

O RDF representa os dados através de triplas RDF, compostas por sujeito (s), predicado (p) e objeto (o) e utilizadas para codificar expressões lógicas simples sobre o

mundo [26]. *Universal Resource Identifier* (URI) são utilizados para identificação dos indivíduos e propriedades de forma única e permitem que conteúdos de fontes diferentes sejam integrados. O sujeito e o predicado da tripla também podem ser *blank nodes* (nós auxiliares anônimos, que não possuem URI), e o objeto também pode ser um literal, permitindo o uso de *datatypes*, como por exemplo números.

O conjunto de triplas forma um grafo RDF. A Figura 2.1 mostra um grafo RDF para o domínio "Fraudes em Licitações Públicas" [17]. O recurso `pfe:Pessoa1` (sujeito), por exemplo, está ligado ao recurso `pfe:EmpresaA` (objeto) através da propriedade `pfe:isResponsibleFor`. O prefixo `pfe` se refere a http://www.pr-owl.org/examples/pr-owl2/ProcurementFraud/ProcurementFraud.owl. Prefixos são utilizados para simplificar a escrita das URIs dos recursos).

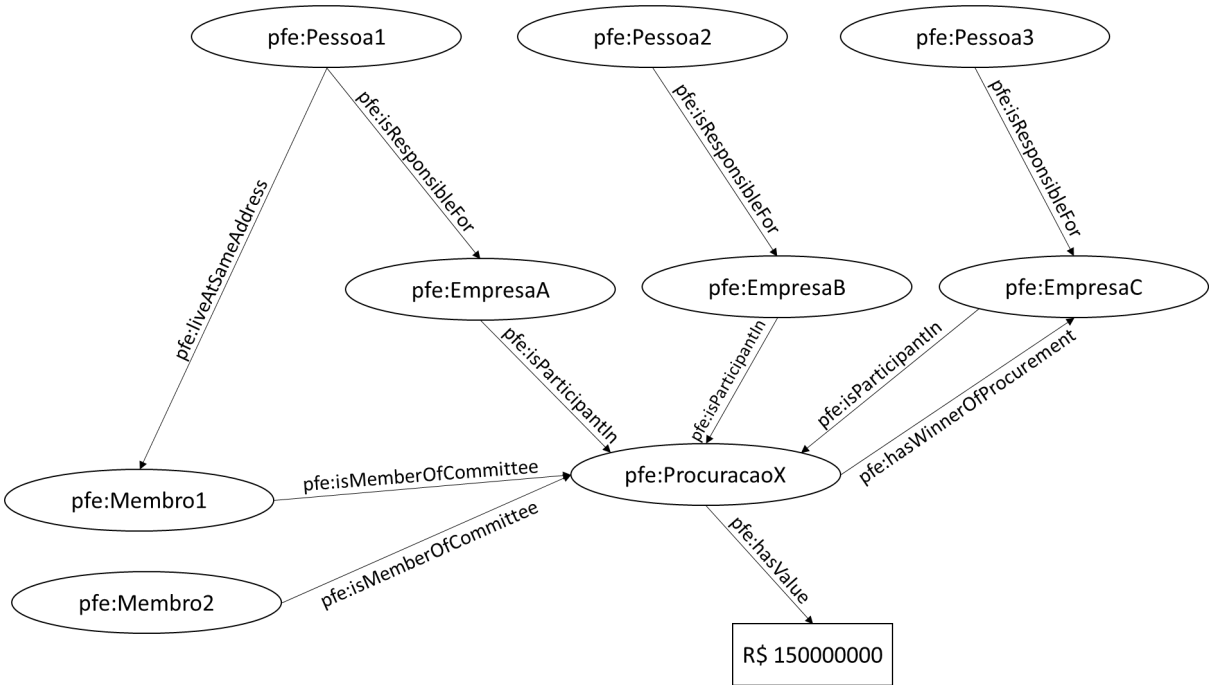

Figura 2.1: Exemplo de grafo RDF para o domínio Fraudes em Licitações Públicas

Um documento em RDF geralmente é serializado utilizando XML, porém há outros formatos disponíveis, como N3, N-Triples e Turtle (*Terse RDF Triple Language*).

O RDF Schema (RDFS) estende o RDF permitindo modelar o conhecimento terminológico de um domínio através de um vocabulário formal pré-definido. O RDF é uma linguagem universal que permite ao usuário descrever os recursos utilizando o seu próprio vocabulário [1]. O RDFS adiciona um vocabulário padronizado que permite a criação de ontologias leves.

Segue uma listagem com algumas das principais construções da linguagem RDFS:

- `rdf:type`: permite definir qual o tipo de um recurso. Ex: `Procurement rdf:type rdfs:Class`;
- `rdfs:Class`: define que o recurso é do tipo Classe;
- `rdfs:subClassOf`: definição de sub classes, permitindo a construção de hierarquias entre classes. Ex: `Humano rdf:subClassOf Mamifero`;
- `rdf:Property`: define que o recurso é do tipo Propriedade;
- `rdfs:subPropertyOf`: definição de sub propriedades;.
- `rdfs:range` e `rdf:domain`: permite que se defina domínio e contra-domínio de propriedades.

2.3 OWL e Lógica Descritiva

A *Web Ontology Language* (OWL) é uma linguagem de representação baseada em lógica formal, utilizada para construção de ontologias complexas. Em 2004, o OWL se tornou a recomendação padrão do W3C para o desenvolvimento de ontologias na Web Semântica, e desde então vem se tornando cada vez mais popular em diversos domínios. O OWL é construído sobre o RDF, e normalmente utiliza RDF/XML para persistência.

O OWL possui três versões, que diferem em poder de expressividade e decidibilidade: OWL *Lite*, OWL DL e OWL *Full*. O OWL *Lite* e o OWL DL possuem suas semânticas baseadas em lógicas descritivas decidíveis. O OWL *Full* possui grande expressividade, permitindo o uso de todas as construções da linguagem OWL em conjunto com as construções do RDFS, porém é indecidível. Uma das razões para a indecidibilidade do OWL *Full* é o fato da separação de tipos não ser obrigatória, tornando possível que classes, indivíduos e propriedades sejam mixadas livremente (um mesmo identificador pode ser utilizado como individuo em uma declaração e como propriedade em outra) [43]. A semântica do OWL Full é baseada em grafos RDF.

Lógicas descritivas são uma família de formalismos de representação de conhecimento que representam o conhecimento de um domínio de aplicação definindo os conceitos relevantes do domínio (Terminologia - TBox), e então usando estes conceitos para especificar propriedades dos objetos e indivíduos que ocorrem no domínio (Base assertiva - ABox) [2]. Existem diversas versões de lógicas descritivas, balanceando expressividade e complexidade de inferência. A $\mathcal{ALC}$ (*Attributive Logic with Complement*) é a lógica descritiva mais básica, possuindo construções para classes, relacionamentos e indivíduos. As outras lógicas descritivas acrescentam construtores à $\mathcal{ALC}$, aumentando a expressividade. O OWL *Lite* se baseia na lógica descritiva $\mathcal{SHIF}$(D), enquanto o OWL DL se baseia em

uma lógica descritiva mais expressiva, a $\mathcal{SHOIN}$(D). Os nomes das lógicas descritivas são baseados nos construtores que elas acrescentam à $\mathcal{ALC}$. $\mathcal{SHOIN}$(D), por exemplo, possui suporte a transitividade entre relacionamentos ($\mathcal{S}$), hierarquia de relacionamentos ($\mathcal{H}$), nominais ($\mathcal{O}$), relacionamentos inversos ($\mathcal{I}$), restrições de cardinalidade ($\mathcal{N}$) e tipos de dados (D). A semântica do OWL DL, baseada em lógica descritiva, é conhecida como Semântica Direta.

Os blocos básicos de construção do OWL são classes, propriedades e indivíduos [43]. Classes são definidas utilizando a construção `owl:Class`. Propriedades de objeto (`owl:ObjectProperty`) são utilizadas para relacionar dois indivíduos, enquanto propriedades de dados (`owl:DatatypeProperty`) relacionam um individuo com um valor de dados. Indivíduos representam objetos no domínio de interesse.

Segue listagem com as principais construções da linguagem OWL:

- `owl:Thing` e `owl:Nothing` são classes pré-definidas, onde a primeira contém todos os indivíduos como instâncias, enquanto a segunda não contém nenhum individuo;
- `owl:sameAs` pode ser utilizado para definir que dois nomes de indivíduos se referem ao mesmo individuo e `owl:differentFrom` que se referem a indivíduos diferentes;
- `owl:subClassOf` pode ser utilizado para definir que uma classe é sub classe de outra e `owl:equivalentClass` para definir que duas classes são equivalentes;
- `owl:disjointWith` pode ser utilizado para definir que os indivíduos de uma classe são disjuntos entre sí;
- `owl:unionOf`, `owl:intersectionOf` e `owl:complementOf` corresponde respectivamente aos operadores lógicos *ou*, *and* e *not* podendo ser utilizado para combinar classes atômicas em classes complexas [43];
- `owl:allValuesFrom` pode ser utilizado para fazer restrições em propriedades, indicando que para uma classe, todos os valores da propriedade informada devem ser de uma determinada classe;
- `owl:someValuesFrom` pode ser utilizado para fazer restrições em propriedades, indicando que para uma classe, deve existir ao menos um valor de uma determinada classe para a propriedade;
- `owl:cardinality` pode ser utilizado para definir a cardinalidade de uma propriedade, enquanto `owl:minCardinality` permite definir a cardinalidade mínima e `owl:maxCardinality` a cardinalidade máxima;
- `owl:hasValue` pode ser utilizada para fazer restrições indicando que para a classe, a propriedade informada deve ter um determinado valor;

- `owl:FunctionalProperty`, `owl:InverseFunctionalProperty`, `owl:TransitiveProperty`, `owl:SymmetricProperty`, podem ser utilizadas respectivamente para definir propriedades como funcionais, funcionais inversas, transitivas e simétricas.

Foram desenvolvidos diversos raciocinadores para OWL DL, dentre as quais o FaCT++ [79], da Universidade de Manchester, o HermiT [75], do Laboratório de Computação da Universidade de Oxford e o Pellet [66], da Clark & Parsia LLC. Estes raciocinadores utilizam algoritmos baseados em *tableau*. Há outras abordagens, como bases de dados dedutivas, baseadas em datalogs (ex: KAON2 [61]), e raciocínio baseado em regras, utilizando materialização (ex: GraphDB [5] e Sesame [8]). Estas abordagens, no entanto, são limitadas a fragmentos menos expressivos do OWL [7].

Em 2012, o W3C, revisou e adicionou novas funcionalidades ao OWL, criando a especificação OWL 2. Entre as novas funcionalidades está a possibilidade de se trabalhar com cadeias de relacionamentos, restrições de cardinalidade mais ricas, propriedades assimétricas, reflexivas e disjuntas, entre outras. Os recursos são identificados em OWL 2 através de *Internationalized Resource Identifiers* (IRI), uma evolução do padrão URI que permite o uso de caracteres de diversos sistemas de escrita existentes. Uma descrição detalhada das novas funcionalidades do OWL 2 pode ser encontrada em [37].

O OWL 2 possui uma versão DL, baseada na lógica descritiva $\mathcal{SROIQ}$(D), mais expressiva que a $\mathcal{SHOIN}$(D), e uma versão *Full*, não decidível. Além destas duas versões, foi incluído o conceito de *profiles*, sub-linguagens (subconjuntos sintáticos), desenvolvidos para aplicações especificas, que possuem propriedades que permitem o desenvolvimento de algoritmos eficientes: os principais tipos de raciocínio podem ser realizados em tempo polinomial. Foram definidos três *profiles*:

- **OWL 2 EL**: Recomendado para aplicações envolvendo ontologias com um grande número de propriedades e classes, como por exemplo aplicações em biomedicina [38]. Se baseia na família de lógica descritiva $\mathcal{EL}++$ (de onde se origina a sigla EL do nome do profile), que provê apenas quantificador existencial [62].
- **OWL 2 QL**: Recomendado para aplicações com grande volume de dados e onde a resposta à queries é a tarefa mais importante [62]. Trabalha com *queries* conjuntivas, e pode ser implementado utilizando bancos de dados relacionais. QL se refere a ao fato da resposta a consultas poder ser implementado através de reescrita de *queries* utilizando uma linguagem de consulta (*Query Language*) relacional [62].
- **OWL 2 RL**: Recomendado para aplicações que precisam de um raciocínio escalável sem sacrificar muito a expressividade [62]. Permite a implementação de algoritmos em tempo polinomial utilizando tecnologias baseadas em regras, operando sobre

bases RDF [38]. RL reflete o fato do raciocínio no profile poder ser implementado utilizando uma linguagem de regras (*Rule Language*).

Há várias sintaxes disponíveis para o OWL 2, apropriadas para propósitos diferentes [42]. A sintaxe funcional [64] é utilizada para propósitos de especificação e provê fundamentos para a implementação em ferramentas OWL 2. A sintaxe RDF/XML [67] traduz as construções do OWL para RDF/XML, e deve ser suportada por todas as ferramentas OWL 2. As referências a construções do OWL 2 neste texto utilizam a sintaxe RDF/XML. Além destas, há as sintaxes Manchester e OWL/XML.

2.4 O Profile OWL 2 RL

O *profile* OWL 2 RL consiste em um subconjunto sintático do OWL 2 que permite implementações utilizando tecnologias baseadas em regras [62]. O *profile* permite a implementação de algoritmos em tempo polinomial em relação ao tamanho da ontologia para os tipos de inferência padrão: verificação de consistência, satisfabilidade de expressões de classe, verificação de instâncias, subsunção de expressões de classe e resolução de *queries* conjuntivas [62].

O OWL 2 RL é baseado em pD* e em *Description Logic Programs* (DLP).

A linguagem pD* estende o modelo teórico RDFS com regras e aplica os resultados a um subconjunto não padronizado do OWL, que inclui as propriedades `FunctionalProperty`, `InverseFunctionalProperty`, `sameAs`, `SymmetricProperty`, `TransitiveProperty`, `inverseOf`, `equivalentClass`, `equivalentProperty`, `hasValue`, `someValuesFrom`, `allValuesFrom`, `differentFrom` e `disjointWith` [78]. Com respeito a este subconjunto da linguagem OWL, pD* representa uma interpretação razoável que é útil para fazer inferências sobre instâncias utilizando a ontologia [78]. O pD* é decidível e possui relativa baixa complexidade [78].

DLP consiste em uma linguagem formada por um subconjunto do OWL DL somado a Datalogs, sendo menos expressivo que ambos os formalismos [43]. A linguagem Datalog foi criada como uma alternativa mais adequada para o uso em bases de dados dedutivas (*Deductive Databases*) que o Prolog. Bases de dados dedutivas utilizam programação lógica para expressar deduções relativas ao conteúdo da base [18]. Uma regra em Datalog é uma implicação lógica que pode conter conjunções, constantes e quantificadores universalmente quantificados, não sendo permitido o uso de disjunções, negações, quantificadores existenciais ou símbolos de funções [43]. Na linguagem DLP, os predicados devem ser binários e o corpo do grafo de variáveis do corpo de cada regra deve ser conectado e aciclico [69]. A inferência em DLP é restrita a base assertiva: apenas fatos podem ser derivados [69].

O *profile* OWL 2 RL inclui diversas restrições ao OWL 2 *Full* aplicadas ao uso e posição de diversas características da linguagem. Sua sintaxe é assimétrica, onde as restrições aplicadas do lado esquerdo (lado do sujeito) em expressões de subclasse são diferentes das aplicadas ao lado direito (lado do objeto). A Figura 2.2 ilustra um exemplo (adaptado de [50]) contendo expressões válidas em OWL 2 RL. As expressões declaram que gatos são animais, que gatos predam apenas animais pequenos e que animais que predam animais pertencem a classe dos predadores.

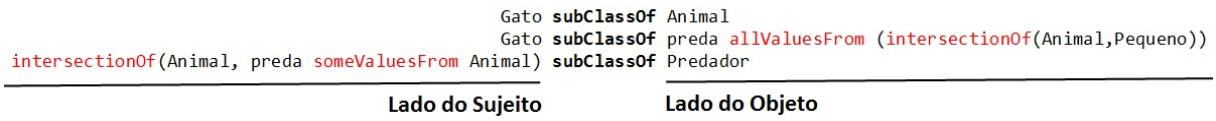

Figura 2.2: Exemplo de expressões OWL 2 RL

A seguir são apresentadas as principais restrições do OWL 2 RL, segundo [43]. O leitor interessado pode encontrar a listagem completa das restrições em [62].

- Em expressões de sub classe, no lado do sujeito, é permitido:
  - nomes de classes, exceto `owl:Thing`;
  - enumerações de indivíduos (`owl:oneOf`);
  - quantificador existencial aplicado a um individuo (`owl:hasValue`); e
  - interseção de classes (`owl:intersectionOf`), união de classes (`owl:unionOf`), e quantificador existencial aplicado a expressões de classe (`owl:someValuesFrom`), desde que as expressões de classe envolvidas também sejam expressões do lado do sujeito. Se `owl:someValuesFrom` for utilizado para propriedades concretas, apenas literais de tipos de dados podem ser utilizados.
- Em expressões de sub classe, no lado do objeto é permitido:
  - nomes de classes, exceto `owl:Thing`;
  - `owl:hasValue` e `owl:allValuesFrom` para propriedades concretas. Para propriedades abstratas, é restrito ao lado do objeto; e
  - restrição de cardinalidade `owl:maxCardinality`, restrita a cardinalidades 0 e 1; para propriedades abstratas, restrições de cardinalidade podem ser utilizadas com expressões do lado do sujeito com objetivos classes.
- `owl:equivalentClass` pode ser usado em expressões de classe, desde que ambos sejam expressões do lados do sujeito ou expressões do lado do predicado. `owl:disjointWith` e `owl:AllDisjointClasses` são restritas ao lado do sujeito.

- `rdfs:domain` e `rdfs:range` pode ser utilizado somente com expressões de classe do lado objeto.
- Declarações de membros de classe de um individuo podem ser utilizadas no lado do objeto de expressões de classe. `owl:NegativePropertyAssertion` não pode ser utilizado.

O W3C estabeleceu uma axiomatização parcial, composta por um conjunto de regras de implicações baseadas na semântica RDF, que pode ser utilizada para a implementação do *profile* OWL 2 RL em tecnologias baseadas em regras. Este conjunto de regras é apresentado como OWL 2 RL/RDF, podendo ser aplicado em ontologias OWL serializadas em RDF (se baseia na semântica RDF). As regras são implicações de 1ª ordem universalmente quantificadas sobre predicados ternários no formato **T(s, p, o)**, onde **s** é o sujeito, **p** o predicado e **o** o objeto. A regra abaixo, por exemplo, diz que se `c1` é uma subclasse de `c2` e `x` é uma instância da classe `c1`, então `c1` também é uma instância da classe `c2`.

```
IF T(?c1, rdfs:subClassOf, ?c2) AND T(?x, rdf:type, ?c1)
THEN T(?x, rdf:type, ?c2)
```

A axiomatização parcial OWL 2 RL/RDF garante principalmente que apenas indivíduos explicitamente declarados serão considerados durante a inferência [60], justificativa por exemplo, para que a construção `sameValuesFrom` não seja permitida do lado direito das expressões de sub-classes, já que ela estaria se referindo a indivíduos não explicitamente declarados [60].

Utilizando o conjunto de regras proposto, a inferência pode ser realizada através de materialização. A materialização consiste em expandir as regras em momento de carga, calculando todas as novas expressões que surgem a partir do conjunto de expressões adicionadas. No exemplo acima, caso tenhamos que a classe `A` é subclasse da classe `B`, ao adicionar na base a tripla `(a, rdf:type, A)`, a tripla `(a, rdf:type, B)` também será adicionada.

A axiomatização OWL 2 RL/RDF, no entanto, não implementa completamente o profile OWL 2 RL. O Teorema PR1 [62] (reproduzido no Apêndice C), define as restrições sintáticas que devem ser satisfeitas para que uma ontologia O_1 implique uma ontologia O_2 na Semântica Direta do OWL 2 (baseada em lógica descritiva), dado que O_2 foi obtida a partir de O_1 aplicando o conjunto de regras OWL 2 RL/RDF (que se baseia na semântica RDF). Tratam-se das seguintes restrições:

- O_1 e O_2 não podem conter IRI utilizadas por mais de um tipo de entidade. Uma IRI de classe, por exemplo, não pode ser utilizada como IRI de individuo;

- O_1 não pode conter axiomas envolvendo as construções `rdfs:subPropertyOf`, `rdfs:domain` e `rdfs:range` aplicadas a anotações[1];
- cada axioma em O_2 deve ser uma assertiva de classe, de propriedade (de objeto ou de dados), ou indicando que recursos se referem ao mesmo individuo (através da construção `owl:sameAs`).

De acordo com o teorema, quando a ontologia é consistente com a definição estrutural do OWL 2 RL, uma implementação adequada, baseada no conjunto de regras OWL 2 RL/RDF, que responda consultas atômicas *ground*, será completa (todas as respostas corretas serão computadas) e consistente (somente respostas corretas serão computadas) [38].

2.5 Bases de Dados RDF

O sucesso do RDF como linguagem de representação de dados semi-estruturados na Web Semântica tem causado uma proliferação de aplicações baseadas em grandes repositórios de dados armazenados em formato RDF [85]. *Triplestores* armazenam as informações através de triplas RDF.

As *triplestores* permitem a integração entre diferentes bases de dados através das URI. O projeto *Linking Open Data* [2], por exemplo, monta um mapa da Web Semântica mostrando como diversas bases de dados semânticos abertas se inter-relacionam através de links RDF, montando uma vasta rede de informações.

Por permitir uma organização do conhecimento de uma forma menos restritiva que as bases de dados relacionais, as bases de dados RDF se tornaram uteis em aplicações que envolvem Big Data. Bases de dados RDF como o OpenLink Virtuoso, o JenaTDB e o GraphDB, permitem a inserção e consulta de bilhões de triplas RDF. Conforme relatado em [83], em testes realizados com as triplestores Oracle Spatial and Graph e AllegroGraph foi possível carregar na base trilhões de triplas RDF.

As bases de dados RDF podem ser implementadas sobre um banco de dados relacional, utilizando tabelas, abordagem utilizada pelo Virtuoso e pelo Oracle Spatial and Graph; ou utilizando armazenagens não relacionais, baseadas em grafos, como as utilizadas no GraphDB, Sesame e Jena TDB. As abordagens não relacionais são comumente conhecidas na literatura como bancos de dados NoSQL - *Not Only SQL*.

[1] Anotações (*annotations) correpondem a construções do OWL que não contribuem para a especificação do conhecimento lógico da ontologia [42], sendo utilizados apenas para fornecer informações adicionais sobre a ontologia, axiomas, entidades, etc.*

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

Triplestores implementando o *profile* OWL 2 RL estão se tornando populares [85]. Exemplos de *triplestores* comerciais que implementam completamente ou parcialmente este *profile* incluem GraphDB [5], Oracle Spatial and Graph [49] e AllegroGraph [44].

O GraphDB, antigo OWLIM, é uma família de componentes de repositório semântico que contém uma base de dados RDF nativa, um raciocinador e um motor de busca [5]. Criado pela Ontotext, o GraphDB é escrito em linguagem Java, e implementa a interface SAIL (*Storage and Inference Layer*) do Sesame [8], um popular framework para processamento de dados RDF. O GraphDB utiliza diversas funcionalidades do Sesame, como por exemplo o modelo e *parser* RDF e o motor de *query* SPARQL [65]. O GraphDB é distribuído em três versões: *Lite*, *Standard* e *Enterprise*. As versões *Standard* e *Enterprise* são comerciais, permitindo o trabalho com bilhões de triplas RDF. A versão *Enterprise* se diferencia por permitir replicação em cluster, possibilitando processamento paralelizado. A versão *Lite*, gratuita, possui os mesmos recursos da versão *Standard*, porém, trabalha com os dados armazenados em memória, tendo sua escalabilidade limitada pela capacidade de armazenagem da máquina do usuário.

O GraphDB permite o trabalho com RDF, RDFS e com os *profiles* OWL 2 RL e OWL 2 QL, além de duas outras versões restritivas não padronizadas do OWL: OWL Horst, baseado no pD* apresentado em [78], e OWL-max, uma combinação entre o OWL-Lite e o RDFS.

O GraphDB baseia a sua implementação de OWL 2 RL no conjunto de regras definidas pelo W3C na especificação OWL 2 RL/RDF e em [71], que especifica como implementar estas regras utilizando RIF (*Rule Interchange Format*). A inferência é feita por materialização no momento em que novas declarações são adicionadas a base: o raciocinador utiliza predominantemente encadeamento para frente para aplicar as regras de inferência selecionadas diretamente sobre as triplas RDF [4]. O GraphDB é consistente e completo com respeito a semântica do OWL 2 RL/RDF, exceto pela falta de suporte a raciocínio com *datatypes* [4].

O GraphDB implementa o profile OWL 2 QL através de raciocínio baseado em regras sobre bases RDF, ao invés de utilizar uma abordagem baseada em reescrita de *queries*, mais adequada ao formalismo. A implementação de OWL 2 QL do GraphDB é incompleta e apresenta diversos problemas, como apresentado em [5].

Capítulo 3

O Formalismo PR-OWL

Este capítulo apresenta a linguagem *Probabilistic* OWL e um levantamento da arte sobre os formalismos existentes para tratamento de incerteza na Web Semântica.

Para permitir o entendimento da linguagem PR-OWL, as Seções 3.1 e 3.2 apresentam, respetivamente, redes bayesianas e redes bayesianas multi-entidades (MEBN), formalismos utilizados para adicionar o tratamento de incerteza ao OWL. A Seção 3.3 apresenta o PR-OWL e sua extensão, o PR-OWL 2. A Seção 3.4 apresenta as implementações de MEBN e PR-OWL no framework UnBBayes. A Seção 3.5 discute outros formalismo para tratamento de incerteza na Web Semântica e compara estes com o novo formalismo proposto neste trabalho, mostrando que novidades do PR-OWL 2 RL justificam sua proposição.

3.1 Redes Bayesianas

Rede bayesiana é um formalismo que se baseia na teoria dos grafos, na probabilidade condicional, na independência condicional e no Teorema de Bayes para criar modelos gráficos probabilísticos utilizados para modelagem e inferência de domínios que envolvem incerteza.

O Teorema de Bayes, apresentado abaixo, permite que se atualize as probabilidades de uma proposição a partir de novas informações. Este teorema é a base de todos os sistemas de Inteligência Artificial para inferência probabilística [72].

Teorema 3.1.1 (Teorema de Bayes) *Seja E uma sequência de evidências e H_k, k = 1,...n uma partição do espaço em hipóteses possíveis, então:*

$$P(H_i|E) = \frac{P(H_i)P(E|H_i)}{\sum_{k=1}^{n}(P(H_k)P(E|H_k))}$$

As redes bayesianas são representadas por um grafo acíclico direcionado (DAG) onde os nós representam variáveis aleatórias, que podem assumir valores discretos ou contínuos. Os arcos representam dependências probabilísticas entre as variáveis: um arco do nó x_i (nó pai) para o nó x_j (nó filho) indica que o valor da variável x_j depende do valor da variável x_i. A dependência é quantificada por uma tabela de probabilidade condicional (CPT). A distribuição de probabilidade conjunta pode ser expressa como um produtório das probabilidades condicionais de cada nó da rede, conforme ilustrado na Equação 3.1.

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|pais(x_i)) \tag{3.1}$$

A Figura 3.1 ilustra o exemplo *Family Out*, apresentado em [20], para exemplificar os conceitos de redes bayesianas. A rede modelada o raciocínio feito por uma pessoa chegando em casa a fim de concluir se sua família se encontra ou não. Quando a família sai, frequentemente deixa as luzes do quintal ligadas e o cachorro fora de casa. O cachorro porém, pode ter sido colocado pela família para fora de casa por estar com problemas estomacais. Escutar latidos do cachorro ao se aproximar da casa pode indicar que o cachorro está fora de casa, porém, estes também podem ser latidos de outros cachorros da vizinhança. Cada uma destes eventos é transformado em uma variável aleatória booleana, representada por um nó na rede bayesiana. A Figura 3.1 apresenta também as CPTs de cada nó, onde LL se refere ao nó "Luzes Ligadas", FS ao nó "Família Saiu" e assim por diante.

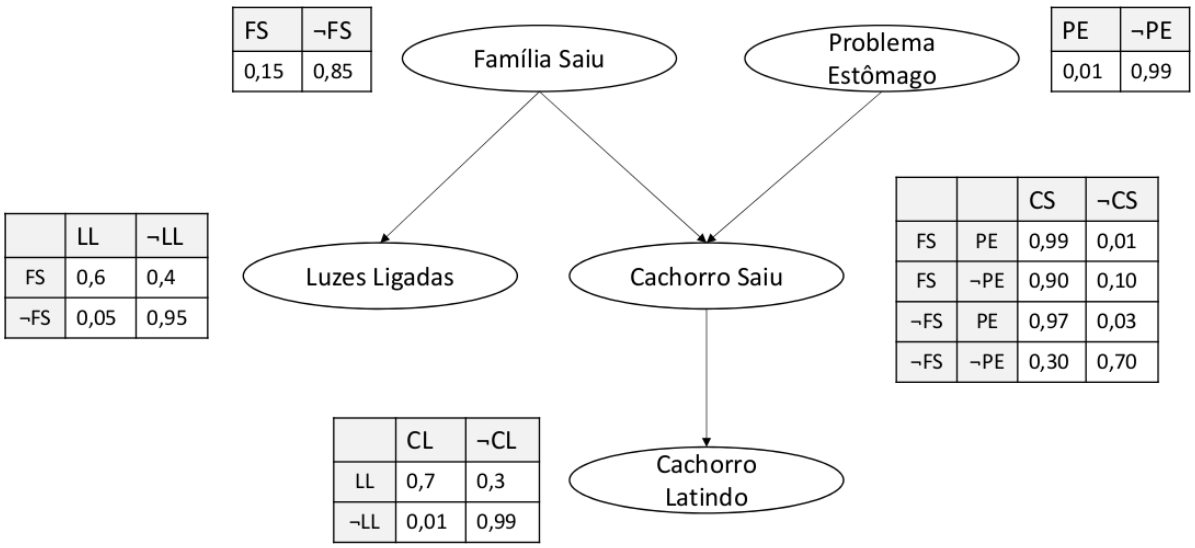

Figura 3.1: Rede bayesiana *Family Out*

A independência condicional ocorre quando uma variável torna-se independente de

outra, dada o valor de uma terceira. A notação $X_J \perp X_L | X_K$ significa que a variável aleatória X_J é condicionalmente independente de X_L dado X_K. Em uma rede bayesiana, cada nó é independente de seus nós não descendentes no grafo dado o estado de seus nós pais [20]. No exemplo *Family Out*, se soubermos que o cachorro está fora de casa, a probabilidade de escutar o latido se torna independente das variáveis aleatórias "Família Saiu" e "Problema Estômago", pois ela pode ser determinada unicamente pela variável "Cachorro Saiu". Esta propriedade permite que a distribuição de probabilidade de um nó seja definida apenas a partir da distribuição de probabilidade de seus nós pais na rede. Uma rede bayesiana com frequência é exponencialmente menor que a distribuição conjunta enumerada explicitamente [72], viabilizando o raciocínio probabilístico em domínios contendo grandes quantidades de variáveis aleatórias.

A independência condicional em redes bayesianas pode ser verificada utilizando o critério de d-separação [45]:

Teorema 3.1.2 (d-separação) *Duas variáveis A e B em um DAG são d-separadas se para todos os caminhos entre A e B existe uma variável intermediária W tal que:*

a) a conexão é serial ou divergente e o estado W é conhecido, ou,

b) a conexão é convergente e nem W ou seus descendentes recebem qualquer evidência.

A Figura 3.2 ilustra este critério. As probabilidades são propagadas por meio de conexões seriais e divergentes (arcos verdes), mas nas conexões convergentes elas avançam apenas caso exista evidência para a variável intermediária W ou para uma de suas descendentes.

A inferência exata em redes bayesianas possui complexidade NP-Hard. Os algoritmos mais conhecidos se baseiam na montagem de agrupamentos (cliques) transformando a rede em uma árvore de junção, na qual as probabilidades podem ser propagadas em tempo polinomial.

Redes bayesianas são utilizadas em diversos tipos de aplicações, como por exemplo, filtro de spams, robótica, sistemas de diagnóstico, análise de mercado financeiro e aplicações militares.

3.2 Redes Bayesianas Multi-Entidades

Redes Bayesianas Multi-Entidades (MEBN) é uma linguagem para representação de bases de conhecimento probabilísticas de primeira ordem [51]. O MEBN estende as redes bayesianas ao incorporar o poder de expressividade da lógica de primeira ordem, resolvendo a principal limitação das redes bayesianas: a impossibilidade de representar situações onde o número de variáveis aleatórias envolvidas é desconhecido.

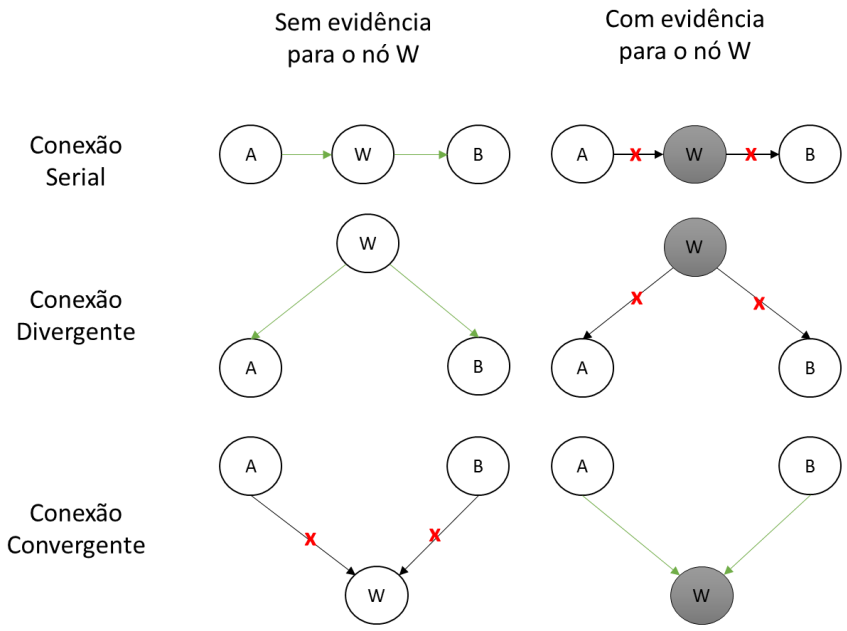

Figura 3.2: Critérios de d-Separação

O MEBN modela o domínio através das entidades envolvidas, seus atributos e seus relacionamentos, funcionando como um *template*, onde a quantidade real de entidades, e seus relacionamentos, só precisam ser conhecidos no momento da instanciação do modelo.

Uma Teoria MEBN (MTheory) é formada por um conjunto de Fragmentos MEBN (MFrags), que satisfazem restrições de consistência que garantem a existência de uma única distribuição conjunta de probabilidade sobre o conjunto de variáveis aleatórias [51].

MFrags são compostas por nós residentes, nós de entrada e nós de contexto. Nós residentes e nós de entrada representam as propriedades e relacionamentos das entidades, possuindo argumentos (variáveis ordinárias) a serem preenchidos com entidades durante a instanciação do modelo. Os nós residentes possuem sua tabela de distribuição de probabilidade local (LPD) definida na MFrag onde se encontra, enquanto o nó de entrada possui sua LPD definida em outra MFrag, na qual ele é residente. Os nós de contexto são expressões da lógica de primeira ordem que definem restrições a serem observadas para que as distribuições de probabilidade locais da MFrag sejam válidas. Caso os nós de contexto não possam ser satisfeitos, será utilizada uma distribuição *default* para os nós da MFrag.

A Figura 3.3 ilustra a MFrag *Front of Enterprise*, do domínio *Fraudes em Licitações Públicas*. Esta ontologia, desenvolvida em parceria com a Controladoria Geral da União (CGU) e apresentada em [17], modela a possibilidade de ter havido fraude em uma licitação pública baseada em informações diversas, como por exemplo, probabilidade do comitê

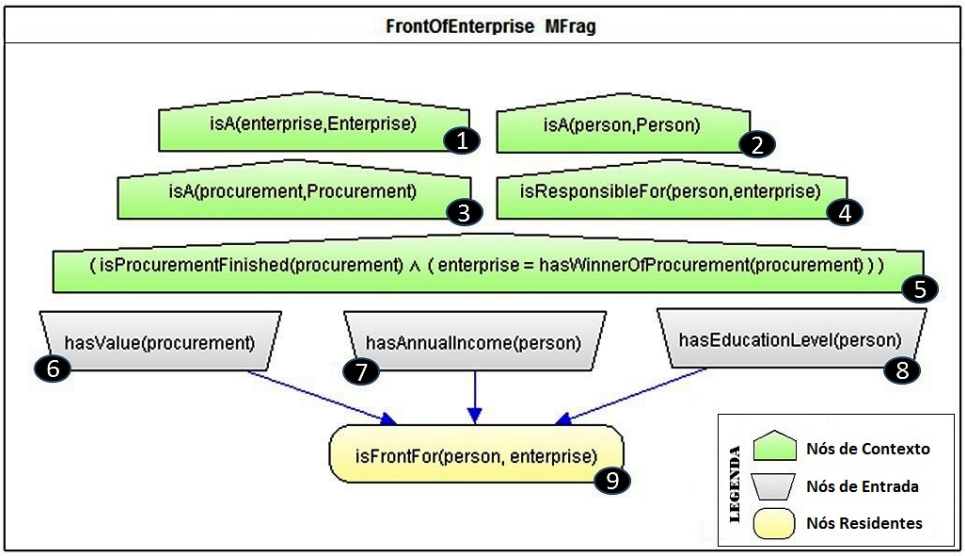

Figura 3.3: MFrag do domínio Fraudes em Licitações Públicas

responsável pela licitação ser suspeito, seja por ter algum dos seus membros relacionados com alguma pessoa das empresas concorrentes ou pelo membro ter um histórico de corrupção, possibilidade da pessoa responsável por uma empresa na licitação ser um laranja e possibilidade de responsáveis por empresas competidoras diferentes estarem relacionados. A Figura 3.4 ilustra a MTheory deste domínio.

A inferência no MEBN se dá pela instanciação da MTheory para responder a questionamentos (*queries*). A partir das *queries* e das entidades e evidências observadas (*findings*), é construída uma rede bayesiana de situação específica (SSBN), rede bayesiana minima suficiente para computar a resposta de uma *query*, onde *query* consiste em obter a distribuição posterior para um conjunto de variáveis aleatórias dado um conjunto de evidências e de variáveis de contexto [55]. Um algoritmo *Bottom-up* para geração de SSBN é apresentado em [51]. O algoritmo parte de um conjunto de questionamentos feitos pelo usuário e das evidências existentes na base para montar a SSBN de forma iterativa, acrescentando em cada iteração os nós pais do conjunto de nós gerado na iteração anterior. Mais detalhes do algoritmo são apresentados na Seção 5.1.

3.3 PR-OWL e PR-OWL 2

A linguagem PR-OWL (Probabilistic OWL), proposta por Costa [21] [23] em 2005, adiciona suporte a incerteza ao OWL. O PR-OWL consiste de um conjunto de classes, subclasses e propriedades que coletivamente formam um framework para a construção de ontologias probabilísticas [23].

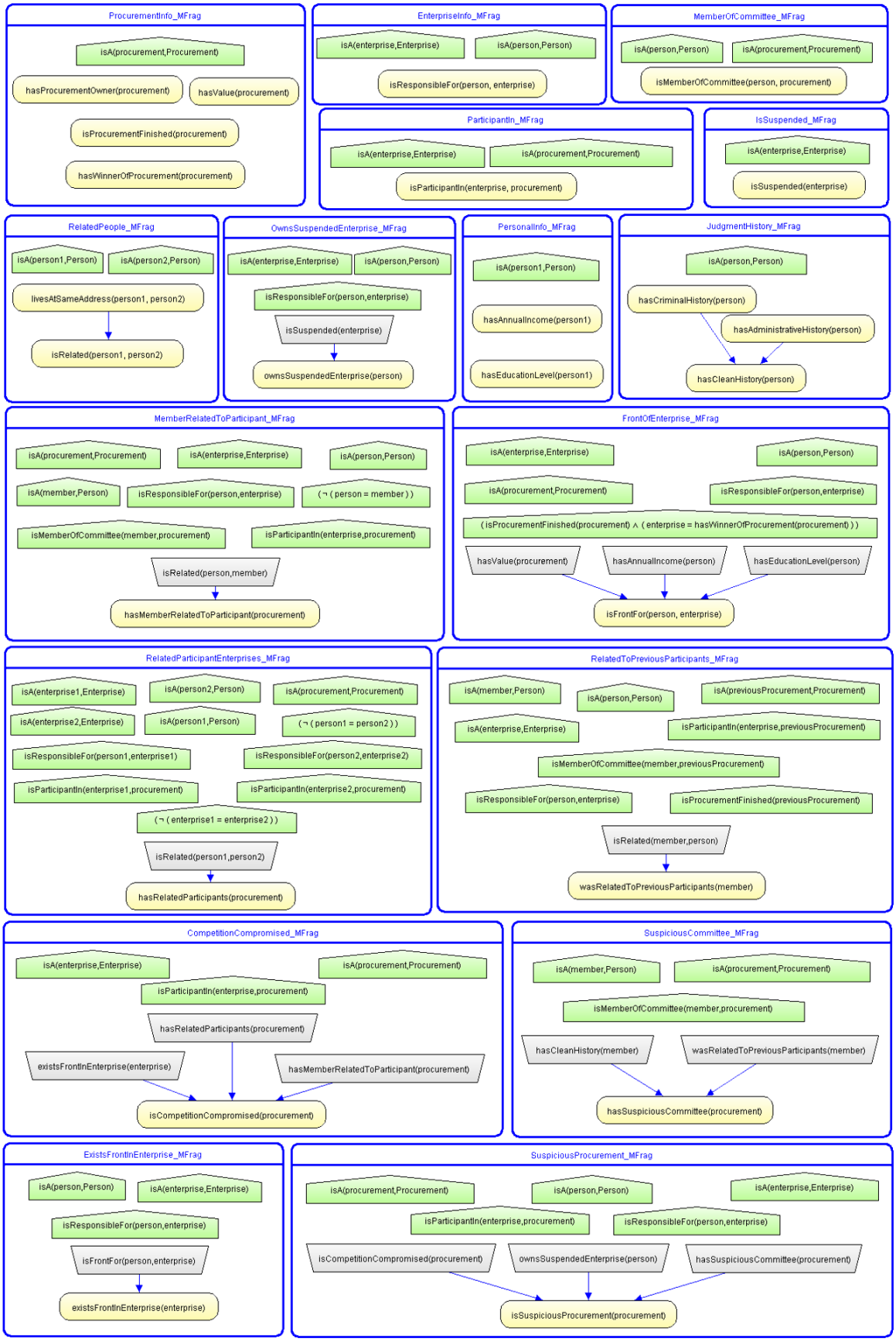

Figura 3.4: MTheory do domínio Fraudes em Licitações Públicas

A Figura 3.5, extraída de [23], apresenta como o PR-OWL modela os principais conceitos envolvidos na definição de uma teoria MEBN. O usuário modela a parte probabilística da ontologia utilizando a classe MTheory, composta por um conjunto de MFrags (conectadas a MTheory por meio de propriedades de objeto `hasMFrag`) que coletivamente devem formar uma MTheory consistente. As MFrags são compostas por nós (residentes, de entrada e de contexto). Os nós são variáveis aleatórias, possuindo uma distribuição de probabilidade vinculada e um conjunto exaustivo de estados possíveis, indivíduos da classe Entity.

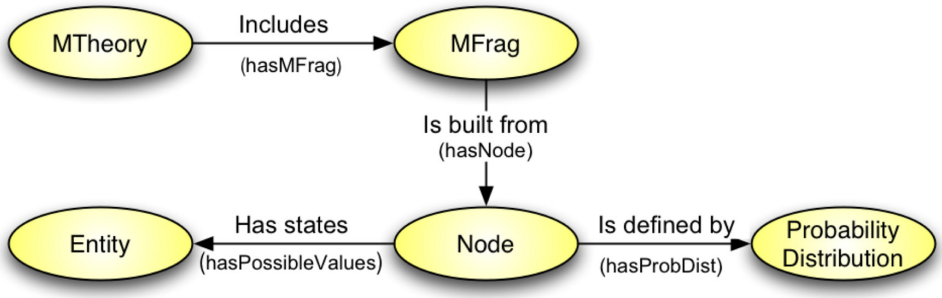

Figura 3.5: Conceitos do PR-OWL

A Figura 3.6, extraída de [23], apresenta os elementos presentes no PR-OWL, incluindo as sub-classes, elementos de suporte e relações necessárias para expressar a estrutura complexa do MEBN utilizando a sintaxe da OWL. O uso de reificações no PR-OWL é necessário porque as propriedades OWL são binárias, enquanto muitas das relações no modelo probabilístico proposto incluem mais de um individuo [23].

O PR-OWL é uma ontologia escrita em OWL DL, definindo cada uma de suas classes utilizando expressões lógicas, conforme ilustrado na Figura 3.7, que mostra as restrições que definem a classe `Domain_MFrag`. Um *reasoner* OWL DL pode ser utilizado para verificar se a ontologia probabilística montada pelo usuário é consistente com a definição do PR-OWL, verificando-se estas restrições.

Em 2011, Carvalho propôs o PR-OWL 2 [14] [9], uma extensão do PR-OWL que resolve duas de suas maiores limitações: a falta de mapeamento formal entre as variáveis aleatórias do PR-OWL e os conceitos definidos no OWL, e a falta de compatibilidade entre os tipos do PR-OWL e os já existentes no OWL [13].

Para resolver a primeira limitação, utiliza-se o relacionamento `definesUncertaintyOf`, que permite fazer a ligação entre uma variável aleatória do PR-OWL e uma propriedade do OWL. Adicionalmente, os relacionamentos `isSubjectIn` e `isObjectIn` são utilizados para definir o domínio e o contra-domínio da variável aleatória utilizando conceitos do OWL. Através desta característica, o PR-OWL 2 facilita a construção de ontologias híbridas, contendo declarações probabilísticas e determinísticas inter-relacionadas.

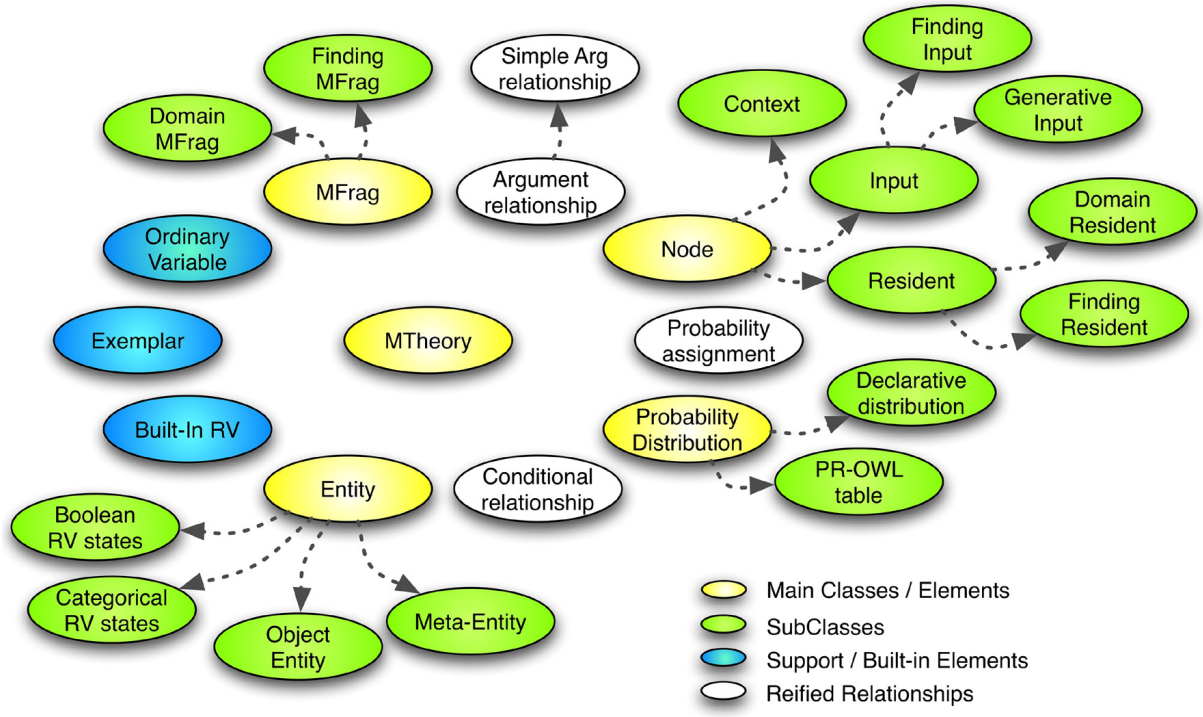

Figura 3.6: Elementos do PR-OWL

Para resolver a segunda limitação, o PR-OWL 2 mapeia os tipos do PR-OWL para tipos já existentes no OWL, eliminando a necessidade da redefinição. A classe `ObjectEntity`, por exemplo, é substituída pela classe `Thing`, enquanto a classe `CategoricalRVStates` é substituída por `DataOneOf`, caso seja necessário enumerar *data types*, ou `ObjectOneOf` para enumerar objetos [9]. `BooleanRVState` pode ser substituído pelo tipo de dados booleano e `MetaEntity` pelos conceitos do OWL [9].

O PR-OWL 2 contém ainda outras melhorias, como a capacidade de trabalhar com polimorfismo nas variáveis aleatórias.

3.4 Implementações do PR-OWL no UnBBayes

O UnBBayes é um framework desenvolvido para se trabalhar com raciocínio probabilístico. Criado pelo Grupo de Inteligência Artificial da UnB (GIA-UnB), e escrito em linguagem Java, o UnBBayes é distribuído como software *open-source* com licença GPL, disponibilizado através do SourceForge[1]. Implementa diversos formalismos, dentre os quais, redes bayesianas, diagramas de influência, Multiply-Sectioned Bayesian Network (MSBN), Object-Oriented Bayesian Network (OOBN), Hybrid Bayesian Network (HBN)

[1]http://sourceforge.net/projects/unbbayes

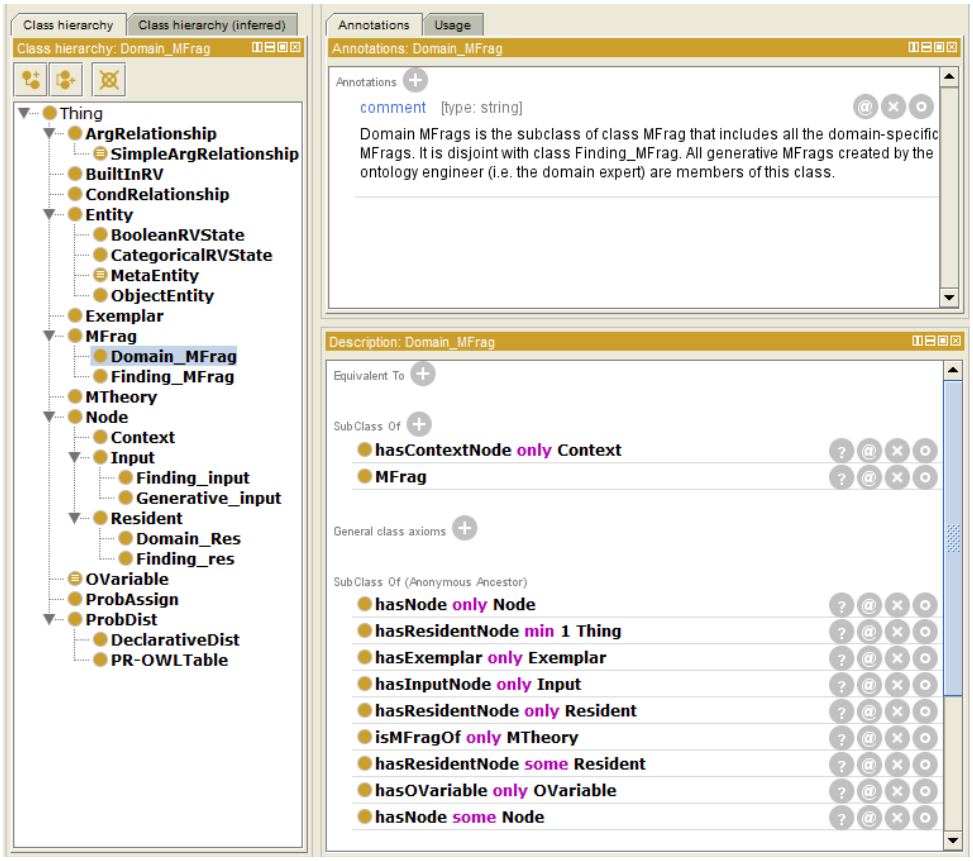

Figura 3.7: Definição da classe Domain_MFrag

e Multi-Entity Bayesian Network (MEBN). A arquitetura atual do UnBBayes é baseada em plug-ins [57], facilitando a implementação de novos formalismos.

O suporte a MEBN e PR-OWL foi desenvolvido em 2008 [12] [22], tendo sido a primeira implementação mundial de MEBN. A implementação incluí uma interface gráfica para a modelagem visual da MTheory (ilustrado na Figura 3.8); um sistema de representação de conhecimento e inferência; e um algoritmo para geração da SSBN, baseado no algoritmo Bottom-Up proposto em [51]. A persistência é feita utilizando o formato PR-OWL, permitindo a criação de ontologias probabilísticas de forma gráfica e intuitiva.

O plug-in de MEBN/PR-OWL utiliza como sistema de representação de conhecimento e inferência lógica o PowerLoom. O PowerLoom fornece linguagem e ambiente para construção de aplicações inteligentes, baseadas em conhecimento [19]. Utiliza como linguagem de representação uma variante do KIF (*Knowledge Interchange Format*), linguagem desenvolvida para a troca de conhecimento entre sistemas de computadores diferentes [34]. O PowerLoom é utilizado para salvar a base assertiva, contendo as entidades e as evidências para uma situação específica e como raciocinador, para avaliar as expressões em lógica de primeira ordem dos nós de contexto.

O povoamento da base de conhecimento é realizado em duas partes: primeiro é feito

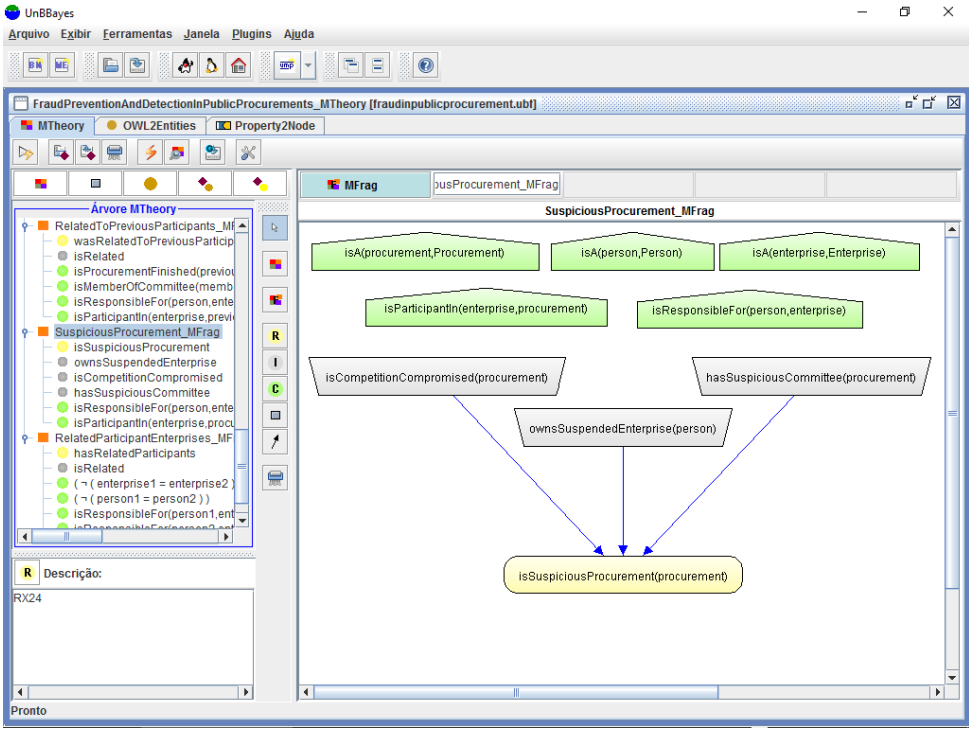

Figura 3.8: Edição de MTheory do UnBBayes

a geração e carregamento do TBox no PowerLoom, traduzindo a MTheory geradora para KIF; depois a ABox é carregada a partir de arquivo KIF informado pelo usuário, ou por meio da interface gráfica do UnBBayes, onde o usuário pode instanciar as classes, relacionamentos e propriedades. O usuário pode gravar arquivos KIF diferentes contendo conjuntos diversos de *findings*, desde que estes sejam compatíveis com a definição da TBox carregada. Como o PowerLoom possui suporte à lógica de primeira ordem, a tradução dos elementos do MEBN para os elementos da base de dados é feita de forma direta. Um formato extra, o UBF (*UnBBayes Format*) é utilizado para armazenar as características gráficas do modelo, como por exemplo, posição e tamanho dos nós na tela de edição, já que o PR-OWL não prevê suporte a armazenamento deste tipo de informação.

A geração da SSBN é feita utilizando uma implementação do algoritmo Bottom-Up proposto em [51], onde a montagem da rede parte dos nós de queries e dos nós de findings, gerando novos nós a partir das relações contidas nas MFrags e nas avaliações dos nós de contexto. O algoritmo é descrito em detalhes no Capítulo 4. O algoritmo implementado permite que apenas uma query seja realizada por inferência. Este algoritmo pode ser facilmente estendido para tratar múltiplas queries.

O plug-in para PR-OWL 2 foi implementado em 2011 [56], como uma validação da proposição do formalismo. Como o PR-OWL 2 permite a modelagem de ontologias híbridas, adicionando as informações sobre incerteza a uma ontologia determinística, surgiu a

necessidade de permitir ao usuário trabalhar simultaneamente com as declarações determinísticas e probabilísticas da ontologia. Para permitir a visualização e edição da parte determinística, optou-se pela integração do Protégé ao UnBBayes. O Protégé [70] é um framework *open-source* bastante popular e maduro utilizado para o trabalho com ontologias. O Protégé (versão 4.1) foi encapsulado como um painel interno no UnBBayes, permitindo que o usuário acesse todas as suas funcionalidades. O usuário modela a parte determinística da ontologia no Protégé, e a parte probabilística na interface de edição de MEBN. Um painel permite que o usuário faça o link entre propriedade OWL e nós residente, indicando que o segundo está definindo a incerteza relacionada ao primeiro. Este link será traduzido para o PR-OWL 2 utilizando as propriedades `definesUncertaintyOf`, `isObjectIn` e `isSubjectIn`. Os paineis podem ser visualizados na Figura 3.9.

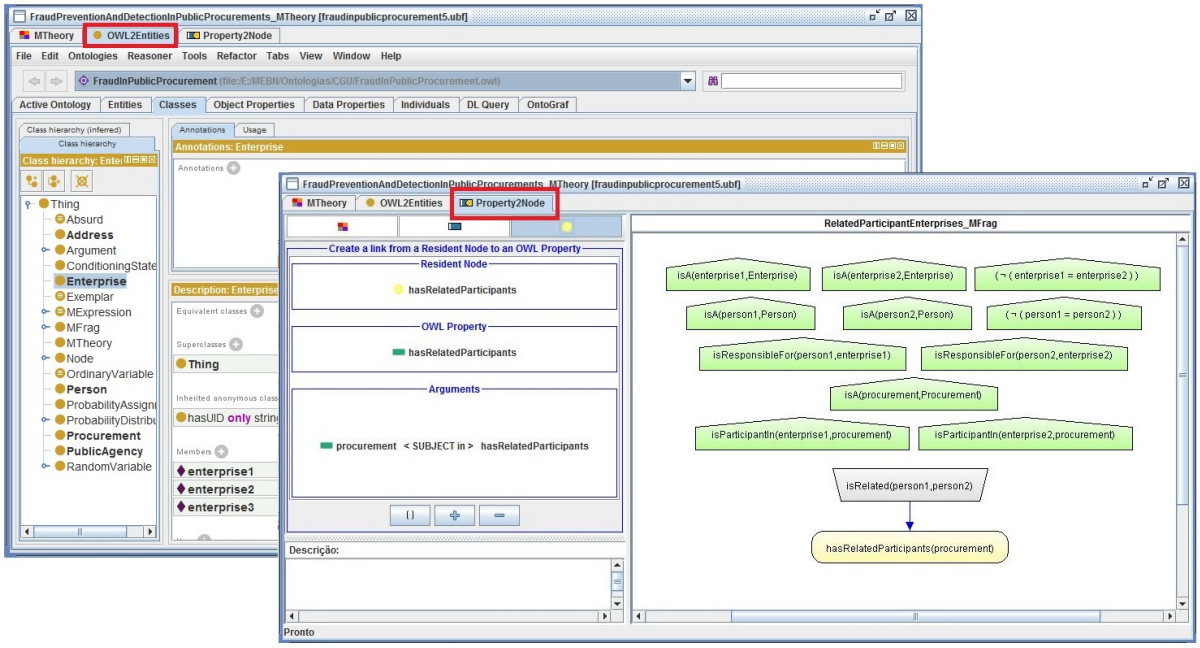

Figura 3.9: Painel do Protégé e de link entre Propriedades

A base de conhecimento passou a ser salva em OWL 2, eliminando a necessidade de gravação de arquivos KIF separados com a ABox. O HermiT [75], raciocinador padrão do Protégé, é utilizado para fazer a busca de informações na base de conhecimento e para avaliar as expressões dos nós de contexto. O HemiT é baseado em lógica descritiva, oferecendo suporte ao OWL 2 DL. Como os nós de contexto do PR-OWL 2 possuem expressividade de lógica de primeira ordem, diversas restrições foram necessárias nos formatos de fórmulas aceitas, conforme descrito na Seção 4.1.

3.5 Trabalhos Correlatos

Diversos formalismos foram propostos para tratar incerteza na Web Semântica. As abordagens baseadas em probabilidade são as mais numerosas. Segundo Predoiu [69] os modelos probabilísticos são uma escolha natural para representar os diferentes tipos de incerteza encontrados frequentemente na Web, e, de fato, foram desenvolvidas diversas extensões probabilísticas para as principais linguagens da Web Semântica e para lógicas descritivas relacionadas a Web Semântica. Este capitulo descreve apenas abordagens baseadas em probabilidade, por estas apresentam semelhanças com o trabalho proposto.

A Tabela 3.1 apresenta (em ordem cronológica) alguns dos principais formalismos criados com o objetivo de tratar incerteza na Web Semântica. Os formalismos são discutidos à seguir.

Tabela 3.1: Formalismos para tratamento de incerteza na Web Semântica

Autor	**Formalismo**	**Suporte Probabilístico**	**Suporte Lógico**
Koller 1997	P-CLASSIC	Redes Bayesianas	CLASSIC
Giuno 2002	P-$\mathcal{SHOQ}$(D)	Teoria da Probabilidade	$\mathcal{SHOQ}$(D)
Lukasiewicz 2002	P-$\mathcal{SHOIN}$(D)	Teoria da Probabilidade	$\mathcal{SHOIN}$(D)
Fukushige 2004	Fukushige 2004	Redes Bayesianas	RDF
Udrea 2005	pRDF	Lógica Probabilística	RDF
Yang 2005	OntoBayes	Redes Bayesianas	OWL
Costa 2005	PR-OWL	MEBN	OWL
Ding 2006	BayesOWL	Redes Bayesianas	OWL
Cozman 2008	CR$\mathcal{ALC}$	Redes Credais	$\mathcal{ALC}$
Carvalho 2011	PR-OWL 2	MEBN	OWL 2

Fukushige [32] propôs uma extensão para representação de incertezas no RDF, composta basicamente por um vocabulário para representação dos elementos de redes bayesianas. Outra abordagem envolvendo RDF, o pRDF (*Probabilistic* RDF), proposto em [80], apresenta uma extensão probabilística formal do RDF [69], formando uma lógica probabilística.

O OntoBayes [84] inclui anotações no OWL para representar probabilidades bayesianas e relacionamentos de dependência e permite inserir informações de incerteza em propriedades, sendo capaz de representar variáveis aleatórias e suas dependências através de distribuições de probabilidade condicional [9]. O raciocínio se dá traduzindo a ontologia para uma rede bayesiana. O BayesOWL [27], também baseado em redes bayesianas, permite a inferência quanto as classes da ontologia. Cada classe é mapeada para um nó no grafo, e links diretos entre os nós são gerados de acordo com um conjunto de regras. O PR-OWL [23] e o PR-OWL 2 [14], discutidos previamente neste capitulo, se baseiam no MEBN, permitindo a representação de modelos probabilísticos de primeira ordem.

Como o OWL é baseado em lógica descritiva, extensões probabilísticas destas são interessantes por poderem ser estendidas para subconjuntos do OWL. A P-CLASSIC [48] é uma extensão probabilística da lógica descritiva CLASSIC que adiciona probabilidades as propriedades formando redes bayesianas. Foi modelada para permitir raciocínio eficiente, sendo a inferência de subsunção de classe realizada em tempo polinomial [69]. CLASSIC, porém, é uma lógica descritiva pouco expressiva, estando distante da expressividade do $\mathcal{SHOIN}$(D), lógica na qual o OWL DL é baseado.

Lukasiewicz apresentou diversas extensões de lógicas descritivas baseadas semanticamente na noção de implicações probabilísticas lexicográficas [54]. O P-$\mathcal{SHOQ}$(D) [35] possui expressividade próxima à do $\mathcal{SHOIN}$(D), não contendo apenas a capacidade de descrever propriedades inversas [69]. Ao P-$\mathcal{SHOQ}$(D) [35], seguiu-se o P-$\mathcal{SHIF}$(D) [54], extensão da lógica descritiva $\mathcal{SHIF}$(D), base do OWL-Lite, e o P-$\mathcal{SHOIN}$(D) [54], extensão do $\mathcal{SHOIN}$(D), base do OWL-DL. Segundo [25], o P-$\mathcal{SHOIN}$(D) é a extensão probabilística para lógica descritiva mais expressiva das abordagens existentes.

Cozman apresenta uma extensão da lógica descritiva $\mathcal{ALC}$ baseada em redes credais, chamada CR$\mathcal{ALC}$ [25], que adota uma semântica baseada em interpretações para a definição de incerteza. Redes credais são modelos gráficos para tratar crenças precisas e imprecisas [24] (redes bayesianas ordinárias permitem apenas o tratamento de crenças precisas). Quanto a expressividade, o CR$\mathcal{ALC}$ mantem todas as construções da $\mathcal{ALC}$, porém permite nomes de conceitos apenas do lado esquerdo das expressões [25].

A justificativa para a criação do PR-OWL 2 RL é a necessidade de representar modelos bayesianos complexos e de se trabalhar com bases assertivas escaláveis, comuns em domínios reais, onde a quantidade de declarações armazenadas em bases de dados pode chegar a milhões. Até onde é do nosso conhecimento, não existem abordagens para tratamento de incerteza na Web Semântica aplicadas diretamente a *triplestores* RDF. Além disso, a maioria dos formalismos para tratamento de incerteza na Web Semântica não possui implementação prática que possa ser utilizada diretamente para a construção de ontologias na Web Semântica, foco deste trabalho. Este trabalho propõe um framework para trabalhar com o novo formalismo proposto, discutindo as questões necessárias para a implementação.

Capítulo 4

Modelagem do PR-OWL 2 RL

As versões atuais do PR-OWL possuem limitações de escalabilidade e expressividade que restringem o seu uso em domínios reais. Este capítulo discute estas limitações e propõe como solução a criação de uma versão do PR-OWL baseada no *profile* OWL 2 RL e em bases de dados RDF (*triplestores*), adequada a domínios com grandes bases assertivas.

O capítulo está organizado da seguinte forma: a Seção 4.1 discute as limitações das versões do PR-OWL e de suas implementações no UnBBayes; a Seção 4.2 introduz a linguagem PR-OWL 2 RL; a Seção 4.3 descreve a sintaxe do formalismo proposto; a Seção 4.4 apresenta os formatos de nós de contexto aceitos no PR-OWL e descreve a avaliação destes utilizando a linguagem SPARQL.

4.1 Limitações das Versões do PR-OWL Existentes

O PR-OWL e sua extensão, o PR-OWL 2, estendem o OWL adicionando suporte a incerteza. Para tal é utilizado o formalismo MEBN, uma lógica probabilística de primeira ordem: o PR-OWL permite a criação de ontologias probabilísticas ao fornecer o ferramental lógico necessário para modelar teorias MEBN utilizando a linguagem OWL.

O OWL 2 DL, versão do OWL utilizada no PR-OWL 2, é baseado na lógica descritiva $\mathcal{SROIQ}$(D). Lógicas descritivas são subconjuntos da lógica de primeira ordem e restringem a sua expressividade em troca de uma maior tratabilidade. O PR-OWL não especifica como deve ser realizado o raciocínio com as expressões em lógica de primeira ordem do MEBN, ficando estas à cargo da implementação. Um caminho natural consiste em utilizar raciocinadores já existentes para OWL, já que a avaliação das fórmulas é feita a partir das expressões terminológicas e assertivas da ontologia.

A implementação da linguagem PR-OWL 2 no UnBBayes utiliza o HermiT [75] como raciocinador OWL 2. O HermiT é utilizado na inferência do MEBN para recuperar informações na base assertiva e para resolver as expressões lógicas presentes nos nós de

contexto. Raciocinadores baseados em calculo tableau, como o Pellet [66], o Racer [41] e o FaCT++ [79], (ambos implementando a lógica descritiva $\mathcal{SROIQ}$(D)) realizam testes de consistência tentando construir um modelo para a base de conhecimento [75]. O HermiT implementa um calculo "Hipertableau", que reduz consideravelmente a quantidade de possíveis modelos a serem considerados [75], o tornando uma alternativa eficiente.

Raciocinadores OWL DL são bons para o trabalho com ontologias complexas, oferecendo respostas completas, porém apresentam problemas quanto a escalabilidade. Donini [28] apresenta duas fontes de complexidade no calculo tableau: o *AND-Branching*, responsável pelo tamanho exponencial de um simples modelo candidato, e o *OR-Branching*, responsável pelo número exponencial de diferentes modelos candidatos. Esta complexidade limita a performance e escalabilidade dos raciocinadores: quanto maior a base de conhecimento, mais tempo e memória serão necessários para a avaliação.

A Figura 4.1 apresenta a complexidade temporal de inferência para as diferentes versões do OWL (Conforme [62]).

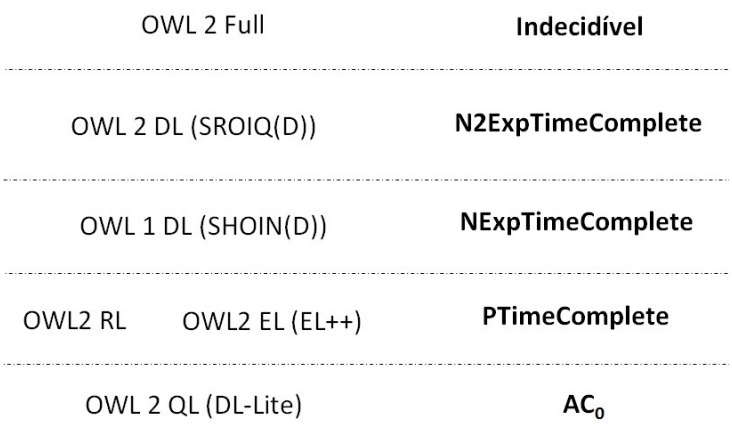

Figura 4.1: Complexidade temporal das diversas versões do OWL

Os principais problemas de inferência para OWL 2 DL (verificação de consistência da ontologia, satisfabilidade e subsunção de expressões de classe e verificação de instâncias) possuem complexidade temporal N2EXPTIME-completo [62]: estão na classe de problemas solucionáveis por algoritmos não determinísticos em tempo ao menos duas vezes exponencial no tamanho da entrada. Como o PR-OWL 2 se baseia no OWL 2 DL, torna-se difícil a criação de implementações escaláveis.

O OWL 1 DL possui complexidade temporal NEXPTIME-completo, e portanto, uma implementação do PR-OWL 1 baseada em um raciocinador OWL DL teria problemas semelhantes à implementação do PR-OWL 2. A implementação do PR-OWL 1 feita no UnBBayes, no entanto, utiliza o PowerLoom para fazer a avaliação das expressões da lógica de primeira ordem. Para tal, a MTheory (TBox) e a base assertiva devem ser convertidas

para o formato KIF e adicionada a base de conhecimento do PowerLoom. O PowerLoom realiza inferência através da dedução natural, utilizando encadeamento *forward* e *backward* para derivar o que logicamente segue dos fatos e regras existentes na base de conhecimento [19]. Apesar de não implementar uma lógica descritiva, o Powerloom possui classificadores que podem classificar hierarquias de conceitos e relacionamentos, utilizando o poder de expressividade da lógica de primeira ordem. Não há um estudo formal quanto a complexidade do raciocínio feito no PowerLoom, o que dificulta um melhor estudo quanto a sua escalabilidade. A dificuldade de integração do conhecimento expresso na ontologia em OWL DL com a base de conhecimento do PowerLoom, somada a falta de integração do PR-OWL com o OWL no nível semântico, dificulta a adoção da linguagem e desta estratégia de implementação em ontologias probabilísticas desenvolvidas a partir de ontologias determinísticas.

Afim de avaliar de forma prática o tamanho da ontologia que o usuário poderia utilizar na implementação de PR-OWL 2 no UnBBayes, foram feitos testes utilizando o Lehigh University Benchmark (LUBM) [40]. Este *benchmark* é bastante utilizado para testes de performance em raciocinadores OWL e *triplestores* RDF. O *benchmark* é composto por uma ontologia em OWL *Lite* modelando um domínio acadêmico, um gerador automático de testes que possibilita a criação de bases ABox com quantidades variáveis de assertivas, e um conjunto de quatorze consultas de complexidades variadas. A Tabela 4.1 apresenta a quantidade de declarações e o tamanho de algumas destas bases de teste.

Tabela 4.1: Tamanho das bases de testes do LUBM

	Tamanho	Inst. Classes	Inst. Propriedades
LUBM 1	8,02 MB	20.659	82.415
LUBM 10	102 MB	263.427	1.052.895
LUBM 100	1,06 GB	2.779.262	11.096.694
LUBM 500	12 GB	13.839.128	55.240.636

Utilizando o UnBBayes, executando em uma computador com processador Intel i5 com 6 GB de memória RAM (3 GB dedicados ao processo) não foi possível carregar e inicializar a máquina de inferência com a versão LUBM 100. A inicialização do HermiT consiste em construir a hierarquia de classes, classificar as propriedades de objeto e de dados, computar as instâncias de todas as classes e propriedades de objeto, e calcular os indivíduos iguais segundo o relacionamento `sameAs` do OWL. Esta inicialização é necessária para permitir a execução posterior das *queries*. Conforme a tabela apresentada, o LUBM 100 contém 2.779.262 instâncias de classes e 11.096.694 instâncias de propriedades. Quando armazenado em um arquivo OWL em formato XML, o LUBM 100 possui apenas 1,06 GB, deixando claro que a estrutura utilizada pelo UnBBayes (que utiliza

internamente o Protégé para armazenar a ontologia) acrescenta um grande *overhead* na implementação do PR-OWL 2.

Além dos problemas de escalabilidade devido á versão do OWL que serve como base para o PR-OWL 2, a implementação do plug-in para a linguagem no UnBBayes possuí problemas na geração da rede bayesiana de situação específica para domínios que possuam grande base assertiva. O algoritmo implementado inicia a sua execução a partir dos nós de *query* do usuário e das evidências existentes na base de conhecimento, gerando, de forma iterativa, os nós ascendentes destes nós, sendo necessário para isto a avaliação dos nós de contexto das MFrags envolvidas. Uma fase de poda posterior retira os nós que não influenciam probabilisticamente os nós de *query*. Para domínios que possuem grande base assertiva, uma grande quantidade de nós de redes possivelmente desconexas serão geradas, e possivelmente descartadas na fase de poda, tornando o algoritmo ineficiente. A complexidade na avaliação das fórmulas dos nós de contexto utilizando máquina de inferência de lógica descritiva tornam o algoritmo ainda mais ineficiente, ao avaliar fórmulas de nós de contexto de partes da rede podadas da SSBN final.

O plug-in desenvolvido também possuí limitações práticas, como por exemplo a necessidade das novas instâncias e relacionamentos serem inseridas manualmente pelo usuário, utilizando a interface do Protégé. Esta tarefa tende a ser lenta e repetitiva quando há uma grande quantidade de inserções a serem efetuadas. O trabalho com ontologias contendo milhares de assertivas só é possível através do uso de API e ferramentas que automatizem esta inserção.

Limitações de expressividade no PR-OWL 2 ocorrem devido as restrições nos formatos dos nós de contexto para permitir a sua avaliação utilizando raciocinadores OWL. Uma MFrag representa uma distribuição de probabilidade de instâncias de seus nós residentes, dados os valores das instâncias dos seus pais no grafo. Para que estas distribuições se apliquem, as restrições dadas pelos nós de contexto devem ser satisfeitas. A implementação do PR-OWL 2 necessitou de simplificações nos formatos de fórmulas aceitos, já que a lógica descritiva do OWL 2 DL é um subconjunto da lógica de primeira ordem. Conforme Matsumoto [56]:

> A lógica descritiva implementa nativamente as variáveis aleatórias built-in do MEBN (i.e. operações como *and*, *or*, *not*, *forAll*, ou *exists* são implementadas nativamente). No entanto, por causa de diferenças de expressividade entre a lógica de primeira ordem (utilizada em expressões das fórmulas dos nós de contexto) e a lógica descritiva, as fórmulas dos nós de contexto não podem ser diretamente mapeadas para consultas na ontologia PR-OWL 2, principalmente por causa das consultas DL não poderem ser feitas para diversas variáveis ordinárias simultaneamente.

A Tabela 4.2 apresenta as restrições nos formatos dos nós de contexto feitas na implementação do PR-OWL 2. Na tabela, ov são as variáveis ordinárias (*ordinary variables*) que

serão preenchidas com entidades durante a avaliação da MFrag, BooleanRV são variáveis aleatórias booleanas (*boolean random variables*), nonBooleanRV são variáveis aleatórias não booleanas e CONST são constantes. Conforme a tabela, esta primeira implementação permite apenas fórmulas simples, sem o uso de conectivos e quantificadores.

Tabela 4.2: Formatos de fórmulas de nós de contexto válidos no plug-in de PR-OWL 2

Fórmula	Negação
$ov_1 = ov_2$	NOT ($ov_1 = ov_2$)
booleanRV(ov_1 [, ov_2 , . . .])	NOT booleanRV(ov_1 [, ov_2, . . .])
ov_0 = nonBooleanRV(ov_1)	NOT (ov_0 = nonBooleanRV(ov_1))
ov_0 = nonBooleanRV(ov_1 [, ov_2, . . .])	
CONST = nonBooleanRV(ov_1 [, ov_2 , . . .])	
nonBooleanRV(ov_1 [, ov_2 , . . .]) = CONST	
nonBooleanRV(ov_1) = ov_0	NOT (nonBooleanRV (ov $_1$) = ov_0)
nonBooleanRV(ov_1 [, ov_2 , . . .]) = ov_0	

4.2 Tratamento de Incerteza na Web Semântica Utilizando o PR-OWL 2 RL

Este trabalho propõe um formalismo escalável para tratamento de incerteza na Web Semântica que permite o trabalho de ontologias com grandes bases assertivas. Para tal, o formalismo estende o PR-OWL 2 utilizando bases de dados RDF (*Triplestores*) em conjunto com o profile OWL 2 RL.

Triplestores possuem suporte apenas a versões do OWL com expressividade limitada, não permitindo o trabalho com o OWL 2 DL, versão utilizada pelo PR-OWL 2, justificando a necessidade de se trabalhar com uma versão mais restritiva. O *profile* OWL 2 RL é implementado por diversas *triplestores* e apresenta boa escalabilidade sem sacrificar muito a expressividade.

O PR-OWL 2 RL consiste em uma ontologia escrita em OWL 2 RL que permite que o usuário modele ontologias probabilísticas utilizando redes bayesianas multi-entidades (MEBN) como suporte à incerteza.

A ontologia probabilística em PR-OWL 2 RL deve seguir as restrições do OWL 2 RL e ser serializada em RDF, permitindo que a inferência seja feita por *triplestores* com suporte ao *profile*. Além disso, a ontologia deve conter uma MTheory válida, que defina uma distribuição conjunta única de probabilidade sobre as suas variáveis aleatórias, para que possa ser feito o raciocínio probabilístico. O raciocínio probabilístico, feito a partir de uma máquina de inferência específica para o PR-OWL 2 RL, consiste em gerar a

rede bayesiana de situação específica a partir das *queries* e das informações da base de conhecimento (explicitas e implícitas, sendo estas últimas geradas a partir da expansão das regras do profile OWL 2 RL).

A proposta do novo formalismo requer:

1. revisar o PR-OWL 2, originalmente escrito em OWL 2 DL, de acordo com as restrições sintáticas do profile OWL 2 RL;
2. definir como fazer a avaliação das fórmulas dos nós de contexto, originalmente em lógica de primeira ordem, utilizando bases de dados RDF;
3. definir um algoritmo escalável para a geração de SSBN, já que o algoritmo Bottom-Up proposto previamente não é adequado para domínios que contenham grandes bases assertivas.

As Seções 4.3 e 4.4 discutem as duas primeiras questões, enquanto o Capítulo 5 discute a terceira.

4.3 Adaptação do PR-OWL 2 ao Profile OWL 2 RL

Como o PR-OWL 2 é escrito em OWL 2 DL, alguns ajustes são necessários para adaptá-lo ao OWL 2 RL, já que este profile impõe restrições sintáticas severas às expressões OWL.

Executando um validador desenvolvido pela Universidade de Manchester [1], foram encontradas as seguintes não conformidades, exemplificadas na Figura 4.2:

1. uso de expressões não permitidas em expressões de superclasse;
2. uso de expressões não permitidas em expressões de subclasse;
3. uso de expressões não permitidas em expressões de equivalência;
4. uso de contradomínios não suportados em propriedades de dados.

A primeira não conformidade ocorre por diversas razões, como por exemplo o uso do quantificador existencial e disjunções (`or`) no lado direito de expressões de subclasse ou em expressões de domínio ou contradomínio, o uso de `owl:Thing` como superclasse ou em expressões de domínio ou contradomínio, e o uso da restrição qualificada `exactly` para limitar a quantidade de objetos de uma propriedade.

A segunda não conformidade ocorre na classe `owl:Thing`, que é definida como uma subclasse da restrição "`hasUID only String`". O OWL não possui a assunção de nome

[1] http://mowl-power.cs.man.ac.uk:8080/validator/

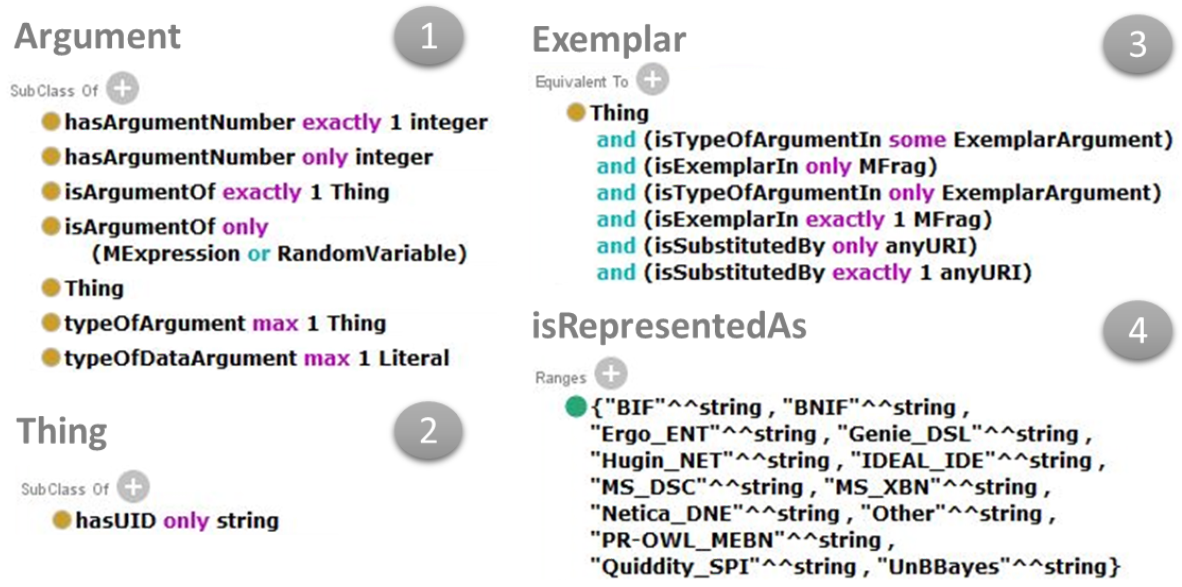

Figura 4.2: Não conformidades do PR-OWL 2 com o profile OWL 2 RL

único: dois indivíduos com URIs diferentes podem se referir ao mesmo elemento. A propriedade `hasUID` é utilizada para garantir que cada individuo em uma ontologia PR-OWL tenha um identificador único, que permita verificar facilmente se indivíduos com nomes diferentes se referem ao mesmo individuo. Esta verificação é necessária para o trabalho com MEBN para garantir que cada estado possível e cada argumento utilizado na instanciação de um nó são únicos.

A terceira não conformidade ocorre em expressões de equivalência de classes (`equivalent`) que incluem conjunções (`and`), expressões de cardinalidade min/max/exactly e quantificadores universal e existencial. O OWL 2 RL é muito restritivo em relação a expressões de equivalência, permitindo apenas classes, interseções e expressões `hasValue`.

Por fim, a quarta não conformidade ocorre na expressão de contradomínio `isRepresentedAs`, onde são listados todos os formatos possíveis para representar as distribuições probabilísticas.

Para adaptar o PR-OWL 2 ao OWL 2 RL, uma possibilidade seria reescrever todas as expressões em OWL 2 DL para OWL 2 RL. Esta alternativa no entanto não é possível devido a menor expressividade do OWL 2 RL. Expressões de subclasse envolvendo quantificador existencial no lado direito, por exemplo, não podem ser expressas no profile. Uma outra opção seria reescrever as expressões possíveis de serem reescritas, cortar as não possíveis e deixar a cargo da máquina de inferência PR-OWL a validação completa da ontologia probabilística. O problema com esta abordagem é que a ontologia resultante é de difícil entendimento e formalização, já que apenas parte das definições lógicas ficam

explicitadas na ontologia. A alternativa escolhida neste trabalho consiste em transformar o PR-OWL 2 em uma ontologia leve, contendo apenas a hierarquia de classes e as definições das propriedades de objeto e de dados (com as definições de domínio e contradomínio). Todo o resto da validação da consistência do modelo probabilístico fica a cargo da máquina de inferência PR-OWL.

O PR-OWL 2 RL, portanto, é composto por um conjunto de classes e propriedades que permitem a modelagem de ontologias probabilísticas utilizando teorias MEBN, sendo necessário que a ontologia modelada pelo usuário seja uma MTheory válida, requisito necessário para permitir a inferência utilizando uma máquina de inferência MEBN. No Apêndice A o leitor pode encontrar a hierarquia de classes do PR-OWL 2 RL e a listagem das propriedades de dados e de objetos existentes na linguagem.

Para cada classe são definidas as superclasses e as classes com as quais ela é disjunta (`Disjoint with`), conforme ilustrado na Figura 4.3, que apresenta as simplificações feitas nas expressões que definem a classe `DeclarativeDistribution`.

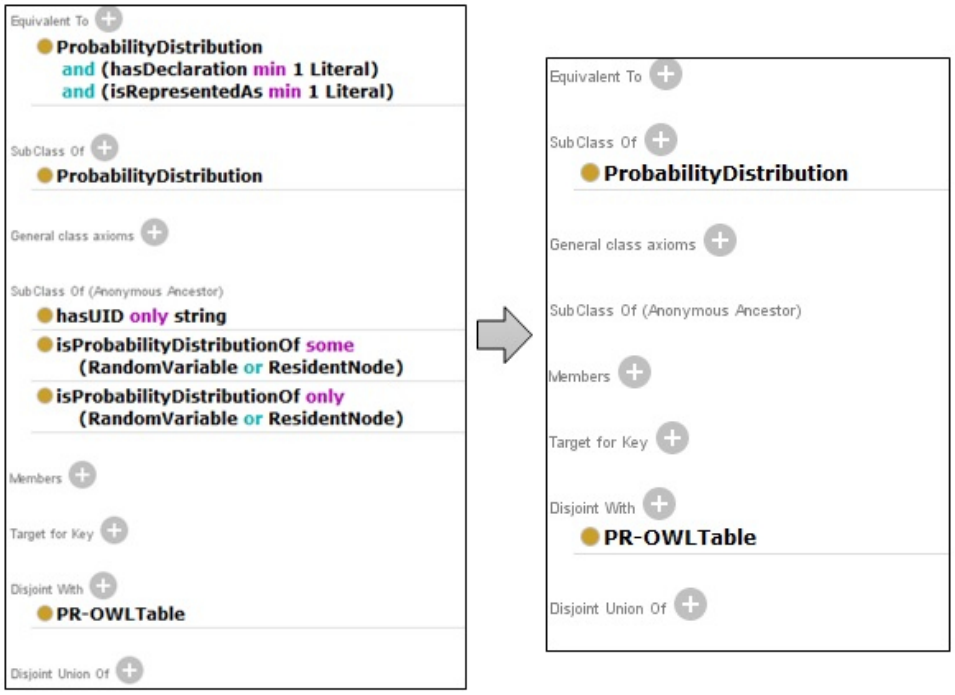

Figura 4.3: Simplificações na classe DeclarativeDistribution feitas no PR-OWL 2 RL

A classe `Absurd`, que era definida apenas como equivalente ao indivíduo `Absurd` foi substituída pela classe `LogicalConstant`, que possui apenas o indivíduo `Absurd`. Isto foi necessário porque o uso de `equivalent` com um `NamedIndividual` não é permitido em OWL 2 RL. Como no PR-OWL 2 são utilizados as constantes `true` e `false` já existentes no OWL, apenas a nova constante `Absurd` foi incluída como individuo desta classe. As outras classes correspondem as já definidas no PR-OWL 2 original.

A propriedade `hasUID` foi retirada da classe `owl:Thing`. A verificação de entidades iguais no PR-OWL 2 RL é feita utilizando a semântica do OWL, onde a partir das bases assertivas e terminológicas o raciocinador verifica se a propriedade `owl:sameAs` é atendida por um par de entidades. Esta alteração retira a obrigatoriedade do usuário do PR-OWL 2 especificar um identificador único para cada classe utilizada como possível estado ou argumento em uma variável aleatória, mas o obriga a especificar explicitamente se dois individuos são iguais (o que pode ser feito através da propriedade `owl:sameAs`) ou diferentes (propriedade `owl:differentFrom`). Uma classe onde todos os indivíduos são diferentes pode fazer esta proposição utilizando a propriedade `owl:AllDiferent`.

A ontologia modelada pelo usuário deve seguir as restrições do profile OWL 2 RL e, adicionalmente, as restrições definidas pelo Teorema PR1. O profile OWL 2 RL é baseado na semântica direta do OWL 2 (uma extensão da semântica da lógica descritiva $\mathcal{SROIQ}$ [63]), enquanto o conjunto de regras OWL 2 RL/RDF é baseado na semântica RDF. O Teorema PR1 [62] define restrições que devem ser satisfeitas de forma que a aplicação do conjunto de regras OWL 2 RL/RDF produza resultados corretos na semântica direta do OWL 2: as *queries* devem ser relacionadas a base assertiva; uma mesma IRI não pode ser utilizada para identificar mais de um tipo de entidade; e as construções `rdfs:subPropertyOf, rdfs:domain e rdfs:range` não são permitidas para anotações. Caso o usuário não se adeque as restrições do teorema, poderá haver inconsistências no raciocínio caso a ontologia seja utilizada em alguma ferramenta OWL 2 RL baseada na semântica direta. Além disso, *triplestores* em conformidade com o profile OWL 2 RL garantem a completude do raciocínio apenas para ontologias adequadas ao profile.

É possível adaptar facilmente a ontologia leve desenvolvida para trabalhar com os outros profiles (OWL 2 QL e OWL 2 EL). Um estudo do trabalho do PR-OWL 2 com estes outros dois profiles pode ser interessante, à medida que estende uma linguagem criada com o objetivo de se tornar padrão na Web Semântica para os profiles oficiais da W3C, utilizados em tipos diferentes de aplicações. O profile OWL 2 QL, por exemplo, pode ser utilizado para permitir o trabalho com ontologias leves em bases de dados tradicionais (baseadas no modelo Entidade-Relacionamento), enquanto o profile OWL 2 EL pode ser utilizado em ontologias que possuem uma TBox grande, porém simples, como é o caso de ontologias de domínio biomédico que especificam grandes hierarquias de classes classificando os elementos da área.

4.4 Definição dos Formatos Válidos para Expressões dos Nós de Contexto

Os formatos das fórmulas dos nós de contexto no PR-OWL 2 RL são definidos de forma que estes possam ser avaliados através de consultas a base de dados RDF, não sendo possível a avaliação de expressões complexas.

A inferência na maioria das *triplestores* é feita por materialização em tempo de carga. A inferência em OWL 2 RL pode ser implementada, com complexidade de tempo polinomial, utilizando materialização a partir do conjunto de regras da axiomatização OWL 2 RL/RDF [62] (dadas as restrições do Teorema PR1 [62]). Para resolver as *queries*, como tanto as triplas da base assertiva quanto as implicadas por estas estarão materializadas na base de dados, não há raciocínio posterior: as *queries* são resolvidas através de buscas na base de dados, de forma similar a feita com SQL em bases de dados relacionais. Otimizações, como por exemplo a criação de índices, podem ser utilizadas para acelerar a avaliação destas.

A Listagem 4.1 apresenta a gramática BNF contendo as fórmulas aceitas no PR-OWL 2 RL. Além dos átomos básicos, são aceitas fórmulas com negação, conjunção e disjunção. A negação é aceita apenas aplicada a átomos. São aceitos os conectivos `AND` e `OR`, novamente aplicados a átomos, não sendo possível incluir sub-expressões contendo o conectivo `AND` dentro de expressões de conjunção e vice-versa. Fórmulas que seguem as restrições definidas pela gramática podem ser avaliadas através de consultas submetidas a *triplestores* RDF.

Listagem 4.1: Gramática BNF para Fórmulas Aceitas no PR-OWL 2 RL

```
1  <atomo>      ::= ov1 == ov2                                          |
2                    booleanRV(ov1 [,ov2 ...])                          |
3                    nonBooleanRV(ov1 [,ov2 ...])  = ov0                |
4                    ov0 = nonBooleanRV(ov1 [,ov2 ...])                 |
5                    nonBooleanRV(ov1 [,ov2 ...])  = CONSTANTE          |
6                    CONSTANTE = nonBooleanRV(ov1 [,ov2 ...])
7  <negacao>    ::= NOT <atomo>
8  <conjuncao>  ::= <atomo> [AND <atomo>]+
9  <disjuncao>  ::= <atomo> [OR <atomo>]+
10 <formula>    ::= <atomo>      |
11                  <negacao>    |
12                  <conjuncao>  |
13                  <disjuncao>
```

A linguagem SPARQL é a linguagem padrão para consultas em *triplestores*. A avaliação dos nós de contexto consiste em validar se um conjunto de valorações para as variáveis ordinárias satisfazem as expressões lógicas contidas no nó, ou recuperar da base valorações que satisfaçam a expressão. Para o primeiro caso, onde há valoração para todas as variáveis ordinárias, é utilizado o comando `ASK` do SPARQL. Este comando retorna `TRUE` caso a informação contida na base valide o questionamento ou `FALSE` caso contrário. Para ilustrar, a Listagem 4.2 mostra a avaliação do nó de contexto `isResponsibleFor`, da MFrag `FrontOfEnterprise`, do domínio "Fraudes em Licitações Públicas", na situação onde a variável ordinária `person` está preenchida com o valor `PERSON_1` e a variável `enterprise` com `ENTERPRISE_1`. A MFrag é ilustrada na Figura 4.4. O comando ASK, quando submetido a *triplestore* irá retornar `TRUE` caso haja a tripla `PERSON_1 isResponsibleFor ENTERPRISE_1` na base.

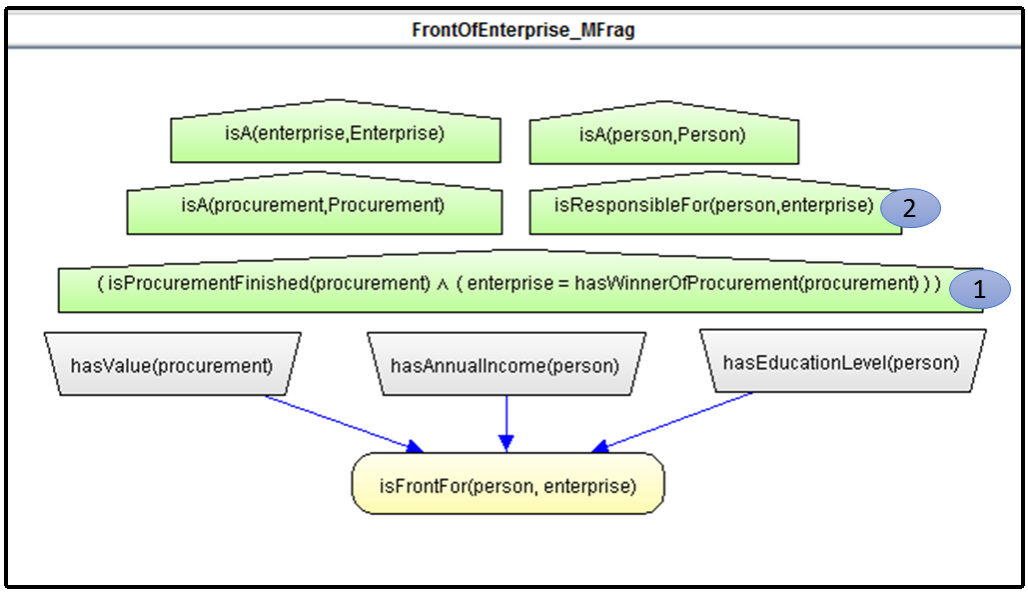

Figura 4.4: MFrag *Front of Enterprise*

A avaliação de nós de contexto utilizando *triplestore* discutida neste trabalho utiliza a assertiva de mundo fechado, onde são consideradas como falsas as informações que não estão contidas na base. Esta restrição é necessária porque, tipicamente, apenas dados positivos podem ser representados em RDF.

Listagem 4.2: Avaliação do nó de contexto isResponsibleFor utilizando o comando ASK

```
1  PREFIX epf:<http://www.pr-owl.org/examples/ProcurementFraud.owl#>
2
3  ASK
4  WHERE {epf:PERSON_1 epf:isResponsibleFor epf:ENTERPRISE_1}
```

Para avaliar expressões onde há variáveis ordinárias com valores desconhecidos, é utilizado a construção `SELECT`. O código abaixo mostra como o nó de contexto 1 da MFrag `FrontOfEnterprise` (Figura 4.4) pode ser avaliado em uma situação onde a variável ordinária `enterprise` é preenchida com a entidade `ENTERPRISE_1`. A *query* busca as licitações já finalizadas que tiveram como vencedora a empresa `ENTERPRISE_1`.

Listagem 4.3: Avaliação de um nó de contexto usando o comando SELECT

```
1  PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX epf:<http://www.pr-owl.org/examples/ProcurementFraud.owl#>
3
4  SELECT ?procurement
5  WHERE {?procurement rdf:type epf:Procurement .
6         ?procurement epf:isProcurementFinished true .
7         epf:ENTERPRISE_1 epf:hasWinnerOfProcurement ?procurement}
```

As avaliações de expressões com o operador `EQUAL TO` envolvendo variáveis aleatórias não booleanas é feita simplesmente submetendo a tripla que corresponde a variável aleatória à base, da mesma forma como ocorre com variáveis aleatórias booleanas (conforme Listagem 4.2). Para variáveis aleatórias booleanas envolvendo mais de dois argumentos, ou variáveis aleatórias não booleanas envolvendo mais de um argumento, é adotada a solução apresentada em [15] para expressar relacionamentos n-arios em OWL, utilizando *blank nodes* para fazer a ligação entre o relacionamento e suas diversas variáveis. A negação pode ser feita utilizando a construção `FILTER NOT EXISTS`. Para resolver expressões com o operador `EQUAL TO` (=) entre duas variáveis ordinárias, pode ser utilizado o comando `owl:sameAs`. A negação novamente é feita acrescentando um `FILTER NOT EXITS`, conforme exemplificado na Listagem 4.4 onde é avaliado um nó de contexto `person1 NOT EQUAL person2`, na situação onde `person1 = PERSON_1`.

Listagem 4.4: Avaliação de um nó de contexto com Equal To entre variáveis ordinárias

```
1  PREFIX owl: <http://www.w3.org/2002/07/owl#>
2  PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3  PREFIX epf:<http://www.pr-owl.org/examples/ProcurementFraud.owl#>
4
5  SELECT ?person2
6  WHERE {?person2 rdf:type epf:Person .
7  FILTER NOT EXISTS(?person2 owl:sameAs PERSON_1)}
```

Expressões com `AND` são avaliadas através da conjunção entre os diversos termos presentes na expressão. Em SPARQL a conjunção é feita utilizando o operador "ponto" (veja

exemplo na Listagem 4.3). Expressões com `OR` são avaliadas através do operador `UNION`. Quando utilizado com o comando `ASK`, o operador `UNION` retorna `TRUE` caso alguma das expressões envolvidas possa ser validada na base. Quando utilizado com o comando `SELECT`, serão retornadas todas as valorações paras as variáveis ordinárias que satisfaçam ao menos uma das expressões.

Capítulo 5

Algoritmos para Geração de SSBN

Este capítulo propõe um novo algoritmo para geração de SSBN que soluciona os problemas de escalabilidade do algoritmo Bottom-Up. A Seção 5.1 apresenta em detalhes o algoritmo Bottom-Up implementado no UnBBayes e discute porque este algoritmo não é adequado para o trabalho com grandes bases assertivas. A Seção 5.2 apresenta um algoritmo anterior implementado no UnBBayes que iniciava a execução apenas a partir da *query* e discute porque este algoritmo não foi utilizado para a criação da versão escalável. A Seção 5.3 apresenta o algoritmo Bayes-Ball utilizado para verificar nós d-conectados a um conjunto de nós objetivo. A Seção 5.4 propõe um novo algoritmo de geração de SSBN baseado no Bayes-Ball, e apresenta um exemplo.

5.1 Algoritmo Bottom-Up para Geração de SSBN

O algoritmo Bottom-Up para construção de SSBN proposto em [51] estende o algoritmo apresentado em [55] para lidar com SSBN possivelmente infinitas. Esta seção apresenta a versão do algoritmo implementada no framework UnBBayes.

O algoritmo inicia com um conjunto de nós de *query* (Q) e um conjunto de nós de evidência (E). Os nós de ambos os conjuntos formam o conjunto inicial S_1. Cada nó é marcado inicialmente como $\neg Finalizado$ (não finalizado). Na primeira iteração, cada nó de S_1 é avaliado, onde a avaliação consiste em recuperar a MFrag onde o nó é residente, avaliar os nós de contexto e gerar os nós pais, baseados nos valores das variáveis ordinárias instanciadas a partir das entidades do nó original. O nó avaliado é marcando como $Finalizado$. Os novos nós gerados são adicionados a um novo conjunto S_2, que será a entrada para a segunda iteração. O processo continua até que o conjunto S_i, avaliado na iteração i, esteja vazio. Um limite de iterações pode ser definido para evitar iterações infinitas.

Os passos do algoritmo são descritos com mais detalhes a seguir. A entrada para o algoritmo é a MTheory geradora, a base de conhecimento contendo as informações assertivas, e as *queries* a serem solucionadas. Uma constante $MAX\_ITERACOES$ indica o número máximo de iterações.

Passo 1 **Inicialização:** Cria o conjunto S_1 contendo os nós de *query* (Q) e os nós de evidência (E). Os nós de evidência são recuperados a partir da base de conhecimento. Faça $i = 1$ (número da iteração). Cada nó de S_1 é marcado como $\neg Finalizado$.

Passo 2 **Construção da Estrutura:** Se $i = MAX\_ITERACOES$, avança para o próximo passo. Se S_i esta vazio, vai para o próximo passo. Caso contrário, cria o conjunto S_{i+1} que irá conter os nós a serem avaliados na próxima iteração. Para cada nó x, pertencente ao conjunto S_i execute os passos 2.1 à 2.4. Faça $i = i + 1$.

2.1 Recupera a MFrag M_x, na qual x é residente;

2.2 Instancia as variáveis da MFrag M_x utilizando os argumentos do nó x. Avalia os nós de contexto de M_x [1]. Nós de contexto que possuem variáveis ordinárias não preenchidas a partir dos argumentos de x são questionadas contra a base de conhecimento por valores para estas que satisfaçam a fórmula do nó de contexto. Caso algum nó de contexto seja avaliado como $false$, x e M_x são marcados para utilizar a distribuição *default* e a avaliação de x é concluída. Caso contrário, a avaliação prossegue. Caso algum nó de contexto não possa ser avaliado por ter alguma variável ordinária não resolvida com a consulta a base de conhecimento, é utilizada uma estratégia de referência incerta, onde o nó de contexto e os possíveis nós pais (para cada possível valoração da variável ordinária não preenchida) viram pais de x. O nó de contexto neste caso funciona como um multiplexador [2];

2.3 São instanciados os nós pais de x dentro da MFrag M_x. Os nós pais devem ser nós residentes ou de entrada. Cada nó criado é marcado como $\neg Finalizado$ e adicionado ao conjunto S_{i+1}. Os valores para as variáveis ordinárias dos nós gerados são definidas a partir dos valores destas variáveis nos nós filhos e a partir dos nós de contexto para variáveis ordinárias não presentes nos nós filhos. Neste passo também é avaliada a recursão, onde um nó residente tem como pai um nó de entrada apontando para o mesmo nó residente, com uma variável ordinária recursiva. O algoritmo busca o valor anterior da variável ordinária recursiva;

[1] No UnBBayes, a avaliação é feita utilizando a máquina de inferência PowerLoom no PR-OWL 1 e HermiT no PR-OWL 2

[2] No UnBBayes, esta abordagem é limitada a nós de contexto no formato "Relacionamento(x) = y". Para nós em outros formatos, o nó não é avaliado e é utilizada a distribuição *default* para x

2.4 Marca o nó x como $Finalizado$;

Passo 3 **Poda da Estrutura:** Remove da SSBN os nós que não influenciam na distribuição de probabilidade dos nós *queries* dado os nós de evidência.

Passo 4 **Construção das Distribuições de Probabilidade:** Constrói as tabelas de probabilidades (CPTs) a partir da avaliação dos pseudocódigos que definem a distribuição de cada nó a partir da quantidade de pais e dos seus estados.

Passo 5 **Compilação e Inicialização da SSBN:** Compila a rede bayesiana e a inicializa com as evidências.

A rede bayesiana montada durante a fase de construção da estrutura é chamada *Grand* BN. Com a *Grand* BN é possível obter as respostas corretas para as *queries*. A fase de poda é importante para remover as nós que são irrelevantes para determinar a distribuição de probabilidade dos nós de *query*, reduzindo o tamanho da rede bayesiana e consequentemente o tempo necessário para construir as CPTs, compilar e inicializar a SSBN. [51] propõe a exclusão dos seguintes nós:

- **Nós d-separados**: nós que são d-separados dos nós de *query* dados os nós de evidência (conforme critério d-separation, descrito na Seção 3.1).
- **Nós estéreis** (*barren nodes*): nós que não possuem descendentes que são *query* ou nós de evidência. Nós estéreis podem depender das evidências, mas eles não contribuem para mudar a probabilidade dos nós objetivo e portanto são computacionalmente irrelevantes [53].
- **Nós ruidosos** (*nuisance nodes*): nós que são computacionalmente relevantes mas não estão em um caminho ativo na propagação das crenças das evidências para os nós de *queries* [53]. São os nós que não possuem como ascendentes nós de *query* ou nós de evidência. Nós ruidosos são computacionalmente relevantes para as *queries* apenas por suas distribuições marginais: se estas forem pré-computadas e armazenadas, eles podem ser removidos da SSBN [3].

O UnBBayes utiliza um algoritmo de árvore de junção para compilar a rede.

O algoritmo Bottom-Up não é adequado para o trabalho com ontologias contendo grandes bases de assertivas porque estas possuem muitos nós de evidência que potencialmente originarão redes desconexas ou nós excluídos na fase de poda. Além da quantidade de espaço necessário para armazenar a estrutura destes nós gerados na fase de construção da *Grand* BN, deve-se considerar no calculo da complexidade o tempo gasto para avaliar a MFrag de cada um destes nós, bem como dos nós gerados por serem pais destes.

[3] Atualmente o UnBBayes não implementa este critério.

5.2 Algoritmo para Geração de SSBN Orientado a *Query*

Em [12] é apresentado um algoritmo para geração de SSBN que parte apenas do nó de *query*, gerando os nós pais com uma estratégia semelhante ao algoritmo Bottom-Up descrito na seção anterior. Este algoritmo foi atualizado em [58] para considerar as evidências que são descendentes dos nós *queries*, que também influenciam na sua probabilidade.

O algoritmo gera a *Grand* BN a partir do nó de *query*, gerando recursivamente os nós pais e os nós filhos de cada nó avaliado, baseando-se nas regras de d-separação. Uma *flag permanente* é utilizada para indicar se o nó gerado será mantido na rede final, sendo setado de acordo com as seguintes regras:

R1 O nó correspondente a *query* é sempre permanente.

R2 Nós de *finding* são sempre permanentes.

R3 Um nó que seja pai de um nó permanente também é permanente, exceto nós pais de *finding*, quando o nó de *finding* for ascendente do nó de *query*.

A avaliação inicia no nó de *query*, avaliando cada nó x conforme abaixo. Para a avaliação da Regra 2 é necessário uma *flag* `AvaliaçãoAbaixoDoNoQuery` que indica se o nó que avaliado é descendente do nó *query*.

- Caso x ainda não tenha sido avaliado abaixo:
  - Verifica se o nó x é *finding*. Em caso positivo, seta x como permanente;
  - Recupera a MFrag M_x, na qual x é residente;
  - Avalia os nós de contexto de M_x que contenham variáveis ordinárias relacionadas as variáveis de x. Caso algum nó de contexto não tenha sido avaliado como $true$, seta x para utilizar a distribuição *default* e encerra a avaliação deste;
  - Gera todos os nós filhos possíveis a partir dos valores das variáveis ordinárias instanciados para as valorações que validam os nós de contexto. Para cada nó filho y criado, chama recursivamente a avaliação do nó. Se algum dos nós filhos estiver marcado como permanente, marca x como permanente;
  - Marca x como avaliado abaixo.
- Caso o nó esteja marcado como permanente e ainda não tenha sido avaliado acima, faz a avaliação acima:

– Gera todos os nós pais possíveis a partir dos valores das variáveis ordinárias instanciados para as valorações que validam os nós de contexto. Para cada nó pai y criado:

  * Caso x não seja *finding*, avalia y e o marca como permanente;
  * Caso x seja *finding* e `AvaliaçãoAbaixoDoNoQuery` seja *true*, marca y como permanente e o avalia.

– Marca o nó x como avaliado acima.

Este algoritmo não mantém os nós estéreis na rede final, já que um nó só será permanente caso seja pai do nó de *query* ou de um nó de *finding*. Esta poda portanto não é necessário na *Grand* BN gerada.

Conforme o algoritmo, um nó pai de um nó de evidência ascendente do nó de *query* não permanece na rede final, exceto caso também seja pai de algum outro nó permanente não *finding*. Isto não está correto de acordo com o critério d-separation, conforme ilustrado na Figura 5.1, onde o nó `Q` corresponde a *query* e o nó `4` a um nó de *finding*. A área cinza mostra os nós considerados como permanentes segundo o algoritmo descrito. Pelo critério de d-separation o nó `6` também deveria fazer parte da rede gerada, pois como o nó `4` é um *finding*, `6` influência `5` por uma conexão convergente, e por consequência, influência `Q` . Esta falha no algoritmo não é de simples correção, pois a avaliação do nó `6` é feita levando-se em consideração apenas as informações passadas a partir do nó `4`.

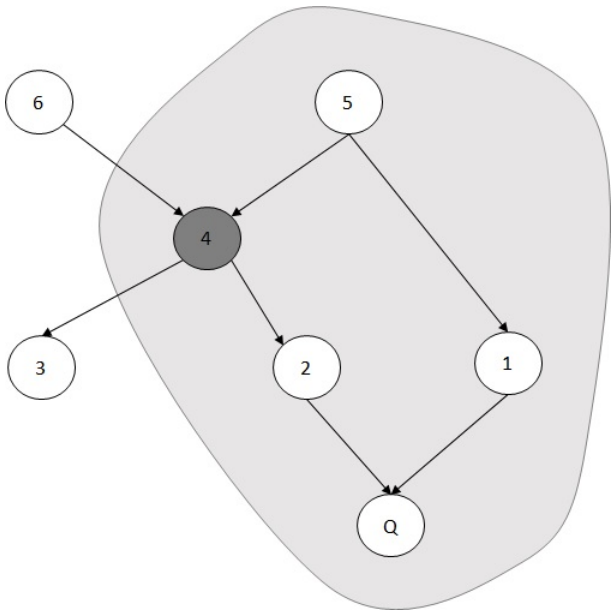

Figura 5.1: Problema no algoritmo de geração de SSBN orientado a *query*

O problema citado, a falta de formalização da corretude do algoritmo descrito e a dificuldade em adaptá-lo para avaliação de múltiplos nós de *query* motivaram a criação de um novo algoritmo para geração de SSBN baseado no Bayes-Ball.

5.3 Algoritmo Bayes-Ball

O algoritmo Bayes-Ball [74] verifica quais nós de uma rede bayesiana são d-conectados a um conjunto de nós objetivos, permitindo a identificação dos nós relevantes. O algoritmo executa em tempo linear em relação ao tamanho da parte ativa do grafo (nós d-conectados aos nós objetivos), sendo consideravelmente mais rápido quando a maioria dos nós são irrelevantes [74] quando comparado com algoritmos que percorrem todo o grafo, como o apresentado em [33].

Seja $B = (N, A)$ uma rede bayesiana, onde N é o conjunto de nós e A o conjunto de arcos. O algoritmo permite que se calcule o conjunto de nós irrelevantes para um conjunto de nós objetivos X_J, dado um conjunto de nós com evidência X_K:

$$N_i(J|K) = \{i \in N : X_i \perp X_J | X_K\} \tag{5.1}$$

O algoritmo Bayes-Ball se baseia na ideia de encontrar os nós que possuem caminho ativo para os nós objetivos através de passagem de uma "bola bayesiana", uma bola virtual que a partir dos nós de objetivo é repassada para outros nós no grafo seguindo um conjunto de regras baseadas no critério de d-separação: dependendo se a bola foi recebida de um nó pai ou de um nó filho, e de se o nó que recebeu a bola é evidência ou não, a bola pode ser repassada para os nós pais, devolvida/repassada para os nós filhos ou bloqueada. A Figura 5.2 ilustra as regras, formuladas abaixo[4]:

R1 Caso a bola seja recebida de um nó pai por um nó para o qual não há evidência, a bola é repassada para todos os nós filhos deste.

R2 Caso a bola seja recebida de um nó filho por um nó para o qual não há evidência, a bola é repassada para todos os nós filhos e pais deste.

R3 Caso a bola seja recebida de um nó pai por um nó de evidência, a bola é repassada para todos os nós pais deste.

R4 Caso a bola seja recebida de um nó filho por um nó de evidência, a bola não é repassada para nenhum nó (a bola é bloqueada).

[4] [74] apresenta uma regra adicional para nós que são funcionalmente dependentes dos nós pais: caso este receba a bola de um nó filho, a bola será repassada apenas para seus nós pais. O algoritmo apresentado nesta seção considera que a rede bayesiana não contém nós funcionalmente dependentes dos nós pais.

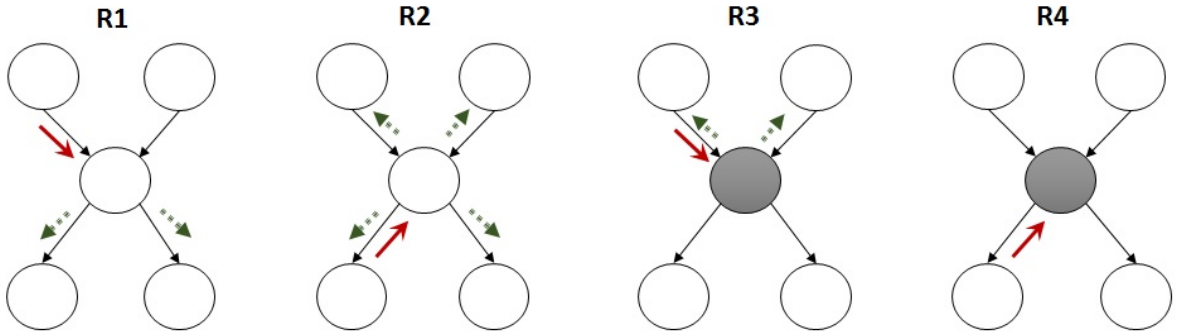

Figura 5.2: Regras do algoritmo Bayes-Ball

A bola é recebida inicialmente por cada um dos nós objetivo repassados por um nó filho virtual. Para cada nó da rede bayesiana utiliza-se uma *flag* indicando se a bola já foi repassada por este nó para os nós pais e outra *flag* indicando se a bola já foi repassada para os nós filhos. Estas *flags* são necessárias para evitar que um arco seja percorrido mais de uma vez e garantem que o algoritmo terminará.

Os nós irrelevantes, $N_i(J|K)$, são os nós não marcados abaixo [74] (ou seja, que não repassaram a bola para os filhos). Conforme o Teorema 5.3.1 (reproduzido de [74], onde pode ser encontrada a prova) o conjunto de nós irrelevantes calculado pelo algoritmo Bayes-Ball corresponde ao conjunto de nós d-separados do conjunto J, dado K:

Teorema 5.3.1 *Dado uma rede bayesiana* $B = (N, A)$ *e* $J, K, L \subseteq N$, $X_J \perp X_L | X_K$ *se e somente se* $L \subseteq N_i(J|K)$.

Como todos os nós visitados que não são evidência são marcados abaixo (conforme pode ser visualizado na Figura 5.2), todos os nós visitados são d-conectados a J, dado K.

A Figura 5.3 mostra o algoritmo aplicado a rede bayesiana "Family Out" (apresentada na Seção 3.1) com J={"LL"} e K={"CL"}. A seta vermelha indica o nó que enviou a bola para o nó avaliado enquanto as setas verdes indicam para onde o nó envia a bola após a recepção. O triangulo para cima marca que já foi avaliado a situação onde o nó envia a bola para os pais, enquanto o triangulo para baixo marca a situação onde a bola já foi enviada para os filhos. No exemplo, a bola bayesiana passa por todos os nós da rede e portanto todos são relevantes para calcular a probabilidade de "LL". Na Figura 5.4 J={"CL"} e K={"FS"}. Para este caso o nó "LL" é irrelevante.

A complexidade do algoritmo é O(|N| + $|A_v|$) [74], onde A_v são os arcos visitados durante a execução do algoritmo. Será necessário passar por todos os nós da rede para inicializar as suas *flags*. No pior caso, o algoritmo é linear no tamanho do grafo [74], já que todos os nós e todos os arcos serão percorridos.

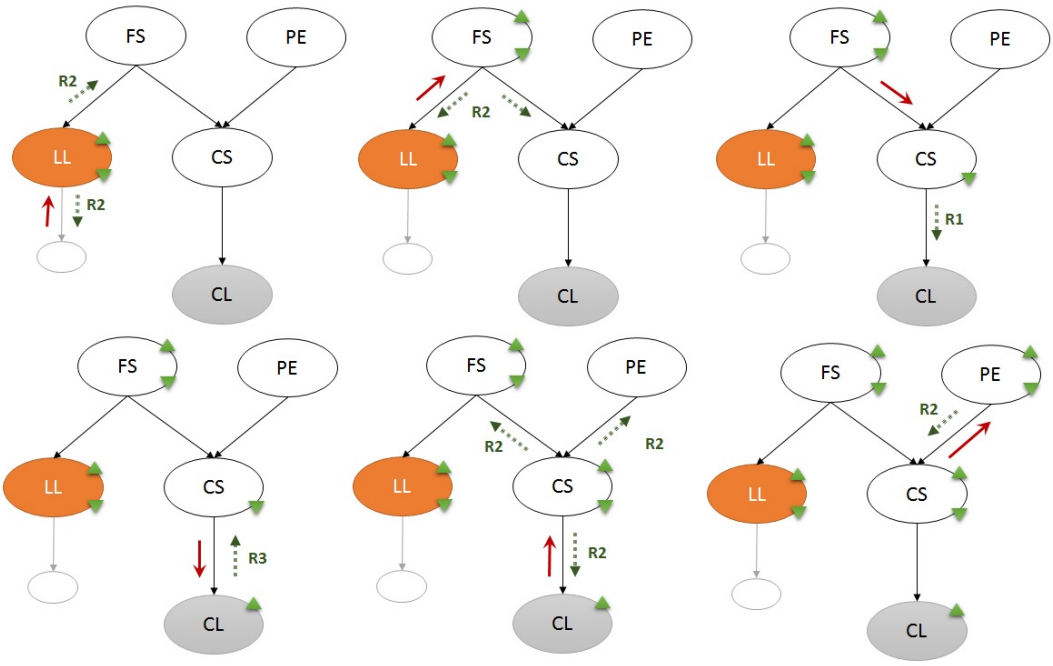

Figura 5.3: Algoritmo Bayes-Ball aplicado à rede *Family Out* - Exemplo 1

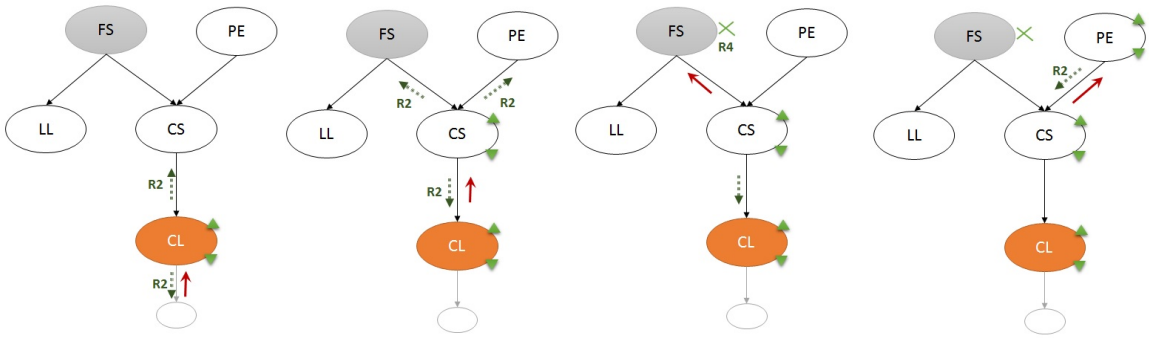

Figura 5.4: Algoritmo Bayes-Ball aplicado à rede *Family Out* - Exemplo 2

5.4 Algoritmo para Geração de SSBN Baseado no Bayes-Ball

O algoritmo de geração de SSBN proposto neste trabalho se baseia no método Bayes-Ball, gerando os nós a partir da visita destes pela bola bayesiana, garantindo que sejam gerados apenas nós d-conectados com os nós de *query*, dado os nós de *finding*.

Os passos gerais do algoritmo são os mesmos do algoritmo Bottom-up apresentado na Seção 5.1, porém com as seguintes modificações:

- No Passo 1 (Inicialização) apenas os nós de *query* são incluídas no conjunto inicial (S_1).
- No Passo 2 (Construção da Estrutura) os nós são criados se baseando nas regras do algoritmo Bayes-Ball.
- No Passo 3 (Poda da Estrutura) não é necessário podar os nós d-separados pois o algoritmo gera apenas nós d-conectados aos nós de *query* dado as evidências.

A seguir será discutido o funcionamento do Passo 2.
Cada nó gerado durante o algoritmo possui os seguintes atributos:

- `receivedBallFromChild`: indica se a bola foi recebida de um nó filho.
- `receivedBallFromParent`: indica se a bola foi recebida de um nó pai.
- `evaluatedTop`: indica se o nó já foi avaliado acima.
- `evaluatedBottom`: indica se o nó já foi avaliado abaixo.
- `isFinding`: indica se o nó é um *finding* (evidência).
- `visited`: indica se o nó foi visitado pela bola bayesiana.

A construção da estrutura é feita conforme o Algoritmo 1. Os seguintes procedimentos são chamados no algoritmo:

- **evaluateNode**: verifica se o nó x é uma evidência na base de conhecimento. Se sim, marca `isFinding`; instancia a MFrag M_x na qual o nó é residente e faz a avaliação dos seus nós de contexto (mesmas considerações do algoritmo Bottom-Up).
- **evaluateTop**: faz avaliação acima. Instância os pais de x, marcando `receivedBallFromChild`. Se y, nó pai de x, é um nó de entrada, a MFrag M_y, onde y é residente será instanciada utilizando para preencher os argumentos de y os valores das variáveis ordinárias em M_x. Se y é um nó residente, apenas o instância, utilizando as valorações para as variáveis ordinárias de M_x. Trata a recursão da mesma forma que o algoritmo Bottom-Up. Adiciona os novos nós gerados à `nodeList`.
- **evaluateBottom**: faz avaliação abaixo. Gera os nós residentes filhos de x em M_x e os nós residentes filhos de y em M_y, onde y é um nó de entrada localizado na MFrag M_y, que faz referência para x. Neste último caso, a MFrag M_y também será instanciada, a partir das variáveis ordinárias de x e dos nós de contexto de M_y. Os novos nós serão criados e adicionados a `nodeList` e será marcado `receivedBallFromParent`.

Algoritimo 1 Algoritmo para construção da Grand BN utilizando Bayes-Ball

Entrada: Objeto SSBN inicializado com as queries
Entrada: Base de conhecimento contendo as evidências
Saída: nodeList, contendo os nós gerados

```
1:
2: createList(nodeList)
3: createList(unvaluatedNodeList)
4:
5: for each node in SSBN.queryList do
6:     node.receivedBallFromChild ← true
7:     unvaluatedNodeList.add(node)
8:     nodeList.add(node)
9:
10: while unvaluatedNodeList.size() > 0 do
11:     for each node in unvaluatedNodeList do
12:
13:         evaluateNode(node)
14:
15:         if node.receivedBallFromChild then
16:             if node.isFinding == false then
17:                 if node.evaluatedTop == false then
18:                     evaluateTop(nodelist, node);
19:                     node.evaluatedTop ← true;
20:                 if node.isEvaluatedBottom == false then
21:                     evaluateBottom(nodeList, node);
22:                     node.evaluatedBottom ← true;
23:
24:         if node.isReceivedBallFromParent then
25:             if node.isFinding == false then
26:                 if node.isEvaluatedBottom == false then
27:                     evaluateBottom(nodelist, node);
28:                     node.evaluatedBottom ← true;
29:             else
30:                 if node.isEvaluatedTop == false then
31:                     evaluateTop(nodeList, node);
32:                     node.evaluatedTop ← true;
33:
34:         node.isEvaluated ← true
35:
36:     unvaluatedList.clear()
37:     for each node in nodeList do
38:         if !node.visited then
39:             unvaluatedList.add(node)
```

Seja G o grafo obtido aplicando-se a MTheory geradora a toda a base assertiva, onde x é pai de y caso haja um arco de x para y na MFrag Mx, ou caso x seja um nó de contexto na MFrag My. Seja Q o conjunto dos nós de *query* e seja F o conjunto dos nós de *finding*. Considerando-se o tempo para avaliar os nós de contexto como constante e igual ao tempo para avaliar um nó residente, temos que a complexidade do algoritmo Bayes-Ball para geração da *Grand* BN é $O(|dcon(Q)|)$, onde $dcon(Q)$ se refere ao conjunto formado pelos nós de Q e pelos nós d-conectados aos nós de Q, dado os nós de F. Como os nós d-conectados a Q estão contidos na lista de descendentes (dsc) e ascendentes (asc) de Q, temos que $|dcon(Q)| <= |(asc(Q) + dsc(Q))|$. A complexidade do algoritmo Bottom-Up para geração da *Grand* BN é $O(|asc(Q) + asc(F)|)$. Como $dcon(Q) \subseteq asc(Q) \cup asc(F)$ temos que a complexidade do algoritmo Bayes-Ball é menor ou igual que a do Bottom-Up. O algoritmo Bayes-Ball será mais eficiente que o Bottom-Up em situações onde a base assertiva contém uma grande quantidade de *findings*, e no entanto a maioria destes são d-separados de Q.

A Figura 5.5 mostra que o algoritmo proposto soluciona a falha apresentada no algoritmo apresentado na Seção 5.2. Ao chegar ao nó de *finding* através do envio da bola feito pelo nó *2* (grafo a), a bola é bloqueada e os pais de `4` não são gerados. Porém, ao chegar recebida do nó `5` (grafo b) a bola é repassada para todos os pais do nó `4`, adicionando o nó `6` a rede.

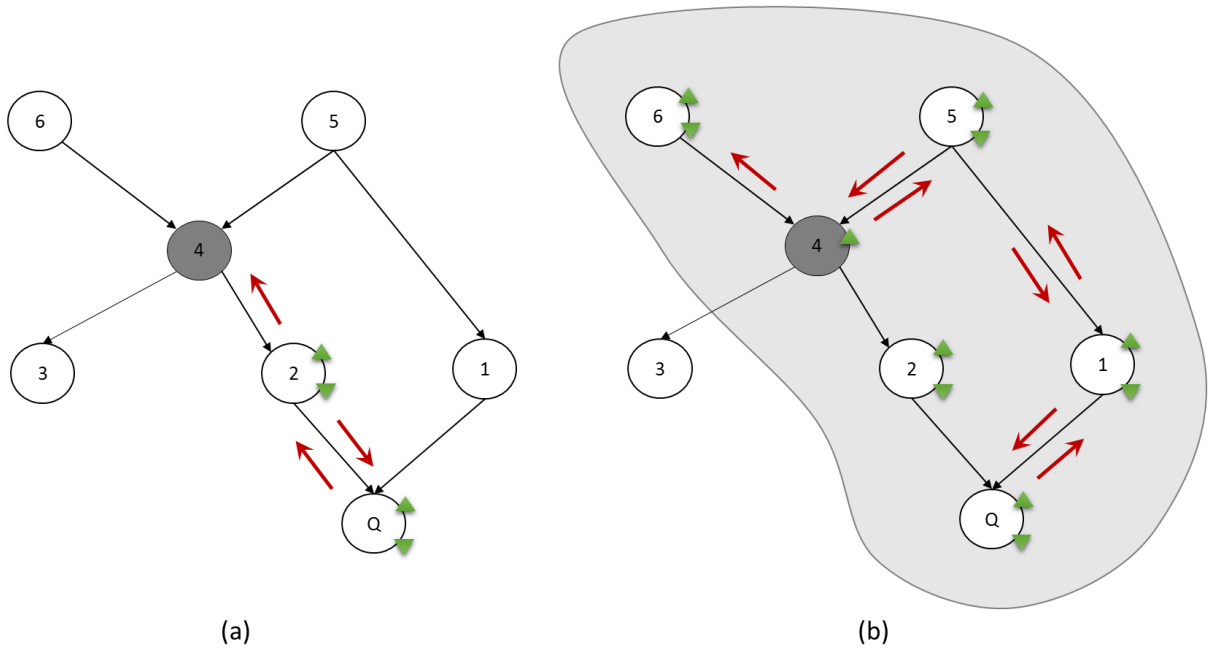

Figura 5.5: Correção da falha do algoritmo de geração de SSBN orientado a *query*

A seguir é apresentado um exemplo mostrando o funcionamento do algoritmo, aplicado a ontologia "Fraudes em Licitações Públicas", cuja MTheory foi ilustrada na Figura 3.4.

A Listagem 5.1 contém trecho da base assertiva (em Turtle) utilizada durante a inferência. Conforme descrito na listagem, a licitação `procurement0`[5] envolve duas empresas, `enterprise0` e `enterprise1`, tendo a `enterprise0` sido a vencedora (conforme propriedade `hasWinnerOfProcurement`, na linha 7). `person0` é responsável pela `enterprise0` e `person1` pela `enterprise1`, enquanto `person13` e `person14` são membros do comitê. `procurement0` já foi concluída, conforme indica a propriedade `isProcurementFinished`, que possui o valor booleano `true` (^^xsd:boolean indica o tipo do literal).

Listagem 5.1: Trecho Base Assertiva

```
1  PREFIX epf:<http://www.pr-owl.org/examples/ProcurementFraud.owl#>
2  PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3
4  epf:procurement0 rdf:type epf:Procurement ;
5                   epf:isProcurementFinished "true"^^xsd:boolean ;
6                   epf:hasValue exp:Lower10k ;
7                   epf:hasProcurementOwner epf:agency0 ;
8                   epf:hasWinnerOfProcurement epf:enterprise0 .
9
10 epf:enterprise0  rdf:type epf:Enterprise;
11                  epf:isSuspended "false"^^xsd:boolean ;
12                  epf:isParticipantIn epf:procurement0 .
13
14 epf:enterprise1  rdf:type epf:Enterprise;
15                  epf:isSuspended "false"^^xsd:boolean ;
16                  epf:isParticipantIn epf:procurement0 .
17
18 epf:person0       rdf:type epf:Person;
19                  epf:hasAnnualIncoming epf:From30kTo60k ;
20                  epf:hasEducationLevel epf:MiddleSchool ;
21                 epf:hasAdmnistrativeHistory epf:NeverInvestigated ;
22                  epf:hasCriminalHistory epf:NeverInvestigated ;
23                  epf:isResponsibleFor epf:enterprise0 .
24
25 epf:person1       rdf:type epf:Person;
26                  epf:hasEducationLevel epf:HighSchool ;
27                  epf:hasAnnualIncoming epf:Lower10k ;
```

[5]Para simplificar a notação, o prefixo epf foi retirado dos nomes das IRI.

```
28                    epf:hasCriminalHistory epf:NeverInvestigated ;
29                   epf:hasAdmnistrativeHistory epf:NeverInvestigated ;
30                    epf:isResponsibleFor epf:enterprise1 .
31
32    exp:person13    rdf:type epf:Person;
33                    epf:hasAnnualIncoming epf:From30kTo60k ;
34                    epf:hasEducationLevel epf:HighSchool ;
35                    epf:hasCriminalHistory epf:NeverInvestigated ;
36                   epf:hasAdmnistrativeHistory epf:NeverInvestigated ;
37                    epf:isMemberOfCommittee epf:procurement0 .
38
39    epf:person14    rdf:type epf:Person;
40                    epf:hasAnnualIncoming epf:From10kTo30k ;
41                    epf:hasEducationLevel epf:HighSchool ;
42                   epf:hasAdmnistrativeHistory epf:NeverInvestigated ;
43                    epf:hasCriminalHistory epf:NeverInvestigated ;
44                    epf:liveAtSameAddress epf:person8 ;
45                    epf:isMemberOfCommittee epf:procurement0 .
```

Seja a *query* `isSuspiciousProcurement(procurement0)`.

No passo de inicialização, é recuperado o nó residente correspondente a *query* e montado um SSBNNode, com a *flag* `receivedBallFromChild` marcada como `TRUE`. O SSBNNode é acrescentado a SSBN.

No passo de construção da *Grand* BN, recupera-se a MFrag onde o nó se encontra. Primeiro verifica-se se há evidência para o nó, consultando a base de conhecimento. Não existe evidência para o nó. O próximo passo consiste em instanciar a MFrag do nó e avaliar os seus nós de contexto. O nó se encontra na MFrag SuspiciousProcurement_MFrag, ilustrada na Figura 5.6.

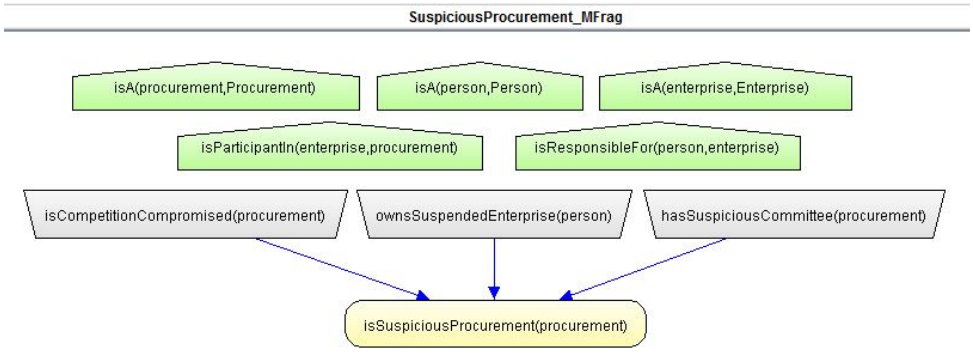

Figura 5.6: MFrag *Suspicious Procurement*

A avaliação do nó `isParticipantIn(?enterprise, procurement0)` (onde o simbolo `"?"` é utilizado para indicar que se deseja descobrir o valor para a variável ordinária `enterprise`) retorna `enterprise0` e `enterprise1` (Linhas 9 e 12 na Listagem 5.1). Avaliando o nó `isResponsibleFor(?person, enterprise0)`, a base de conhecimento retorna `person0`, enquanto para `isResponsibleFor (?person, enterprise1)` é retornado `person1`.

Na primeira iteração, há apenas um nó a ser avaliado. O nó `isSuspiciousProcurement(procurement0)` recebeu a bola bayesiana do filho, e, por não ser *finding*, deve repassar a bola de volta para os filhos e para os pais. As valorações para as variáveis ordinárias da MFrag originam os nós pais ilustrados na Figura 5.7, que são criados com a *flag* `receiveBallFromChild = TRUE`. O nó `isSuspiciousProcurement` não possui filhos, e portanto a avaliação abaixo não gera nós. Observe que os filhos não devem ser procurados apenas na MFrag SuspiciousProcurement, mas também em todas as outras MFrags onde o nó seja nó de entrada.

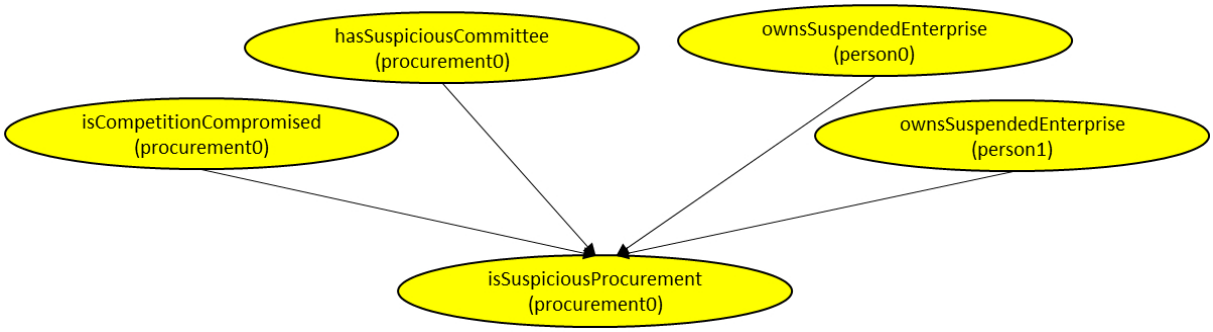

Figura 5.7: Pais do nó isSuspiciousProcurement(procurement0)

Após o final da primeira iteração, temos na SSBN os nós da Figura 5.8, com os valores para as *flags* conforme indicado.

isSuspiciousProcurement (procurement0)	isCompetitionCompromised (procurement0)	hasSuspiciousCommittee (procurement0)	ownsSuspendedEnterprise (person0)	ownsSuspendedEnterprise (person1)
ReceivedBallFromChild = T	ReceivedBallFromChild = T	ReceivedBallFromChild = T	ReceivedBallFromChild = T	ReceivedBallFromChild = T
ReceivedBallFromParent = F	ReceivedBallFromParent = F	ReceivedBallFromParent = F	ReceivedBallFromParent = F	ReceivedBallFromParent = F
EvaluatedTop = T	EvaluatedTop = F	EvaluatedTop = F	EvaluatedTop = F	EvaluatedTop = F
EvaluatedBottom = T	EvaluatedBottom = F	EvaluatedBottom = F	EvaluatedBottom = F	EvaluatedBottom = F
isFinding = F	isFinding = F	isFinding = F	isFinding = F	isFinding = F
Visited = T	Visited = F	Visited = F	Visited = F	Visited = F

Figura 5.8: Nós gerados após a primeira iteração

Monta-se uma nova lista contendo os nós que estão com `visited = FALSE` e se inicia a avaliação de cada um destes nós.

Na avaliação do nó `ownsSuspendedEnterprise(person0)` é verificado que não há evidência na base. Recupera-se a MFrag do nó e valida-se os nós de contexto, obtendo a valoração `enterprise0` para a variável ordinária `enterprise`. Como o nó recebeu a bola de um filho, deverão ser gerados os pais e filhos. É gerado o nó `isSuspended(enterprise0)` como pai, conforme ilustrado na Figura 5.9.

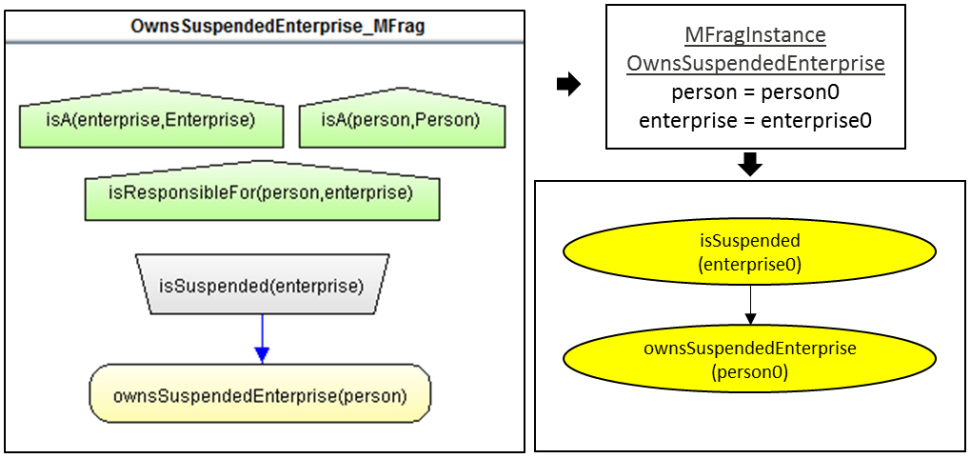

Figura 5.9: Avaliação do nó ownsSuspendedEnterprise(person0)

O nó `ownsSuspendedEnterprise(person0)` não possui nós filhos na MFrag `ownsSuspendedEnterprise`, mas possui na MFrag `SuspiciousProcurement`, onde é um nó de entrada (Veja Figura 5.6). É gerada uma nova instância da MFrag `SuspiciousProcurement` com a variável ordinária `person` instanciada para `person0`. Os nós de contexto da MFragInstance são instanciados, obtendo as valorações `enterprise = enterprise0` e `procurement = procurement0`. Com estas valorações, é gerado apenas o filho `isSuspiciousProcurement(procurement0)`. Verifica-se que já existe um SSBNNode para este nó, e portanto não é gerado um objeto novo. Apenas atualiza-se as *flags* do SSBNNode para indicar que agora a bola foi recebida do pai, e a *flag* `visited` é marcada como FALSE para permitir a avaliação do nó novamente.

A avaliação do nó `ownsSuspendedEnterprise(person1)` funciona da mesma forma.

A Figura 5.10 mostra os nós pais gerados para os nós `hasSuspiciousCommittee(procurement0)` e `isCompetitionCompromised(procurement0)`. Ambos os nós geram apenas o nó `isSuspiciousProcurement(procurement0)` como filho.

Após o final da segunda iteração, temos na SSBN os nós da Figura 5.11. Os valores em vermelho são as *flags* alteradas de nós já existentes antes da iteração. Novamente será montada uma lista contendo apenas os nós não visitados para a próxima iteração. A avaliação prosseguirá até chegar a uma iteração onde não haja nós não visitados.

A Figura 5.12 mostra a SSBN final gerada, inclusive após a fase de poda.

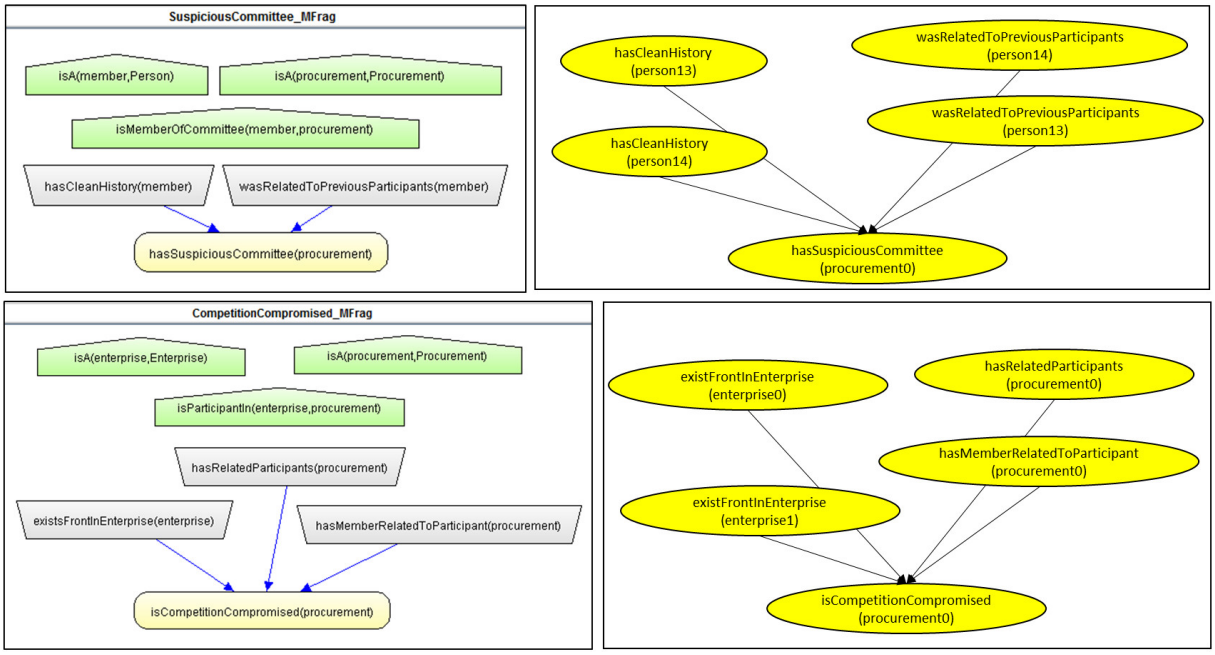

Figura 5.10: Pais gerados para os nós hasSuspiciousCommittee e isCompetitionCompromised

isSuspiciousProcurement (procurement0)	**isCompetitionCompromised (procurement0)**	**hasSuspiciousCommittee (procurement0)**	**ownsSuspendedEnterprise (person0)**	**ownsSuspendedEnterprise (person1)**
ReceivedBallFromChild = T ReceivedBallFromParent = T EvaluatedTop = T EvaluatedBottom = T isFinding = F Visited = F	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = T EvaluatedBottom = T isFinding = F Visited = T	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = F EvaluatedBottom = F isFinding = F Visited = F	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = T EvaluatedBottom = T isFinding = F Visited = T	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = T EvaluatedBottom = T isFinding = F Visited = T
isSuspended (enterprise0)	**isSuspended (enterprise1)**	**hasCleanHistory (person13)**	**hasCleanHistory (person14)**	**hasRelatedParticipants (procurement0)**
ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = F EvaluatedBottom = F isFinding = F Visited = F	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = F EvaluatedBottom = F isFinding = F Visited = F	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = F EvaluatedBottom = F isFinding = F Visited = F	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = F EvaluatedBottom = F isFinding = F Visited = F	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = F EvaluatedBottom = F isFinding = F Visited = F
existFrontInEnterprise (enterprise0)	**existFrontInEnterprise (enterprise1)**	**hasMemberRelatedToParticipants (person0)**	**wasRelatedToPreviousParticipant (person13)**	**wasRelatedToPreviousParticipant (person14)**
ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = T EvaluatedBottom = T isFinding = F Visited = F	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = F EvaluatedBottom = F isFinding = F Visited = F	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = F EvaluatedBottom = F isFinding = F Visited = F	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = F EvaluatedBottom = F isFinding = F Visited = F	ReceivedBallFromChild = T ReceivedBallFromParent = F EvaluatedTop = F EvaluatedBottom = F isFinding = F Visited = F

Figura 5.11: Nós gerados após a segunda iteração

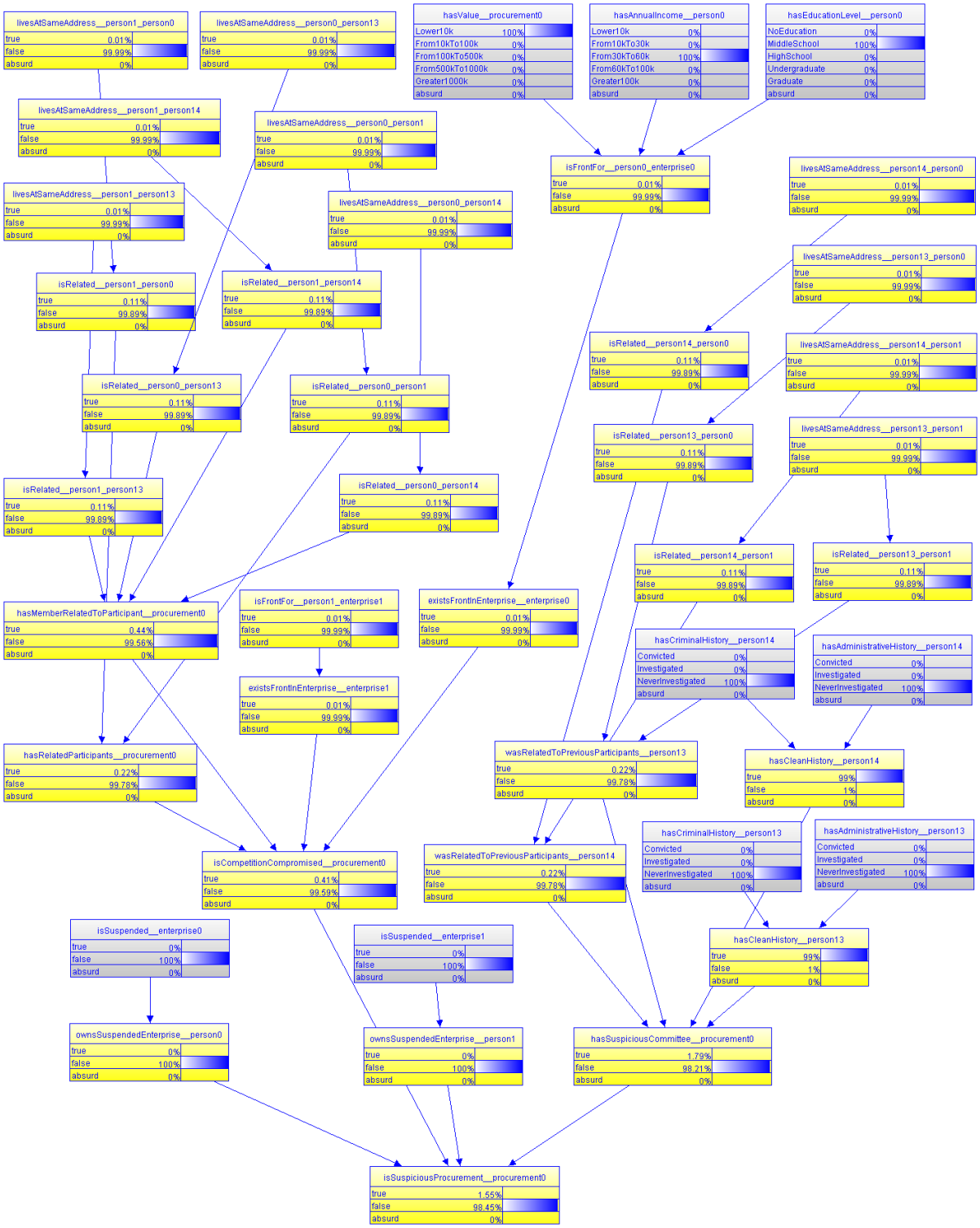

Figura 5.12: SSBN gerada para o exemplo apresentado

Capítulo 6

Avaliação do Novo Formalismo

Este capítulo faz um estudo da escalabilidade e da expressividade da solução apresentada. Afim de avaliar de forma prática o formalismo proposto, foi desenvolvido um plug-in para o PR-OWL 2 RL no framework UnBBayes. O plug-in é descrito na Seção 6.1. A Seção 6.2 faz uma avaliação da escalabilidade da implementação da linguagem PR-OWL 2 RL (com o algoritmo Bayes-Ball) utilizando o estudo de caso "Fraude em Licitações Públicas" e compara com os resultados obtidos utilizando a implementação de PR-OWL 2 (com o algoritmo Bottom-Up). A Seção 6.3 faz uma comparação entre as versões do PR-OWL existentes, discutindo as contribuições da nova linguagem.

6.1 Implementação do PR-OWL 2 RL no UnBBayes

A implementação da linguagem PR-OWL 2 RL no UnBBayes permite que o usuário modele uma MTheory utilizando a interface gráfica do usuário (GUI) do framework, salve em formato PR-OWL 2 RL e faça inferência utilizando uma *triplestore* como base de conhecimento.

A Figura 6.1 ilustra o fluxo de trabalho com a ferramenta. A modelagem da TBox, contendo tanto a parte probabilística quanto a determinística pode ser feita no UnBBayes. Alternativamente, a modelagem da parte determinística pode ser realizada por um software externo, como por exemplo o Protégé. O usuário deve garantir que a ontologia segue as restrições do *profile* OWL 2 RL, conforme apresentado na Seção 2.4. Ferramentas de validação podem ser utilizadas com este fim, como a disponibilizada em http://mowl-power.cs.man.ac.uk:8080/validator/. A edição da parte probabilística é feita através da criação da MTheory geradora, utilizando o UnBBayes. A GUI de edição do UnBBayes deixa transparente para o usuário a sintaxe do PR-OWL 2 RL.

Após finalizada a modelagem, a ontologia é salva em PR-OWL 2 RL e deve ser carregada no repositório semântico. O usuário então é capaz de fazer o povoamento da base

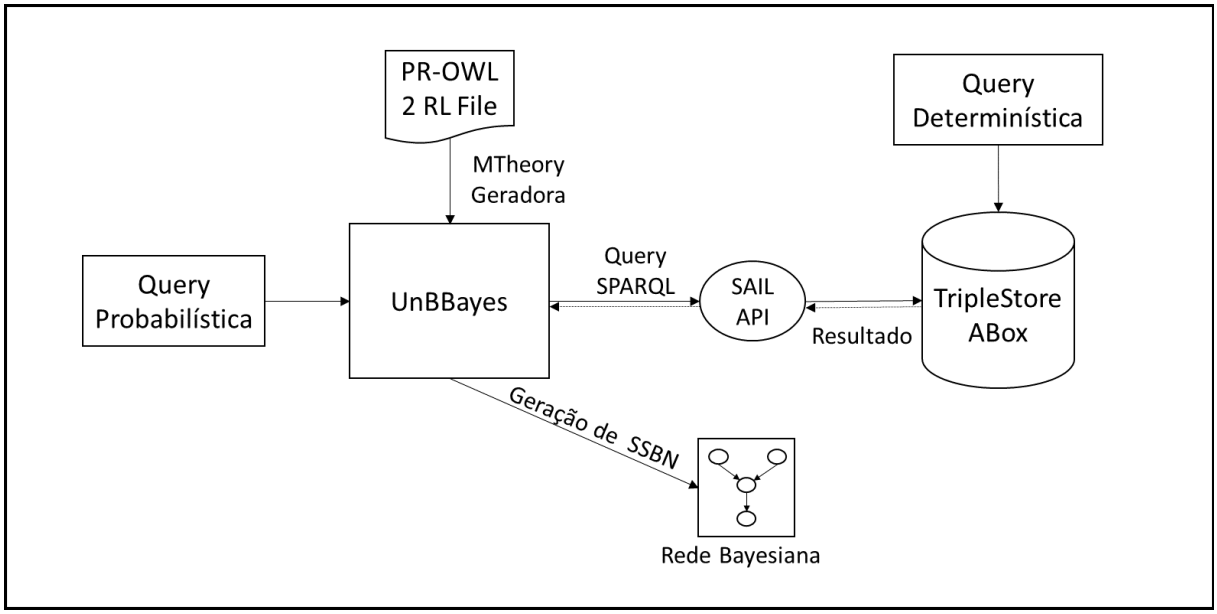

Figura 6.1: Geração de SSBN a partir de *query* do usuário

assertiva (ABox) diretamente no repositório semântico. O *triplestore* é responsável pela inferência na parte determinística da ontologia, que pode ser feita utilizando a estratégia de materialização, durante a carga da base assertiva.

As *queries* determinísticas podem ser feitas utilizando a interface fornecida pelo repositório semântico, possivelmente utilizando a linguagem SPARQL. As *queries* probabilísticas são respondidas pela interface do UnBBayes, onde a SSBN será montada a partir da MTheory e da base de conhecimento. Para tal, o usuário deve previamente carregar o arquivo PR-OWL 2 RL, contendo a definição da ontologia e efetuar a conexão com a base de dados. O UnBBayes utiliza a conexão para buscar as evidências e avaliar as fórmulas dos nós de contexto (utilizando queries na linguagem SPARQL). A conexão é feita utilizando a interface SAIL, do Sesame, que abstrai os detalhes de armazenamento e inferência. Esta interface é implementada por diversas *triplestores*, permitindo o trabalho do UnBBayes em conjunto com outras opções de bases de dados.

Para a implementação do plug-in do UnBBayes foi escolhido o GraphDB Lite como *triplestore*. Foram feitos testes com o Sesame, Jena TDB e GraphDB Lite, selecionados por serem gratuitas e bastante utilizadas entre a comunidade da Web Semântica. Os testes foram feitos utilizando o LUBM, verificando a capacidade de armazenamento e raciocínio destas. Apesar de ambas as versões terem sido capazes de carregar o LUBM 100, em uma máquina i5 com 6Gb de memória, apenas o GraphDB respondeu corretamente as quatorze *queries* do *benchmark*. Não foi possível carregar as versões de teste maiores do LUBM no GraphDB Lite, já que a versão deste utilizada nos testes, carrega toda a base

na memória da máquina do usuário, estando portanto o tamanho da base limitado ao tamanho desta memória. A troca da versão Lite do GraphDB por uma versão comercial é simples, já que ambas utilizam a mesma interface.

A arquitetura do plug-in desenvolvido é apresentada em detalhes no Apêndice B.

6.2 Avaliação do PR-OWL 2 RL Utilizando o Domínio Fraude em Licitações Públicas

Para testar a escalabilidade da implementação feita no UnBBayes foi utilizada a ontologia "Fraude em Licitações Públicas". Esta seção mostra os resultados destes testes e faz uma comparação entre a escalabilidade do plug-in de PR-OWL 2 e do plug-in de PR-OWL 2 RL do UnBBayes.

Devido a dificuldade em conseguir dados reais para realizar testes de escalabilidade, optamos por criar um gerador de base assertiva. A partir de parâmetros informados pelo usuário, o gerador cria uma base com a quantidade de entidades desejada. Os relacionamentos e atributos das entidades são definidos de forma aleatória, baseando-se nas probabilidades presentes na ontologia. Desta forma, cria-se para ontologias probabilísticas um benchmark semelhante ao LUBM, utilizado para ontologias determinísticas, possibilitando testes com tamanhos de bases incrementalmente maiores. O gerador de base assertiva está disponível no SourceForge [1]. Para os testes apresentados nesta seção, foram geradas bases contendo 15, 50, 100, 1.000, 10.000, 100.000 e 1.000.000 de pessoas, que serão tratadas como Base 15, Base 50, Base 100, etc. Os parâmetros utilizados para a criação das bases são apresentados na Tabela 6.1. A Base 15 corresponde a base utilizada no exemplo da Seção 5.4, onde foram consideradas 2 membros e 2 empresas por licitação. Na Base 50, considerou-se 3 membros e 3 empresas e na bases seguintes 4 membros/ licitação e 4 empresas/licitação.

Tabela 6.1: Parâmetros para criação das bases de teste utilizadas

	Pessoas	Licitações	Empresas	Mesmo Endereço
Base 15	15	1	2	3
Base 50	50	2	5	5
Base 100	100	2	5	10
Base 1.000	1.000	20	50	100
Base 10.000	10.000	200	500	1.000
Base 100.000	100.000	2.000	5.000	10.000
Base 1.000.000	1.000.000	20.000	50.000	100.000

[1] https://sourceforge.net/projects/unbbayes/files/

Os testes foram feitos utilizando um computador com processador i5 (modelo 4460, com dois núcleos de 3,20 GHz) e 8 GB de memória RAM, executando o Windows 10 (versão 64 bits). Foram alocados 5 GB para a máquina virtual Java executando o UnBBayes.

Para os testes com o plug-in de PR-OWL 2 RL foi utilizada a *triplestore* GraphDB Lite, versão 6.6.3, e o algoritmo Bayes-Ball. Para cada base de teste, foi criado um repositório OWL 2 RL no GraphDB, feito as cargas do arquivo contendo a ontologia PR-OWL 2 RL, do arquivo contendo a ontologia "Fraudes em Licitações Públicas" e do arquivo contendo a base assertiva, em formato Turtle (ambas as tarefas foram realizadas utilizando a interface web GraphDB Free Workbench). A Tabela 6.2 mostra a quantidade de declarações em cada arquivo Turtle utilizado para carregar a base assertiva, a quantidade de declarações (incluindo as implicitas, geradas pelo processo de materialização) contidas em cada base após a carga dos três arquivos (PR-OWL 2 RL, Ontologia e Base Assertiva) e os tempos de carga para cada base.

Tabela 6.2: Tempo de carga das bases na *triplestore*

	Declarações Turtle	Declarações Base	Tempo Carga(s)
Base 15	97	7.908	0,1
Base 50	296	8.220	0,1
Base 100	574	8.657	0,3
Base 1.000	5.514	16.716	0,4
Base 10.000	55.104	98.760	2,5
Base 100.000	551.004	887.758	25,0
Base 1.000.000	5.510.004	8.807.760	272,1

Para os testes com o plug-in de PR-OWL 2 foi utilizado o raciocinador HermiT (versão 1.3.1) e o algoritmo Bottom-Up. Foi criado um arquivo OWL/XML para cada base, contendo a MTheory geradora e a base assertiva.

O critério utilizado para os testes foi o tempo gasto para responder a *query* `isSuspiciousProcurement(procurement0)`. A Tabela 6.3 descreve os resultados (tempos totais em segundos) de ambos os algoritmos para as Bases 0, 15, 50 e 100. A Base 0 corresponde a uma base contendo apenas as entidades da Base 15, sem nenhum relacionamento entre elas. Não foi considerado o tempo necessário para carregar a base assertiva na *triplestore*, pois esta carga ocorre apenas uma vez, e não antes de cada *query*.

Tabela 6.3: Tempo total de geração da SSBN nos plug-ins do UnBBayes

	Base 0	Base 15	Base 50	Base 100
PR-OWL 2	0,5	760,4	2.335,7	5.529,9
PR-OWL 2 RL	0,062	1,5	5,0	201,7

A Figura 6.2 mostra o gráfico contendo o crescimento dos tempos de avaliação utilizando o plug-in do PR-OWL 2 e o plug-in do PR-OWL 2 RL. Conforme ilustrado, na execução do algoritmo utilizando o plug-in do PR-OWL 2, com o HermiT como raciocinador e o algoritmo Bottom-Up para geração da SSBN, o tempo de execução cresce mais rapidamente que na execução do plug-in de PR-OWL 2 RL. Isto se deve ao fato do algoritmo Bottom-Up partir de todas as evidências da base, e ao tempo maior gasto pelo raciocinador OWL 2 DL para responder as *queries* em comparação com a *triplestore*.

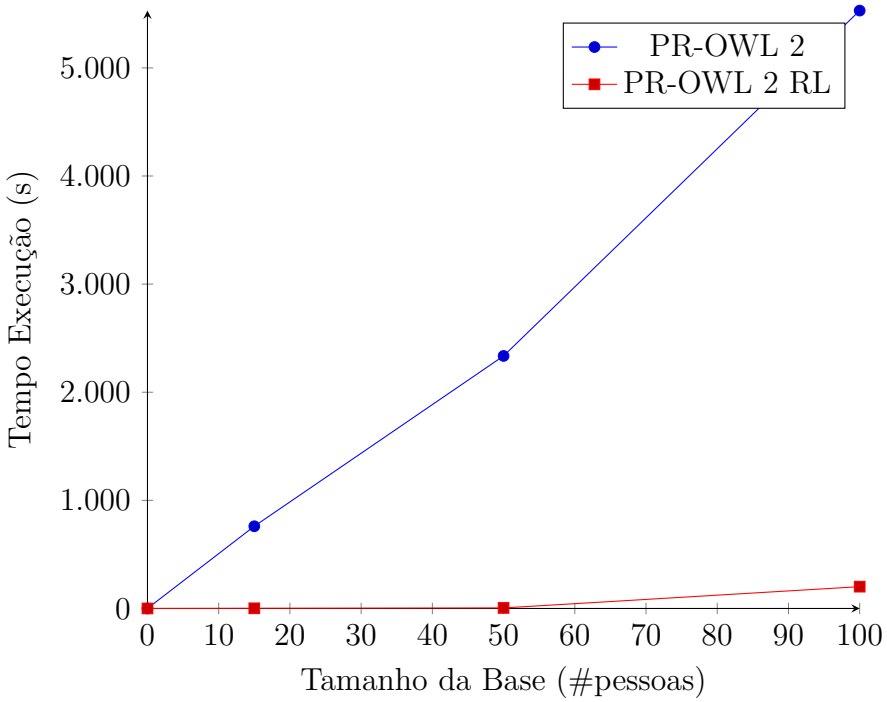

Figura 6.2: Gráfico comparativo da geração da SSBN nos plug-ins do UnBBayes

Como o algoritmo Bottom-Up parte de cada evidência presente na base assertiva, quanto maior for esta base, maior será a quantidade de nós no conjunto inicial e consequentemente maior a quantidade de nós pais gerados na SSBN. Para gerar os nós em uma MFrag é necessário avaliar todos os nós de contexto desta para verificar se a geração é realmente possível, e utilizando quais valores para as variáveis ordinárias. Portanto, a maior quantidade de nós também fará com que mais nós de contexto precisem ser avaliados. O algoritmo Bayes-Ball utiliza o critério de d-separação para gerar apenas nós que influenciam probabilisticamente o nó de *query*, dado os nós de evidência.

As *queries* submetidas ao raciocinador HermiT demoram mais para ser respondidas do que as submetidas a *triplestore* porque a inferência é feita em tempo de *query* e é baseada em uma lógica descritiva com complexidade exponencial. Na *triplestore* utilizada, a inferência é realizada em tempo de carga, por materialização, em tempo polinomial. A *query* consiste apenas em uma busca na base de dados.

A Tabela 6.4 ilustra a quantidade de nós gerados após os passos de inicialização (Passo 1), geração da *Grand* BN (Passo 2) e poda da rede (Passo 3) para ambos os plug-ins para a Base 15. Conforme pode ser observado na tabela, o plug-in de PR-OWL 2 RL montou uma SSBN com um nó a mais. Isto ocorreu devido a dois problemas na implementação de PR-OWL 2 no UnBBayes, um originando a geração de três nós a menos e o outro de quatro nós a mais. Os problemas são explicados a seguir.

Tabela 6.4: Números de nós gerados para Base 15

	Passo 1	Passo 2	Passo 3
Bottom-Up	78	116	45 [2]
Bayes-Ball	1	44	44

O primeiro problema se deve ao não suporte aos conectivos *AND* e *OR*. Ao avaliar nós de contexto com estes conectivos, a implementação de PR-OWL 2 considera que os nós foram avaliados como *false*. O nó `isFrontFor(person0, enterprise0)` causa a avaliação dos nós de contexto da MFrag `FrontOfEnteprise`. Como um dos nós possui o conectivo *AND* (nó 1 na Figura 4.4), o nó `isFrontFor` é definido para utilizar a distribuição default, e, consequentemente, seus pais (`hasAnnualIncome_person0`, `hasEducationLevel_person0` e `hasValue_procurement0`) não são gerados. Há evidência para todos estes nós pais, de forma que eles deveriam fazer parte da SSBN.

O segundo problema se refere a restrição na avaliação de nós `NOT`. A implementação não avalia expressões com `NOT`, porém, diferente da avaliação dos conectivos `AND` e `OR`, estes nós são avaliados como *true*. Assim, o nó de contexto `NOT(person1 = person2)` na MFrag `RelatedParticipantEnterprises` foi avaliado indevidamente como *true*, gerando quatro nós pais indevidos, conforme ilustrado na Figura 6.3.

As Tabelas 6.5 e 6.6 mostram a quantidade de nós gerados para as bases 50 e 100. O teste com a Base 1.000 utilizando o plug-in do PR-OWL 2 não foi concluído no tempo limite de 10 horas, e por isto, os testes seguintes não foram efetuados neste plug-in.

Tabela 6.5: Números de nós gerados para Base 50

	Passo 1	Passo 2	Passo 3
Bottom-Up	238	311	83
Bayes-Ball	1	82	80 [2]

[2]A diferença na quantidade de nós na rede final se deve a duas simplificações na implementação do algoritmo Bottom-Up no plug-in de PR-OWL 2, descritas no texto.

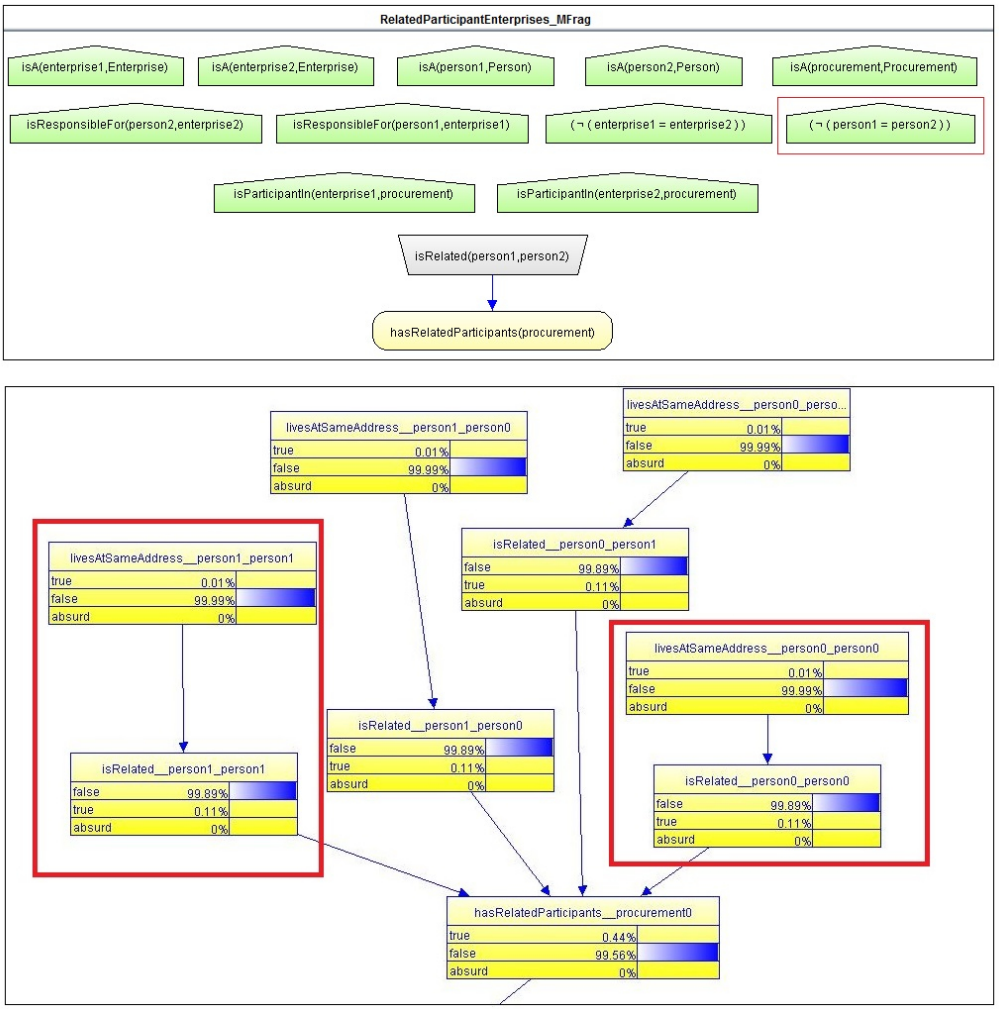

Figura 6.3: Falha na SSBN gerada pelo algoritmo Bottom-Up do UnBBayes

Tabela 6.6: Números de nós gerados para Base 100

	Passo 1	Passo 2	Passo 3
Bottom-Up	451	552	115
Bayes-Ball	1	117	113 [2]

A Tabela 6.7 mostra o tempo gasto (em segundos) para executar a *query* `isSuspicious-Procurement(procurement0)` em cada uma das bases no plug-in PR-OWL 2 RL. Na tabela, o Passo 4 corresponde a criação das CPT e o Passo 5 a compilação da rede.

Os testes a partir da Base 1.000 foram executados em tempo aproximadamente constante. Isto se deve ao fato da quantidade de relacionamentos efetivos considerados na SSBN gerada serem aproximadamente constantes, já que considerou-se a mesma quantidade de membros e de empresas por licitação (4 empresas/licitação e 4 membros/licitação). A geração da SSBN no algoritmo Bayes-Ball considera apenas as informações da

Tabela 6.7: Tempo de execução para *queries* utilizando plug-in PR-OWL 2 RL

	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5	Tempo Total
Base 1.000	0,01	3,0	0,002	233,0	303,2	539,0
Base 10.000	0,04	2,6	0,003	229,2	310,3	542,2
Base 100.000	0,03	2,0	0,003	234,2	301,3	537,6
Base 1.000.000	0,03	8,4	0,001	243,4	313,1	565,0

base assertiva relacionadas a *query* que se deseja resolver, e não toda a informação da base, como no algoritmo Bottom-Up. Portanto, apesar de se aumentar a base assertiva, como a quantidade destes relacionamentos se manteve constante nas bases maiores que a Base 1.000, o tempo necessário para resolver a *query* se manteve aproximadamente constante. Aumentando a quantidade de empresas/licitação e de membros/licitação para 5, a geração de SSBN não pôde ser executada com sucesso nas configurações especificadas porque o espaço alocado não foi suficiente para gerar e compilar a tabela de probabilidade do nó `hasMemberRelatedToParticipant`, gerado com aproximadamente 55 pais `isRelatedTo`.

6.3 Comparação Entre as Versões do PR-OWL

A Tabela 6.8 compara as duas versões do PR-OWL existentes com as características do PR-OWL 2 RL.

Tabela 6.8: Tabela comparativa entre as versões do PR-OWL

	PR-OWL	PR-OWL 2	PR-OWL 2 RL
Linguagem de Representação	OWL DL	OWL 2 DL	OWL 2 RL
Nós de Contexto	Lógica 1ª Ordem	Lógica 1ª Ordem	Restrito
Raciocínio Híbrido	Não	Sim	Sim
Incerteza em propriedades OWL	Não	Sim	Sim
Uso de Tipos Primitivos OWL	Não	Sim	Sim
Suporte a Polimorfismo	Não	Sim	Sim

O PR-OWL original foi escrito em OWL DL[3]. O autor cita a necessidade de futuramente fazer restrições na linguagem para evitar a indecidibilidade das *queries* e garantir a tratabilidade [23]. O PR-OWL 2 foi desenvolvido em OWL 2 DL, sendo feitas algumas concessões para manter a linguagem restrita a esta expressividade. Como exemplo, a propriedade `definesUncertaintyOf` deveria ligar um individuo (variável aleatória no modelo MEBN) a uma propriedade (propriedade da ontologia determinística, a qual se quer definir a incerteza), porém, o OWL 2 DL não permite que propriedades tenham como contradomínio outra propriedade. Isto foi solucionado transformando `definesUncertaintyOf`

[3]O PR-OWL foi proposto quando o OWL 1 ainda era a linguagem padrão para criação de ontologias

em uma propriedade de dados, ligando a variável ordinária a uma *string* contendo a URI da propriedade, ficando a cargo da máquina de inferência do PR-OWL 2 fazer o correto tratamento desta propriedade.

O PR-OWL 2 RL é escrito utilizando o *profile* OWL 2 RL, se baseando na semântica RDF, permitindo o uso em conjunto com *triplestores*. Para tal, o PR-OWL 2 foi transformado em uma ontologia leve, onde a verificação quanto a MTheory ser válida fica a cargo da máquina de inferência PR-OWL 2 RL. A ontologia probabilística escrita pelo usuário também deve seguir as restrições do profile OWL 2 RL, descritas na Seção 2.4, para que o raciocínio determinístico seja completo e consistente. *Triplestores* permitem o trabalho com grafos RDF arbitrários, permitindo que qualquer ontologia seja inserida na base, inclusive ontologias OWL 2 Full. A implementação do profile OWL 2 RL no entanto, garante apenas que a inferência será consistente (gerará apenas respostas corretas), mas não completa (todas as respostas corretas serão geradas) [38] quando aplicada a bases não adequadas ao profile.

A expressividade dos nós de contexto do PR-OWL e do PR-OWL 2 é em lógica de primeira ordem, ficando a implementação do raciocínio destes a cargo da máquina de inferência à ser desenvolvida, já que raciocinadores de OWL são baseadas em lógica descritiva. Os formatos de fórmulas válidas nos nós de contexto no PR-OWL 2 RL são restritos para se adequarem ao profile OWL 2 RL e ao raciocínio utilizando bases de dados RDF. As fórmulas aceitam os operadores *and*, *or* e *not* (com restrições), conforme Gramática BNF apresentada na Seção 4.4.

O PR-OWL possibilita apenas raciocínio probabilístico, enquanto o PR-OWL 2 permite raciocínio híbrido, característica mantida no PR-OWL 2 RL, através do uso da propriedades `definesUncertaintyOf` em conjunto com as propriedades `isSubjectIn` e `isObjectIn`. Da mesma forma, o uso de tipos primitivos OWL e o suporte ao polimorfismo, funcionalidades acrescentadas no PR-OWL 2, são mantidas no PR-OWL 2 RL.

A Tabela 6.9 compara as implementações disponíveis para o PR-OWL e PR-OWL 2 no framework UnBBayes com a implementação do novo plug-in para PR-OWL 2 RL.

No PR-OWL, a linguagem de representação KIF, utilizada no PowerLoom, se juntou ao OWL DL, já que a base assertiva é expressa em KIF, separada da ontologia OWL original. A linguagem PR-OWL possui construções para armazenar as evidências utilizando o formato OWL, porém estas não são utilizadas na implementação, dificultando o trabalho em domínios reais, expressos nos formatos padronizados da Web Semântica. No PR-OWL 2 esta limitação foi sanada, sendo tanto a TBox quanto a ABox armazenadas utilizando-se

[4]O PowerLoom é um sistema de representação de conhecimento e inferência (KR&R - *Knowledge Representation and Inference*

Tabela 6.9: Tabela comparativa entre as implementações do PR-OWL no UnBBayes

	PR-OWL	PR-OWL 2	PR-OWL 2 RL
Linguagem de Representação	OWL DL/KIF	OWL 2 DL	OWL 2 RL
Tipo do Raciocinador	Sistema KR&R [4]	*Raciocinador* DL	*Triplestore*
Reasoner Utilizado	PowerLoom	HermiT	GraphDB
Nós de Contexto	Lógica 1ª Ordem	Restrito - Ad Hoc	Restrito
Raciocínio Híbrido	Não	Sim	Sim
Incerteza em propriedades OWL	Não	Sim	Sim
Uso de Tipos Primitivos OWL	Não	Sim	Sim
Suporte a Polimorfismo	Não	Não	Não
Escalabilidade	Milhares	Milhares	Milhões

OWL. O PR-OWL 2 RL mantém esta característica, armazenando a ABox na *triplestore*, utilizando OWL serializado como RDF.

Os nós de contexto na implementação do PR-OWL aceitam fórmulas em lógica de primeira ordem, avaliadas pelo PowerLoom, havendo restrições apenas na avaliação de fórmulas com variáveis não preenchidas em momento de avaliação. No PR-OWL 2 são permitidas apenas as fórmulas nos formatos apresentados na Tabela 4.2, cuja avaliação foi implementada no UnBBayes. Os formatos de fórmulas válidos do PR-OWL 2 RL são baseados na forma como é feito o raciocínio no OWL 2 RL/RDF e como são feitas as *queries*, utilizando o SPARQL em bases de dados RDF.

O suporte a polimorfismo não foi implementado no PR-OWL 2 e no PR-OWL 2 RL, sendo um trabalho futuro.

O maior ganho do PR-OWL 2 RL é em relação à escalabilidade, onde há a proposta de se trabalhar com milhões de triplas RDF. Apesar das *triplestores* atuais permitirem trabalhar com bilhões de triplas RDF, o software utilizado no protótipo, a versão *Lite* do GraphDB, carrega a base de dados em memória ao iniciar o seu processamento, limitando o tamanho da base. Mesmo com esta limitação, os testes apresentados na Seção 6.2 mostraram que o plug-in do PR-OWL 2 RL, em conjunto com o novo algoritmo Bayes-Ball proposto, é mais escalável que o plug-in do PR-OWL 2. O usuário interessado em trabalhar com bases maiores poderá utilizar a versão comercial do GraphDB, ou de uma outra *triplestore* que utilize a interface `SAIL` do Sesame.

Como a inferência no MEBN consiste na geração de uma rede bayesiana, ainda poderão ocorrer problemas de escalabilidade devido à inferência nestas. O algoritmo de árvore de junção implementado no UnBBayes, por exemplo, apresenta problemas quando são gerados nós contendo uma grande quantidade de pais, que originarão cliques muito grandes. Esta restrição pode limitar o tamanho da base, dependendo do domínio utilizado. É necessário um trabalho futuro no estudo e implementação de algoritmos aproximados para lidar com a SSBN gerada.

Capítulo 7

Conclusões

Esta dissertação descreve um novo formalismo para tratamento de incerteza na Web Semântica, que utilizando uma versão menos expressiva do OWL, o profile OWL 2 RL, em conjunto com bases de dados RDF (*triplestores*) permite o trabalho com ontologias probabilísticas que contenham grandes bases assertivas. Este capítulo descreve a avaliação e as considerações finais sobre o trabalho desenvolvido.

Os objetivos específicos propostos na Seção 1.4 foram cumpridos da seguinte forma:

- Foi criada uma nova versão do PR-OWL que permite o uso da profile OWL 2 RL (que possuí complexidade temporal polinomial para os principais tipos de raciocínios) e bases de dados RDF (*triplestores*).
- Foi proposta uma gramática formal para produção das expressões em lógica de primeira ordem aceitas em nós de contexto que podem ser avaliada utilizando *triplestores*. Foi discutido como estas expressões podem ser avaliadas utilizando *queries* SPARQL.
- Foi proposto um algoritmo escalável para geração de SSBN adequado a domínios contendo grande base assertiva.
- Foi feito uma implementação do PR-OWL 2 RL e do algoritmo de geração de SSBN baseado no Bayes-Ball no UnBBayes. Foi feita a avaliação empírica do novo formalismo por meio de testes com o domínio "Fraudes em Licitações Públicas".

7.1 Contribuições

Este trabalho contribuiu com a proposição de um formalismo escalável para tratamento de incerteza na Web Semântica que permite o trabalho com ontologias probabilísticas contendo grandes bases assertivas. O formalismo proposto, o PR-OWL 2 RL representa

a incerteza através da tradução do domínio para um modelo MEBN, onde a inferência ocorre através da geração de SSBN. Este trabalho apresenta a sintaxe da nova linguagem e os formatos válidos para os nós de contexto, avaliados através de consultas a base de dados utilizando a linguagem SPARQL. Outra contribuição do trabalho é a proposição de um algoritmo escalável para geração de SSBN baseado no algoritmo Bayes-Ball. Este novo algoritmo pode ser utilizado em conjunto com as versões anteriores do PR-OWL e com o MEBN propriamente dito.

Foi desenvolvido um plug-in para o *framework* de código aberto UnBBayes implementando o formalismo, de forma que este pode ser avaliado e efetivamente utilizado. O plug-in está disponível no SourceForge (`https://sourceforge.net/projects/unbbayes/`). O Apêndice B contém detalhes sobre a implementação. Foi implementado também o algoritmo de geração de SSBN baseado no Bayes-Ball. Como esta implementação foi feita no plug-in de PR-OWL, ela está disponível para ambos os plug-ins de PR-OWL, PR-OWL 2 e PR-OWL 2 RL.

Foram publicados dois artigos científicos em conferências internacionais relacionados ao trabalho descrito nesta dissertação:

- *Scalable uncertainty treatment using triplestores and the OWL 2 RL profile*, 18th International Conference on Information Fusion, FUSION 2015, Washington, DC, USA, 6-9 de julho, 2015 [73]
- *PR-OWL 2 RL - A Language for Scalable Uncertainty Reasoning on the Semantic Web information*, 11th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2015) co-located with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, USA, 12 de outubro, 2015 [29]

O primeiro artigo foca em justificar a criação do novo formalismo, descrevendo as limitações do PR-OWL 2 e propondo o trabalho com OWL 2 RL e *triplestores*, enquanto o segundo procura descrever as restrições de expressividade necessárias para o novo formalismo.

7.2 Considerações Finais

A linguagem PR-OWL 2 RL é uma ontologia leve, adequada ao profile OWL 2 RL. Para isto, as expressões do PR-OWL 2 original definindo restrições lógicas quanto às classes da ontologia foram retiradas, restando apenas a hierarquia de classes e as definições de domínio e contra-domínio das propriedades. A verificação quanto à modelagem correta da MTheory utilizando a linguagem fica a cargo da máquina de inferência desenvolvida.

Apesar de não permitir expressar todas as fórmulas da lógica de primeira ordem, a gramática BNF proposta para os nós de contexto é expressiva o suficiente para o trabalho com a maioria das ontologias probabilísticas existentes. Estas construções tornam possível o trabalho com a ontologia probabilística desenvolvida para o domínio *Maritime Domain Awareness*, apresentada e [10] e [11] e com a ontologia "Fraudes em Licitações Públicas", apresentada em [17]. Elas ainda incluem todos os formatos existentes na implementação atual do PR-OWL 2 no UnBBayes, que utiliza o HermiT, raciocinador do OWL 2 para avaliar os nós de contexto. Costa [21] já havia discutido em seu trabalho original sobre o PR-OWL a possibilidade de criar versões mais restritivas da linguagem para garantir tratabilidade. Restrições de expressividade em prol da tratabilidade são comuns na lógica descritiva e na Web Semântica, e tem se mostrado um bom caminho para permitir o raciocínio prático com estes formalismos.

A linguagem PR-OWL 2 RL não substitui a linguagem PR-OWL 2. As características de cada uma fazem com que sejam adequadas a tipos de domínios diferentes. O PR-OWL 2 é recomendado para ontologias complexas, que contenham expressões complexas porém bases assertivas simples. O PR-OWL 2 RL, por outro lado, é ideal para o trabalho com ontologias leves, que podem ter bases assertivas volumosas. Estas ontologias são muito comuns na Web Semântica e são a base de projetos de *Linked Data*. Em [36], por exemplo, é proposto o profile OWL 2 LD, uma linguagem baseada no OWL 2 RL robusta e de fácil implementação para o trabalho com Linked Data, mantendo as construções do OWL mais utilizadas na Web. Além disso, bases de dados RDF são boas alternativas para o trabalho com *Big Data* por permitirem uma fácil integração entre bases diferentes, podendo o PR-OWL 2 RL ser utilizado para trabalhar com incerteza nestas bases.

O novo algoritmo de geração de SSBN proposto se baseia no algoritmo Bayes-Ball, utilizado para verificação de nós relevantes aos nós objetivos. O algoritmo proposto parte apenas dos nós de *query*, gerando os nós da SSBN de forma iterativa, avaliando os pais ou os filhos do nó analisado baseando-se nas regras do algoritmo Bayes-Ball. Conforme demonstrado nos testes, o algoritmo é mais escalável que o anterior, possibilitando a inferência em domínios com grande base assertiva, situação que o anterior não permitia. Foram feito testes com bases contendo milhões de assertivas para validar o algoritmo. Este trabalho melhora a escalabilidade na geração da SSBN, porém ainda poderão ocorrer problemas com a inferência na rede bayesiana gerada devido a quantidade de nós e arcos e aos algoritmos utilizados para realizar a inferência nesta. O algoritmo de árvore de junção, implementado no UnBBayes, apresenta problemas para redes bayesianas com cliques muito grandes.

7.3 Limitações e Trabalhos Futuros

A escalabilidade do novo formalismo se deve a utilização de uma versão do OWL 2 com menor complexidade e ao novo algoritmo de geração de SSBN que gera apenas nós d-conectados aos nós de *query*. O ganho de escalabilidade do algoritmo baseado no Bayes-Ball em relação ao algoritmo Bottom-Up no entanto, depende do domínio e da forma como este é modelado na MTheory geradora. Domínios que possuam as entidades da base assertiva fortemente relacionadas entre si, poderão gerar *Grand* BN semelhantes as geradas pelo algoritmo Bottom-Up. Como trabalho futuro, pode-se estudar padrões de modelagem que permitam melhor escalabilidade da ontologia probabilística desenvolvida, além de serem feitos novos testes com outros domínios e com dados reais do domínio "Fraudes em Licitações Públicas".

Algumas questões ainda precisam ser melhor estudadas em relação a rede bayesiana de situação específica, pois a inferência com esta ainda depende da sensibilidade do algoritmo utilizado ao tamanho e conexões da rede. O UnBBayes utiliza o algoritmo de árvore de junção [46] [47], um algoritmo de inferência exata que apresenta problemas ao trabalhar com cliques grandes. Algoritmos de inferência aproximada parecem ser uma abordagem mais razoável, à serem consideradas em trabalhos futuros.

Assim como este trabalho propõe estender o PR-OWL 2 ao profile OWL 2 RL, a adequação ao profile OWL 2 QL também pode ser analisada em trabalhos futuros. O OWL 2 QL baseia a sua inferência na reescrita das *queries* de forma que estas recuperem as informações explicitas e implícitas da base. A ontologia leve proposta para o PR-OWL 2 RL não atende todas as restrições do profile OWL 2 QL, além de ser necessário analisar as restrições para as fórmulas dos nós de contexto, a fim de criar uma versão do PR-OWL 2 para este profile. Pode-se ainda estudar restrições em ontologias probabilísticas criadas em PR-OWL 2 que permitam o aumento da escalabilidade, mesmo utilizando raciocinadores OWL 2 DL. A partir de uma formalização dos formatos de expressões em lógica de primeira ordem aceitos no PR-OWL 2 pode-se analisar otimizações na avaliação destas.

Como trabalhos futuros, pode-se ainda avaliar a possibilidade de transformar o gerador de código para a ontologia "Fraudes em Licitações Públicas" em um *benchmark* para ontologias probabilísticas, semelhante a forma como o LUBM é utilizado como *benchmark* para ontologias em OWL e RDF.

Referências

[1] Grigoris Antoniou, Paul T. Groth, Frank van Harmelen, e Rinke Hoekstra. *A Semantic Web Primer, 3rd Edition*. MIT Press, 2012. 10

[2] Franz Baader e Werner Nutt. Basic description logics. Em Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, e Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, páginas 43–95. Cambridge University Press, 2003. 11

[3] Tim Berners-Lee, James Hendler, e Ora Lassila. The semantic web. Em *Scientific American*, volume 285, páginas 34–43. 2001. 1, 9

[4] Barry Bishop e Spas Bojanov. Implementing OWL 2 RL and OWL 2 QL rule-sets for OWLIM. Em Michel Dumontier e Mélanie Courtot, editors, *OWLED*, volume 796 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011. 18

[5] Barry Bishop, Atanas Kiryakov, Damyan Ognyanoff, Ivan Peikov, Zdravko Tashev, e Ruslan Velkov. OWLIM: A family of scalable semantic repositories. *Semantic Web*, 2(1):33–42, 2011. 4, 13, 18

[6] Christian Bizer, Tom Heath, e Tim Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009. 3, 9

[7] Jürgen Bock, Peter Haase, Qiu Ji, e Raphael Volz. Benchmarking owl reasoners. Em *Proc. of the ARea2008 Workshop, Tenerife, Spain (June 2008)*, 2008. 13

[8] Jeen Broekstra, Arjohn Kampman, e Frank Van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF Schema. Em *The Semantic Web—ISWC 2002*, páginas 54–68. Springer, 2002. 13, 18

[9] Rommel N. Carvalho. *Probabilistic Ontology: Representation and Modeling Methodology*. Tese de Doutorado, George Mason University, Fairfax, VA, USA, 2011. 3, 25, 26, 30

[10] Rommel N. Carvalho, Paulo Cesar G. da Costa, Kathryn B. Laskey, e Kuo-Chu Chang. PROGNOS: predictive situational awareness with probabilistic ontologies. Em *13th Conference on Information Fusion, FUSION 2010, Edinburgh, UK, July 26-29, 2010*, páginas 1–8, 2010. 74

[11] Rommel N. Carvalho, Richard Haberlin, Paulo C. G. da Costa, Kathryn B. Laskey, e Kuo-Chu Chang. Modeling a probabilistic ontology for maritime domain awareness. Em *FUSION*, páginas 1–8. IEEE, 2011. 74

[12] Rommel N. Carvalho, Marcelo Ladeira, Laécio L. Santos, Shou Matsumoto, e Paulo C. G. Costa. A GUI tool for plausible reasoning in the semantic web using MEBN. Em *Innovative Applications in Data Mining*, páginas 17–45. 2009. 2, 27, 48

[13] Rommel N. Carvalho, Kathryn B. Laskey, e Paulo C. G. Costa. Compatibility formalization between PR-OWL and OWL. Em *Proceedings of the First International Workshop on Uncertainty in Description Logics (UniDL) on Federated Logic Conference (FLoC) 2010*, Edinburgh, UK, July 2010. 25

[14] Rommel N. Carvalho, Kathryn B. Laskey, e Paulo C. G. Costa. PR-OWL 2.0–bridging the gap to OWL semantics. Em *Uncertainty Reasoning for the Semantic Web II*, páginas 1–18. Springer, 2013. 25, 30

[15] Rommel N. Carvalho, Kathryn B. Laskey, e Paulo Cesar G. da Costa. PR-OWL 2.0 - bridging the gap to OWL semantics. Em *Proceedings of the 6th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2010), collocated with the 9th International Semantic Web Conference (ISWC-2010), Shanghai, China, November 7, 2010*, páginas 73–84, 2010. 43

[16] Rommel N. Carvalho, Kathryn B. Laskey, e Paulo Cesar G. da Costa. Uncertainty modeling process for semantic technology. *PeerJ PrePrints*, 4:e2045, 2016. 6

[17] Rommel N. Carvalho, Shou Matsumoto, Kathryn B. Laskey, Paulo C. G. da Costa, Marcelo Ladeira, e Laécio L. Santos. Probabilistic ontology and knowledge fusion for procurement fraud detection in brazil. Em *Uncertainty Reasoning for the Semantic Web II, International Workshops URSW 2008-2010 Held at ISWC and UniDL 2010 Held at FLoC, Revised Selected Papers*, páginas 19–40, 2013. 3, 6, 10, 22, 74

[18] Stefano Ceri, Georg Gottlob, e Letizia Tanca. *Logic Programming and Databases*. Springer, 1990. 14

[19] Hans Chalupsky, Robert M. MacGregor, e Thomas Russ. Powerloom manual. http://www.isi.edu/isd/LOOM/PowerLoom/documentation/manual.pdf, 2010. 4, 27, 34

[20] Eugene Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991. 20, 21

[21] Paulo C. G. Costa. *Bayesian Semantics for the Semantic Web*. Tese de Doutorado, George Mason University, Fairfax, VA, USA, July 2005. 1, 2, 23, 74

[22] Paulo C. G. Costa, Marcelo Ladeira, Rommel N. Carvalho, Kathryn B. Laskey, Laécio L. Santos, e Shou Matsumoto. A first-order Bayesian tool for probabilistic ontologies. Em David Wilson e H. Chad Lane, editors, *FLAIRS Conference*, páginas 631–636. AAAI Press, 2008. 2, 27

[23] Paulo C. G. Costa, Kathryn B. Laskey, e Kenneth J. Laskey. PR-OWL: a Bayesian ontology language for the semantic web. Em *Uncertainty Reasoning for the Semantic Web I: ISWC International Workshops, URSW 2005-2007, Revised Selected and Invited Papers*, páginas 88–107. Springer-Verlag, 2008. 2, 23, 25, 30, 69

[24] Fabio G. Cozman. Credal networks. *Artificial intelligence*, 120(2):199–233, 2000. 31

[25] Fabio G. Cozman e Rodrigo B. Polastro. Loopy propagation in a probabilistic description logic. Em *Scalable Uncertainty Management*, páginas 120–133. Springer, 2008. 2, 31

[26] Richard Cyganiak, David Wood, e Markus Lanthaler. RDF 1.1 concepts and abstract syntax (W3C Recommendation). http://www.w3.org/TR/rdf11-concepts/, 2014. Acesso em 01/02/2015. 1, 10

[27] Zhongli Ding, Yun Peng, e Rong Pan. BayesOWL: Uncertainty modeling in semantic web ontologies. *Soft Computing in Ontologies and Semantic Web*, páginas 3–29, 2006. 1, 30

[28] Francesco M. Donini. Complexity of reasoning. Em *The Description Logic Handbook: Theory, Implementation, and Applications*, páginas 96–136, 2003. 33

[29] Laécio L. dos Santos, Rommel N. Carvalho, Marcelo Ladeira, Weigang Li, e Gilson Libório Mendes. PR-OWL 2 RL - A language for scalable uncertainty reasoning on the semantic web information. Em *Proceedings of the 11th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2015) co-located with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, USA, October 12, 2015.*, páginas 14–25, 2015. 73

[30] Amira Essaid e Boutheina B. Yaghlane. BeliefOWL: An evidential representation in OWL ontology. Em *Proceedings of the Fifth International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009), Washington DC, USA, October 26, 2009*, volume 527 of *CEUR Workshop Proceedings*, páginas 77–80. CEUR-WS.org, 2009. 2

[31] The Apache Software Foundation. Apache Jena - TDB. http://jena.apache.org/documentation/tdb/, 2015. Acesso em 01/02/2015. 4

[32] Yoshio Fukushige. Representing probabilistic knowledge in the Semantic Web. Em *Proceedings of the W3C Workshop on Semantic Web for Life Sciences*, October 2004. 30

[33] Dan Geiger, Thomas Verma, e Judea Pearl. d-separation: From theorems to algorithms. Em *UAI '89: Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence, Windsor, Ontario, Canada, August 18-20, 1989*, páginas 139–148, 1989. 50

[34] Michael R. Genesereth, Richard E Fikes, et al. *Knowledge Interchange Format-version 3.0: Reference manual*. Computer Science Department, Stanford University, 1992. 27

[35] Rosalba Giugno e Thomas Lukasiewicz. P-SHOQ (D): A probabilistic extension of SHOQ (D) for probabilistic ontologies in the semantic web. Em *JELIA*, volume 2, páginas 86–97. Springer, 2002. 1, 31

[36] Birte Glimm, Aidan Hogan, Markus Krötzsch, e Axel Polleres. OWL: yet to arrive on the web of data? Em *WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012*, 2012. 74

[37] Christine Golbreich e Evan K. Wallace. OWL 2 Web Ontology Language new features and rationale (W3C Recommendation). http://www.w3.org/TR/owl2-new-features/, 2012. Acesso em 01/06/2016. 13

[38] W3C OWL Working Group. OWL 2 Web Ontology Language document overview (W3C Recommendation). http://www.w3.org/TR/owl2-overview/, 2012. Acesso em 01/02/2015. 13, 14, 17, 70

[39] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, 1995. 9

[40] Yuanbo Guo, Zhengxiang Pan, e Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2):158–182, 2005. 34

[41] Volker Haarslev e Ralf Möller. RACER system description. Em *Automated Reasoning, First International Joint Conference, IJCAR 2001, Siena, Italy, June 18-23, 2001, Proceedings*, páginas 701–706, 2001. 33

[42] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, e Sebastian Rudolph. OWL 2 Web Ontology Language primer (W3C Recommendation). https://www.w3.org/TR/2012/REC-owl2-primer-20121211/, 2012. Acesso em 01/07/2016. 14, 17

[43] Pascal Hitzler, Markus Krützsch, e Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC Press, 1 edition, 2010. 9, 11, 12, 14, 15

[44] Franz Inc. AllegroGraph RDFStore Web 3.0's Database. http://franz.com/agraph/allegrograph/, 2015. Acesso em 01/02/2015. 4, 18

[45] Finn V Jensen. *An introduction to Bayesian networks*, volume 210. UCL press London, 1996. 21

[46] Finn V Jensen, Steffen L Lauritzen, e Kristian G Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational statistics quarterly*, 1990. 75

[47] Finn Verner Jensen, Kristian G. Olesen, e Stig K. Andersen. An algebra of bayesian belief universes for knowledge-based systems. *Networks*, 20(5):637–659, 1990. 75

[48] Daphne Koller, Alon Levy, e Avi Pfeffer. P-CLASSIC: A tractable probablistic description logic. *AAAI/IAAI*, 1997:390–397, 1997. 2, 31

[49] Vladimir Kolovski, Zhe Wu, e George Eadon. Optimizing enterprise-scale OWL 2 RL reasoning in a relational database system. Em *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, volume 6496 of *Lecture Notes in Computer Science*, páginas 436–452. Springer, 2010. 5, 18

[50] Markus Krötzsch. OWL 2 profiles: An introduction to lightweight ontology languages. Em *Reasoning Web. Semantic Technologies for Advanced Query Answering - 8th International Summer School 2012, Vienna, Austria, September 3-8, 2012. Proceedings*, páginas 112–183, 2012. 15

[51] Kathryn B. Laskey. MEBN: a language for First-Order Bayesian knowledge bases. *Artificial Intelligence*, 172(2-3):140–178, 2008. 2, 4, 21, 22, 23, 27, 28, 45, 47

[52] Kenneth Laskey e Kathryn B. Laskey. Uncertainty reasoning for the World Wide Web: Report on the URW3-XG incubator group. URW3-XG, W3C, 2008. 2

[53] Yan Lin e Marek J. Druzdzel. Computational advantages of relevance reasoning in bayesian belief networks. Em *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Brown University, Providence, Rhode Island, USA, August 1-3, 1997*, páginas 342–350, 1997. 47

[54] Thomas Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852–883, 2008. 31

[55] Suzanne M. Mahoney e Kathryn B. Laskey. Constructing Situation Specific Belief Networks. Em *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998*, páginas 370–37, 1998. 23, 45

[56] Shou Matsumoto. *Framework Based in Plug-ins for Reasoning with Probabilistic Ontologies*. Dissertação de Mestrado, Universidade de Brasília, Brasília, Brasil, 2011. 3, 28, 35, 89

[57] Shou Matsumoto, Rommel N. Carvalho, Marcelo Ladeira, Paulo C. G. Costa, Laecio L. Santos, Danilo Silva, Michael Onishi, Emerson Machado, e Ke Cai. UnBBayes: a Java framework for probabilistic models in AI. *Java in Academia and Research. iConcept Press. http://unbbayes. sourceforge. net*, 2011. 2, 27

[58] Shou Matsumoto e Laécio Lima dos Santos. *Framework para redes bayesianas multientidades e ontologias probabilisticas*. Trabalho de Graduação, Universidade de Brasília, Brasília, Brasil, 2008. 48

[59] Deborah L. McGuinness e Frank van Harmelen. OWL Web Ontology Language overview (W3C Recommendation). http://www.w3.org/TR/owl-features/, 2004. Acesso em 01/02/2015. 1

[60] Georgios Meditskos e Nick Bassiliades. Dlejena: A practical forward-chaining OWL 2 RL reasoner combining jena and pellet. *J. Web Sem.*, 8(1):89–94, 2010. 16

[61] Boris Motik. KAON2 - scalable reasoning over ontologies with large data sets. *ERCIM News*, 2008(72), 2008. 13

[62] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, e Carsten Lutz Achille Fokoue. OWL 2 Web Ontology Language profiles (W3C Recommendation). http://www.w3.org/TR/owl2-profiles/, 2012. Acesso em 01/06/2016. 13, 14, 15, 16, 33, 40, 41, 95, 96

[63] Boris Motik, Peter F. Patel-Schneider, e Bernardo Cuenca Grau. OWL 2 Web Ontology Language Direct Semantics (W3C Recommendation). https://www.w3.org/TR/owl2-direct-semantics/, 2012. Acesso em 01/07/2016. 40

[64] Boris Motik, Peter F. Patel-Schneider, e Bijan Parsia. OWL 2 Web Ontology Language structural specification and functional-style syntax (W3C Recommendation). https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/, 2012. Acesso em 01/07/2016. 14, 95

[65] Ontotext. Site do GraphDB. http://www.ontotext.com/products/ontotext-graphdb/, 2014. Acesso em 01/02/2015. 18

[66] Bijan Parsia e Evren Sirin. Pellet: An OWL DL reasoner. Em *Third International Semantic Web Conference-Poster*, volume 18, 2004. 13, 33

[67] Peter F. Patel-Schneider e Boris Motik. OWL 2 Web Ontology Language: mapping to rdf graphs (W3C Recommendation). http://www.w3.org/TR/2012/REC-owl2-mapping-to-rdf-20121211/, 2012. Acesso em 01/07/2016. 14, 96

[68] Axel Polleres, Aidan Hogan, Renaud Delbru, e Jürgen Umbrich. RDFS and OWL reasoning for linked data. Em *Reasoning Web. Semantic Technologies for Intelligent Data Access - 9th International Summer School 2013, Mannheim, Germany, July 30 - August 2, 2013. Proceedings*, páginas 91–149, 2013. 9

[69] Livia Predoiu e Heiner Stuckenschmidt. Probabilistic extensions of semantic web languages - a survey. Em *The Semantic Web for Knowledge and Data Management: Technologies and Practices*. Idea Group Inc, 2008. 14, 30, 31

[70] Protégé. The Protégé ontology editor and knowledge acqisition system. http://protege.stanford.edu/, 2000. 29

[71] Dave Reynolds. OWL 2 RL in RIF (W3C Recommendation). http://www.w3.org/TR/rif-owl-rl/, 2013. Acesso em 01/02/2015. 18

[72] Stuart J. Russell e Peter Norvig. *Artificial Intelligence - A Modern Approach*. Pearson Education, 3 edition, 2010. 19, 21

[73] Laécio L. Santos, Rommel N. Carvalho, Marcelo Ladeira, Li Weigang, Kathryn B. Laskey, e Paulo C. G. Costa. Scalable uncertainty treatment using triplestores and the OWL 2 RL profile. Em *18th International Conference on Information Fusion, FUSION 2015, Washington, DC, USA, July 6-9, 2015*, páginas 924–931, 2015. 73

[74] Ross D. Shachter. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). Em *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998*, páginas 480–487, 1998. 5, 50, 51

[75] Rob Shearer, Boris Motik, e Ian Horrocks. HermiT: A highly-efficient OWL reasoner. Em Catherine Dolbear, Alan Ruttenberg, e Ulrike Sattler, editors, *OWLED*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008. 3, 13, 29, 32, 33

[76] Giorgos Stoilos, Giorgos B. Stamou, Vassilis Tzouvaras, Jeff Z. Pan, e Ian Horrocks. Fuzzy OWL: Uncertainty and the semantic web. Em *OWLED*, 2005. 1

[77] Andrew S. Tanenbaum. *Computer networks (4. ed.)*. Prentice Hall, 2002. 8

[78] HJ ter Horst. Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):79–115, 2005. 14, 18

[79] Dmitry Tsarkov e Ian Horrocks. FaCT++ description logic reasoner: System description. Em Ulrich Furbach e Natarajan Shankar, editors, *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4130 of *Lecture Notes in Computer Science*, páginas 292–297. Springer, 2006. 13, 33

[80] Octavian Udrea, Yu Deng, Edward Hung, e V. S. Subrahmanian. Probabilistic ontologies and relational databases. Em *Proceedings of the OTM Confederated International Conferences (CoopIS, DOA, and ODBASE)*, volume 3760, páginas 1–17, Agia Napa, Cyprus, 2005. Lecture Notes in Computer Science. 30

[81] W3C. SPARQL 1.1 overview (W3C Recommendation). http://www.w3.org/TR/sparql11-overview/, 2013. Acesso em 01/02/2015. 9

[82] W3C. Página da web da World Wide Web Consortium. http://www.w3.org/, 2014. Acesso em 01/02/2015. 2

[83] W3C. Large triple stores. https://www.w3.org/wiki/LargeTripleStores, 2015. Acesso em 01/02/2015. 17

[84] Y. Yang e J. Calmet. OntoBayes: An ontology-driven uncertainty model. Em *Proceedings of the International Conference on Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC)*, Vienna, Austria, 2005. 1, 30

[85] Yujiao Zhou, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, e Jay Banerjee. Making the most of your triple store: query answering in OWL 2 using an RL reasoner. Em Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo A. Baeza-Yates, e Sue B. Moon, editors, *WWW*, páginas 1569–1580. International World Wide Web Conferences Steering Committee / ACM, 2013. 17, 18

Apêndice A

Sintaxe do PR-OWL 2 RL

Este apêndice descreve a sintaxe do PR-OWL 2 RL.

A.1 Classes

A Figura A.1 apresenta a hierarquia de classes do PR-OWL 2 RL.

A.2 Propriedades

A Tabela A.1 apresenta as propriedades de objeto presentes na nova linguagem, enquanto a Tabela A.2 apresenta as propriedades de dados. A última coluna de ambas as tabelas indica as características das propriedades, onde "F" indica Funcional e "I" indica Inverso-Funcional (não há relacionamentos transitivos, simétricos, assimétricos, reflexivos ou irreflexivos).

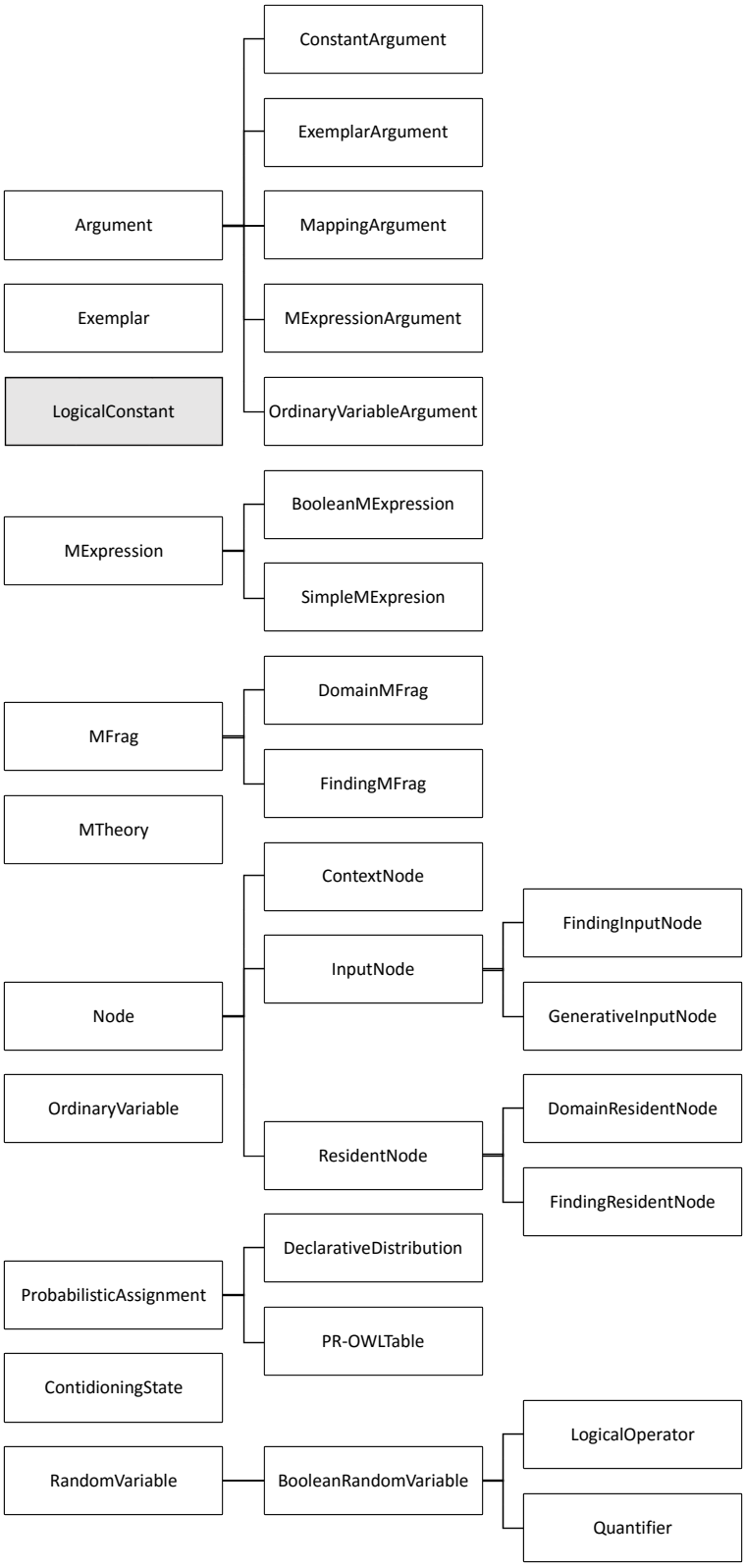

Figura A.1: Hierarquia de classes do PR-OWL 2 RL

Tabela A.1: Propriedades de objeto do PR-OWL 2 RL

	Propriedade	Domínio	Contra-Domínio	
1	hasArgument	•	Argument	I
1.1	hasArgumentA	MExpression	•	I
1.2	hasArgumentB	RandomVariable	•	I
2	hasConditioningState	ProbabilisticAssignment	ConditioningState	
3	hasContextNode	DomainMFrag	ContextNode	
4	hasResidentNode	MFrag	ResidentNode	
5	hasInputNode	MFrag	InputNode	
6	hasMExpression	Node	MExpression	F
7	hasExemplar	MFrag	Exemplar	
8	hasMFrag	MTheory	MFrag	
9	hasNode	MFrag	Node	I
10	hasOrdinaryVariable	MFrag	OrdinaryVariable	
11	hasParent	ResidentNode	NodeThatCanBeParent	
12	hasParentName	ConditioningState	NodeThatCanBeParent	F
13	hasParentState	ConditioningState	•	F
14	hasProbabilisticAssignment	PR-OWL Table	ProbabilisticAssignment	I
15	hasProbabilisticDistribution	•	ProbabilisticDistribution	F
15.1	hasProbabilisticDistributionA	RandomVariable	•	F
15.2	hasProbabilisticDistributionB	ResidentNode	•	F
16	hasStateName	ProbabilisticAssignment	•	F
17	typeOfArgument	Argument	•	F
18	typeOfMExpression	MExpression	RandomVariable	F
19	isPossibleObjectValueOf	•	RandomVariable	
20	isArgumentOf	Argument	•	I
20.1	isArgumentOfA	•	MExpression	I
20.2	isArgumentOfB	•	RandomVariable	I
21	isConditioningStateOf	ConditioningState	ProbabilisticAssignment	
22	isContextNodeIn	ContextNode	DomainMFrag	
23	isExemplarIn	Exemplar	MFrag	
24	isInputNodeIn	InputNode	MFrag	F
25	isMExpressionOf	MExpression	Node	
26	isMFragOf	MFrag	MTheory	
27	isNodeIn	Node	MFrag	F
28	isOrdinaryVariableIn	OrdinaryVariable	MFrag	F
29	isParentOf	NodeThatCanBeParent	ResidentNode	
30	isProbabilisticAssignmentIn	ProbabilisticAssignment	PR-OWLTable	F
31	isProbabilisticDistributionOf	ProbabilisticDistribution	•	F
31.1	isProbabilisticDistributionOfA	•	RandomVariable	F
31.2	isProbabilisticDistributionOfB	•	ResidentNode	F
32	isResidentNodeIn	ResidentNode	MFrag	F
33	isTypeOfArgumentIn	•	Argument	I
34	isTypeOfMExpression	RandomVariable	MExpression	

Tabela A.2: Propriedades de dados do PR-OWL 2 RL

	Propriedade	Domínio	Contra-Domínio	
1	definesUncertaintyOf	RandomVariable	anyURI	F
2	hasArgumentNumber	Argument	nonNegativeInteger	F
3	hasDeclaration	DeclarativeDistribution	string	F
4	hasPossibleValues	RandomVariable	anyURI	
5	hasStateProbability	ProbabilisticAssignment	float	F
6	isObjectIn	MappingArgument	anyURI	F
7	isRepresentedAs	MappingArgument	•	F
8	isSubjectIn	MappingArgument	anyURI	F
9	isSubstitudedBy	•	anyURI	
9.1	isSubstitudedByA	Exemplar	•	
9.2	isSubstitudedByB	OrdinaryVariable	•	
10	typeOfDataArgument	ConstantArgument	Literal	F

Apêndice B

Implementação do PR-OWL 2 RL no UnBBayes

Este apêndice descreve a implementação do plug-in de PR-OWL 2 RL, desenvolvido para o UnBBayes. A Seção 6.1 descreve e justifica as escolhas feitas para a implementação do plug-in. A Seção B.1 descreve a arquitetura em plug-ins do UnBBayes. E finalmente, a Seção B.2 descreve a arquitetura do plug-in desenvolvido.

B.1 Arquitetura do UnBBayes

O UnBBayes possui sua arquitetura baseada em plug-ins, utilizando como base o Java Plugin Framework (JPF) [1]. O JPF, que possuí licença LGPL, permite a construção de projetos Java escaláveis e com b aixo custo de manutenção e provê mecanismos para descoberta e carregamento dinâmico sob demanda. Utilizando o JPF, o UnBBayes permite que novas implementações sejam desenvolvidas de forma independente.

O plug-in `core` do UnBBayes contém as funcionalidades básicas, como por exemplo a implementação dos algoritmos para inferência em redes bayesianas. O acoplamento entre os plug-ins é feito através de pontos de extensão, métodos gancho bem definidos que podem ser estendidos por outros plug-ins para adicionar novas características e funcionalidades. A Figura B.1 ilustra os pontos de extensão do plug-in `core` e do plug-in `MEBN`.

O plug-in `MEBN` estende o ponto de extensão `Module` do plug-in `core`. Este ponto de extensão deve ser implementado por plug-ins relativamente auto-suficientes, com janelas internas invocadas ao acionar os menus ou botões da barra de ferramentas principal do UnBBayes.

Os pontos de extensão oferecidos pelo plug-in `MEBN` são os seguintes:

[1] http://jpf.sourceforge.net/

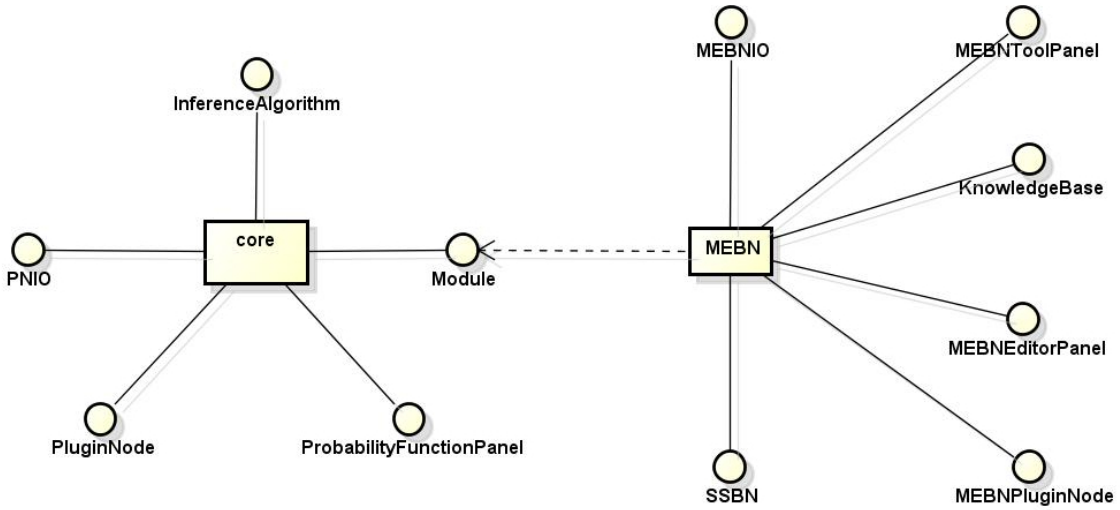

Figura B.1: Pontos de extensão dos plug-ins core e MEBN do UnBBayes

- `MEBNIO`: deve ser estendido para implementar alternativas para a persistência do modelo MEBN.
- `SSBN`: deve ser estendido para implementar algoritmos para geração de redes bayesianas específicas de situação.
- `MEBNToolPanel`: deve ser estendido para implementar novas barras de ferramentas de edições do modelo MEBN.
- `KnowledgeBase`: deve ser estendido para implementar novas bases de conhecimento.
- `MEBNEditorPanel`: deve ser estendido para implementar novos painéis de edição do modelo MEBN.
- `MEBNPluginNode`: deve ser estendido para implementar novos tipos de nós.

B.2 Arquitetura do Plug-in para PR-OWL 2 RL

A implementação do PR-OWL 2 RL no UnBBayes consiste em três tarefas: implementação da base de conhecimento utilizando triplestores; implementação do novo algoritmo para geração de SSBN; e persistência utilizando o PR-OWL 2 RL.

A Figura B.2 mostra o diagrama UML contendo as classes que implementam a persistência em PR-OWL 2 RL. Estas classes estendem a implementação anterior de PR-OWL 2 e utilizam a API OWL 2 e a API do Protégé para trabalhar com o OWL. O arquivo OWL 2 RL é salvo em formato RDF/XML. Abaixo segue a descrição da funcionalidade de

cada classe nova. O leitor interessado em uma descrição aprofundada da implementação de PR-OWL 2 no UnBBayes poderá consultar [56].

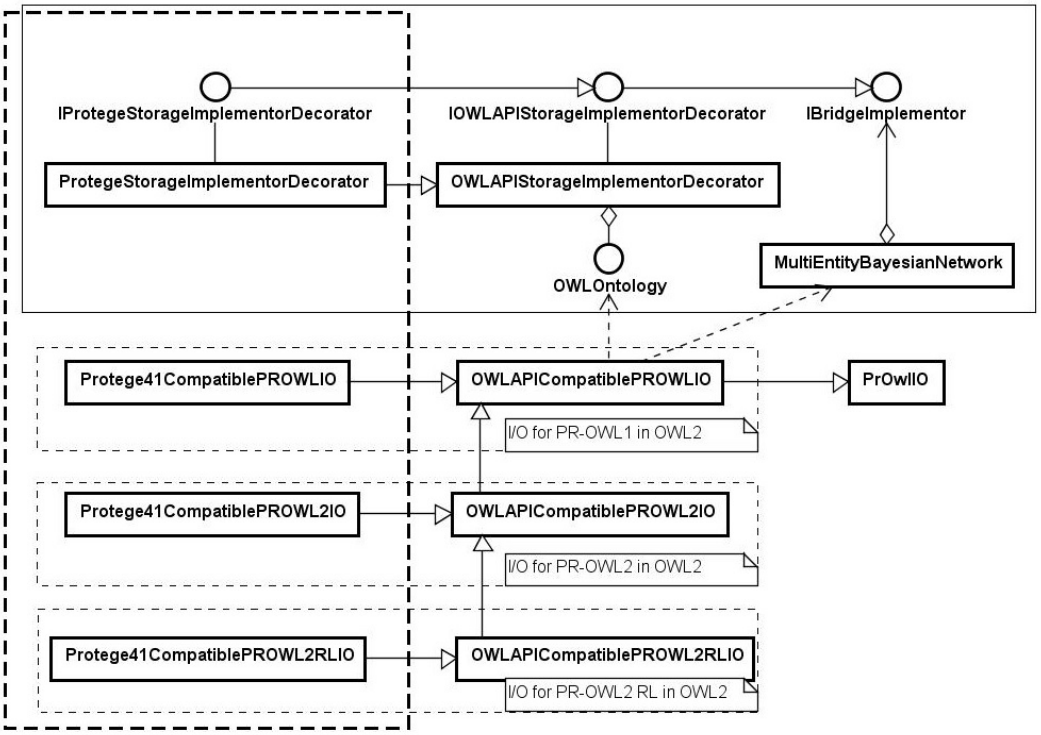

Figura B.2: Arquitetura das classes de I/O em PR-OWL 2 RL

- **Classe MultiEntityBayesianNetwork**: encapsula a MTheory geradora. É composta internamente por uma ou mais MFrags, que conterão os nós do modelo (nós residente, nós de entrada e nós de contexto).
- **Classe PrOwlIO**: implementa a interface MebnIO para carregar e salvar arquivos no formato PR-OWL. Utiliza as classes `LoaderPrOWLIO` e `SaverPrOWLIO` para cada uma das tarefas.
- **Classe Protege41CompatiblePROWL2RLIO**: carrega classes adicionais específicas do Protégé 4.1 e delega posteriormente para as rotinas `OWLAPICompatiblePROWL2RLIO`.
- **Classe OWLAPICompatiblePROWL2RLIO**: responsável pelo carregamento de ontologias PR-OWL 2 RL utilizando a API OWL 2.
- **Classe OWLAPIStorageImplementorDecorator**: responsável por implementar mecanismos de persistência de dados da MTheory como ontologia OWL. Torna a `OWLOntology` (ontologia encapsulada por classe da OWLAPI) acessível pela MTheory de forma indireta.

- **Classe ProtegeStorageImplementorDecorator**: contém objetos específicos do Protégé, como o *reasoner*.

A Figura B.3 descreve a arquitetura da base de conhecimento utilizando *triplestores*.

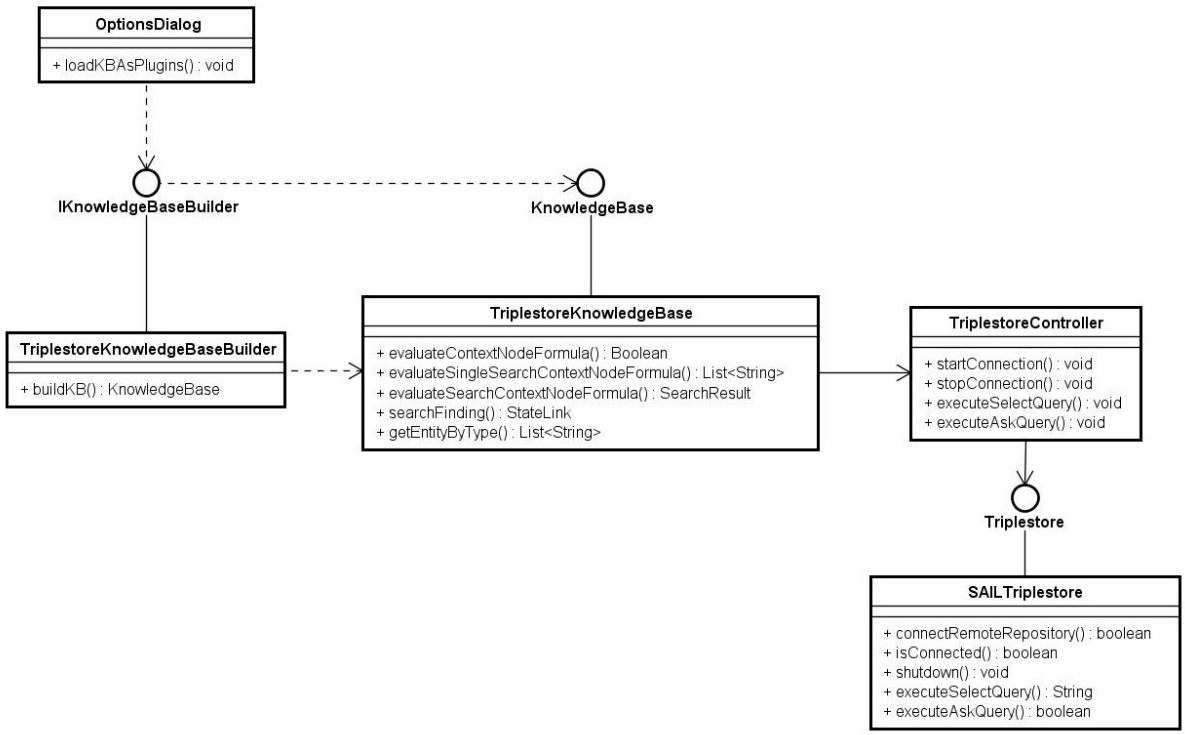

Figura B.3: Arquitetura da base de conhecimento utilizando *triplestores*

- **Classe OptionsDialog**: caixa de diálogo que permite ao usuário selecionar entre as bases de conhecimento disponíveis, e entre os algoritmos de geração de SSBN disponíveis.
- **Classe TriplestoreKnowledgeBaseBuilder**: responsável por criar o objeto `KnowledgeBase` referente a *triplestore*. Implementa `IKBOptionPanelBuilder` oferecendo a GUI para seleção das opções necessárias para se conectar a base.
- **Classe TriplestoreKnowledgeBase**: implementa a interface `KnowledgeBase` para *triplestore*. A busca de informações na base é feita através de consultas a base, mediadas pelo `TriplestoreController`.
- **Classe TriplestoreController**: controlador utilizado para fazer a conexão com a base de dados propriamente dita. Permite que opções a implementação utilizando o SAIL sejam implementadas.

- **Classe SAILTriplestore**: implementação da base de dados utilizando a interface SAIL do Sesame.

A Figura B.4 mostra a arquitetura da implementação do algoritmo Bayes Ball para geração de SSBN. O algoritmo foi implementado no plug-in de MEBN, podendo ser utilizado por ambas as implementações de PR-OWL.

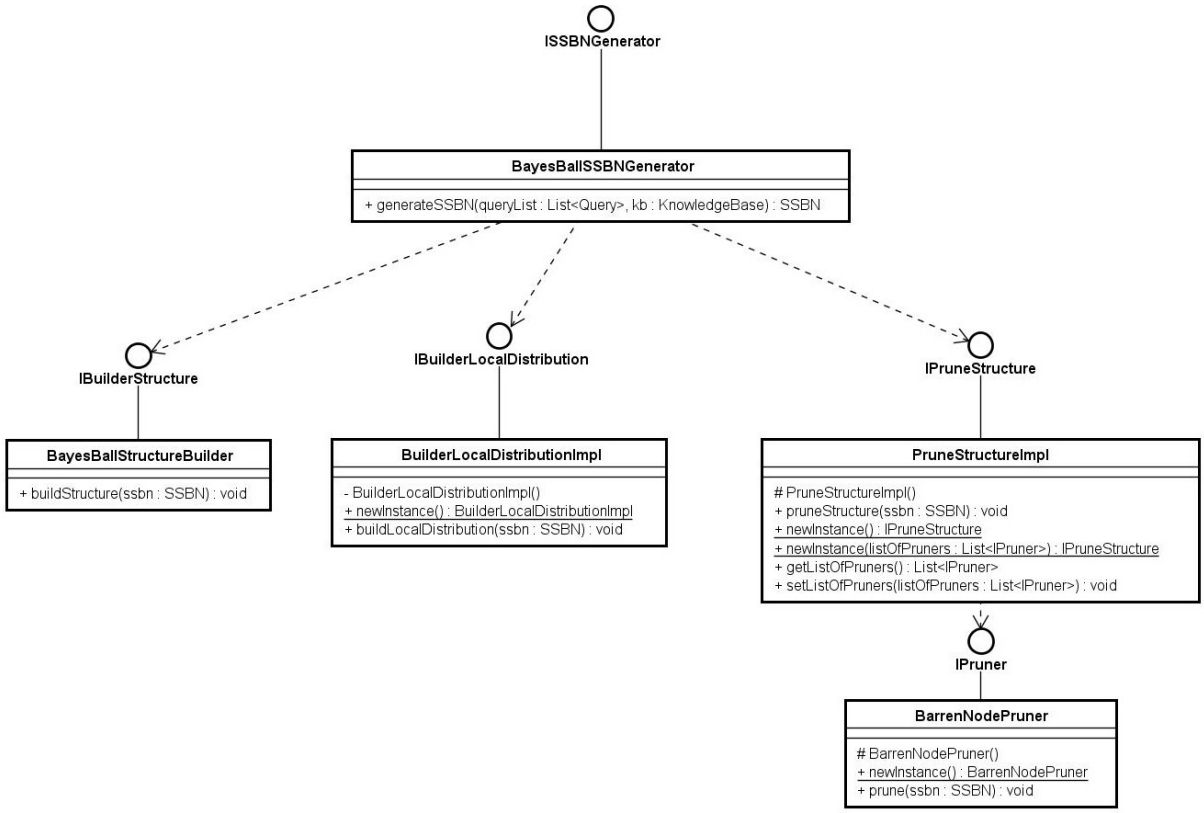

Figura B.4: Arquitetura do algoritmo Bayes-Ball

- **Classe BayesBallSSBNGenerator**: implementação do algoritmo Bayes-Ball para geração de SSBN. Implementa o método `generateSSBN` da interface `ISSBNGenerator`, retornando a SSBN gerada a partir do conjunto de queries e da base de conhecimentos.
- **Classe BayesBallStructureBuilder**: implementa a construção da Grand-BN no algoritmo Bayes-Ball, gerando os nós da rede baseando-se nas regras do algoritmo Bayes-Ball.
- **Classe BuilderLocalDistributionImpl**: implementa geração das distribuições locais de probabilidade a partir dos pseudocódigo de cada nó residente, das valo-

rações para as variáveis ordinárias e dos pais de cada nó. Não houve modificações nesta classe para a implementação do plug-in de PR-OWL 2 RL.

- **Classe PruneStructureImpl**: implementa as podas na rede. Executa de forma encadeada um conjunto de classes que implementam a interface `IPruner`, contendo o método `prune`, que recebe como entrada a SSBN a ser podada, e remove desta os nós com as características desejadas.
- **Classe BarrenNodePruner**: implementa a poda dos nós estéreis (nós que não possuem nós de *query* ou de *finding* como descendentes).

A Figura B.5 mostra a arquitetura da classe responsável por fazer a construção da GrandBN.

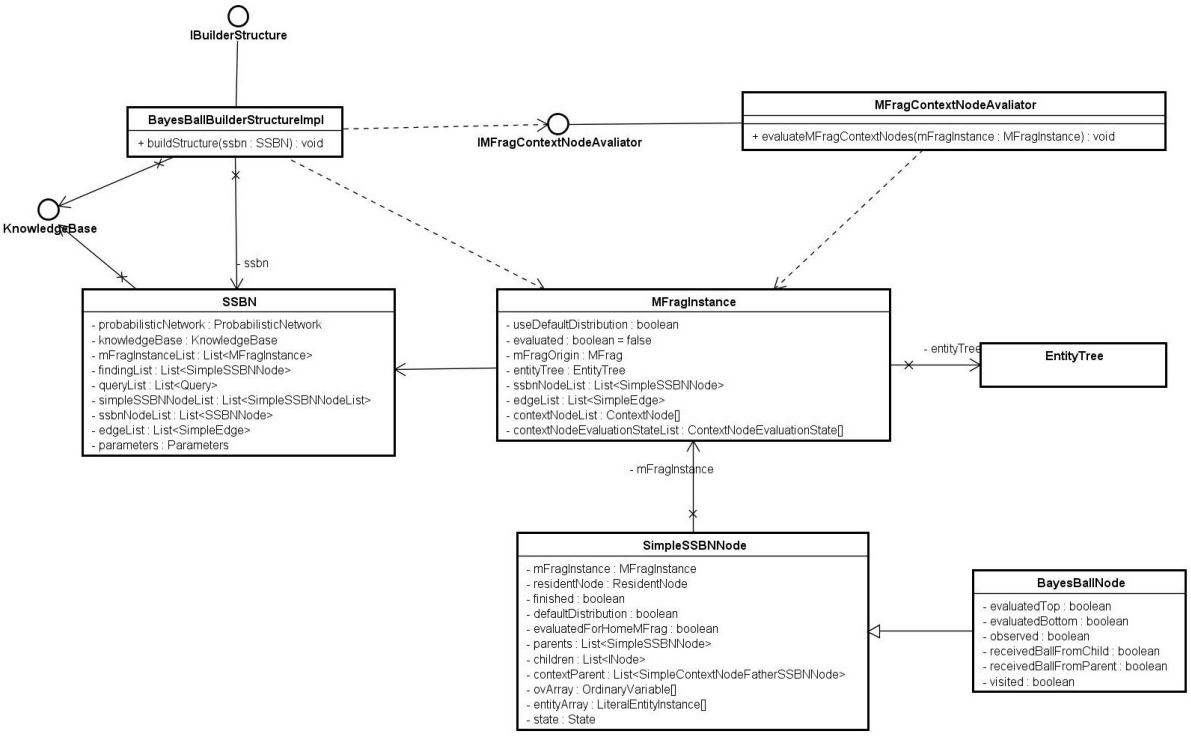

Figura B.5: Arquitetura da construção da Grand BN no algoritmo Bayes-Ball

- **Classe BayesBallBuilderStructureImpl**: construção da Grand-BN no algoritmo Bayes-Ball, gerando os nós da rede baseando-se nas regras do algoritmo Bayes-Ball.
- **Classe SimpleSSBNNode**: nó da SSBN com informações básicas para a execução do algoritmo. Simplifica o SSBNNode ao conter apenas atributos necessários a construção da SSBN: não contém tabelas de probabilidade, por exemplo.

- **Classe BayesBallNode**: estende `SimpleSSBNNode` adicionando informações necessárias para executar o algoritmo Bayes-Ball.
- **Classe SSBN**: contém a SSBN. Durante o passo de inicialização, construção da rede e poda, contém um conjunto de objetos da classe `SimpleSSBNNode`, que são convertidos em objetos `SSBNNode` no passo de construção de CPT, onde um objeto `ProbabilisticNode` é vinculado a cada `SSBNNode`.
- **Classe MFragContextNodeAvaliator**: classe responsável por fazer a avaliação dos nós de contexto da `MFragInstance`.
- **Classe MFragInstance**: MFrag instanciada com valorações para as variáveis ordinárias. As combinações de valorações possíveis estão contidas na `EntityTree` vinculada a MFragInstance.
- **Classe EntityTree**: valorações possíveis para as variáveis ordinárias na `MFragInstance`.

Afim de aumentar a escalabilidade do algoritmo de geração de SSBN, foram necessárias modificações na estratégia de avaliação dos nós de contexto. O algoritmo anteriormente fazia a avaliação de cada nó de contexto da MFrag de forma independente, considerando para a avaliação de cada nó apenas as valorações para variáveis ordinárias que originaram a criação da MFragInstance. O resultado final da avaliação consistia na interseção entre os valores válidos para cada variável ordinária encontrada em cada nó de contexto.

O algoritmo foi alterado para considerar na avaliação de cada nó de contexto apenas os valores já considerados válidos pela avaliação dos nós de contexto anteriores. Além disso, foi criada uma *schedulagem* que permite definir em qual ordem os nós de contexto devem ser avaliados de forma que sejam avaliado primeiro os nós para os quais já estão disponíveis mais informações. Esta alteração minimiza a quantidade de triplas retornadas pelas *queries* submetidas a base de conhecimento.

O plug-in de PR-OWL 2 RL deve ser instalado em uma distribuição do UnBBayes contendo o plug-in core e os plug-ins MEBN e de PR-OWL 2. A Figura B.6 mostra a tela para configuração da base de conhecimento, onde o usuário pode escolher utilizar a *triplestore* e conectar-se a base, oferecendo os dados para conexão. Há uma aba onde o usuário pode escolher entre o algoritmo Bottom-Up e o Bayes-Ball para gerar a SSBN.

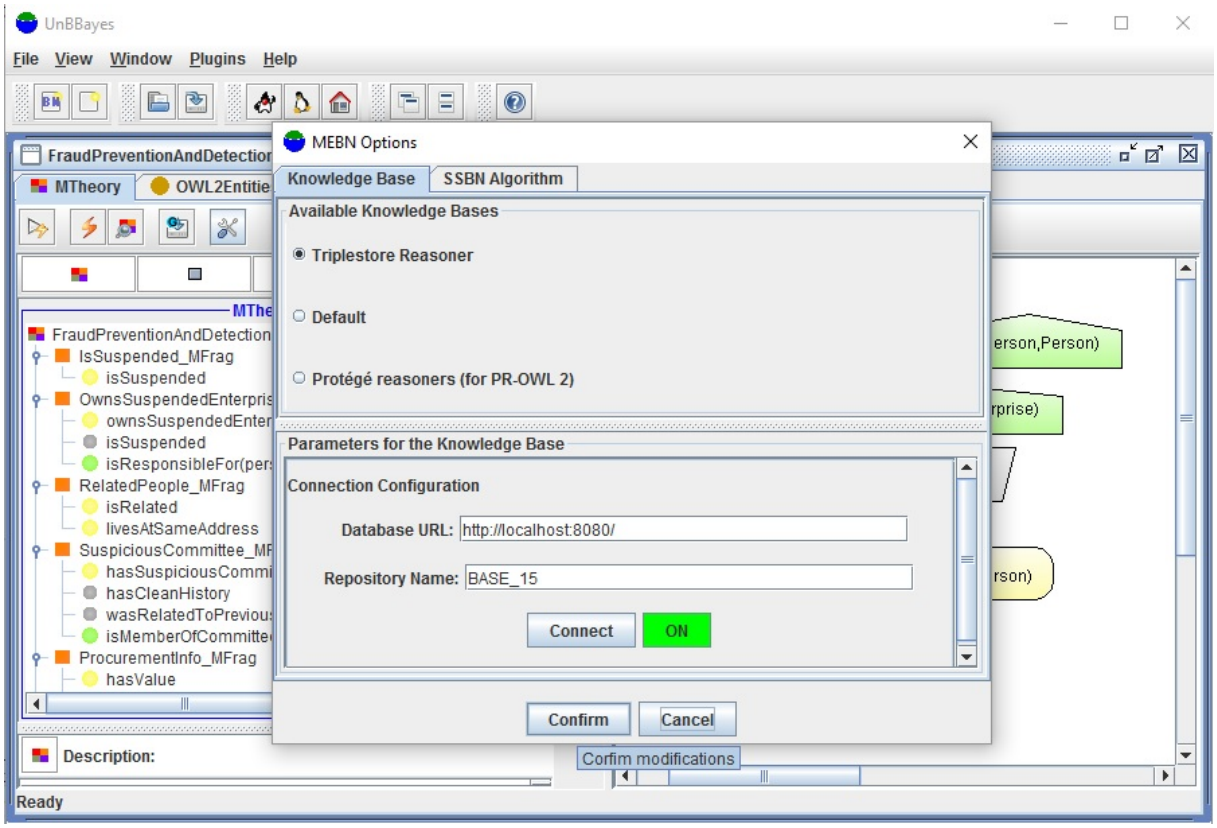

Figura B.6: Painel de configuração da *triplestore*

Apêndice C

Teorema PR1

Este apêndice reproduz o Teorema PR1 apresentado em [62]. O teorema utiliza a sintaxe funcional do OWL 2, definida em [64]. Segue descrição das construções utilizadas no teorema:

- `ClassAssertion( C a )` : assertiva de classe, indicando que o individuo `a` pertence a classe `C`. Ex:

  ```
  ClassAssertion(:Pessoa :Maria);
  ```

- `ObjectPropertyAssertion( OP` a_1 a_2 `)`: assertiva de propriedade de objetos. Ex:

  ```
  ObjectPropertyAssertion(:ehPaiDe :Joao :Pedro);
  ```

- `DataPropertyAssertion( DP a v )`: assertiva de propriedade de dados. Ex:

  ```
  DataPropertyAssertion( :possuiIdade :Pedro "18"^^xsd:integer);
  ```

- `SameIndividual(` a_1 `...` a_n `)`: assertiva indicando que dois ou mais indivíduos se referem ao mesmo individuo. Ex:

  ```
  SameIndividual( a:Pedro a:Pedro_Alvares )
  ```

- `SubAnnotationPropertyOf(`AP_1 AP_2`)`: indica que a anotação de propriedade AP_1 é uma subpropriedade da anotação de propriedade AP_2. Anotações de propriedade são utilizadas para adicionar comentários a propriedades da ontologia.

- `AnnotationPropertyDomain(AP U)`: indica que o domínio da anotação de propriedade `AP` é a IRI `U`.

- `AnnotationPropertyRange(AP U)`: indica que o contra-domínio da anotação de propriedade `AP` é a IRI `U`.

Segue o teorema.

Teorema C.0.1 (PR1)

Seja R as regras OWL 2 RL/RDF definidas pela W3C em [62]. Seja O_1 e O_2 ontologias OWL 2 RL satisfazendo as seguintes propriedades:

- *nem O_1 e nem O_2 contém uma IRI utilizada por mais de um tipo de entidade;*
- *O_1 não pode contém axiomas* `SubAnnotationPropertyOf`*,* `AnnotationPropertyDomain` *e* `AnnotationPropertyRange`*; e*
- *cada axioma em O_2 é uma assertiva de uma das formas especificadas abaixo, onde a_1, a_2,..., a_n são indivíduos:*

  `ClassAssertion`*(C a_1) onde C é uma classe,*

  `ObjectPropertyAssertion`*(OP a_1 a_2) onde OP é uma propriedade de objeto,*

  `DataPropertyAssertion`*(DP a v) onde DP é uma propriedade de dado, ou*

  `SameIndividual`*(a_1 ... a_n)*

Além disso, seja RDF(O_1) e RDF(O_2) traduções de O_1 e O_2, respectivamente, em grafos RDF como especificado em [67]; e seja FO(RDF(O_1)) e FO(RDF(O_2)) tradução destes grafos em teorias de primeira ordem nas quais as triplas são representadas utilizando o predicado `T`*, ou seja,* `T(s, p, o)` *representa uma tripla RDF com sujeito* `s`*, predicado* `p` *e objeto* `o`*. Então O_1 implica O_2 sob a Semântica Direta do OWL 2 se e somente se FO(RDF(O_1)) $\cup$ R implicam FO(RDF(O_2)) sob a semântica padrão da lógica de primeira ordem.*

A prova do teorema pode ser encontrada em [62].