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

Geração Automática de Ontologias Probabilísticas a partir de um modelo UMP-ST

Diego Marques de Azevedo

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Marcelo Ladeira

Brasília
2017

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

Geração Automática de Ontologias Probabilísticas a partir de um modelo UMP-ST

Diego Marques de Azevedo

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Marcelo Ladeira (Orientador)
CIC/UnB

Prof. Dr. Dr.

Prof. Dr. Rodrigo Bonifácio
Coordenador do Bacharelado em Ciência da Computação

Brasília, 14 de Marco de 2017

Dedicatória

Dedico esse estudo e trabalho a Deus, que me conduziu e me deu forças para chegar até aqui, a minha família e Alyne Alves dos Santos, pelo apoio, paciência e incentivo.

Agradecimentos

Agradeço ao professor Marcelo Ladeira pelo empenho e paciência, ao mestre Laécio Lima dos Santos que durante a pesquisa se dispôs a tirar dúvidas e ministrar algumas aulas, ao professor Rommel Novaes Carvalho pelo apoio, aos colegas de curso Lucas Amaral e Daniella Albuquerque dos Angelos pela ajuda durante o realização do trabalho.

Resumo

O URP-ST é uma metodologia baseada no processo unificado que orienta o engenheiro de ontologias durante a construção de ontologias probabilísticas por meio de uma série de etapas que englobam desde a modelagem até a realização de inferências. A etapa de modelagem é definida pelo UMP-ST, uma metodologia iterativa e incremental voltada para a maioria das tecnologias semânticas. Uma delas é o PR-OWL, uma linguagem para a representação do MEBN. A modelagem de ontologias probabilísticas a partir do UMP-ST utilizando MEBN/PR-OWL pode ser realizada no UnBBayes, um framework para a construção gráfica de modelos probabilísticos e a realização de raciocínio plausível. Apesar da orientação dada pelo UMP-ST, a modelagem de ontologias probabilísticas é uma tarefa penosa e repetitiva. Durante a implementação do modelo, é necessário a construção da ontologia a partir do zero utilizando um determinada tecnologia semântica, além da modelagem feita no UMP-ST. Uma integração apropriada que ajude o usuário a implementar a ontologia, tal como um estrutura intermediária, agilizaria e facilitaria a sua implementação. Esse trabalho propõe um plug-in Java para o UnBBayes com o objetivo de automatizar o mapeamento de uma ontologia modelada via UMP-ST em um modelo MEBN, permitindo ao usuário realizar inferências probabilísticas em ontologias com representação de conhecimento com ou sem incerteza probabilística.

Palavras-chave: Geração Automática, Ontologia Probabilística, UMP-ST, URP-ST, MEBN, PR-OWL, UnBBayes

Abstract

The URP-ST is a methodology based on the unified process that guides the ontology engineer in how to design Probabilistic Onologies. The UMP-ST is an incremental and iterative approach that covers the modeling step related to the URP-ST. It is a general methodology for the majority of the existing semantic technologies which support uncertainty. One of them is the PR-OWL, a language for MEBN representation. The modeling of probabilistic ontologies from the UMP-ST using MEBN / PR-OWL can be performed in UnBBayes, a framework for building probabilistic graphical models and performing plausible reasoning. Despite the guidance given by the UMP-ST, the implementation of a PO is a painful and repetitive task. During the implementation of the model, it is necessary to build the ontology from the zero using a specific semantic technology, even if the user models the PO in UMP-ST. A proper integration that helps the user to implement the PO, such as an intermediate structure, would expedite and facilitate its implementation. This work presents an automatic way to generate POs using MEBN representation from the UMP-ST model by mapping the elements of both sides. This is an extension of the UMP-ST to generate POs to an specific formalism and it is developed as a Java plug-in for UnBBayes.

Keywords: Automatic Generation, Probabilistic Ontology, UMP-ST, URP-ST, MEBN, PR-OWL, UnBBayes

Sumário

Lista de Figuras

Capítulo 1

Introdução

A incerteza contida em domínios abrangentes tem sido tratada por ontologias com diferentes formalismos. Parte dessas linguagens utilizam o raciocínio probabilístico como um importante meio para tratar incerteza, dentre elas está o *Probabilistic Web Ontology Language* (PR-OWL) [1]. O PR-OWL tem como base o *Multi-Entity Bayesian Networks* (Redes Bayesianas Multi-Entidades) (MEBN), um formalismo que une o poder de expressão da *First Order Logic* (Lógica de Primeira Ordem) (FOL) e a capacidade de realizar raciocínio plausível baseado em *Bayesian Network* (Rede Bayesiana) (BN).

Apesar do raciocínio probabilístico ser uma das abordagens mais promissoras para tratar a incerteza em ontologias, pouco apoio é dado devido ao nível de dificuldade relativo a execução e construção de ontologias probabilísticas [2]. Carvalho [3] propôs uma metodologia geral baseada no *Unified Process* (Processo Unificado) (UP) cuja finalidade é tornar o processo de construção de *Probabilistic Ontology* (Ontologia Probabilística) (PO) menos complexo e de melhor compreensão. O *Uncertainty Reasoning Process for Semantic Technologies* (URP-ST) é um *framework* que abrange grande parte das tecnologias semânticas e orienta esse processo de construção por meio de uma série de etapas tais como modelagem, povoamento da base de conhecimento e a realização de inferências sobre o modelo.

O *Uncertainty Modeling Process for Semantic Technologies* (Processo de Modelagem de Incertezas para Tecnologias Semânticas) (UMP-ST) é uma abordagem iterativa e incremental voltada para a modelagem de ontologias probabilísticas. Isso é feito por meio de um conjunto de disciplinas que definem o *Probabilistic Ontology Modeling Cycle* (POMC). Carvalho define o POMC como um tipo de ciclo de vida do projeto composto por quatro disciplinas que auxiliam a modelagem de incerteza e a realização de raciocínio plausível. Essas disciplinas compreendem Levantamento de Requisitos, Analise & Design, Implementação e Teste [3].

O UnBBayes[1] é um *framework* de código aberto cuja arquitetura oferece suporte a integração de *plug-ins*. Tais *plug-ins* dão suporte a diversos métodos com diferentes formalismos baseados em redes bayesianas [4], entre eles está a modelagem de processos baseada no UMP-ST. Basicamente, o *plug-in* UMP-ST, engloba as disciplinas de Levantamento de Requisitos e Analise & Design [5]. Entretanto, apesar da limitação do *plug-in* UMP-ST, as demais fases relativas ao POMC e ao URP-ST são tratadas pelo UnBBayes por meio do auxílio de outros *plug-ins*.

Mesmo com o suporte dado pelo UnBBayes à metodologia de construção definida pelo URP-ST, os artefatos produzidos durante a disciplina de modelagem não estão diretamente relacionados com a disciplina de implementação. Isso torna o processo de desenvolvimento repetitivo e penoso, já que não existe uma integração apropriada entre o *plug-in* UMP-ST e o *plug-in* que dá suporte a construção de ontologias baseada no MEBN/PR-OWL, por exemplo.

1.1 Motivação e Justificativa

A geração automática de um modelo intermediário da PO a partir da modelagem feita no UMP-ST agilizaria e facilitaria a implementação de ontologias probabilísticas. Basicamente, isso seria viável a partir do mapeamento entre os elementos do UMP-ST e os elementos da tecnologia semântica selecionada pelo engenheiro de ontologias.

Para este trabalho, foi selecionado o PR-OWL devido a literatura substancial sobre o que é [1, 6, 7], como é implementada [8, 9, 10], onde pode ser aplicada [11, 12, 13, 14, 15, 16], e o suporte oferecido pelo UnBBayes à construção de PO junto a modelagem no UMP-ST [5, 17, 18]. Além disso, sua implementação é de fácil acesso já que é possível acessar o código do UnBBayes.

A representação das regras definidas durante a disciplina de Análise & Design do UMP-ST, por ser apenas textual, não é computacionalmente inteligível. Assim, a representação das relações de influência definidas entre os elementos do modelo não é plenamente compreendida. Logo, para que a geração de ontologias probabilísticas por meio da geração do modelo em PR-OWL seja viável, faz-se necessário a formalização das regras de maneira a facilitar o entendimento por máquina.

1.2 Objetivos

O foco deste trabalho é propor e implementar um mapeamento dos elementos do UMP-ST em elementos MEBN/PR-OWL de forma a gerar um modelo MEBN/PR-OWL, com

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

ou sem a supervisão do usuário, a partir de um modelo de ontologia especificada por UMP-ST.

De forma geral, o trabalho pretende colaborar no processo de construção de ontologias probabilísticas ao permitir uma fácil integração entre as ferramentas que dão suporte a modelagem e implementação baseada no UMP-ST.

Os objetivos específicos são:

- criar um mapeamento entre os elementos modelados no UMP-ST e os elementos do MEBN/PR-OWL;
- propor uma representação da regra presente na disciplina de Analise & Design no UMP-ST para que seja computacionalmente viável;
- implementar um algoritmo que permita a geração da PO a partir do modelo UMP-ST no UnBBayes e avaliar a sua corretude por meio de estudo de caso.
- o modelo UMP-ST, *de persi*, não especifica distribuições de probabilidades, mas apenas relações entre entidades do domínio e a organização delas. No mapeamento para MEBN/PR-OWL esses elementos podem ser utilizados para a criação de variáveis aleatórias;
- O modelo MEBN/PR-OWL a ser gerado, necessariamente, deve atender aos critérios definidos por Laskey [2] para garantir a unicidade da distribuição de probabilidade conjunta representada.

1.3 Contribuições

O trabalho propõe um novo formalismo a partir do modelo UMP-ST. A fundamentação para o mapeamento tem como objetivo tornar os conceitos definidos no UMP-ST mais claros. Isso é feito por meio de definições que, posteriormente, serão utilizadas na formalização proposta para as regras e na formalização proposta para o mapeamento. Para isso, serão criados outros símbolos, além dos definidos para o UMP-ST e para o MEBN.

A partir da formalização das regras e do mapeamento, é proposto o algoritmo para geração de PO em PR-OWL. O algoritmo é implementado por meio de um *plug-in* que irá gerar, a partir do modelo construído no *plug-in* UMP-ST, um modelo para o *plug-in* MEBN capaz de ser lido e editado. Caso o engenheiro de ontologias queira gerar de forma automática o modelo MEBN, é necessário a edição das regras de acordo com a formalização proposta. Isso é feito por meio de novos painéis acoplados aos painéis do *plug-in* UMP-ST.

A persistência do modelo UMP-ST, com ou sem edição das regras no novo formalismo, é feita em um arquivo texto, no formato XML, com extensão ".ump". O arquivo é implementado a partir da definição de *tags* para armazenamento de informações dos elementos definidos no *plug-in*.

1.4 Organização do Documento

O primeiro capítulo faz o levantamento do estado da arte, onde é apresentado a Ontologia Probabilística e o MEBN/PR-OWL como alternativa ao tratamento de incerteza. Os capítulos seguintes propõe a formalização da regra e do mapeamento junto ao algoritmo para geração do modelo em MEBN/PR-OWL; construção do modelo no UMP-ST por meio do formalismo proposto; a implementação do *plug-in* para a geração da ontologia probabilística no MEBN/PR-OWL; ao final, discorre sobre as limitações e possíveis expansões do formalismo e do algoritmo.

Segue a descrição detalhada de cada capítulo:

- O Capítulo 2 apresenta os conceitos de Ontologia e Ontologia Probabilística, utilizados como forma para representação do domínio de conhecimento. Além disso, são apresentados conceitos e algumas definições do MEBN, visto como formalismo para tratamento de incerteza em Ontologias Probabilísticas representadas pelo PR-OWL e PR-OWL 2.

- O Capítulo 3 apresenta a metodologia utilizada no trabalho.

- O Capítulo 4 apresenta como é feito a geração automática da PO em PR-OWL. Para isso, são apresentados os fundamentos para o mapeamento de modelos UMP-ST para MEBN/PR-OWL, onde são definidos os elementos necessários para construção do modelo UMP-ST; formalização das regras no UMP-ST; formalização do mapeamento entre os dois modelos; o algoritmo para geração de PO em PR-OWL e implementação do *plug-in* no UnBBayes;

- O Capítulo 5 apresenta a definição do modelo de fraude em licitações públicas, utilizado neste capítulo como estudo de caso para geração da ontologia probabilística.

- O Capítulo 6 apresenta as considerações finais e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Representação de Ontologias Probabilísticas em PR-OWL

2.1.1 Redes Bayesianas

Bayesian Network (Rede Bayesiana) (BN), em IA, denota um modelo compacto de raciocínio probabilístico para a representação e manipulação de conhecimento com incerteza [19]. A modelagem e realização de inferência probabilística em domínios com incerteza se baseiam em conceitos que englobam teoria de grafos, probabilidade condicional, independência condicional e teorema de Bayes [20]. Em BN, o Teorema de Bayes é utilizado como método para atualização de crença dado a incidência de novas evidências sobre o modelo.

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

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

$P(H_i|E)$ denota a probabilidade *a posteriori* da hipótese H_i dado a sequência de evidências definida por E. $\sum_{k=1}^{n}(P(H_k)P(E|H_k))$ caracteriza a probabilidade total *a priori* de se obter E, onde $P(H_k)$ representa a crença sobre H_k e $P(E|H_k)$ define a probabilidade condicional de E dado como ocorrido H_k. De forma similiar, $P(H_i)$ representa a crença sobre H_i e $P(E|H_i)$ define a probabilidade condicional de E ocorrido H_i.

A representação de informações em redes bayesianas é feita por meio de um *Directed Acyclic Graph* (Grafo Acíclico Dirigido) (DAG), onde cada nó caracteriza variáveis aleatórias, que podem assumir valores discretos ou contínuos. Os arcos denotam depen-

dências probabilísticas entre as variáveis definidas pelos nós. Seja um arco do nó x_j para o nó x_i, a relação definida pela orientação do arco indica que o valor da variável x_i (nó filho) dependa do valor de x_j (nó pai). A dependência probabilística entre o valor dos dois nós é quantificada em uma *Conditional Probability Table* (Tabela de Probabilidade Condicional) (CPT) e a distribuição de probabilidade conjunta pode ser calculada a partir do produtório das probabilidades condicionais de cada nó x_i da rede, tal como visto na fórmula:

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

Logo, é possível determinar a distribuição de probabilidade conjunta por meio da definição das probabilidades condicionais locais.

A Figura 2.1 apresenta o modelo *Family Out* proposto em [21]. Nessa rede o objetivo é verificar se uma dada família está ou não em casa. Sabe-se que quando a família sai, as luzes do quintal provavelmente estão ligadas e o cachorro se encontra fora de casa. Além disso, um dos motivos pelos quais o cachorro pode estar fora de casa é caso ele esteja com problemas estomacais. Caso haja latido do cachorro é provável que ele esteja fora de casa, entretanto pode ser provável também que o latido seja de cachorros próximo a casa. Durante a construção de uma rede bayesiana, estes eventos são convertidos em variáveis aleatórias do tipo booleano e representam os nós da rede. Junto à criação das variáveis aleatórias é definida a distribuição de probabilidade condicional relativa a cada variável, essa distribuição é representada pelas CPT, que podem ser construídas ao se perceber regularidades estatísticas, por exemplo. Nas CPT definidas na Figura 2.1 a representação das variáveis relativas a cada nó da rede é feita por meio das letras iniciais de cada variável.

Não é necessário definir nas CPT todas as combinações possíveis entre as variáveis. O que torna o raciocínio probabilístico sobre BN computacionalmente viável em domínios com grande quantidade de VA. Isso é possível porque existem variáveis aleatórias condicionalmente independentes entre si, isto é, a probabilidade de um dado evento não é afetada pela ocorrência de outro evento condicionalmente independente. Esse tipo de variável pode ser facilmente identificada por meio dos arcos ligados a outros nós. Variáveis conectadas por arcos são condicionalmente dependentes, já as que não estão são condicionalmente independentes. Isso é apresentado de forma mais clara ou se perceber que no modelo *Family Out*, caso o cachorro esteja fora de casa, a probabilidade de escutar o latido é unicamente determinada pela variável "Cachorro Saiu". Observando essa propriedade, é possível determinar a distribuição de probabilidade de um nó apenas a partir da definição da distribuição de probabilidade de seus nós pais.

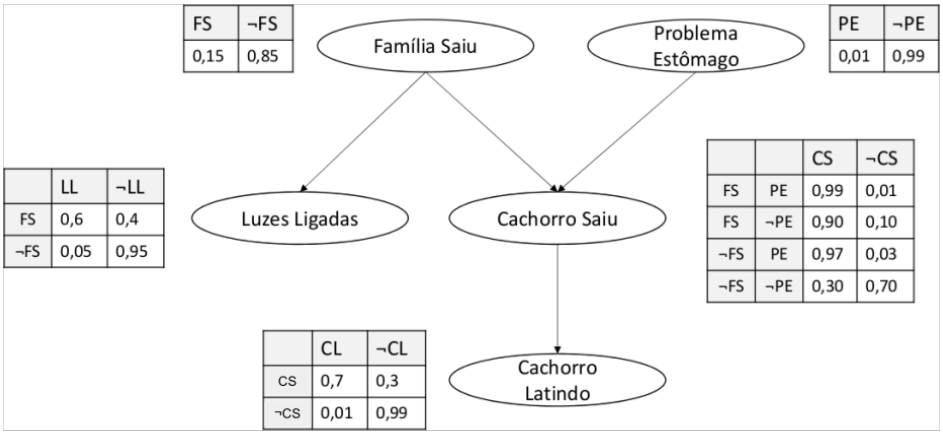

Figura 2.1: Rede Bayesiana *Family Out*

Dentre as aplicações de BN estão a interpretação de linguagens, reconhecimento de imagens, aplicações militares, sistemas de diagnóstico, análise de mercado financeiro e filtro de *spams*.

2.1.2 Redes Bayesianas Multi-Entidades

Multi-Entity Bayesian Networks (Redes Bayesianas Multi-Entidades) (MEBN) estende as redes bayesianas de modo a tratar sua impossibilidade ao representar domínios cuja quantidade de condicionantes *a priori* são desconhecidos [22, 2]. Para isso Laskey [2] incorpora à BN a expressividade da lógica de primeira ordem. O domínio passa a ser representado pelas entidades envolvidas, seus atributos e relacionamentos. A quantidade de condicionantes só é conhecida durante a instanciação do modelo. Entende-se por variável aleatória (VA) os atributos e relacionamentos das entidades participantes do domínio. Basicamente, o MEBN funciona como um *template* voltado para a representação do domínio de aplicação, uma linguagem para a representação de bases de conhecimento probabilística de primeira ordem [2].

No MEBN, o conhecimento sobre o domínio é expresso por meio de uma coleção de fragmentos MEBN (MFrags) que em conjunto formam a *MEBN Theory* (Teoria MEBN) (MTheory). Esse conjunto de MFrags deve satisfazer determinadas condições de consistência que garantem a existência de uma distribuição de probabilidade conjunta única sobre suas variáveis aleatórias [2]. Essas variáveis também podem ser condicionadas a elas mesmas direta ou indiretamente.

Cada MFrag representa uma distribuição de probabilidade condicional sobre uma quantidade potencial infinita de instâncias de variáveis aleatórias residentes limitadas pelo contexto definido na MFrag [23]. Assim como em BN, uma MFrag contém nós or-

ganizados em um grafo dirigido. Cada nó representa uma variável aleatória (VA) e são classificados em nós *residentes*, nós de *entrada* e nós de *contexto*.

A Figura 2.2 apresenta uma MFrag presente no domínio de Fraude em Licitações Públicas [5]. Essa ontologia tem como objetivo detectar fraudes em licitações baseadas em diversas informações, tais como, a possibilidade de uma pessoa que possuiu uma empresa suspensa influenciar na ocorrência de fraude na licitação a qual faz parte. Além disso, poderia existir o caso em que, durante o processo de licitação, as pessoas envolvidas possuam uma relação de proximidade, seja por parentesco ou morarem no mesmo endereço por exemplo, o que potencialmente prejudicaria a integridade do processo.

Os nós de *contexto* (nós 1, 2 e 3): são expressões lógicas de primeira ordem que definem restrições a serem satisfeitas para que as distribuições de probabilidade locais da MFrag sejam válidas. Eles podem assumir os valores *true, false* e *absurd*. Esse ultimo é usado quando uma expressão não possui nenhum sentido ou o valor envolve parâmetros inválidos.

Os nós de *entrada* (nó 4): são variáveis cujos valores já foram definidos em uma MFrag pertencente a MTheory, mas que podem influenciar na distribuição de probabilidade dos nós residentes. Para isso, cada nó de entrada antes deve ter sido um nó residente.

Os nós *residentes* (nó 5): são variáveis que possuem a distribuição local de probabilidade definida pela CPT. Seus valores são definidos a partir do valor do nó de seus pais, que podem ser nós de entrada e, até mesmo, outros nós residentes. Para que haja consistência na MTheory, cada nó *residente* deve ser definido como tal em apenas uma MFrag, denotada como *home* MFrag.

Os nós 1 e 2 definem as variáveis ordinárias, isto é, variáveis que podem ser substituídas por entidades do domínio expresso durante a instanciação do modelo. Essas variáveis são utilizadas como argumentos em nós residentes.

As definições apresentadas a seguir formalizam elementos presentes no MEBN.

- *Símbolos de variáveis (ordinárias):* São utilizados tal como na FOL e podem ser substituídos por entidades não-específicas do domínio. São escritas como *strings* alfanuméricas iniciadas com letra minúscula. Exemplo, `person`;

- *Símbolos de constantes:* são entidades particulares do domínio. São escritos como *strings* alfanuméricas iniciadas com letra maiúscula. Exemplo, `Maria`;

- *Símbolos de identificadores únicos:* A mesma entidade pode ser representada por símbolos de constantes distintos. O MEBN evita ambiguidade atribuindo símbolos de identificadores únicos para cada entidade. Eles são os *possíveis valores* para variáveis aleatórias. Eles podem assumir dois tipos:

- *Símbolos de valores verdade e o símbolo indefinido:* Os símbolos reservados `T`, `F` e $\perp$ são constantes lógicas, onde o símbolo $\perp$ denota indefinido ou contraditório. Os símbolos `T` e `F` denotam valores verdade atribuídos a uma hipótese;
- *Símbolos de identificadores de entidades:* Existe um conjunto finito $\mathcal{E}$ de símbolos de identificadores de entidades. No MEBN, eles são usados como rótulos para entidades. São escritos como numerais ou *strings* alfanuméricos iniciados por ponto de exclamação;

- *Conectivos lógicos e operador de igualdade:* Os símbolos de conectivos lógicos $\neg$, $\wedge$, $\vee$, $\Rightarrow$, e $\Leftrightarrow$, junto com a relação de igualdade $=$, são símbolos utilizados tal como operadores de "negação", "e", "ou", "$\Rightarrow$", "$\Leftrightarrow$"e "="na FOL e definem variáveis aleatórias reservadas com valores determinados pela semântica.

- *Quantificadores:* Os símbolos $\forall$ e $\exists$ são usados no MEBN na construção de variáveis aleatórias que representam sentenças contendo quantificadores na FOL;

- *Símbolos de variáveis aleatórias de domínio-específico:* São escritas como *strings* alfanuméricas iniciadas com letra maiúscula. Cada variável aleatória está associada a um conjunto de *possíveis valores*. Caso esse conjunto esteja contido em $\{\texttt{T}, \texttt{F}, \perp\}$, a variável aleatória é chamada de variável aleatória *lógica*. Caso o conjunto esteja contido em $\mathcal{E} \cup \perp$, então a variável aleatória é chamada de variável aleatória *fenomenal*. Variáveis aleatórias *lógicas* correspondem a predicados e variáveis aleatórias *fenomenais* correspondem a funções na FOL;

- *Símbolos exemplares:* São símbolos utilizados como variáveis vinculadas ao contradomínio de quantificadores. São escritos como *strings* alfanuméricas iniciados com \$.

Dado os símbolos definidos no MEBN:

1. Um termo MEBN de variável aleatória é um símbolo de variável aleatória ψ seguido de uma lista parametrizada de argumentos separados por vírgulas, onde cada argumento pode ser uma variável, símbolo de constante ou termos (recursivos) de variáveis aleatórias;
2. Seja ϱ uma constante e θ uma variável ordinária, ϕ é uma fórmula se:
   (a) Predicado e função são definidos por $\psi(\theta_1, ..., \theta_n)$;
   (b) Termo t é uma variável aleatória ψ, uma constante ϱ ou uma variável ordinária θ;

(c) Termo exemplar σ é utilizado com quantificadores a partir da substituição de θ_i por σ em $\psi(..., \theta_i, ...)$;

(d) Conectivos lógicos possuem dois argumentos definidos pelas fórmulas ϕ_1 e ϕ_2;

(e) Igualdade possui dois argumentos definidos por dois termos t_1 e t_2;

(f) Quantificadores possuem dois argumentos, um termo exemplar σ e uma fórmula ϕ, onde ϕ é definido a partir de $\psi(..., \sigma, ...)$;

(g) Referência indireta $\phi(t_1, ..., t_k)$ é definida por uma fórmula ϕ formada pelos termos $t_1, ..., t_k$ presentes como argumento na variável aleatória ψ.

Laskey define uma MFrag como:

Definição 1 [Laskey [2]] Uma MFrag $\mathcal{F} = (\mathcal{C}, \mathcal{I}, \mathcal{R}, \mathcal{G}, \mathcal{D})$ consiste em um conjunto finito $\mathcal{C}$ de VA de *contexto*; um conjunto finito $\mathcal{I}$ de VA de *entrada*; um conjunto finito $\mathcal{R}$ de VA *residentes*; um fragmento de grafo $\mathcal{G}$; e um conjunto $\mathcal{D}$ de distribuição local, um para cada membro de $\mathcal{R}$. Os conjuntos $\mathcal{C}, \mathcal{I}$ e $\mathcal{R}$ são disjuntos. O fragmento de grafo $\mathcal{G}$ é um grafo acíclico dirigido cujos nós possuem correspondência de um-para-um com as VA em $\mathcal{I} \cup \mathcal{R}$, onde as VA em $\mathcal{I}$ correspondem aos nós raízes em $\mathcal{G}$. As distribuições locais especificam as distribuições locais de probabilidade para as VA residentes.

A partir da definição da MFrag $\mathcal{F}$, é definido um conjunto de ligação $\mathcal{B}$ que associa símbolos de identificadores únicos com variáveis ordinárias presentes em $\mathcal{F}$ por meio da variável aleatória ψ.

Definição 2 [Laskey [2]] Seja $\mathcal{F}$ uma MFrag contendo variáveis ordinárias $\theta_1, ..., \theta_k$, e seja $\psi(\theta)$ uma variável aleatória residente em $\mathcal{F}$ que depende de algumas ou todas as variáveis θ_i.

2a. Um conjunto de ligação $\mathcal{B} = \{(\theta_1 : \varepsilon_1), (\theta_2 : \varepsilon_2), ..., (\theta_k : \varepsilon_k)\}$ para $\mathcal{F}$ é um conjunto de pares ordenados que associa um símbolo de identificador único ε_i com cada variável ordinária θ_i em $\mathcal{F}$. O símbolo de constante ε_i é chamado de *binding* para variável θ_i determinada por $\mathcal{B}$. Não é necessário que ε_i seja distinto.

2b. Seja $\mathcal{B} = \{(\theta_1 : \varepsilon_1), (\theta_2 : \varepsilon_2), ..., (\theta_k : \varepsilon_k)\}$ um conjunto de ligação para $\mathcal{F}$, $\psi(\varepsilon)$ é uma instância de ψ obtida pela substituição de ε_i por cada ocorrência de θ_i em $\psi(\theta)$.

Dado que existam MFrags definidas de acordo com a Definição 1, são estabelecidas algumas condições para a formação de MTheories.

Definição 3 [Laskey [2]] Seja $\mathcal{T}' = \{\mathcal{F}_1, \mathcal{F}_2, ...\}$ um conjunto de MFrags. Seja $V_{\mathcal{T}'}$ o conjunto de termos de VA contidos em $\mathcal{F}_i$, e seja $N_{\mathcal{T}'}$ o conjunto de instâncias de VA formadas a partir de $V_{\mathcal{T}'}$ por meio da substituição dos argumentos das variáveis aleatórias em $V_{\mathcal{T}'}$ pelos identificadores únicos de $\{T, F, \bot\} \cup \xi$. $\mathcal{T}'$ é uma MTheory simples se obedece às seguintes condições:

1. Sem ciclos. Nenhuma VA pode ser antecessora de si mesma;
2. Home MFrag única. Para cada $\phi(\alpha) \in N_{\mathcal{T}'}$, existe exatamente uma MFrag $F_{\phi(\alpha)} \in \mathcal{T}'$, chamada de home MFrag de $\phi(\alpha)$, tal que $\phi(\alpha)$ é uma instância de um VA residente $\phi(\theta)$ de $F_{\phi(\alpha)}$.

A partir da definição de $\mathcal{T}'$, o MEBN expande a representação do seu domínio ao tratamento de situações que envolvem recursividade, geralmente presentes em domínios com padrões temporais ou domínios onde existe uma dependência entre diferentes instâncias de variáveis aleatórias do mesmo nó residente [2, 24, 22]. Além disso, observa-se que a constituição das relações de dependência entre as variáveis aleatórias não permite a ocorrência de dependências circulares de modo a evitar a construção de grafos cíclicos.

O item 2 da Definição 3 pode ser relevado dado que existam subtipos de entidade aplicados a diferentes MFrags. Sobre essa configuração, as instâncias de variáveis aleatórias podem ser independentes para subtipos distintos. Na versão polimórfica do MEBN, seria conveniente que as distribuições locais para cada subtipo de entidade relativa a VA residente fossem definidas em MFrags diferentes [2].

As MFrags dão ao MEBN meios flexíveis para a representação de conhecimento sobre assuntos específicos, entretanto o maior ganho em expressividade desse modelo é revelado ao agregar os padrões de conhecimento definidos em cada MFrag. A junção desses padrões em uma teoria permite a aplicação de raciocínio sobre situações específicas e consequentemente, o refinamento do modelo a partir de aprendizado [24]. A obediência às condições de formação de uma MFrag observando a consistência do modelo permite a construção de uma MTheory tal como na Figura 2.6 e2.7.

A MTheory gerada a partir do conjunto de MFrags ilustra regularidades estatísticas que caracterizam um domínio. O raciocínio sobre cenários específicos por meio do MEBN ocorre por meio da instanciação do modelo ao representar informações específicas intrínsecas ao cenário observado. Para que os questionamentos sobre a MTheory (*queries*) sejam computados e assim obtenham uma resposta, é construída uma *Situation-Specific*

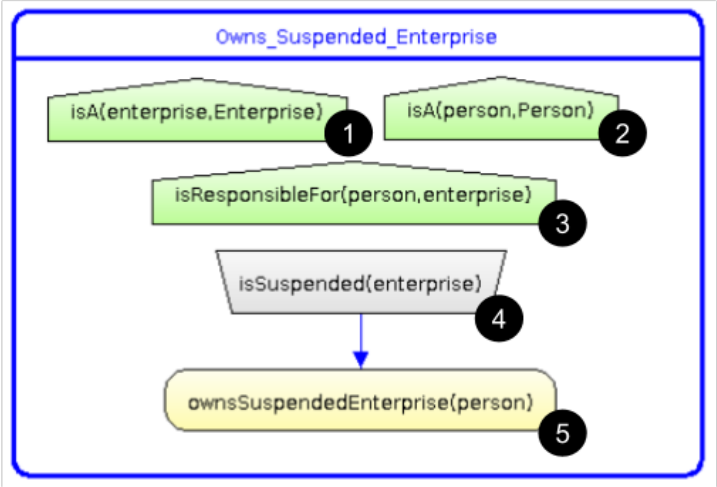

Figura 2.2: MFrag do Domínio de Fraudes em Licitações Públicas

Bayesian Network (Rede Bayesiana de Situação Específica) (SSBN), rede bayesiana mínima formada a partir das instâncias de variáveis aleatórias. Cada *query* consiste em obter um conjunto de valores *a posteriori* para as variáveis aleatórias ligadas ao domínio característico a partir do conjunto de evidências e variáveis de contexto [25].

A expansão desse modelo ao englobar novas evidências é feita por meio da inclusão de *findings*, MFrags compostas apenas por um nó de entrada e um nó residente. Após inclusão de novos nós residentes, a distribuição de cada variável ligada a situação específica é atualizada [24, 22].

A lógica MEBN é capaz de representar qualquer sentença da lógica de primeira ordem ao englobar o conjunto de *built-in* MFrags [24, 2]. Esse conjunto de MFrags embute em sua implementação quantificadores, referência indireta e MFrags booleanos (compostos por conectivos como "AND"e "OR"). Além disso, no MEBN é possível declarar entidades com tipos distintos, o que tornaria necessário a definição de nós residentes referenciando todos os tipos possíveis no domínio. Isso poderia ser feito por meio da criação de nós residentes *isA* junto a definição da distribuição da entidade no domínio [22].

2.1.3 Ontologia e Ontologia Probabilística

Uma ontologia é uma especificação formal e explicita de uma conceituação compartilhada. Conceituação refere-se a um modelo abstrato de algum fenômeno no mundo, identificado por conceitos relevantes desse fenômeno. Explicito refere-se ao tipo de conceito utilizado, e as restrições sobre o seu uso são explicitamente definidos. Formal refere-se ao fato de que a ontologia deve ser compreendida por um computador. Compartilhável reflete a noção de que uma ontologia captura o conhecimento consensual, ou seja, não é restrito a um indivíduo, mas aceito por um grupo [26].

Uma ontologia pode ter várias formas, porém é necessário a inclusão de um vocabulário de termos e especificações de seus significados. Isso inclui definições e uma indicação de como os conceitos estão inter-relacionados e coletivamente impõe uma estrutura no domínio e limita as possíveis interpretações dos termos [27].

Uma dos tipos descritos em [27] define que uma ontologia é capaz de representar os principais conceitos, seus atributos e relacionamentos, por meio de um vocabulário que pode ser compartilhado por diferentes agentes. A ideia principal é que esses agentes acessem diferentes tipos de informação sobre o domínio do discurso e seja capaz de fazer afirmações sobre esse conhecimento, além de fazer perguntas a outros agentes sobre informações de interesse.

Na Web Semântica uma ontologia é definida como um artefato de engenharia voltado para a especificação formal de significados e associações entre *tags*, provendo um vocabulário de termos, cujas novas terminologias podem ser formuladas por meio das já existentes. Além disso, a especificação formal das entidades permite identificar relacionamentos em termos que pertencem a outras ontologias [4, 3]. Ela possui basicamente dois componentes distintos:

- "classes": nomes dos conceitos importantes ao domínio;
  - Ex: **Humano**: conceito sobre membros de uma categoria de animais. **Adulto**: conceito sobre seres vivos com idade superior a determinado valor relativa a uma categoria do ser vivo. **Huma_Adulto**: conceito sobre **Humanos** com idade superior a determinada idade.
- "propriedades": conhecimentos e restrições sobre o domínio;
  - Ex: Todos os **Humanos** são **Homens** ou **Mulheres**.

A *Semantic Web* (Web Semântica) (SW) [28] tem como objetivo estender a Web tradicional explicitando informações sobre a semântica dos conteúdos de tal forma que as máquinas possam compreender informações antes compreendidas apenas por humanos. Para isso, o conhecimento seria representado, principalmente, por meio de ontologias.

A *World Wide Web Consortium* (W3C), organização criada para a padronização de protocolos e para incentivo à interoperabilidade entre sites na Web [29], definiu vários padrões para SW. Dentre os padrões está o *Web Ontology Language* (OWL), linguagem padrão para a representação do conhecimento na Web Semântica. O OWL permite a modelagem e inferência em ontologias complexas ao se basear na expressividade da lógica descritiva.

A representação de informação por meio de linguagens baseadas no uso de ontologias além de permitir a representação de domínios determinísticos, deve ser capaz de representar domínios que envolvam incerteza. Nestes domínios, podem existir, além de domínios estocásticos por natureza, informações ambíguas, inconsistentes, incompletas, vagas, ou incertas por ignorância do modelador. Diversos formalismos foram propostos para tratar a incerteza intrínseca a essa forma de representação, o que permitiu a criação de novas linguagens tais como o PR-OWL [1, 3], OntoBayes [30], BayesOWL [31], P-CLASSIC [32], e extensões probabilísticas de lógicas descritivas de SHIF(D) e SHOIN(D) [33].

A representação de incerteza por meio da abordagem probabilística considera aspectos relacionados à frequência, crença, ocorrência ou relações de causa e efeito sobre um evento. Neste caso, ao considerar a teoria da probabilidade, a representação da incerteza é caracterizada por valores entre 0 e 1; a validação e a inferência sobre o modelo pode ser feita com o auxílio de modelos bayesianos.

O uso da abordagem probabilística é promissor ao considerar suas vantagens. Dentre elas destacam-se a aplicação de operações sobre um domínio fechado, isto é, entre 0 e 1; restrições suficientes para manter um resultado computável; a capacidade de atualização das ontologias por meio da observação de fatos; a instância pertencer a apenas um conjunto, ou seja, uma instância não será "meio"falsa e "meio"verdadeira caso apresente o valor de "0.5", ela fará parte da verdade na metade das ocasiões observadas [22, 3].

Entretanto, mesmo com suas vantagens, uma linguagem para representação de ontologias não se torna factível caso não haja suporte de um modelo computacional de edição e inferência. Neste caso, o uso do UnBBayes, por meio do plug-in UnBBayes-MEBN, se torna um diferencial já que o uso do *plug-in* por meio *framework* permite a inferência e edição de ontologias em PR-OWL [4].

2.1.4 PR-OWL e PR-OWL 2

A abordagem probabilística relacionada a ontologia é definida por Costa [1] como uma *Probabilistic Ontology* (Ontologia Probabilística) (PO). Ontologia probabilística é uma representação explicita e formal de conhecimento sobre um domínio de aplicação. Isso inclui elementos que caracterizam as entidades presentes no domínio tais como, seu tipo, propriedades e relacionamentos, processos e eventos, conhecimento inconclusivo, ambíguo, incompleto, pouco confiável, ou dissonante, além da incerteza sobre todas as formas de conhecimento citadas anteriormente [1, 6].

O *Probabilistic Web Ontology Language* (PR-OWL) consiste de um conjunto de classes, subclasses e propriedades que coletivamente formam um *framework* para a construção de ontologias probabilísticas [6]. Ao representar a incerteza, o PR-OWL realiza a aplicação lógica do MEBN e necessita de máquina de inferência MEBN para processar sua

sintaxe. Com isso, o PR-OWL não apenas provê uma representação consistente e que pode ser reusada por diferentes sistemas probabilísticos, mas também permite aplicações para realizar um raciocínio plausível sobre o domínio de forma eficiente [1].

A Figura 2.3 apresenta os principais elementos que compõe a estrutura do PR-OWL ao modelar uma MTheory. Neste caso, a parte probabilística da ontologia será modelada utilizando a classe `MTheory` composta por um conjunto de indivíduos da classe `MFrags` conectadas a MTheory por meio da propriedade de objeto `hasMFrag`. O conjunto de MFrags deve coletivamente formar uma MTheory consistente. Cada MFrag é composta por indivíduos da classe `Node` (nós residentes, de entrada ou de contexto). Os indivíduos dessa classe são variáveis aleatórias que possuem um conjunto mutuamente exclusivo e coletivamente exaustivo de estados possíveis. Cada nó referencia seus estados possíveis (indivíduos da classe `Entity`) por meio da propriedade de objeto `hasPossibleValues`. A distribuição probabilística condicional ou incondicional (representada por indivíduos da classe *ProbabilityDistribution*) é vinculada a cada indivíduo da classe `Node` por meio da propriedade de objeto `hasProbDist`.

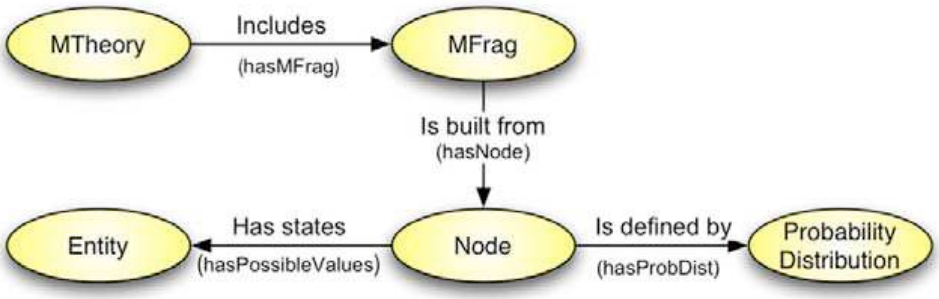

Figura 2.3: Modelo Simples do PR-OWL

Basicamente, o PR-OWL adiciona à sintaxe do OWL elementos necessários à representação de incerteza por meio de estruturas complexas do MEBN, tornando-se um formato para aplicação lógica MEBN em artefatos da Web Semântica [34].

Apesar de estender o OWL ao compreender a lógica MEBN, o PR-OWL carece de uma maior compatibilidade entre a integração do conhecimento em OWL para o conhecimento em PR-OWL/MEBN. Tendo em vista essa problema, foi proposto o *Probabilistic Web Ontology Language 2* (PR-OWL 2) [35, 3]. O PR-OWL 2 tem como objetivo resolver as limitações que envolvem a falta de mapeamento formal entre as variáveis aleatórias do PR-OWL e os conceitos definidos em OWL, e a falta de compatibilidade entre os tipos do PR-OWL e os existentes no OWL [36].

A falta de mapeamento formal entre as duas estruturas foi tratado ao se utilizar o relacionamento `definesUncertaintyOf`, o qual liga uma variável aleatória do PR-OWL a uma propriedade do OWL. Além disso, para que haja a referência aos conceitos do

OWL, o domínio e o contra-domínio das variáveis aleatórias são definidos por meio dos relacionamentos `isSubjectIn` e `isObjectIn`. Dessa forma, o PR-OWL 2 permite a construção de ontologias híbridas ao dar suporte a declarações probabilísticas e determinísticas inter-relacionadas.

A falta de mapeamento entre os tipos referentes às duas linguagens, foi tratada ao substituir a classe `ObjectEntity` (PR-OWL) pela classe `Thing` (OWL). Para que seja possível enumerar *data types* ou objetos por meio de `ObjectOneOf` (OWL), a classe `CategoricalRVStates` (PR-OWL) é substituída pela classe `DataOneOf` (OWL). Além disso, estados booleanos ligados às variáveis aleatórias por meio da classe `BooleanRVStates` podem ser substituídos pelo tipo de dados booleano e `MetaEntity` ligados aos conceitos do OWL [3].

2.2 Modelagem de PO em Tecnologias Semânticas

Com o objetivo de suprir a falta de orientação durante a construção de ontologias probabilísticas, Carvalho [3, 37] propôs uma metodologia geral que facilita o processo de construção baseada em uma abordagem semelhante ao *Unified Process* (Processo Unificado) (UP).

O *Unified Process* (Processo Unificado) (UP) [38, 39, 40] é um processo amplamente aplicado em engenharia de software e, basicamente, possui três características: é interativo e incremental; centrado na arquitetura; focado em riscos. A construção do projeto é dividida em ciclos iterativos, no qual o produto entregue a cada iteração pode ser um conhecimento adicional sobre os requisitos do projeto e arquitetura ou código executável. A arquitetura do sistema é o resultado da incorporação incremental de produtos de cada ciclo. Como parte fundamental do sistema, a arquitetura é um modelo que define a estrutura da informação, suas possíveis operações e sua organização [41]. O UP trata os riscos a partir da sua priorização. Riscos maiores são priorizados em detrimento dos menores durante a inicio da implementação. Assim, caso um aspecto crítico do sistema falhe, é melhor que se tenha conhecimento de qual seja o quão mais cedo possível. Isso irá permitir que o projeto seja revisto ou cancelado antes que uma grande quantidade de recursos seja perdida caso o projeto se torne inviável.

O UP define o ciclo de vida do projeto em quatro fases: Concepção; Elaboração; Construção; Transição. Geralmente a primeira fase é a mais curta. Durante a Concepção, o objetivo principal é definir o propósito do projeto, seu escopo, os riscos envoltos e quais são seus principais requisitos. Durante a Elaboração, a maioria dos requisitos devem estar definidos, os riscos presentes no projeto também e a arquitetura deve ser definida e validada. A fase de Construção é a mais longa e engloba a maior parte do desenvolvimento

do projeto. Essa fase geralmente é composta por iterações onde ao final de cada é entregue parte do código executável. A fase de Transição abrange a implantação do sistema, neste os usuários são treinados e o *feedback* inicial é dado para que o sistema seja aperfeiçoado.

Além das fases, o UP define disciplinas. Cada disciplina presente no ciclo de vida do projeto descreve uma sequência de atividades produzidas por atores que resultam em artefatos. As disciplinas presentes no UP são definidas em Requisitos, Analise & Design, Implementação, Teste e Implantação. Em Requisitos, são definidos os objetivos do sistema, basicamente, o que ele deve fazer baseado nas informações recolhidas do cliente. Em Analise & Design, é definido como será implementado o sistema. Em Implementação, será desenvolvido o código necessário para a produção do sistema, tendo em vista os requisitos levantados durante a primeira disciplina. Em Teste, o código será verificado e validado. Durante a Implantação, o sistema será entregue ao usuário final.

2.2.1 URP-ST

De forma geral, o processo de construção de ontologias probabilísticas ocorre em três estágios: modelagem do domínio, povoamento do modelo ao adicionar informações sobre situações específicas e fazer inferências sobre o modelo ao adicionar informações sobre a base de conhecimento [37]. Nesse sentido, o *Uncertainty Reasoning Process for Semantic Technologies* (URP-ST) [3, 37, 5] é uma abordagem que visa facilitar o processo de construção de PO ao seguir esses três estágios e permitir o raciocínio plausível em aplicações que usam tecnologias semânticas. Na Figura 2.4 é apresentado esses três estágios.

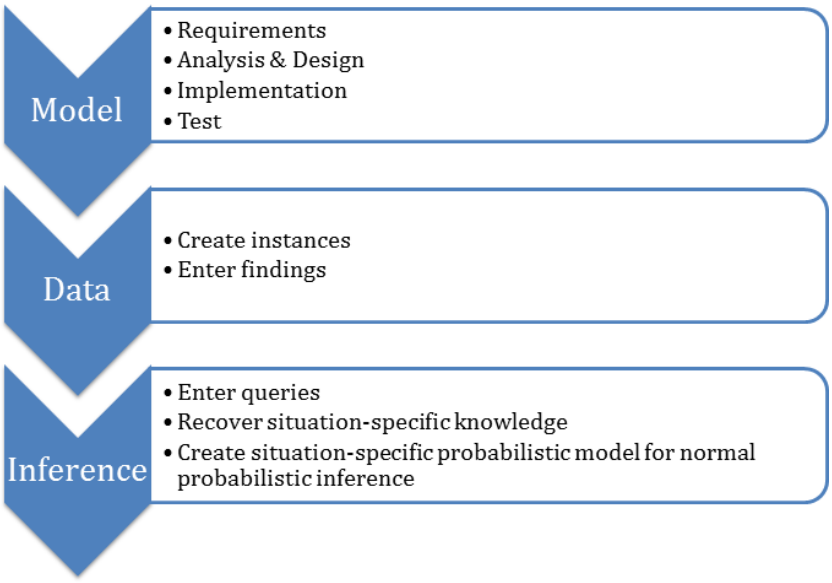

Figura 2.4: *Uncertainty Reasoning Process for Semantic Technologies*

Dentro do URP-ST, o primeiro estágio visa construir uma representação do domínio de modo a entender, explicar, prever ou simular seus aspectos intrínsecos. Para isso, é necessário que o modelo compreenda as entidades que caracterizam o domínio, seus atributos, os relacionamentos que elas podem ter umas com as outras, os processos em que todos eles podem participar e as regras que denotam o comportamento de cada um. Além disso, o modelo deve englobar as incertezas presentes no domínio de aplicação [37, 3].

Já o seu segundo estágio tem como objetivo a realização de inferências sobre o modelo (terceiro estágio). Para isso, seu domínio de aplicação deve englobar situações específicas definidas por meio do povoamento da base de conhecimento [37]. Isso é feito por meio da criação de instâncias que serão utilizadas sobre as variáveis aleatórias e a adição de *findings* sobre a MTheory caso a ontologia seja construída utilizando o PR-OWL/MEBN. O modelo deve descrever como diferentes conceitos pertencentes a ontologia interagem com a presença da incerteza. Isso é feito por meio do conhecimento de fatos que irão guiar o processo de construção.

No último estágio, o modelo deve oferecer uma resposta ao permitir a realização de perguntas (*queries*) e o seu tratamento por máquinas de inferências [37]. Com isso, é avaliado a corretude do modelo em diversos cenários. Entretanto, neste caso, diferentemente de tecnologias semânticas que permitem apenas inferências determinísticas, inferências baseadas em raciocínio probabilístico podem retornar resultados parciais, isto é, não são plenamente garantidos.

2.2.2 UMP-ST

Carvalho [3, 37, 5] divide a modelagem de PO em quatro disciplinas. Essas disciplinas compõe um ciclo de modelagem de ontologias probabilísticas baseado no UP e formam a metodologia geral que define o *Uncertainty Modeling Process for Semantic Technologies* (Processo de Modelagem de Incertezas para Tecnologias Semânticas) (UMP-ST). Assim como no UP, o UMP-ST possui uma abordagem iterativa e incremental, porém, apesar de conter todas as fases relativas ao UP (com modificações que permitem a modelagem de ontologias), ele foca apenas em quatro disciplinas: Requisitos, Analise & Design do modelo, Implementação e Testes [37].

A ideia do UMP-ST é aprimorar a ontologia de forma iterativa a partir de uma modelagem incremental do domínio. Assim, o ciclo de aprendizado definido por uma iteração permitiria o aperfeiçoamento do modelo durante a próxima iteração. Por essa razão, cada fase do UMP-ST é composta por quatro disciplinas que caracterizam seu ciclo de modelagem.

A Figura 2.5 apresenta o *Probabilistic Ontology Modeling Cycle* (POMC). Basicamente, o ciclo definido pelo POMC descreve a ordem natural com que os artefatos de cada

disciplina são produzidos ao seguir a metodologia definida pelo UMP-ST. Entretanto, assim como no UMP-ST, a interação entre cada disciplina presente no POMC não é rígida tal como no modelo em cascata (o qual obedece em sua forma mais simples uma ordem puramente sequencial) [42]. É possível durante a modelagem da ontologia, que falhas identificadas em Testes sejam tratadas diretamente por meio da redefinição das regras durante a disciplina de Analise & Design [37, 3].

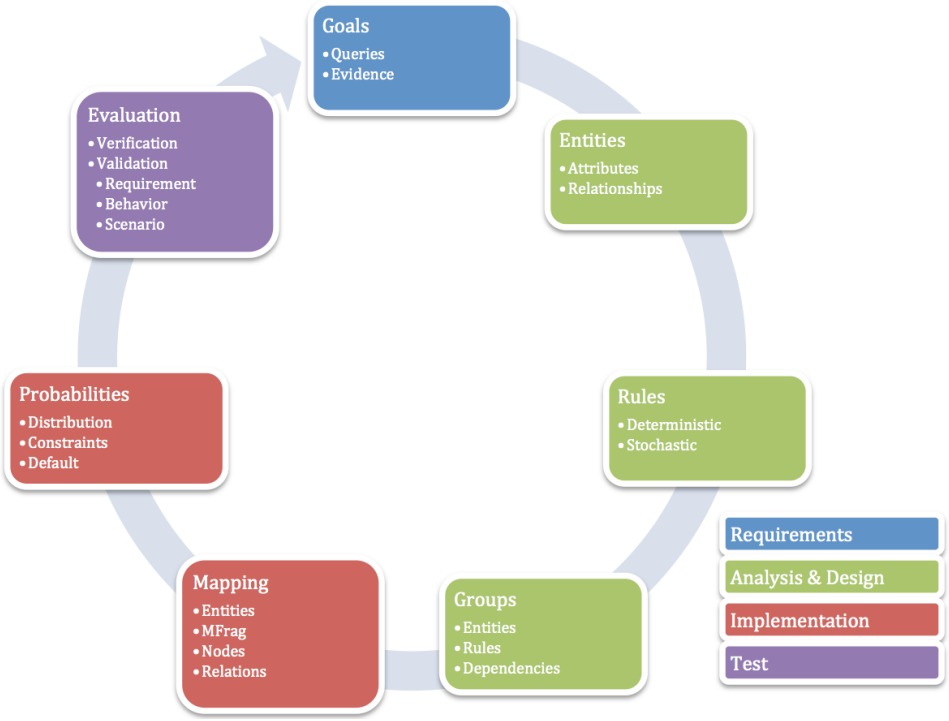

Figura 2.5: *Probabilistic Ontology Modeling Cycle*

A disciplina de Requisitos, nomeada como *Goals* (em azul), é composta pela definição dos objetivos gerais, os quais o modelo deve responder ao se fazer a inferência sobre sua estrutura semântica [37]. Ao final da criação do modelo de fraude em licitações públicas vista como estudo de caso, por exemplo, o modelo deve ser capaz de responder, com determinado grau de certeza, se em determinada licitação existe algum indicativo de fraude. Para isso, cada objetivo é definido por um conjunto de situações específicas que, primeiramente, devem ser tratadas e um conjunto de evidências que fazem parte do domínio de aplicação e darão suporte a construção do modelo. Assim, durante Requisitos são definidos os objetivos, as situações específicas (denominadas *queries*) e as evidências que orientarão o engenheiro de ontologias durante a modelagem.

A disciplina de Análise & Design (em verde) é responsável pela definição semântica do modelo. Nessa disciplina serão descritos as entidades presentes no domínio de aplicação, seus atributos, o relacionamento entre eles, e o conjunto de regras que os envolvem e

caracterizam o domínio [3, 5]. Tais definições são aspectos gerais intrínsecos ao domínio e portanto independem da tecnologia semântica a ser utilizada.

Durante a Implementação (em vermelho) é necessário a escolha de uma linguagem especifica que represente a ontologia probabilística [37]. Essa linguagem, além de oferecer suporte a incerteza, deve ser semanticamente rica ao ser capaz de representar os elementos definidos da disciplina anterior. Assim, na Implementação, basicamente é feito um mapeamento entre os elementos do domínio e sua representação na linguagem escolhida. Para o estudo de caso analisado nesse trabalho, o mapeamento é feito para o PR-OWL, porém sua representação não está restrita a apenas essa linguagem [43].

Por último, de modo a validar o modelo, são feitos os testes. A disciplina de Testes (em roxo) avalia se o modelo desenvolvido na disciplina de Implementação é capaz de atender os objetivos definidos inicialmente. Isso é feito por meio da análise do conjunto de resultados obtidos a partir da inferência definida pelas regras (em Analise & Design). Dessa forma, o engenheiro de ontologia irá verificar se o modelo atende o conjunto de regras e se o conjunto de regras atendem os objetivos.

O UMP-ST foca na apresentação de técnicas que identificam e especificam regras de cunho estocástico [37, 3], apesar de também oferecer suporte a especificação de regras de cunho determinístico. Regras desse tipo são definidas pelo UMP-ST a partir da especificação da relação de dependência entre suas propriedades. A intensidade com que é dada cada relação é definida por meio de parâmetros que compõe a distribuição de probabilidade local.

Alterações no modelo ao identificar falhas ou devido a possíveis expansões do domínio de aplicação são facilitadas pela rastreabilidade de seus elementos. A rastreabilidade promove a identificação de um conjunto de elementos presentes no modelo que devem ser analisados ou modificados, garantindo a sua total cobertura. O UMP-ST permite a rastreabilidade dos artefatos de cada disciplina ao construir uma *Requirements Traceability Matrix* (Matriz de Rastreabilidade de Requisitos) (RTM). Dessa forma, ao modificar uma regra definida em Analise & Design é possível inferir quais são as entidades, relacionamentos, grupos e objetivos que são influenciados por sua definição.

2.2.3 Ferramenta para construção de PO utilizando PR-OWL

A primeira implementação mundial de MEBN e PR-OWL foi desenvolvida no *framework* UnBBayes a partir da cooperação entre o Grupo de Inteligência Artificial da Universidade de Brasília (GIA-UnB) e a Universidade George Mason [4]. O UnBBayes oferece suporte a criação de modelos MEBN graficamente e permite salvá-los utilizando PR-OWL, além de realizar inferências sobre a ontologia probabilística montando SSBN [10]. Entretanto, o UnBBayes não está limitado a esse único propósito. O seu foco principal está em oferecer

um *framework*, utilizando a arquitetura de *plug-ins*, para edição de modelos probabilísticos e inferência [22].

Além da representação em PR-OWL, o UnBBayes possui suporte ao PR-OWL 2, permitindo que ontologias probabilísticas sejam criadas a partir de ontologias determinísticas provindas do OWL [22, 3]. A edição de ontologias determinísticas no UnBBayes é feita por meio da sua integração com o Protegé [44] e adicionada ao domínio de aplicação por meio da sua representação em PR-OWL 2.

Ontologias probabilísticas podem ser construídas com o auxílio do *plug-in* UMP-ST. A implementação do *plug-in* tem como objetivo atacar a complexidade na criação de PO, a dificuldade na manutenção e evolução da PO existente, e a falta de uma ferramenta centralizada para a documentação de ontologias [5]. A modificação do modelo construído é facilitada por meio da identificação da rastreabilidade dos elementos envolvidos. O *plug-in* UMP-ST promove a rastreabilidade dos artefatos criados a partir de cada disciplina por meio da criação implícita da RTM. Entretanto, o *plug-in* oferece suporte apenas às disciplinas de Requisitos e Análise & Design. Por ser uma metodologia geral, o UMP-ST permite que o engenheiro de ontologia escolha a linguagem para a representação da PO e, consequentemente, faça um mapeamento entre a estrutura já criada e a que será desenvolvida. Nesse trabalho, o modelo ontológico será desenvolvido por meio do auxílio do UnBBayes (onde serão utilizados o *plug-in* MEBN/PR-OWL 2 e o *plug-in* UMP-ST).

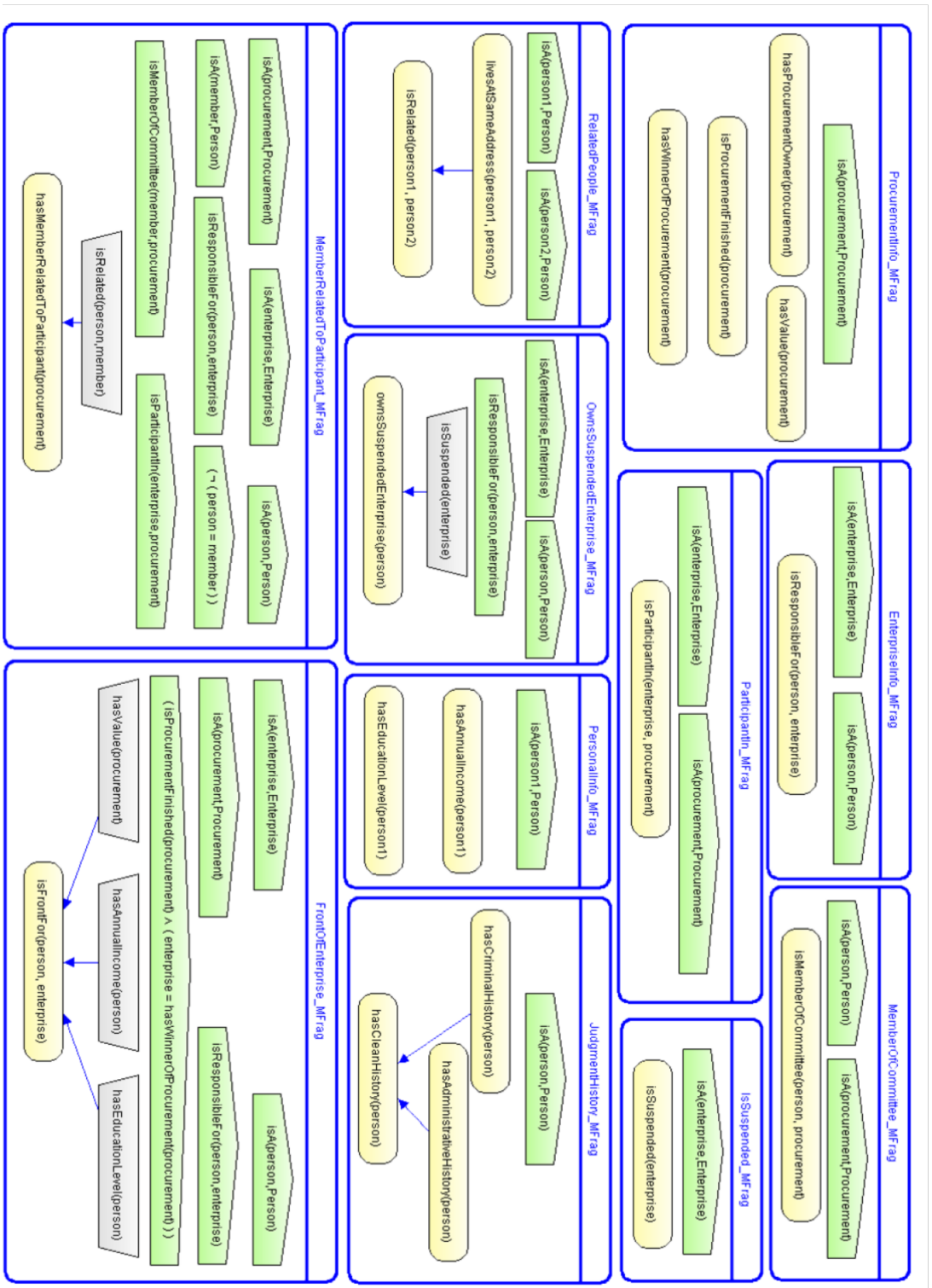

Figura 2.6: MTheory de Fraudes em Licitações Públicas Parte A

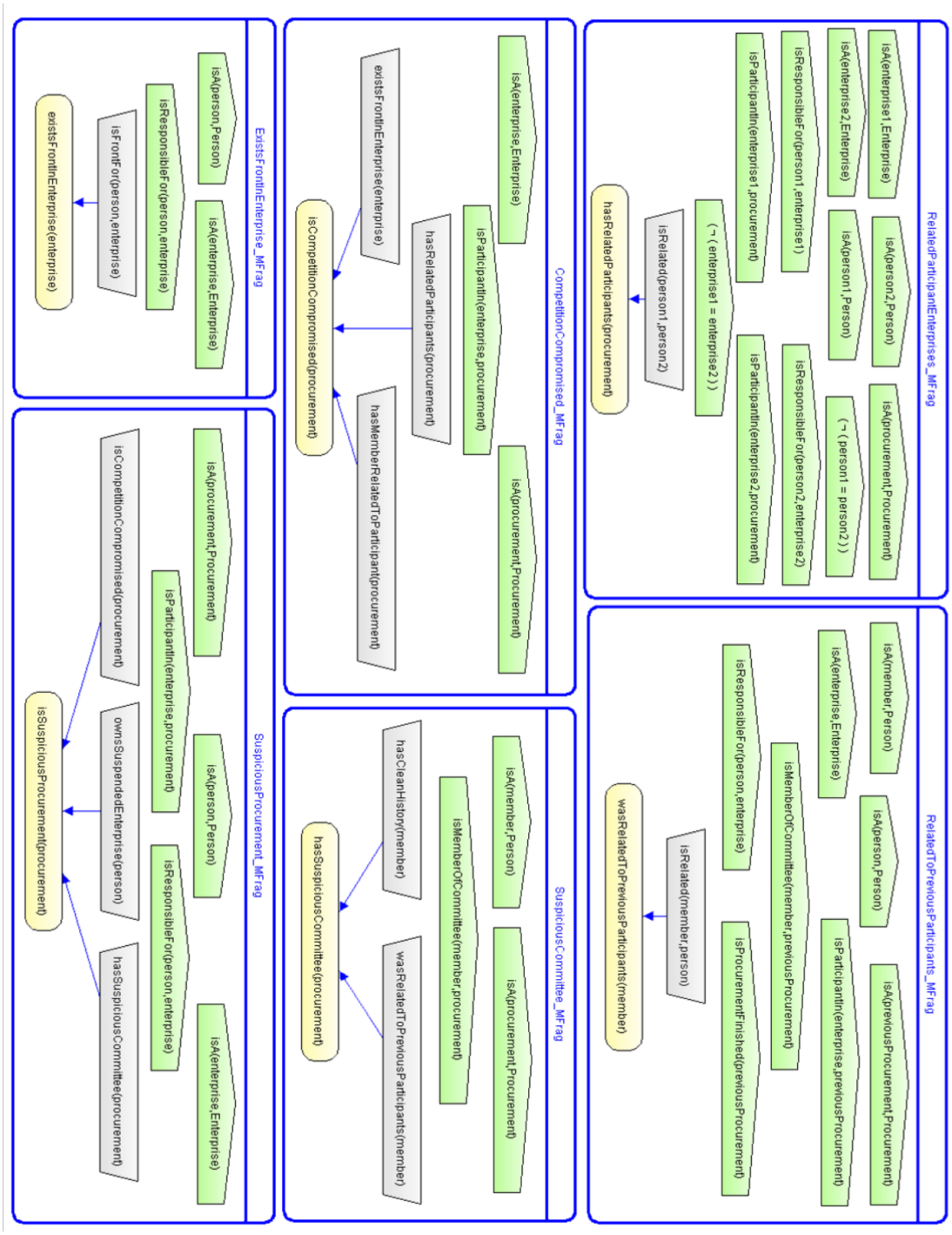

Figura 2.7: MTheory de Fraudes em Licitações Públicas Parte B

Capítulo 3

Metodologia

3.1 Metodologia

Para que uma ontologia probabilística seja criada utilizando a metodologia proposta pelo UMP-ST, é necessário que a linguagem escolhida seja semanticamente rica e capaz de representar incerteza [3, 5]. Ao escolher a linguagem, o modelo deve ser implementado por meio do mapeamento de elementos entre as duas estruturas.

Sendo definido o PR-OWL como linguagem, durante a pesquisa foi analisado a viabilidade do mapeamento entre o modelo definido no UMP-ST e a sua representação no MEBN/PR-OWL. Atualmente, o mapeamento e, posteriormente, a geração do modelo ocorrem manualmente durante a disciplina de Implementação. A geração automática da PO e, consequentemente, o seu mapeamento em PR-OWL tem como objetivo agilizar esse processo, evitando com que o modelo seja construído manualmente do zero. Entretanto, mesmo que o modelo seja gerado de forma automática, ele poderá ser modificado pelo engenheiro de ontologia por meio de uma ferramenta capaz de editar ontologias probabilísticas em PR-OWL.

Durante a construção de regras em Análise & Design, são definidas as relações de dependência entre os elementos presentes no domínio. Para que haja consistência local no modelo é importante com que o engenheiro de ontologias teste a regra por meio da utilização de uma linguagem especifica e avalie se o seu comportamento ocorreu como esperado. A relação de dependência definida durante a construção das regras mostra o quanto determinado evento pode influenciar na ocorrência de outro, seja de modo determinístico ou estocástico.

Após o mapeamento da estrutura básica do modelo, tais como entidades, atributos, relacionamentos e grupos, a geração da PO deve conter a relação de dependência entre

esses elementos. Logo, é importante que a descrição das regras seja computacionalmente compreendida. O UMP-ST apenas define as regras por meio da sua descrição textual, o que torna a sua identificação automática mais complexa e suscetível a erros. Por isso, durante a pesquisa foram analisadas alternativas à formalização de regras de modo que sua definição seja facilmente identificada pelo computador.

O mapeamento dos elementos básicos do UMP-ST junto a definição de regras ainda não são capazes de gerar um novo modelo em PR-OWL. A estrutura representada no MEBN/PR-OWL contém uma diferenciação entre as suas variáveis, classificando-as em nós de contexto, de entrada e residente. Apenas o mapeamento do modelo parcial por meio da formalização da regra não permite a classificação dos elementos do UMP-ST definidos como variáveis aleatórias. Portanto, é necessário que haja um outro recurso durante a geração da PO que permita a seleção dessas variáveis e as classifique de acordo com os termos do PR-OWL/MEBN.

Esse trabalho propõe o primeiro algoritmo de seleção para mapeamento da ontologia em MEBN/PR-OWL [45]. Para isso, o algoritmo considera que todas as regras que definem o domínio de aplicação foram especificadas de acordo com a nova formalização. O algoritmo é dividido em passos. Cada passo descreve um critério utilizado para a seleção das variáveis e sua classificação. Após o mapeamento e a análise do modelo pelo algoritmo, é gerada uma ontologia probabilística em PR-OWL.

Os recursos que viabilizam a geração automática do modelo em PR-OWL foram codificados por meio de um novo *plug-in* que estende o UMP-ST, permitindo a análise prática dessa nova abordagem. O *plug-in* foi desenvolvido em JAVA utilizando como base o *Java Plugin Framework* (JPF) e está disponível no SourceForge[2].

Além da codificação do *plug-in* para a geração automática de PO em PR-OWL, foi elaborado um novo formato baseado em XML para a persistência da estrutura anterior e posterior à nova abordagem de geração da PO. Basicamente, são armazenados os artefatos produzidos durante a disciplina de Requisitos e Análise & Design em conjunto com a formalização das regras.

Por fim, a solução proposta foi avaliada por meio de testes utilizando a ontologia de Fraudes em Licitações Públicas [37, 5]. Essa ontologia foi construída com o auxílio do UMP-ST e, posteriormente, implementada em PR-OWL. Além disso, relações de dependência definidas por meio da distribuição probabilística em outro modelo em MEBN, tal como em [23], promoveram uma análise mais abrangente do algoritmo ao verificar a definição de regras capazes de oferecer suporte a construção de relações mais complexas.

[2] `https://sourceforge.net/projects/unbbayes/`

Capítulo 4

Geração Automática de PO em PR-OWL

O mapeamento dos elementos definidos na modelagem de uma PO, segundo a metodologia UMP-ST, para um modelo PR-OWL é apresentada nesse capítulo. O mapeamento entre esses elementos será feito, basicamente, a partir da semelhança entre os conceitos que definem seus elementos.

4.1 Fundamentos para o mapeamento de modelos UMP-ST em modelo MEBN/PR-OWL

A seguir, são apresentados os conceitos dos elementos presentes nessas duas abordagens e, em seguida, é apresentado na Figura 4.1 o mapeamento geral feito a partir da correlação de seus conceitos.

Entidade: em ontologia probabilística, uma entidade representa qualquer conceito (real ou fictício, concreto ou abstrato) que pode ser descrito e fundamentado em um domínio de aplicação [1]. O UMP-ST representa uma entidade, tal como na ontologia probabilística.

Entidades definidas no UMP-ST são mapeadas para entidades no MEBN/PR-OWL. Durante este mapeamento, não há uma distinção entre diferentes elementos pertencentes a mesma Entidade no UMP-ST. O MEBN relaciona cada elemento da Entidade a um identificador único. Esse mesmo princípio é adotada pelo PR-OWL. Neste caso, identificadores únicos serão atribuídos aos elementos da Entidade apenas quando tratados no MEBN/PR-OWL.

Atributo: denota um conjunto de características pertencentes a uma Entidade [3, 2]. No UMP-ST, cada atributo deve ser criado apenas quando houver uma entidade. No

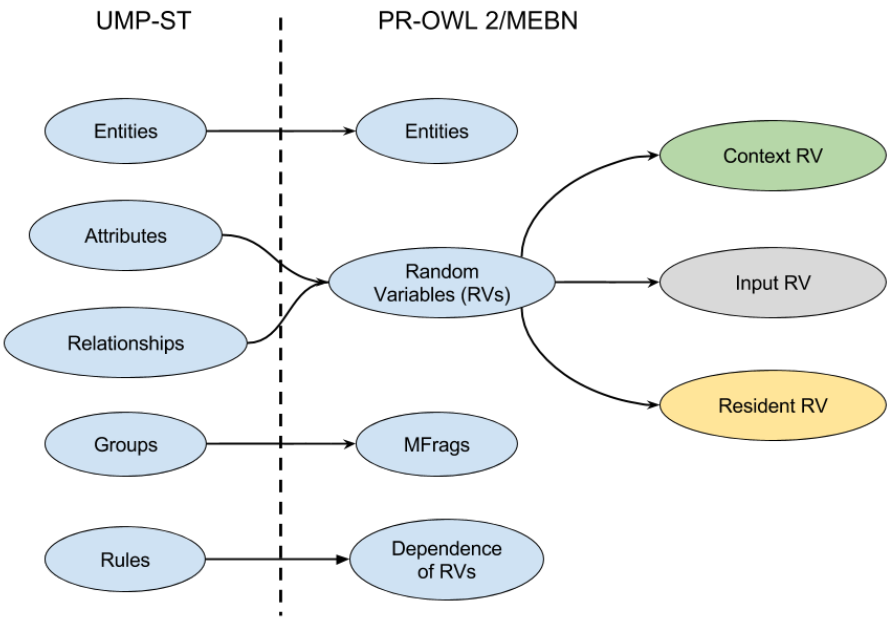

Figura 4.1: Mapeamento dos Elementos do UMP-ST para MEBN/PR-OWL

MEBN e PR-OWL, os atributos podem ser definidos como variáveis aleatórias ou variáveis ordinárias.

Relacionamento: o MEBN e o PR-OWL tratam o mundo como sendo composto de entidades que possuem atributos e relações entre outras entidades [1, 2]. Para Carvalho [3], o relacionamento é definido como uma relação (ou ligação) entre duas ou mais Entidades. Entretanto, é comum a existência de relacionamentos cujas relações são binárias, onde o primeiro argumento pertence ao domínio da relação, e o segundo ao seu contra-domínio [3].

O MEBN e PR-OWL também é capaz de representar funções. Uma função se diferencia de uma relação ao possuir elementos únicos dado os elementos anteriores. Por exemplo, a função `EvaluateEnterprise`, definida em [3], aplica-se a um conjunto de regras de pontuação para computar a pontuação final recebida por uma empresa quando participa de uma licitação específica. Para uma dada empresa e licitação existem exatamente uma pontuação; portanto, o relacionamento é funcional. Tanto a relação e a função são vistas pelo MEBN e PR-OWL como variáveis aleatórias [2]. Relacionamentos cujo argumento são outros relacionamentos, tal como uma referência indireta definida em [2], não são suportados pelo UMP-ST.

O UMP-ST representa o relacionamento como uma ligação entre duas ou mais Entidades, onde a distinção entre uma relação e uma função depende do contexto na qual é aplicado. Essa distinção é feita pelo engenheiro de ontologias. O mapeamento de relacionamentos do UMP-ST para o MEBN/PR-OWL será feito por meio da criação de

variáveis aleatórias no MEBN/PR-OWL que, posteriormente, serão classificadas como VA de contexto, de entrada ou residente.

As seguintes definições formalizam conceitos utilizados no UMP-ST.

Definição 4 Um relacionamento $r(E_1, ..., E_n)$ consiste em uma ligação entre uma ou mais Entidades E_i presentes no domínio, tal que $1 \leq i \leq n$.

Regras: cada regra no UMP-ST pode ser classificada como determinística ou estocástica e é definida como um conjunto de relacionamentos, entidades e atributos que especificam uma relação de dependência presente no domínio de aplicação. O UMP-ST permite que cada regra possa ser dependente de uma ou mais regras distintas. Assim, uma sucessão de eventos determinados por uma regra pode ser influenciada por um conjunto de regras distintas entre sí.

Definição 5 Uma tupla $\rho = (E, A, R, L, \kappa)$ representa uma regra e consiste em um conjunto finito $E = \{E_1, .., E_n, ..., E_k\}$ de entidades; um conjunto finito A de atributos; um conjunto finito R de relacionamentos, tal que $r = r(E_1, ..., E_n)$, $r \in R$; um conjunto finito L de regras dependentes, tal que, $\forall \rho^{'} \in L$, $\rho^{'} \neq \rho$; uma relação de dependência κ do tipo determinística ou estocástica.

A definição de κ possui um formato livre no UM-ST. Ela apenas relaciona os elementos da regra que possuem uma dependência entre si. O UMP-ST não estabelece um método formal para a sua definição, por isso, no *plug-in* UMP-ST [5, 18], κ é definida por um texto, cabendo ao engenheiro de ontologias interpretá-la e implementá-la manualmente. Como parte deste trabalho, posteriormente, κ será formalizada.

Grupos: cada grupo no UMP-ST abrange um conjunto de entidades, atributos, relacionamentos e regras pertencentes a um domínio específico. Basicamente, seu objetivo é organizar o modelo construído em nichos. A intersecção de dois ou mais grupos não necessariamente será vazia. O mapeamento dos grupos para o MEBN/PR-OWL é feito por meio da criação de MFrags [3, 5].

Definição 6 Uma tupla $g = (E, A, R, P)$ representa um grupo e consiste em um conjunto finito e não-vazio E de entidades; um conjunto finito A de atributos; um conjunto finito e não-vazio R de relacionamentos; um conjunto finito P de regras;

6a. Seja $P = \{\rho_1, ..., \rho_n\}$ e $\rho_j = (E_j, A_j, R_j, L_j, \kappa)$ uma regra, tal que $1 \leq j \leq n$; $E_p = \cup_{j=1}^{n} E_j$ é o conjunto de todas as Entidades em P.

6b. E_s é o conjunto de Entidades, onde $E_s \subseteq E$ e $E_s \cap E_p = \oslash$.

6c. $R_p = \cup_{j=1}^{n} R_j$ é o conjunto de todos os relacionamentos em P.

6d. R_s é o conjunto de relacionamentos, onde $R_s \subseteq R$ e $R_s \cap R_p = \oslash$.

6e. $A_p = \cup_{j=1}^{n} A_j$ é o conjunto de todos os atributos em P.

6f. A_s é o conjunto de atributos, onde $A_s \subseteq A$ e $A_s \cap A_p = \oslash$.

Dado que todos os elementos definidos durante a disciplina de Análise & Design são agrupados em uma quantidade finita de grupos, o modelo definido no UMP-ST e utilizado durante o mapeamento para o modelo MEBN será formado pelo conjunto de todos os grupos.

Definição 7 O modelo $\mathcal{M} = \{g_1, ..., g_n\}$ é um conjunto finito e não-vazio de grupos definidos pela tupla $g_i = (E_i, A_i, R_i, P_i)$, tal que $1 \leq i \leq n$.

7a. $R_{\mathcal{M}} = \cup_{i=1}^{n} R_i$ é o conjunto de todos os relacionamentos definidos em $\mathcal{M}$.

Apesar do UMP-ST ser um metodologia geral de modelagem para a maioria das tecnologias semânticas existentes que oferecem suporte a representação de incerteza [3, 5], o PR-OWL não estabelece uma ligação formal entre seus termos e os elementos do UMP-ST. Assim, se torna impossível para os mecanismos de inferência identificar a ligação entre as duas estruturas. Em um cenário melhor, esses mecanismos podem apenas "achar"que eles são semelhantes, pois possuem sintaxes semelhantes [3]. Sendo assim, o trabalho proposto foca na formalização do mapeamento entre o UMP-ST e o PR-OWL/MEBN, permitindo a geração de uma ontologia probabilística a partir do modelo UMP-ST.

4.2 Formalização de Regras no UMP-ST

A construção de regras que fazem parte do domínio de aplicação e, especificamente, a definição de suas relações de dependência devem ser feitas sobre uma estrutura capaz de ser representada computacionalmente. A partir da forma textual, os elementos definidos na regra assumem uma nova organização capaz de ser processada e, posteriormente, auxiliar na geração do modelo em PR-OWL. Para isso, junto a formalização da regra, por meio da Definição 5, é adicionada uma nova definição referente à relação de dependência e a criação de novos símbolos.

- *Símbolos de identificadores:* Símbolos de identificadores são utilizados como argumentos associados a Entidades definidas na regra. Eles são escritos como uma *string* alfanumérica iniciados com letra minúscula, exemplo `person`. Os símbolos de identificadores utilizados na formalização da regra são diferentes dos símbolos de identificadores único definidos no MEBN;

- *Símbolos de constantes:* São Entidades presentes no UMP-ST. Eles são escritos como uma *string* alfanumérica iniciados com letra maiúscula, exemplo `Person`. Durante a formalização da relação de dependência, símbolos constantes serão definidos a partir da sua associação com Entidades presentes na regra. Os símbolos de constantes utilizados na formalização da regra são diferentes dos símbolos de constantes definidos no MEBN;

- *Conectivos lógicos e operador de igualdade:* Os símbolos de conectivos lógicos $\neg$, $\wedge$, $\vee$, $\Rightarrow$, e $\Leftrightarrow$, junto com a relação de igualdade $=$ são definidos como na FOL, respectivamente, como operadores de "negação", "e", "ou", "$\Rightarrow$", "$\Leftrightarrow$"e "$=$". Eles são utilizados na construção de fórmulas durante a definição da relação de dependência. As expressões lógicas são escritas utilizando a notação pré-fixada.

- *Quantificadores:* Os símbolos $\forall$ e $\exists$ são definidos como $\forall$ e $\exists$ na lógica de primeira ordem e são utilizados na construção de fórmulas durante a definição da relação de dependência. Eles são escritos utilizando a notação pré-fixada.

- *Símbolos de relacionamentos de identificadores:* Os símbolos de relacionamentos de identificadores consistem em uma ligação entre um ou mais identificadores. Eles são escritos como uma *string* alfanumérica iniciados com letra minúscula, exemplo `owsSuspendedEnterprise(person)`. Os símbolos de relacionamentos de identificadores irão definir predicados e funções na construção de fórmulas durante a definição da relação de dependência.

- *Símbolos exemplares:* Os símbolos exemplares são utilizados como identificadores vinculados ao contra-domínio de quantificadores. Eles são escritos como uma *string* alfanumérica iniciados com letra minúscula. Os símbolos exemplares utilizados na formalização da regra são diferentes dos símbolos exemplares definidos no MEBN.

Definição 8 Seja $\rho = (E, A, R, L, \kappa)$ uma regra, um conjunto de ligação de identificadores $O = \{(o_1 : E_1), ..., (o_n : E_n), ..., (o_k : E_k)\}$ é um conjunto finito de pares ordenados que associa cada identificador o_i com uma Entidade $E_i \in E$. Os identificadores o_i em $(o_i : E_i) \in O$ são distintos entre si.

8a. Seja $r = r(E_1, ..., E_n)$ um relacionamento em R; $r_\lambda(o_1, ..., o_n)$ é um relacionamento de identificador formado a partir de r ao substituir cada ocorrência de E_i por o_i, tal que $(o_i : E_i) \in O$ e $1 \leq i \leq n$.

8b. Seja $r(E_1, ..., E_n)$ um relacionamento em R, tal que $E_1 = ... = E_n$; seja $r_\lambda = r_\lambda(o_1, ..., o_n)$ e $r'_\lambda = r_\lambda(o_{n+1}, ..., o_{2n})$ relacionamentos de identificadores formados a partir de r, dado $(o_i : E_i) \in O$ e $1 \leq i \leq k$, onde $k = 2n$; r_λ e r'_λ serão distintos.

8c. R_λ é um conjunto finito de relacionamentos de identificadores, tal que $r_\lambda \in R_\lambda$.

Seja o conjunto de ligação qualquer O composto pela associação do identificador `person` com a Entidade `Person` definida na regra, tal que (`person` : `Person`) $\in O$; o relacionamento definido na regra `ownsSuspendedEnterprise(Person)` será um relacionamento de identificador r_λ ao substituir `Person` por `person`. Dessa forma, r_λ = `ownsSuspendedEnterprise(person)`. Cada par ordenado $(o_i : E_i) \in O$ será definido pelo usuário durante a construção de relações de dependência.

Definição 9 Seja $\rho = (E, A, R, L, \kappa)$ uma regra; O um conjunto de ligação de identificadores, tal que $(o_i : E_i) \in O$; $R_\mathcal{M}$ o conjunto de todos os relacionamentos definidos no modelo $\mathcal{M}$; N é um conjunto finito de fórmulas da lógica de primeira ordem, tal que:

1. Identificador o e constante c são definidas como um identificador o_i associado a Entidade $E_i \in E$, onde $(o_i : E_i) \in O$; c é uma constante c_i definida a partir da associação com a Entidade $E_i \in E$;

2. Predicado e função são definidos como um relacionamento de identificador $r_{\lambda_\mathcal{M}}(o_1, ..., o_n)$ formado a partir do relacionamento $r(E_1, ..., E_n) \in R_\mathcal{M}$ ao substituir cada ocorrência de $E_i \in E$ pelo identificador o_i, onde $(o_i : E_i) \in O$;

3. Termo t é definido como uma identificador o, uma constante c, um predicado ou uma função $r_{\lambda_\mathcal{M}}$;

4. Termo exemplar ν é utilizado com quantificadores a partir da substituição de o_i por ν em $r_{\lambda_\mathcal{M}}(..., o_i, ...)$, tal que $1 \leq i \leq n$;

5. Conectivos lógicos possuem dois argumentos definidos por μ_1 e μ_2, fórmulas em N;

6. Igualdade possui dois argumentos definidos por dois termos t_1 e t_2;

7. Quantificadores possuem dois argumentos definidos por um termo exemplar ν e uma fórmula μ, onde μ é definida a partir de $r_{\lambda_{\mathcal{M}}}(..., \nu, ...)$;

Seja μ uma fórmula definida em N, μ será utilizada como condição necessária para que a relação de dependência seja aplicada na regra.

Definição 10 Seja R_λ o conjunto de relacionamentos de identificadores,

1. U é um conjunto de causas, tal que $U \subseteq R_\lambda$;
2. V é um conjunto de efeitos, tal que $V \subset R_\lambda$;
3. $U \cap V = \oslash$.
4. $\Omega \subseteq U \times V$ é um conjunto de relações causais, tal que $\omega_i = (u, v)$ é uma relação causal em Ω formada por uma causa $u \in U$ e um efeito $v \in V$.

Relacionamentos de identificadores definidos como causa $u \in U$ e efeito $v \in V$ especificam dependências $\omega_i \in \Omega$ definidas pela regra. O conjunto de relações causais Ω deve ser definido pelo engenheiro de ontologias durante a especificação da relação de dependência κ.

A definição da relação de dependência κ deve ser composta pelos elementos dos conjuntos O, N, U, V e Ω.

Definição 11 Seja $\rho = (E, A, R, L, \kappa)$ uma regra, a tupla $\kappa = (O, N, U, V, \Omega)$ é uma relação de dependência composta por um conjunto finito e não-vazio O de ligação de identificadores; um conjunto finito N de fórmulas; um conjunto finito e não-vazio U de causas; um conjunto finito V de efeitos; um conjunto finito e não-vazio Ω de relações causais.

A partir da definição da relação de dependência κ, é possível construir regras no UMP-ST além da sua especificação textual. Uma das regras definidas na ontologia para detecção de fraude em licitação pública proposta por Carvalho [3, 37, 5] tem como objetivo identificar pessoas responsáveis por empresas suspensas. Em sua forma escrita, a regra ρ descreve que:

1. "If the responsible person of an enterprise is also responsible for another enterprise which is suspended, then is more likely that this person owns suspended enterprise."

> Se a pessoa responsável por uma empresa é também responsável por outra empresa que é suspensa, então é provável que essa pessoa possua uma empresa suspensa.

No UMP-ST, essa regra participa do domínio definido pelo grupo g `Owns Suspended Entreprise`. De acordo com a formalização proposta, os elementos presentes na regra ρ são organizados tal como ilustrado na Figura 4.2.

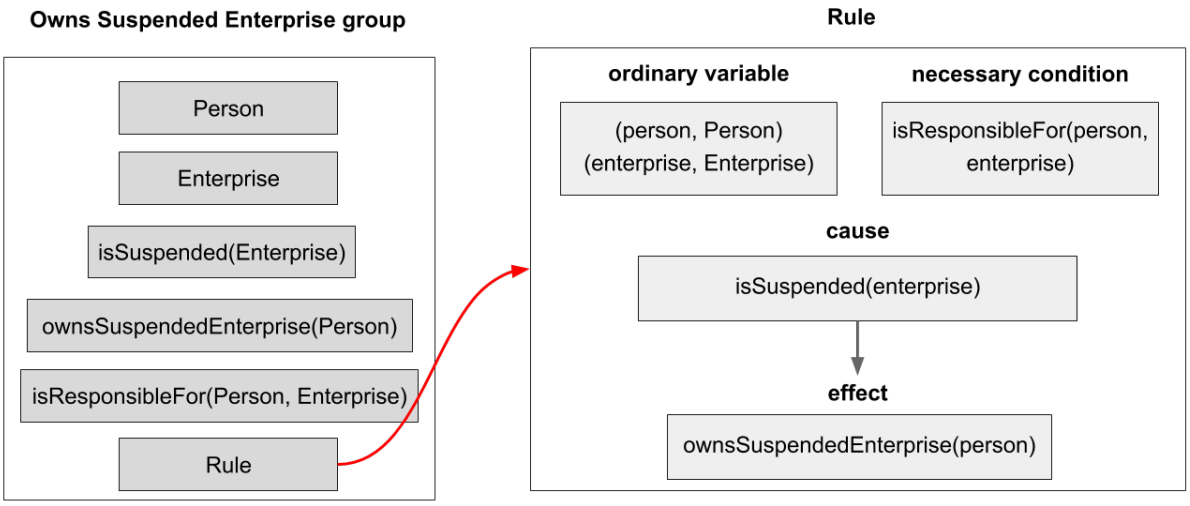

Figura 4.2: Grupo e Formalização da Regra

Seja o grupo `Owns Suspended Entreprise` $g = (E_g, A_g, R_g, P_g)$, onde a regra definida ρ pertence ao conjunto de regras P_g presente no grupo; E_g e R_g serão compostos, tal como na Figura 4.2, por entidades e relacionamentos presentes na regra. Assim, $E_g = \{$`Person`, `Enterprise`$\}$ e $R_g = \{$`isSuspended(Enterprise)`, `ownsSuspendedEnterprise(Person)`, `isResponsibleFor(Person, Enterprise)`$\}$; $A_g = \oslash$. A composição de entidades, relacionamentos e atributos do grupo não é restrito apenas aos elementos que pertencem às regras. É possível que o grupo g contenha elementos em E_g, A_g e R_g além dos que foram definidos em P, tal como E_s, A_s e R_s, Definição 6.

A organização da regra segue a formalização proposta na definição de κ. A partir da sua forma textual, a regra $\rho = (E, A, R, L, \kappa)$ terá o conjunto de entidades definido por $E = \{$`Person`, `Enterprise`$\}$; o conjunto de atributos $A = \oslash$; o conjunto de relacionamentos $R = \{$`isSuspended(Enterprise)`, `ownsSuspendedEnterprise(Person)`, `isResponsibleFor(Person, Enterprise)`$\}$; o conjunto de regras dependentes $L = \oslash$. A definição dos elementos que constituem o modelo é feito manualmente a partir do entendimento do engenheiro de ontologias. Por isso, a especificação desses elementos está sujeita a erros.

A partir dos elementos em E, R e A, é determinada a relação de dependência κ. Seja $\kappa = (O, N, U, V, \Omega)$, a condição necessária definida por uma fórmula em N para que

uma pessoa possua uma empresa suspensa, tal como descrito na regra, é que a pessoa sob suspeita seja responsável por uma empresa. Essa condição é expressa por meio do relacionamento `isResponsibleFor(Person, Enterprise)`. Segundo a definição de N, relacionamentos r que participam de N assumem a forma $r_{\lambda_{\mathcal{M}}}(o_1, ..., o_i)$. Para isso, é preciso que existam identificadores que sejam associados às Entidades `Person` e `Enterprise` presentes em r. Então, é definido o conjunto de ligação de identificadores O, tal que $O = \{(\texttt{person} : \texttt{Person}), (\texttt{enterprise} : \texttt{Enterprise})\}$. Logo, a fórmula em N será definida pela sentença `isResponsibleFor(person, enterprise)`.

A relação causal $\omega \in \Omega$ presente em κ é definida pelo par ordenado (u, v) formado pelo conjunto de causas $u \in U$ e pelo conjunto de efeitos $v \in V$. A partir da especificação da regra, sabe-se que se uma pessoa responsável por uma empresa possui outra empresa que está suspensa, então é provável que a empresa a qual a pessoa é responsável também esteja suspensa. Essa relação é determinada pelos relacionamentos `isSuspended(Enterprise)` e `ownsSuspendedEnterprise(Person)`. Os elementos de U e V serão definidos a partir dos relacionamentos de identificadores formados pela substituição das Entidades `Enterprise` e `Person` presentes nestes relacionamentos com os elementos definidos no conjunto de identificadores O. Assim, $U = \{\texttt{isSuspended(enterprise)}\}$ e $V = \{\texttt{ownsSuspendedEnterprise(person)}\}$; a relação causal será $\omega = (\texttt{isSuspended(enterprise)}, \texttt{ownsSuspendedEnterprise(person)})$.

4.3 Mapeamento de Modelos UMP-ST para Modelos MEBN/PR-OWL

A formalização do mapeamento entre os elementos do UMP-ST para os elementos do MEBN/PR-OWL, tal como ilustrado na Figura 4.1, será feita por meio de funções cujos domínios estão contidos no modelo $\mathcal{M} = \{g_1, ..., g_n\}$ e contra-domínios na MTheory $\mathcal{T}' = \{\mathcal{F}_1, ..., \mathcal{F}_n\}$.

Seja um grupo $g_i \in \mathcal{M}$ definido pela tupla $g_i = (E_i, A_i, R_i, P_i)$, tal que $1 \leq i \leq n$:

1. E_i é um conjunto finito e não-vazio de Entidades;
2. A_i é um conjunto finito de atributos;
3. R_i é um conjunto finito e não-vazio de relacionamentos, tal que $r \in R_i$, $r = r(E_1, ..., E_n)$ e $E_i = \{E_1, ..., E_n, ..., E_m\}$;
4. $P_i = \{\rho_1, ..., \rho_n\}$ é um conjunto finito de regras composto pela tupla $\rho_j = (E_j, A_j, R_j, L_j, \kappa_j)$, tal que $\rho_j \in P_i$ e $1 \leq j \leq n$;
5. $E_j \subseteq E_i$, $A_j \subseteq A_i$ e $R_j \subseteq R_i$;

6. $L_j \subset P_i$ é um conjunto de regras dependentes de ρ_j, tal que $\forall \rho \in L_j, \rho \neq \rho_j$;
7. $\kappa_j = (O, N, U, V, \Omega)$ é uma relação de dependência presente na regra ρ_j.
8. N é um conjunto finito de fórmulas μ expressas a partir dos símbolos definidos na formalização da regra;
9. $O = \{(o_1 : E_1), ..., (o_n : E_n)\}$ é um conjunto finito de ligação de identificadores, tal que cada identificador o_k está associado a uma Entidade $E_k \in E_j, 1 \leq k \leq n$;
10. U é um conjunto de causas definido a partir do conjunto de relacionamentos de identificadores R_λ, Definição 8 e 10;
11. V é um conjunto de efeitos definido a partir do conjunto de relacionamentos de identificadores R_λ, Definição 8 e 10;
12. $\Omega \subseteq U \times V$ é um conjunto de relações causais definido por $\omega = (u, v)$, tal que $u \in U$, $v \in V$ e $U \cap V = \oslash$.

Seja uma MFrag $\mathcal{F}_i \in \mathcal{T}'$ definida pela tupla $\mathcal{F}_i = (\mathcal{C}_i, \mathcal{I}_i, \mathcal{R}_i, \mathcal{G}_i, \mathcal{D}_i)$:

1. $\mathcal{C}_i$ é um conjunto finito de VA de *contexto* definido por fórmulas ϕ expressas por símbolos do MEBN;
2. $\mathcal{I}_i$ é um conjunto finito de VA de *entrada*;
3. $\mathcal{R}_i$ é um conjunto finito de VA *residentes*;
4. $\mathcal{G}_i$ é um fragmento de grafo acíclico dirigido cujos nós possuem correspondência de um-para-um com as VA em $\mathcal{I}_i \cup \mathcal{R}_i$;
5. $\mathcal{D}_i$ é um conjunto de distribuição local, uma para cada VA residente.

O mapeamento do grupo $g_i \in \mathcal{M}$ no UMP-ST para a MFrag $\mathcal{F}_i \in \mathcal{T}'$ no MEBN é feito a partir do mapeamento dos conjuntos definidos em suas tuplas.

Definição 12 Seja N um conjunto de fórmulas definidas em κ, e seja $\mathcal{C}$ um conjunto de variáveis aleatórias de contexto definidas em $\mathcal{F}$. A função $f : N \rightarrow \mathcal{C}$ mapeia a fórmula $\mu \in N$ para $\phi \in \mathcal{C}$ se:

1. $f(o) = \theta$, se o for um identificador em O, tal que $(o : E) \in O$, para alguma variável ordinária θ, onde $(\theta : \varepsilon) \in \mathcal{B}$, Definição 2;
2. $f(c) = \varrho$, se c for uma constante em μ, para alguma contante ϱ definida em ϕ;
3. $f(r_{\lambda_\mathcal{M}}(o_1, ..., o_n)) = \psi(f(o_1), ..., f(o_n))$, se $r_{\lambda_\mathcal{M}}$ for um relacionamento de identificador definido em μ, para alguma variável aleatória ψ definida em ϕ;

4. $f(t_\mu) = t_\phi$, se t_μ for um termo em μ definido a partir de o, c e $r_{\lambda_\mathcal{M}}(o_1, ..., o_n)$, para algum termo t_ϕ em ϕ definido a partir de $f(o), f(c)$ e $f(r_{\lambda_\mathcal{M}}(o_1, ..., o_n))$;

5. $f(t'_\mu = t''_\mu) = (f(t'_\mu) = f(t''_\mu))$;

6. $f(\mu_1 \vee \mu_2) = f(\mu_1) \vee f(\mu_2)$; $f(\mu_1 \wedge \mu_2) = f(\mu_1) \wedge f(\mu_2)$; $f(\mu_1 \Rightarrow \mu_2) = f(\mu_1) \Rightarrow f(\mu_2)$; $f(\mu_1 \Leftrightarrow \mu_2) = f(\mu_1) \Leftrightarrow f(\mu_2)$ e $f(\neg\mu_1) = \neg f(\mu_1)$;

7. $f(\forall(\nu, \mu)) = \forall(f(\nu), f(\mu))$, se ν for um termo exemplar em $r_{\lambda_\mathcal{M}}(..., \nu, ...)$ e $r_{\lambda_\mathcal{M}}(..., \nu, ...)$ seja definido em μ.

8. $f(\exists(\nu, \mu)) = \exists(f(\nu), f(\mu))$, se ν for um termo exemplar em $r_{\lambda_\mathcal{M}}(..., \nu, ...)$ e $r_{\lambda_\mathcal{M}}(..., \nu, ...)$ seja definido em μ.

O mapeamento das fórmulas definidas em N para as fórmulas definidas em $\mathcal{C}$ por meio da função $f : N \rightarrow \mathcal{C}$ não inclui a referência indireta definida por $\phi(t_1, ..., t_k)$, tal que $t_1, ..., t_k$ são termos definidos no MEBN presentes como argumentos na variável aleatória ψ.

Seja o conjunto de fórmulas $N = \{$`isResponsibleFor(person, enterprise)`$\}$ definido no exemplo ilustrado na Figura 4.2; o mapeamento de N para $\mathcal{C}$ será dado por $f($`isResponsibleFor(person, enterprise)`$) =$ `isResponsibleFor`$($$f($`person`$)$, $f($`enterprise`$)))$; $f($`person`$) =$ `person` e $f($`enterprise`$) =$ `enterprise` serão variáveis ordinárias presentes na variável aleatória de contexto `isResponsibleFor(person, enterprise)`.

No MEBN, os símbolos de identificadores de entidades $\varepsilon_1, ..., \varepsilon_n$ são usados como rótulos para tipo entidades. Seja $\mathcal{B} = \{(\theta_1 : \varepsilon_1), ..., (\theta_n : \varepsilon_n)\}$ um conjunto de ligação em $\mathcal{F}$, e seja $\psi(\theta_1, ..., \theta_n)$ uma variável aleatória em $\mathcal{F}$; a cada ocorrência de θ_i presente em ψ, θ_i poderá ser substituído por ε_i, Definição 2.

Definição 13 Seja $O = \{(o_1 : E_1), ..., (o_n : E_n)\}$ um conjunto de ligação de identificadores e $\mathcal{B} = \{(\theta_1 : \varepsilon_1), ..., (\theta_n : \varepsilon_n)\}$ um conjunto de ligação em $\mathcal{F}$. A função h mapeia o_i e E_i em $(o_i : E_i) \in O$ para alguma variável aleatória de contexto `isA`$(\theta_i, E_i) \in \mathcal{C}$ se, dado $1 \leq i \leq n$:

1. $\mathcal{O} = \{o_1, ..., o_n\}$ é um conjunto de identificadores, se o_i é um identificador em $(o_i : E_i) \in O$;

2. $\Theta = \{\theta_1, ..., \theta_n\}$ é um conjunto de variáveis ordinárias, se θ_i é uma variável ordinária em $(\theta_i : \varepsilon_i) \in \mathcal{B}$;

3. $s(o_i) = \theta_i$ é uma função, se $o_i \in O$, para algum $\theta_i \in \Theta$;

4. $E = \{E_1, ..., E_n\}$ é um conjunto de Entidades, se E_i é uma Entidade em $(o_i : E_i) \in O$;

5. $\mathcal{E} = \{\varepsilon_1, ..., \varepsilon_n\}$ é um conjunto de identificadores de entidades, se ε_i é um identificador de entidade em $(\theta_i : \varepsilon_i) \in \mathcal{B}$;

6. $y(E_i) = \varepsilon_i$ é uma função, se $E_i \in E$ for um tipo entidade utilizado por algum identificador de entidade $\varepsilon_i \in \mathcal{E}$ como rótulo;

7. $h(s(o_i), E_i) = \texttt{isA}(\theta_i, E_i)$, se o par ordenado $(s(o_i) : y(E_i)) \in \mathcal{B}$.

Seja $\psi(\theta)$ uma variável aleatória definida em $\mathcal{F}$, tal que θ seja uma variável ordinária definida na VA de contexto $\texttt{isA}(\theta, E_i)$; $\psi(\varepsilon)$ será uma instância de ψ obtida pela substituição de θ por ε se $\varepsilon \in \mathcal{E}$, onde $\mathcal{E}$ é o conjunto de identificadores de entidades que utilizam como rótulo o tipo entidade E_i.

Seja $O = \{(\texttt{person} : \texttt{Person}), (\texttt{enterprise} : \texttt{Enterprise})\}$ o conjunto de identificadores ilustrado pela Figura 4.2. Segundo a Definição 13, $\mathcal{O} = \{\texttt{person}, \texttt{enterprise}\}$ será o conjunto de identificadores formado a partir de O; $\Theta = \{\texttt{person}, \texttt{enterprise}\}$ será o conjunto de variáveis ordinárias presentes em $\mathcal{B}$; a função $s : \mathcal{O} \rightarrow \Theta$ mapeia os identificadores em $\mathcal{O}$ para variáveis ordinárias em Θ; $E = \{\texttt{Person, Enterprise}\}$ é o conjunto de Entidades presentes em O; $\mathcal{E} = \{\varepsilon_1, ..., \varepsilon_n\}$ será o conjunto de identificadores de entidades definidas pelo usuário durante a instanciação do modelo MEBN, tal que cada ε_i poderá ter como rótulo Entidades em E; a função $y : E \rightarrow \mathcal{E}$ mapeia Entidades em E para identificadores de entidades ε_i que definem como rótulo `Person` ou `Enterprise`. Dado os conjuntos e funções definidas, $h(\texttt{person}, \texttt{Person}) = \texttt{isA(person, Person)}$ e $h(\texttt{enterprise}, \texttt{Enterprise}) = \texttt{isA(enterprise, Enterprise)}$.

No MEBN, a dependência entre variáveis aleatórias é expressa por arcos no fragmento de grafo $\mathcal{G}$ e definida em VA residentes por meio da distribuição local em $\mathcal{D}$. Seja $\mathcal{I} \cup \mathcal{R}$ o conjunto de variáveis aleatórias presentes em $\mathcal{G}$, sabe-se que para cada VA de entrada em $\mathcal{I}$, existe uma VA residente definida em uma *home* MFrag $\mathcal{F}^{'}$, tal que $\mathcal{F} \neq \mathcal{F}^{'}$.

A relação de dependência κ, definida na formalização da regra, é composta por um conjunto de relações causais Ω. Cada relação causal $\omega_i \in \Omega$ é formada por relacionamentos de identificadores em R_λ definidos como causa em U ou efeito em V, tal que $\Omega \subseteq U \times V$. Para que haja o mapeamento de Ω para $\mathcal{G}$, é necessário que os elementos da relação causal

U e V sejam mapeados para elementos do fragmento do grafo $\mathcal{I}$ e $\mathcal{R}$.

O mapeamento dos relacionamentos em U e V será feito por:

- Uma função que mapeia cada identificador o_i em $r_\lambda(..., o_i, ...)$ para alguma variável ordinária θ_i definida em $\texttt{isA}(\theta_i : E_i) \in \mathcal{C}$, tal que $1 \leq i \leq n$;
- Uma função que mapeia cada relacionamento r_λ para alguma variável aleatória em $\mathcal{I}$ ou em $\mathcal{R}$.

Definição 14 Seja $\{r_\lambda(o_1, ..., o_i), ..., r_\lambda(o_1, ..., o_j)\}$ um conjunto de relacionamentos de identificadores, onde $1 \leq i, j \leq n$. Seja $\mathcal{C}$, $\mathcal{I}$ e $\mathcal{R}$ conjuntos definidos na MFrag $\mathcal{F}$.

14a. $\mathcal{C} = \{\texttt{isA}(\theta_1 : E_1), ..., \texttt{isA}(\theta_n : E_n)\}$ é um conjunto formado de acordo com a Definição 13;

14b. $\mathcal{I} = \{\psi_I(\theta_1, ..., \theta_i), ..., \psi_I(\theta_1, ..., \theta_j)\}$ é um conjunto de VA de entrada, tal que cada ψ_I é VA residente em $\mathcal{F}^{'}$, com $\mathcal{F}^{'} \neq \mathcal{F}$;

14c. $\mathcal{R} = \{\psi(\theta_1, ..., \theta_i), ..., \psi(\theta_1, ..., \theta_j)\}$ é um conjunto de VA residentes;

14d. $\upsilon(o_k) = \theta_k$, se o_k for um identificador em r_λ, para algum θ_k em $\texttt{isA}(\theta_k : E_k) \in \mathcal{C}$, tal que $1 \leq k \leq n$;

14e. $\chi(r_\lambda(\upsilon(o_1), ..., \upsilon(o_k))) = \psi^{'}(\theta_1, ..., \theta_k)$, onde $1 \leq k \leq n$, se r_λ for um relacionamento de identificador, para algum $\psi^{'} = \psi_I$ em $\mathcal{I}$ ou $\psi^{'} = \psi$ em $\mathcal{R}$.

A distribuição local de cada VA residente não será definida durante o mapeamento do modelo UMP-ST para o modelo MEBN. Dessa forma, $\mathcal{D} = \oslash$ em $\mathcal{F}$ e $\mathcal{F}^{'}$.

Dado o exemplo definido pela regra 1 e ilustrado na Figura 4.2, o conjunto de relacionamentos $\{\texttt{isSuspended(enterprise)}, \texttt{ownsSuspendedEnterprise(person)}\} = U \cup V$. Pela Definição 13, $\mathcal{C} = \{\texttt{isA(person, Person)}, \texttt{isA(enterprise, Enterprise)}\}$. Então, o mapeamento dos elementos de $U \cup V$ será definido pelo mapeamento de identificadores em variáveis ordinárias, onde $\upsilon(\texttt{enterprise}) = \texttt{enterprise}$ e $\upsilon(\texttt{person}) = \texttt{person}$; pelo mapeamento de relacionamentos para VA, tal que $\chi(\texttt{isSuspended}(\upsilon(\texttt{enterprise}))) =$ `isSuspended(enterprise)` e $\chi(\texttt{ownsSuspendedEnterprise}(\upsilon(\texttt{person}))) =$ `ownsSuspendedEnterprise(person)`. As variáveis aleatórias definidas estarão em $\mathcal{I} \cup \mathcal{R}$.

Um fragmento de grafo $\mathcal{G}$ em $\mathcal{F}$ pode ser definido por $(N_\mathcal{G}, E_\mathcal{G})$, tal que $N_\mathcal{G}$ é um conjunto formado por $\mathcal{I} \cup \mathcal{R}$ e $E_\mathcal{G}$ é um conjunto de arcos definido pelo par ordenado (n, m), onde $n, m \in N_\mathcal{G}$. A partir do mapeamento na Definição 14, é possível que cada VA em $\mathcal{I}$ e $\mathcal{R}$ seja uma variável aleatória mapeada a partir de um relacionamento definido

em U e V. Assim, cada nó em $N_{\mathcal{G}}$ será uma VA mapeada a partir de um relacionamento em U e V.

Definição 15 Seja $\mathcal{G} = (N_{\mathcal{G}}, E_{\mathcal{G}})$ um fragmento de grafo acíclico dirigido em $\mathcal{F}$, tal que $N_{\mathcal{G}} = \mathcal{I} \cup \mathcal{R}$ é um conjunto de nós e $E_{\mathcal{G}}$ é um conjunto formado por pares ordenados (n, m), onde $n, m \in N_{\mathcal{G}}$. Seja $\Omega \subseteq U \times V$ um conjunto de relações causais. A função $\beta : \Omega \rightarrow E_{\mathcal{G}}$ mapeia Ω para $E_{\mathcal{G}}$ se:

1. $\chi(r_\lambda(\upsilon(o_1), ..., \upsilon(o_k)) = \psi'(\theta_1, ..., \theta_k)$, se $r_\lambda(o_1, ..., o_k) \in U$, para algum $\psi'(\theta_1, ..., \theta_k) \in \mathcal{I}$ ou $\psi'(\theta_1, ..., \theta_k) \in \mathcal{R}$, segundo a Definição 14;

2. $\chi(r_\lambda(\upsilon(o_1), ..., \upsilon(o_k)) = \psi'(\theta_1, ..., \theta_k)$, se $r_\lambda(o_1, ..., o_k) \in V$, para algum $\psi'(\theta_1, ..., \theta_k) \in \mathcal{I}$ ou $\psi'(\theta_1, ..., \theta_k) \in \mathcal{R}$, segundo a Definição 14;

3. $\beta(u, v) = (n, m)$, se $u \in U$ e $v \in V$ mapeados por υ e χ, para algum $n, m \in \mathcal{I} \cup \mathcal{R}$;

O conjunto de relações causais é mapeado para o fragmento de grafo a partir do mapeamento das arestas definido por $\beta : \Omega \rightarrow E_{\mathcal{G}}$. A classificação da variável aleatória ψ' em residente ou de entrada será feita posteriormente durante o análise do modelo pelo algoritmo.

Dado o exemplo da regra 1 e o conjunto $\mathcal{I} \cup \mathcal{R}$ mapeado a partir de $U \cup V$, o mapeamento de Ω para $E_{\mathcal{G}}$ será definido por β(`isSuspended(enterprise)`, `ownsSuspendedEnterprise(person)`) = (`isSuspended(enterprise)`, `ownsSuspendedEnterprise(person)`). Assim, $N_{\mathcal{G}}$ = {`isSuspended(enterprise)`, `ownsSuspendedEnterprise(person)`} e $E_{\mathcal{G}}$ = {(`isSuspended(enterprise)`, `ownsSuspendedEnterprise(person)`)}.

As funções definidas mapeiam elementos da relação de dependência $\kappa = (O, N, U, V, \Omega)$ para elementos da MFrag $\mathcal{F}_i$. Mesmo que os elementos em κ estejam contidos na regra $\rho_j = (E_j, A_j, R_j, L_j, \kappa)$ e os conjuntos definidos na tupla estejam no grupo $g_i = (E_i, A_i, R_i, P_i)$, tal que $\rho_j \in P_i$; não há o mapeamento de todos os elementos do grupo g_i para a MFrag $\mathcal{F}_i$. Existem elementos definidos no grupo que não estão na regra e, consequentemente, não serão mapeados a partir de κ.

Seja E_s um conjunto de Entidades que não estão em P_i, Definição 6. O mapeamento de E_s para a MFrag $\mathcal{F}_i$ será feito a partir da definição de um conjunto O_x estendido de ligação de identificadores, tal que $(o : E) \in O_x$ e $E \in E_s$.

Definição 16 Seja E_s o conjunto de Entidades e R_s o conjunto de relacionamentos definidos tal como na Definição 6.

16a. $O_x = \{(o_1 : E_1), ..., (o_m : E_m)\}$ é um conjunto estendido de ligação de identificadores, tal que o_k é um identificador e $E_k \in E_s$ é uma Entidade, onde $1 \leq k \leq m$.

16b. Seja $r = r(E_1, ..., E_m)$ um relacionamento em R_s, tal que $E_k \in E_s$ e $1 \leq k \leq m$; o relacionamento de identificador $r_\lambda^{'} = r_\lambda(o_1, ..., o_m)$ será obtido a partir da substituição de cada E_k em r pelo identificador o_k definido em $(o_k : E_k) \in O_x$.

16c. $R_\lambda^{'}$ é o conjunto de relacionamentos de identificadores, tal que $r_\lambda^{'} \in R_\lambda^{'}$.

O mapeamento de O_x será feito pela função h, Definição 13. Neste caso, h irá mapear cada elemento de $(o_k : E_k) \in O_x$ para alguma VA de contexto $\texttt{isA}(\theta_k : E_k) \in \mathcal{C}$, tal que $(\theta_k : \varepsilon_k) \in \mathcal{B}^{'}$ e $1 \leq k \leq m$. $\mathcal{B}^{'}$ será um novo conjunto de ligação em $\mathcal{F}_\rangle$, onde cada identificador de entidade ε_i utiliza como rótulo o tipo entidade definido a partir de E_k.

Sejam os relacionamentos de identificadores $r_\lambda(o_1, ..., o_i), ..., r_\lambda(o_1, ..., o_j)$ formados a partir dos relacionamentos $r(E_1, ..., E_i), ..., r(E_1, ..., E_j)$ em R_s, de acordo com a Definição 16b. O mapeamento do conjunto formado a partir desses relacionamentos de identificadores, dado o mapeamento de O_x e $1 \leq i, j, \leq m$, será feito por meio das funções υ e χ especificadas na Definição 14.

A partir da formalização proposta, é possível fazer algumas afirmações sobre a classificação das variáveis aleatórias definidas em $\mathcal{F}_i$. A Preposição 1 segue das Definições (1, 14e e 15). Ela afirma que, dado o mapeamento de uma variável aleatória $\psi^{'}$ a partir de um relacionamento de identificador r_λ; para toda variável aleatória de entrada $\psi^{'}$ mapeada como VA de entrada em $\in \mathcal{I}$, r_λ será causa em $U^{'}$.

Proposição 1. Seja $\chi(r_\lambda(\upsilon(o_1), ..., \upsilon(o_k))) = \psi^{'}(\theta_1, ..., \theta_k)$ e $U^{'} \subseteq U$; $\forall \psi^{'}(\theta_1, ..., \theta_k) \in \mathcal{I}, r_\lambda(o_1, ..., o_k) \in U^{'}$.

Prova 1.

1. Seja $\mathcal{G} = \{N_\mathcal{G}, E_\mathcal{G}\}$ um fragmento de grafo acíclico dirigido, tal que $N_\mathcal{G} = \mathcal{I} \cup \mathcal{R}$ e $E_\mathcal{G} = \{(n_1, m_1), ..., (n_k, m_k)\}$, onde $n_i, m_i \in N_\mathcal{G}$ e $1 \leq i \leq k$;
2. Dado que $\psi^{'} = \psi^{'}(\theta_1, ..., \theta_k)$, $\psi^{'} = \psi_I$, se $\psi^{'} \in \mathcal{I}$ ou $\psi^{'} = \psi$, se $\psi^{'} \in \mathcal{R}$;
3. Seja n_i nó pai de m_i, n_j será raiz em $\mathcal{G}$ se $n_j \neq m_i$, $1 \leq i, j \leq k$;
4. Seja n_j raiz, $n_j = \psi_I$;
5. Seja $\Omega = \{(u_1, v_1), ..., (u_k, v_k)\}$, tal que $u_i \in U$ e $v_i \in V$, onde $1 \leq i \leq k$;
6. $\beta(u_j, v_i) = (n_j, m_i)$, tal que $u_j \in U^{'}$;
7. $\beta(\chi(r_\lambda(\upsilon(o_1), ..., \upsilon(o_k))), v_i) = (\psi_I, m_i)$, para ψ_I mapeado a partir de $r_\lambda(o_1, ..., o_k)$.

Apesar da afirmação presente na Preposição 1, não é válido afirmar que todas as causas em U serão mapeadas para VA de entrada em $\mathcal{I}$, pois um nó n_i pode ser distinto de um nó n_j raiz.

A Proposição 2 segue da Preposição 1 e das Definições (14 e 16c). Ela afirma que, para todo relacionamento de identificador $r_\lambda^{'}$, $r_\lambda^{'}$ será mapeado para alguma VA residente $\psi^{'} \in \mathcal{R}$.

Proposição 2. Seja $\chi(r_\lambda^{'}(\upsilon(o_1), ..., \upsilon(o_k))) = \psi^{'}(\theta_1, ..., \theta_k); \forall r_\lambda^{'}(o_1, ..., o_k) \in R_\lambda^{'}, \psi^{'}(\theta_1, ..., \theta_k) \in \mathcal{R}$.

Prova 2.

1. Dado que, $\forall \psi^{'}(\theta_1, ..., \theta_k) \in \mathcal{I}, r_\lambda(o_1, ..., o_k) \in U^{'}$;
2. Dado que $R_\lambda^{'} \not\subseteq R_\lambda$ e, por isso, $U \not\subseteq R_\lambda^{'}$.
3. Então, $\chi(r_\lambda^{'}(\upsilon(o_1), ..., \upsilon(o_k))) = \psi^{'}(\theta_1, ..., \theta_k)$ tal que $\psi^{'}(\theta_1, ..., \theta_k) \notin \mathcal{I}$.
4. Logo, $\psi^{'}(\theta_1, ..., \theta_k) \in \mathcal{R}$.

A Preposição 3 segue das Definições (1, 14e e 15). Ela afirma que, para todo relacionamento de identificador $r_\lambda \in V$ será mapeado para alguma VA residente $\psi^{'} \in \mathcal{R}$.

Proposição 3. Seja $\chi(r_\lambda(\upsilon(o_1), ..., \upsilon(o_k))) = \psi^{'}(\theta_1, ..., \theta_k); \forall r_\lambda(o_1, ..., o_k) \in V, \psi^{'}(\theta_1, ..., \theta_k) \in \mathcal{R}$.

Prova 3.

1. Seja $\mathcal{G} = \{N_\mathcal{G}, E_\mathcal{G}\}$ um fragmento de grafo acíclico dirigido, tal que $N_\mathcal{G} \subseteq \mathcal{I} \cup \mathcal{R}$ e $E_\mathcal{G} = \{(n_1, m_1), ..., (n_k, m_k)\}$, onde $n_i, m_i \in N_\mathcal{G}$ e $1 \leq i \leq k$;
2. Dado que $\psi^{'} = \psi^{'}(\theta_1, ..., \theta_k)$, $\psi^{'} = \psi_I$, se $\psi^{'} \in \mathcal{I}$ ou $\psi^{'} = \psi$, se $\psi^{'} \in \mathcal{R}$;
3. Seja n_i nó pai de m_i; n_j será raiz em $\mathcal{G}$ se $n_j \neq m_i$, $1 \leq i, j \leq k$;
4. Seja n_j raiz, $n_j = \psi_I$;
5. Seja N_r o conjunto de nós raízes, tal que $n_j \in N_r$; então, $N_r = \mathcal{I}$.
6. Seja N_s um conjunto de nós, tal que $N_s \subset N_\mathcal{G}, N_s \cap N_r = \oslash$;
7. Dado que, $\mathcal{I} \cap \mathcal{R} = \oslash$; então $N_s = \mathcal{R}$;
8. Seja $\beta(u_i, v_i) = (n_i, m_i)$, tal que $u_i \in U$ e $v_i \in V$, onde $1 \leq i \leq k$;
9. $n_i \in N_\mathcal{G}$ e $m_i \in N_s$.

Seja um relacionamento de identificador $r_\lambda \in U$; r_λ pode ser mapeado tanto para VA residente $\psi' \in \mathcal{R}$ como para VA de entrada $\psi^{'} \in \mathcal{I}$. Por meio das proposições, ainda não é possível classificar ψ'. Entretanto, sabe-se que se $\psi^{'} \in I$ em $\mathcal{F}$, existe uma VA residente $\psi^{'} = \psi$ em $\mathcal{F}^{'}$. Para isso, é necessário a análise do modelo $\mathcal{M}$.

4.4 Algoritmo para Geração de PO em PR-OWL

Dado o modelo $\mathcal{M}$ e as funções definidas, o objetivo do algoritmo será automatizar o mapeamento e a classificação de todos os elementos do grupo g_i a partir da análise do modelo e a aplicação das proposições apresentadas. Ao final, o conjunto de MFrags composto por F_i deverá formar uma MTheory $\mathcal{T}'$ capaz de ser lida e editada pelo *plug-in* UnBBayes-MEBN.

Dado $g_i = (E_i, A_i, R_i, P_i)$ um grupo, tal que $P_i = \{\rho_1, ..., \rho_n\}$ é um conjunto de regras formado por $\rho_j = (E_j, A_j, R_j, L_j, \kappa_j)$, onde $1 \leq j \leq n$:

1. $R_{\lambda j}$ será o conjunto de relacionamentos de identificadores definido a partir de R_j e o conjunto de ligação de identificadores O_j em κ_j;

2. R'_λ será o conjunto de relacionamentos de identificadores definido a partir de R_s e o conjunto estendido de ligação de identificadores O_x;

3. N_j será um conjunto de fórmulas definido em κ_j e $N_p = \cup_{j=1}^{n} N_j$ será o conjunto de todas as fórmulas em P_i.

Seja $R_{\lambda p} = \cup_{j=1}^{n} R_{\lambda j}$ o conjunto de todos os relacionamentos de identificadores e $O_p = \cup_{j=1}^{n} O_j$ o conjunto de ligação de todos os identificadores em P_i. O conjunto de todos os relacionamentos de identificadores em g_i será $R'_\lambda \cup R_{\lambda p}$ e o conjunto de ligação de todos os identificadores em g_i será $O_x \cup O_p$.

Seja $\Omega_j \subseteq U_j \times V_j$ o conjunto de relações causais em κ_j:

1. $U_p = \cup_{j=1}^{n} U_j$ o conjunto de todas as causas, tal que $U_p \subseteq R_{\lambda p}$;

2. $V_p = \cup_{j=1}^{n} V_j$ o conjunto de todos os efeitos, onde $V_p \subset R_{\lambda p}$;

3. O conjunto de todas as relações causais em P_i será denotado por $\Omega_p \subseteq U_p \times V_p$.

N_p e $O_x \cup O_p$ serão mapeados segundo as Definições 12 e 13 para $\mathcal{C}_i$. O conjunto de todas relações causais Ω_p será mapeado segundo a Definição 15 para o conjunto de arcos $E_{\mathcal{G}i}$, onde $\mathcal{G}_i = (N_{\mathcal{G}i}, E_{\mathcal{G}i})$ será o fragmento de grafo em $\mathcal{F}_i$ e $N_{\mathcal{G}i} = \mathcal{I}_i \cup \mathcal{R}_i$ será o conjunto de todos os nós. O mapeamento do conjunto $R'_\lambda \cup R_{\lambda p}$ será feito por meio de três critérios de seleção.

No **Primeiro Critério de Seleção**, cada relacionamento de identificador em R'_λ será mapeado para alguma VA residente em $\mathcal{R}_i$, segundo a Preposição 2.

No **Segundo Critério de Seleção**, cada relacionamento de identificador em $R_{\lambda p}$ será mapeado para alguma VA residente em $\mathcal{I}_i$ ou de entrada em $\mathcal{R}_i$, segundo as Preposições 1 e 3.

No **Terceiro Critério de Seleção**, cada relacionamento de identificador em $R_{\lambda p}$ que não foi mapeado durante o Segundo Critério será mapeado manualmente para alguma VA residente em $\mathcal{I}_i$ ou de entrada em $\mathcal{R}_i$.

O algoritmo, de modo geral, é definido da seguinte forma:

Dado $\mathcal{M} = \{g_1, ..., g_k\}$ e $\mathcal{T}' = \{\mathcal{F}_1, ..., \mathcal{F}_k\}$. Para cada g_i, $1 \leq i \leq k$:

1: Mapeia N_p e $O_x \cup O_p$ para $\mathcal{C}_i$.
• Cada $(o : E) \in O_x \cup O_p$ será mapeado para algum $\texttt{isA}(\theta : E)$, segundo a Definição 13. • Cada $\mu \in N_p$ será mapeado para algum $\phi \in \mathcal{C}_i$, segundo a Definição 12;

2: Primeiro Critério de Seleção:
• Cada $r'_\lambda(o_1, ..., o_n) \in R'_\lambda$ será mapeado para algum $\psi(\theta_1, ..., \theta_n) \in \mathcal{R}_i$, segundo a Preposição 2.

3: Segundo Critério de Seleção:
• Cada $(u, v) \in \Omega_p$ será mapeado para $E_{\mathcal{G}i}$, tal que:
◦ $r_\lambda(o_1, ..., o_n) \in V_p$ será mapeado para algum $\psi(\theta_1, ..., \theta_n) \in \mathcal{R}_i$, segundo a Preposição 3;
◦ $r_\lambda(o_1, ..., o_n) \in U_p$ será mapeado para algum $\psi_I(\theta_1, ..., \theta_n) \in \mathcal{I}_i$, segundo a Preposição 1, dado que existe algum $\psi(\theta_1, ..., \theta_n) \in \mathcal{R}_j$ mapeado a partir de algum relacionamento de identificador $r'_\lambda(o_1, ..., o_n) \in R'_{\lambda j}$ ou $r_\lambda(o_1, ..., o_n) \in R_{\lambda pj}$ semelhante. $R_{\lambda pj}$ é o conjunto de todos os relacionamentos de identificadores em P_j. $R'_{\lambda j}$ é o conjunto de todos os relacionamentos de identificadores formados a partir de R_s e O_x em g_j. $\mathcal{R}_j$ é o conjunto de VA residentes em $\mathcal{F}_j$, onde $1 \leq j \leq n$ e $i \neq j$.
◦ $r_\lambda(o_1, ..., o_n) \in U_p$ será mapeado para $\psi(\theta_1, ..., \theta_n) \in \mathcal{R}_i$ caso seja o único relacionamento de identificador definido no modelo.

4: Terceiro Critério de Seleção:
• Cada $r_\lambda(o_1, ..., o_n) \in R_{\lambda p}$ será mapeado manualmente para algum $\psi(\theta_1, ..., \theta_n) \in \mathcal{R}_i$ ou algum $\psi_I(\theta_1, ..., \theta_n) \in \mathcal{I}_i$.

Seja $r'_\lambda = r'_\lambda(o_1, ..., o_n)$, tal que $r'_\lambda \in R'_{\lambda j}$, um relacionamento de identificador formado a partir de $r_1 = r_1(E_1, ..., E_n)$, onde $r_1 \in R_s$ em g_j. Seja $r_\lambda = r_\lambda(o_1, ..., o_n)$, tal que $r_\lambda \in R_{\lambda pj}$, um relacionamento de identificador formado a partir de $r_2 = r_2(E_1, ..., E_n)$,

onde $r_2 \in R_p$ em g_j. É dito que, $r_\lambda(o_1, ..., o_n) \in U_p$ em g_i é semelhante a r'_λ ou r_λ em g_j, caso $r_\lambda(o_1, ..., o_n) \in U_p$ seja formado a partir de $r = r(E_1, ..., E_n) \in R_i$, sendo $r = r_1$ ou $r = r_2$, $1 \leq j \leq n$ e $i \neq j$.

Um relacionamento de identificador $r_\lambda = r_\lambda(o_1, ..., o_n)$, tal que $r_\lambda \in V_p$, será único no modelo $\mathcal{M}$, se r_λ não é semelhante a $r'_\lambda \in R'_{\lambda j}$ e a $r_\lambda \in R_{\lambda pj}$, para $i \neq j$.

Durante o mapeamento no Terceiro Critério de Seleção, cada relacionamento de identificador $r_\lambda(o_1, ..., o_n) \in R_{\lambda p}$ será inserido em uma lista de diretivas. Seja $\mathcal{D}_\mathcal{M}$ uma lista de diretivas do modelo definida durante o mapeamento pelo usuário, $\mathcal{D}_\mathcal{M}$ será formado pelo grupo o qual r_λ é definido, r_λ e a VA o qual foi mapeado.

Definição 17 Seja um grupo $g \in \mathcal{M}$, um relacionamento de identificador $r_\lambda = r_\lambda(o_1, ..., o_n)$, onde $r_\lambda \in R_{\lambda p}$, e uma VA residente ou de entrada $\psi'(\theta_1, ..., \theta_n)$. $\mathcal{D}_\mathcal{M}$ é uma lista de diretivas do modelo, tal que $(g, r_\lambda, \psi') \in \mathcal{D}_\mathcal{M}$.

Dado a lista de diretivas do modelo $\mathcal{D}_\mathcal{M}$, o algoritmo será dividido em duas fases:

Fase 1: Caso $\mathcal{D}_\mathcal{M} = \oslash$, o algoritmo passará pelos procedimentos (1, 2, 3). Ao final, caso o algoritmo passe pelo Terceiro Critério de Seleção (procedimento 4), cada tupla (g_i, r_λ, ψ') será inserida em $\mathcal{D}_\mathcal{M}$;

Fase 2: Caso $\mathcal{D}_\mathcal{M} \neq \oslash$, o algoritmo irá executar os procedimentos listados (1, 2), porém adicionará um novo procedimento após o mapeamento de cada $r_\lambda \in V_p$ no Segundo Critério de Seleção.
∘ Dado $(g, r_\lambda, \psi') \in \mathcal{D}_\mathcal{M}$ e g_i, tal que $1 \leq i \leq k$. Cada r_λ será mapeado para $\psi' = \psi$ caso o $\psi' \in \mathcal{R}_i$ ou $\psi' = \psi_I$ caso $\psi_I \in I_i$.

Caso o algoritmo execute a Fase 2 e o usuário mude ou insira elementos no modelo, é possível que o algoritmo passe pelo Terceiro Critério de Seleção (procedimento 4) e adicione novas tuplas em $\mathcal{D}_\mathcal{G}$. O algoritmo de seleção e mapeamento é apresentado na Figura 4.5. Nela, alguns processos são intrínsecos a Fase.

Durante a Fase 2 na Figura 4.5, ao verificar se a lista está vazia ou não, o algoritmo irá executar o "Mapeamento de VA de Contexto"(procedimento 1) e o "Primeiro Critério de Seleção"(procedimento 2) normalmente. A lista de diretivas do modelo $\mathcal{D}_\mathcal{G}$, produzida durante a Fase 1, será utilizada durante o "Segundo Critério de Seleção"(procedimento 3), pois de acordo com a Preposição 1, apenas relacionamentos de identificadores $r_\lambda \in U_p$ podem ser mapeados como VA residente $\psi \in \mathcal{R}_i$ ou de entrada em $\mathcal{I}_i$. Dado o mapeamento feito a partir de $\mathcal{D}_\mathcal{G}$, o algoritmo segue a execução normalmente verificando a existência de relacionamentos de identificadores que não foram mapeados durante o "Terceiro Critério de Seleção"(procedimento 4).

4.5 Implementação do Plug-in no UnBBayes

A implementação do *plug-in* para geração da ontologia probabilística a partir do UMP-ST é dividida em cinco tarefas que serão detalhadas no Anexo I.3, são elas:

- Extensão de classes do *plug-in* UMP-ST para edição das relações de dependência;
- Extensão de classes do *plug-in* MEBN para mapeamento de classes do *plug-in* UMP-ST;
- Implementação da *Graphical User Interface* (interface gráfica do usuário) (GUI) para edição das relações de dependência;
- Implementação do algoritmo de mapeamento e seleção;
- Persistência do modelo no *plug-in* UMP-ST e geração do modelo MEBN/PR-OWL utilizando a *Application Program Interface* (API) PR-OWL 2.

A partir dos pontos de extensão do *plug-in* UMP-ST e do *plug-in* MEBN, são criadas algumas classes. A partir dessas classes, é possível a edição das relacões de dependência. O mapeamento entre os dois modelos é feito por meio de métodos implementados no controlador `MappingController`. A geração da ontologia probabilística a partir do UMP-ST é ativada pelo botão criado no novo painel para edição das relações de dependência.

A persistência em PR-OWL 2 é feita por meio de um arquivo ".ubf"e ".owl"criados a partir da API PR-OWL 2. O formato UBF é utilizado pelo UnBBayes como um arquivo de projeto e está intimamente acoplado a um arquivo PR-OWL 2. Além disso, o modelo criado no *plug-in* UMP-ST, com ou sem a edição das relações de dependência, será salvo em um arquivo no formato XML.

A Figura 4.3 apresenta o diagrama UML com as classes que implementam o mapeamento dado pelo algoritmo. Nele, a classe `UMPSTProject` está no *plug-in* UMP-ST e a classe `MultiEntityBayesianNetwork` está no *plug-in* do PR-OWL 2; as classes `FirstCriterionOfSelection`, `SecondCriterionOfSelection`, `ThirdCriterionOf Selection` e `MappingController` são criadas.

- **UMPSTProject**: encapsula todo o modelo construído no *plug-in* UMP-ST. É composto internamente por mapas contendo objetos relativos aos Objetivos, Hipóteses, Entidades, Atributos, Relacionamentos, Regras e Grupos do modelo;
- **MultiEntityBayesianNetwork**: classe de modelo que representa uma MEBN e trafega por todos os componentes do *plug-in* MEBN;

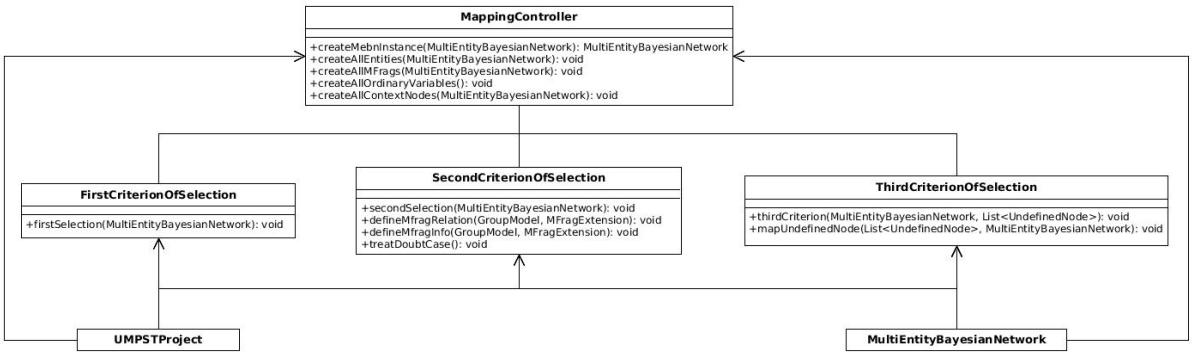

Figura 4.3: Diagrama dos Critérios de Mapeamento

- **FirstCriterionOfSelection:** mapeia para nós residentes relacionamentos de identificadores definidos em apenas um grupo;
- **SecondCriterionOfSelection:** mapeia para nós residentes ou de entrada relacionamentos presentes na relação causal;
- **ThirdCriterionOfSelection:** mapeia para nós residentes ou de entrada relacionamentos que ainda não foram definidos;
- **MappingController:** controladora que gerencia os critérios de seleção e contém métodos para mapeamento de identificadores e fórmulas para nós de contexto.

Dentre as classes apresentadas no diagrama da Figura 4.3, estão os seguintes métodos:

- `createMebnInstance(MultiEntityBayesianNetwork)`: cria uma instância de `MultiEntityBayesianNetwork` para definição da MTheory;
- `createAllEntities(MultiEntityBayesianNetwork)`: mapeia todas as Entidades do modelo UMP-ST para entidades na MTheory;
- `createAllMFrags(MultiEntityBayesianNetwork)`: mapeia todas os grupos do modelo UMP-ST para MFrags na MTheory;
- `createAllOrdinaryVariables(MultiEntityBayesianNetwork)`: mapeia todos os identificadores para nós de contexto `isA` na MTheory;
- `createAllContextNodes(MultiEntityBayesianNetwork)`: mapeia todas as fórmulas para nós de contexto na MTheory;
- `firstSelection(MultiEntityBayesianNetwork)`: busca relacionamentos que foram instanciados apenas uma vez em um grupo. Caso encontre, mapeia para nó residente;

- `secondSelection(MultiEntityBayesianNetwork)`: verifica nos grupos mapeados para MFrags, as regras que possuem relações de dependência;
- `defineMfragRelation(GroupModel, MFragExtension)`: verifica relações causais na relação de dependência e mapeia para nós de entrada relacionamentos que foram mapeados como nó residente em outras MFrags;
- `defineMfragInfo(GroupModel, MFragExtension)`: busca relacionamentos que não são definidos em uma relação de dependência. Caso encontre, mapeia para nó residente;
- `treatDoubtCase()`: mapeia para nó de entrada relacionamentos que não foram mapeados anteriormente, caso existam nós residentes definidos posteriormente a `define MfragRelation(GroupModel, MFragExtension)`;
- `thirdCriterion(MultiEntityBayesianNetwork, List<UndefinedNode>)`: verifica lista de relacionamentos que não foram mapeados (nós em dúvida). Chama método do `MappingController` para que usuário classifique os nós;
- `mapUndefinedNode(List<UndefinedNode>, MultiEntityBayesianNetwork)`: mapeia nós em dúvida de acordo com a opção dada pelo usuário.

Para a edição das relações de dependência, foi criado um novo painel, posterior a implementação dos grupos. A edição é necessária caso o usuário queira gerar a MTheory a partir do UMP-ST. Foram criados quatro painéis, dentre os quais, um que lista todas as regras e dá ao usuário a opção de gerar a MTheory, e os outros que permitem ao usuário editar os identificadores, fórmulas e relações causais. A Figura 4.4 apresenta os painéis implementados para a edição da relação de dependência, os detalhes da implementação da GUI é detalhada no Anexo I.3.

Após a geração da MTheory, é necessário que o engenheiro de ontologias defina a distribuição local relativo a cada nó residente por meio do *plug-in* MEBN. Feito isso, o modelo gerado será capaz de instanciar suas variáveis aleatórias e formar uma Rede Bayesiana de Situação Específica (SSBN).

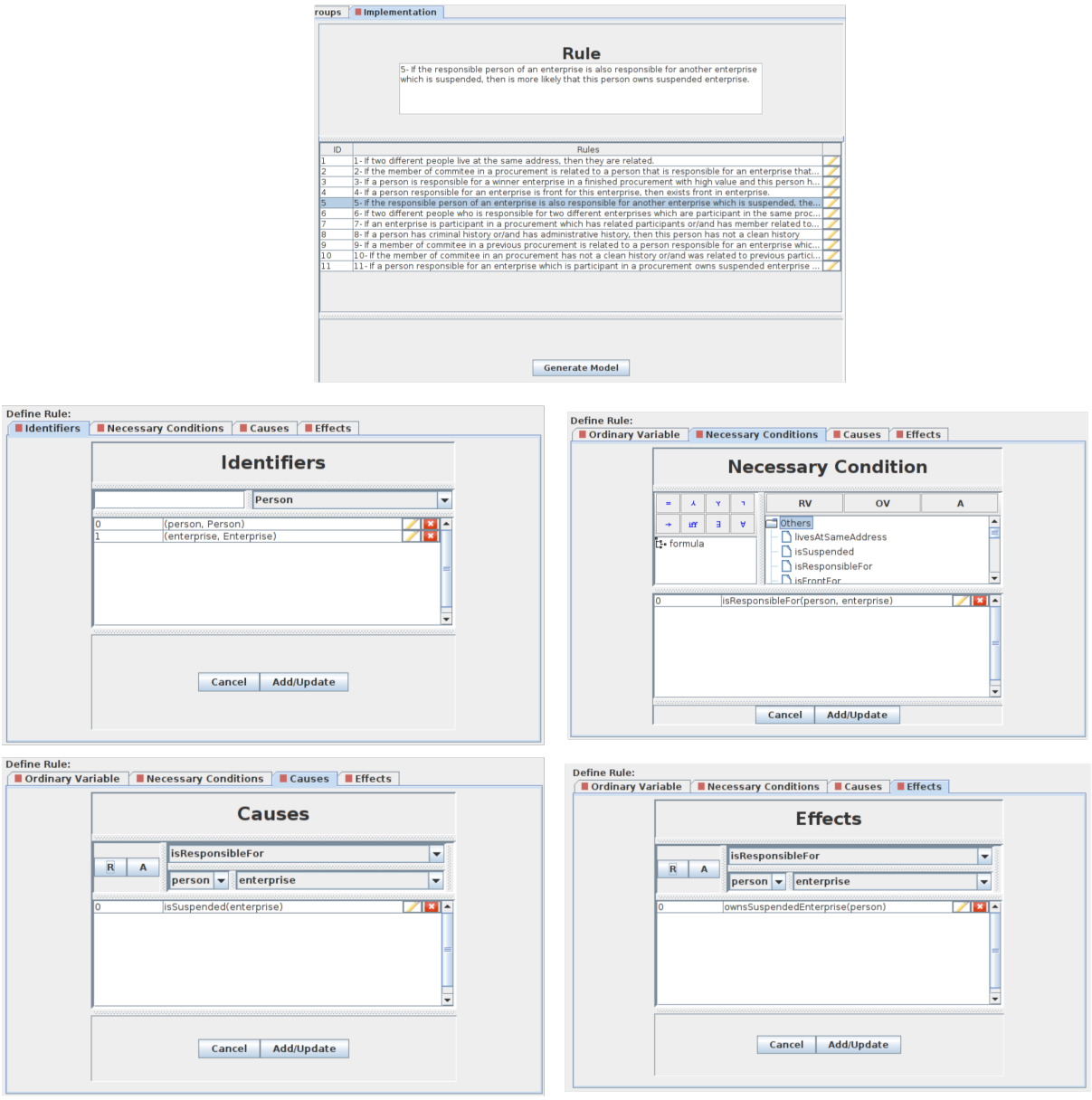

Figura 4.4: Painéis para Edição da Relação de Dependência

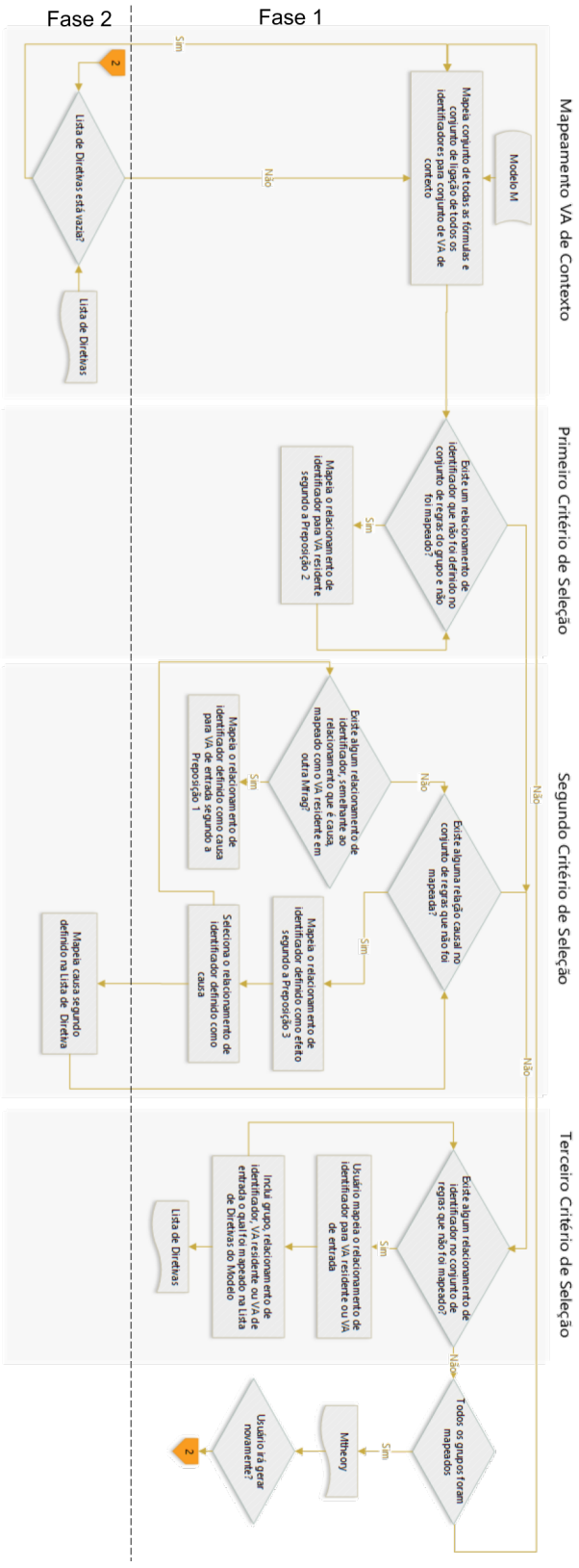

Figura 4.5: Algoritmo de Seleção e Mapeamento

Capítulo 5

Construção de Modelo Utilizando o Domínio de Fraude em Licitações Públicas

A ontologia probabilística de *Fraude em Licitações Públicas* foi proposta por Carvalho [37, 3] e desenvolvida em parceria com a Controladoria Geral da União (CGU). Ela é apresentada nas Figuras 2.6 e 2.7, e será utilizada como estudo de caso para modelagem e geração automática do modelo MEBN/PR-OWL. Neste capítulo, será apresentado parte desse modelo e como é possível, a partir do formalismo proposto no Capítulo 4, gerar de forma automática a ontologia probabilística descrita.

5.1 Descrição do Modelo

A ontologia probabilística utilizada é baseada nos princípios estabelecidos pela Lei brasileira de equidade entre concorrentes. Essa lei proibi a discriminação de potenciais fornecedores por agentes do órgão público. Dessa forma, os agentes também não podem ter um relacionamento com fornecedores que participam da licitação ou dar informações com intuito de beneficiar as empresas concorrentes. O comitê que participa da licitação também deve ser composto por agentes com histórico limpo, isto é, sem condenações administrativas ou por crime. Essas limitações irão determinar os objetivos, as *queries* (perguntas de interesse) e as evidências definidas no modelo. Um das *queries* definida no modelo questiona:

- Is there any relation between members of the committee and the enterprises that participated in previous procurements?

A partir daí Carvalho [3] propõe as seguintes evidências:

> ◦ Look for member and responsible person of an enterprise who are relatives (mother, father, brother, or sister);
>
> ◦ Look for member and responsible person of an enterprise who live at the same address;

Dado esses Requisitos, a próxima disciplina irá identificar os elementos do domínio de aplicação. Em Análise & Design, são definidos as entidades, atributos, relacionamentos, regras e grupos. Segundo Carvalho [3], é visto que:

> "A **Person** has a **name**, a **mother** and a **father** (also **Person**). A **Person** also has an **Education** and **livesAt** a certain **Address**. These entities can be grouped as **Personal Information**. A **PublicServant** is a **Person** who **worksFor** a **PublicAgency**, which is a Government Agency. Every public **Procurement** is owed by a **PublicAgency**, has a committee formed by a group of **PublicServants**, and has a group of **participants**, which are **Enterprises**. The entities just described can be grouped as **Procurement Information**. Every **Enterprise** has at least one **Person** that is **responsible** for its legal acts."
>
> "An **Enterprise** also has an identification number, the General List of Contributors **CGC**, which can be used to inform that this **Enterprise** is suspended from procuring with the public administration, **isSuspended**. These are grouped as the **Enterprise Information**."

Durante a implementação, Carvalho irá definir algumas limitações para o modelo, impostas pelo UnBBayes. Uma delas é considerar que uma entidade `Person` pode trabalhar para uma entidade `PublicAgency`. Dessa forma, o modelo não irá conter `PublicAgency` como entidade. Além disso, será definido na base de conhecimento que todas as entidades `Person` e `Enterprise` serão identificadas pelo seu nome; não serão considerados o `CPF` e o `CGC`. Detalhes da implementação serão vistos em [3].

As regras apresentadas estabelecem padrões vistos no domínio de aplicação e irão definir, posteriormente, as relações de dependências definidas pela formalização. Carvalho, observa que:

1. "If two different people live at the same address, then they are related."

2. "If a member of commitee in a previous procurement is related to a person responsible for an enterprise which is participant in a finished previous procurement, then this member was related to previous participants."

É possível que durante a modelagem, o engeheiro de ontologias não siga a ordem pré-estabelecida pelo POMC, Seção 2.2.2. Por isso, as regras podem ser criadas, implementadas e testadas. Assim, entidades, atributos, relacionamentos e grupos podem ser inseridos no modelo, modificados ou excluídos durante o processo de construção.

A partir do modelo detalhado em [3], é são identificados novos relacionamentos e grupos, além dos que foram vistos. Eles serão utilizados, posteriormente, na formalização e geração da ontologia probabilística descritos nas seções seguintes.

5.2 Definição no Formalismo

Seja um modelo $\mathcal{M} = \{g_1, ..., g_n\}$ definido de acordo com o formalismo proposto no Capítulo 4,

$\mathcal{M} = \{$`ProcurementInformation`, `EnterpriseInformation`, `PersonalInformation`, `RelatedToPreviousParticipants`$\}$

O modelo descrito não abrange todo o domínio definido em *Fraude em Licitações Públicos*. Os grupos definidos em $\mathcal{M}$ serão utilizados na geração de parte da ontologia probabilística proposta por Carvalho [3]. Entretanto, a análise feita nesse estudo de caso poderá ser estendida a todo o domínio de aplicação visto nas Figuras 2.6 e 2.7. A descrição de todo o domínio pelo modelo será especificada como caso de teste para o *plug-in* de geração automática.

Seja um grupo $g_i = (E_i, A_i, R_i, P_i)$ descrito na Definição 6, onde $1 \leq i \leq 4$:

1. $g_1 =$ `ProcurementInformation` $= \{E_1, A_1, R_1, P_1\}$, tal que:

$E_1 = \{$`Person`, `Enterprise`, `Procurement`$\}$,
$A_1 = \oslash$,
$R_1 = \{$`isParticipantIn(Enterprise, Procurement)`,
`isMemberOfCommitee(Person, Procurement)`,
`isProcurementFinished(Procurement)`$\}$,
$P_1 = \oslash$

Seja E_s o conjunto de entidades especificado na Definição 6, O_x o conjunto estendido de ligação de identificadores e R'_λ o conjunto de relacionamentos de identificadores descritos na Definição 16.

(a) Em g_1, O_x e R'_λ serão definidos como:

$O_x = \{$`(person:Person)`, `(procurement:Procurement)`,
`(enterprise:Enterprise)`$\}$, dado $E_s = E_1$;
$R'_\lambda = \{$`isParticipantIn(enterprise, procurement)`,

`isMemberOfCommitee(person, procurement)`,
`isProcurementFinished(procurement)`}

2. $g_2 = \texttt{EnterpriseInformation} = \{E_2, A_2, R_2, P_2\}$, tal que:

$E_2 = \{\texttt{Person}, \texttt{Enterprise}\}$,
$A_2 = \oslash$,
$R_2 = \{\texttt{isResponsibleFor(Person, Enterprise)}\}$
$P_2 = \oslash$

   (a) Em g_2, O_x e R'_λ serão definidos como:

$O_x = \{\texttt{(person:Person)}, \texttt{(enterprise:Enterprise)}\}$, dado $E_s = E_2$;
$R'_\lambda = \{\texttt{isResponsibleFor(person, enterprise)}\}$.

3. $g_3 = \texttt{PersonalInformation} = \{E_3, A_3, R_3, P_3\}$, tal que:

$E_3 = \{\texttt{Person}\}$,
$A_3 = \oslash$,
$R_3 = \{\texttt{livesAtSameAddress(Person, Person)}, \texttt{isRelated(Person, Person)}\}$,
$P_3 = \{\rho_1\}$

ρ_1 é descrito na Regra 1. Seja $\rho_j = (E_j, A_j, R_j, L_j, \kappa_j)$ uma regra e $\kappa_j = (O, N, U, V, \Omega)$ uma relação de dependência, tal como proposto na Definição 5 e 11:

   (a) $\rho_1 = (E_1, A_1, R_1, L_1, \kappa_1)$, tal que:

$E_1 = E_3$,
$A_1 = \oslash$,
$R_1 = R_3$,
$L_1 = \oslash$,
$\kappa_1 = (O, N, U, V, \Omega)$, tal que:

$O = \{$(person1:Person), (person2:Person)$\}$,
$N = \{\neg$(person1, person2)$\}$,
$U = \{$livesAtSameAddress(person1, person2)$\}$,
$V = \{$isRelated(person1, person2)$\}$,
$\Omega = \{($livesAtSameAddress(person1, person2),
isRelated(person1, person2))$\}$

4. $g_4 =$ RelatedToPreviousParticipants $= (E_4, A_4, R_4, P_4)$, tal que:

$E_4 = \{$Person, Enterprise, Procurement$\}$,
$A_4 = \emptyset$,
$R_4 = \{$isParticipantIn(Enterprise, Procurement),
isMemberOfCommitee(Person, Procurement),
isProcurementFinished(Procurement),
isResponsibleFor(Person, Enterprise),
isRelated(Person, Person),
wasRelatedToPreviousParticipants(Person)$\}$,
$P_4 = \{\rho_1\}$

(a) $\rho_1 = (E_1, A_1, R_1, L_1, \kappa_1)$, tal que:

$E_1 = E_4$,
$A_1 = \emptyset$,
$R_1 = R_4$,
$L_1 = \emptyset$,
$\kappa_1 = (O, N, U, V, \Omega)$, tal que:

$O = \{$(member,Person), (person,Person),
(previousProcurement, Procurement),
(enterprise, Enterprise)$\}$,
$N = \{$isMemberOfCommitee(member, previousProcurement),
isResponsibleFor(person, enterprise),
isParticipantIn(enterprise, previousProcurement),
isProcurementFinished(previousProcurement)$\}$,
$U = \{$isRelated(member, person)$\}$,
$V = \{$wasRelatedToPreviousParticipants(member)$\}$,
$\Omega = \{$(isRelated(member, person),
wasRelatedToPreviousParticipants(member))$\}$

5.3 Geração do Modelo

A partir do modelo $\mathcal{M} = \{g_1, g_2, g_3, g_4\}$ e da MTheory $\mathcal{T}' = \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4\}$, a geração do modelo MEBN será feito por meio do algoritmo apresentado em na Seção 4.4. Sabe-se que a lista de diretivas $\mathcal{D}_\mathcal{M} = \oslash$, por isso o algoritmo irá passar pelos procedimentos listados na Fase 1 (Seção 4.4).

Dado o grupo g_1:

1: Mapeia O_x para $\mathcal{C}_1$.
- Cada $(o : E) \in O_x$ será mapeado para algum isA$(\theta : E)$, segundo a Definição 13;

◦ (person:Person) será mapeado para isA(person, Person),
◦ (procurement:Procurement) será mapeado para isA(procurement, Procurement),
◦ (enterprise:Enterprise) será mapeado para isA(enterprise:Enterprise).

2: Primeiro Critério de Seleção:
- Cada $r'_\lambda(o_1, ..., o_n) \in R'_\lambda$ será mapeado para algum $\psi(\theta_1, ..., \theta_n) \in \mathcal{R}_i$, segundo a Preposição 2;

- `isParticipantIn(enterprise, procurement)` será mapeado para `isParticipantIn(enterprise, procurement)`,
- `isMemberOfCommitee(person, procurement)` será mapeado para `isMemberOfCommitee(person, procurement)`,
- `isProcurementFinished(procurement)` será mapeado para `isProcurementFinished(procurement)`.

O algoritmo não irá passar pelos procedimentos 3 (Segundo Critério de Seleção) e 4 (Terceiro Critério de Seleção), pois $P_1 = \oslash$. Após o grupo g_1 ser mapeado, a MFrag $\mathcal{F}_1$ gerada é ilustrada na Figura 5.1.

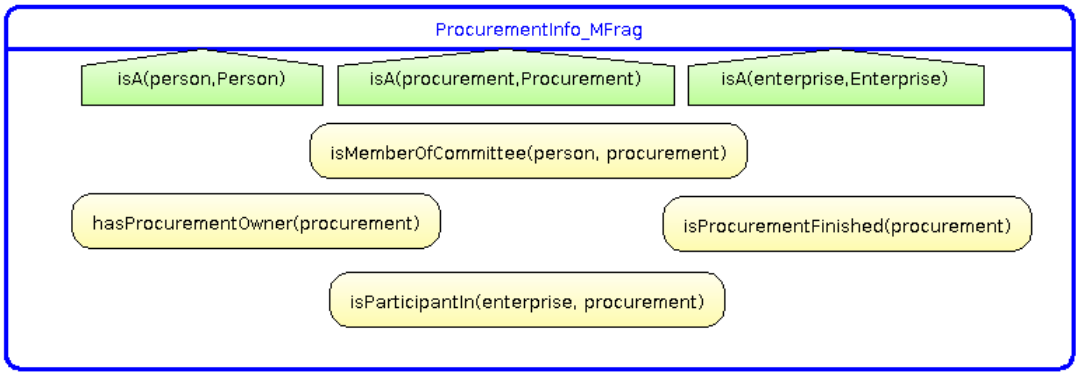

Figura 5.1: MFrag Gerada a partir do grupo ProcurementInfo

Dado o grupo g_2, o algoritmo irá passar pelos mesmos procedimentos descritos para o mapeamento do grupo g_1. Da mesma forma, os procedimentos 3 (Segundo Critério de Seleção) e 4 (Terceiro Critério de Seleção) não serão executados, pois $P_2 = \oslash$. Após o grupo g_2 ser mapeado, a MFrag $\mathcal{F}_2$ gerada é ilustrada na Figura 5.2.

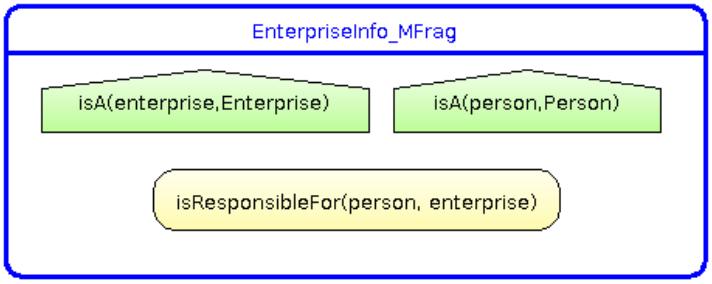

Figura 5.2: MFrag Gerada a partir do grupo EnterpriseInfo

O mapeamento do grupo g_3 será feito pelo algoritmo da seguinte forma:

1: Mapeia N_p e O_p para $\mathcal{C}_3$.

• Cada $(o : E) \in O_p$, onde $O_p = O$ será mapeado para algum `isA`$(\theta : E)$, segundo a Definição 13;

> ◦ `(person1:Person)` será mapeado para `isA(person1, Person)`,
> ◦ `(person2:Person)` será mapeado para `isA(person2, Person)`.

• Cada $\mu \in N_p$ será mapeado para algum $\phi \in \mathcal{C}_3$, segundo a Definição 12;

> ◦ $\neg$`(person1, person2)` será mapeado para $\neg$`(person1, person2)`.

Em g_3, $R'_\lambda = \oslash$, pois não há relacionamentos em R_s, Definição 16. Neste caso, o algoritmo não irá passar pelo Primeiro Critério de Seleção.

3: Segundo Critério de Seleção:

• Cada $(u, v) \in \Omega_p$ será mapeado para $E_{\mathcal{G}3}$, tal que:

◦ $r_\lambda(o_1, ..., o_n) \in V_p$ será mapeado para algum $\psi(\theta_1, ..., \theta_n) \in \mathcal{R}_3$, segundo a Preposição 3;

> ◦ `isRelated(person1, person2)` será mapeado para `isRelated(person1, person2)`.

◦ $r_\lambda(o_1, ..., o_n) \in U_p$ não será mapeado para uma VA de entrada $\psi_I(\theta_1, ..., \theta_n) \in \mathcal{I}_3$, pois não há um relacionamento de identificador semelhante a $r_\lambda(o_1, ..., o_n) \in U_p$ nos grupos g_1, g_2 e g_4;

◦ $r_\lambda(o_1, ..., o_n) \in V_p$ será mapeado para $\psi(\theta_1, ..., \theta_n) \in \mathcal{R}_3$;

> ◦ `livesAtSameAddress(person1, person2)` será mapeado para `livesAtSameAddress(person1, person2)`.

Após o grupo g_3 ser mapeado, a MFrag $\mathcal{F}_3$ gerada é ilustrada na Figura 5.3.

Dado o grupo g_4, o algoritmo irá passar pelo procedimento 1, onde irá mapear cada $(o : E) \in O_p$, tal que $O_p = O$, para alguma VA de contexto `isA`$(\theta : E) \in C_4$ e cada fórmula $\mu \in$

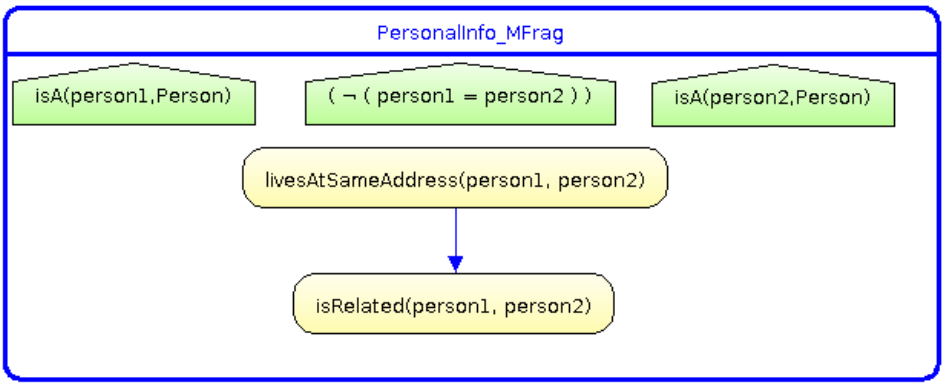

Figura 5.3: MFrag Gerada a partir do grupo PersonalInfo

N_p para alguma fórmula $\phi \in \mathcal{C}_3$. O algoritmo não irá passar pelo procedimento 2 (Primeiro Critério de Seleção), pois $R_s = \oslash$. Em seguida, ele irá mapear os relacionamentos de identificadores no procedimento 3 da seguinte forma:

3: Segundo Critério de Seleção:

• Cada $(u, v) \in \Omega_p$ será mapeado para $E_{\mathcal{G}3}$, tal que:

◦ $r_\lambda(o_1, ..., o_n) \in V_p$ será mapeado para algum $\psi(\theta_1, ..., \theta_n) \in \mathcal{R}_3$, segundo a Proposição 3;

◦ `wasRelatedToPreviousParticipants(member)` será mapeado para `wasRelatedToPreviousParticipants(member)`.

◦ $r_\lambda(o_1, ..., o_n) \in U_p$ será mapeado para uma VA de entrada $\psi_I(\theta_1, ..., \theta_n) \in \mathcal{I}_3$, pois existe um relacionamento de identificador semelhante a $r_\lambda(o_1, ..., o_n) \in U_p$ no grupo g_3;

◦ `isRelated(person1, person2)` será mapeado para `isRelated(person1, person2)`.

Após o mapeamento do grupo g_4, a MFrag $\mathcal{F}_4$ gerada é ilustrada na Figura 5.4.

A definição das regras por meio da formalização proposta é feita por meio de painéis desenvolvidos no *plug-in* para geração do modelo MEBN/PR-OWL. Caso algum grupo contenha regras que não foram definidas de acordo com a nova formalização, o modelo gerado não será capaz de ser instanciado, e consequentemente, responder a *querie* definida durante a disciplina de Requisitos.

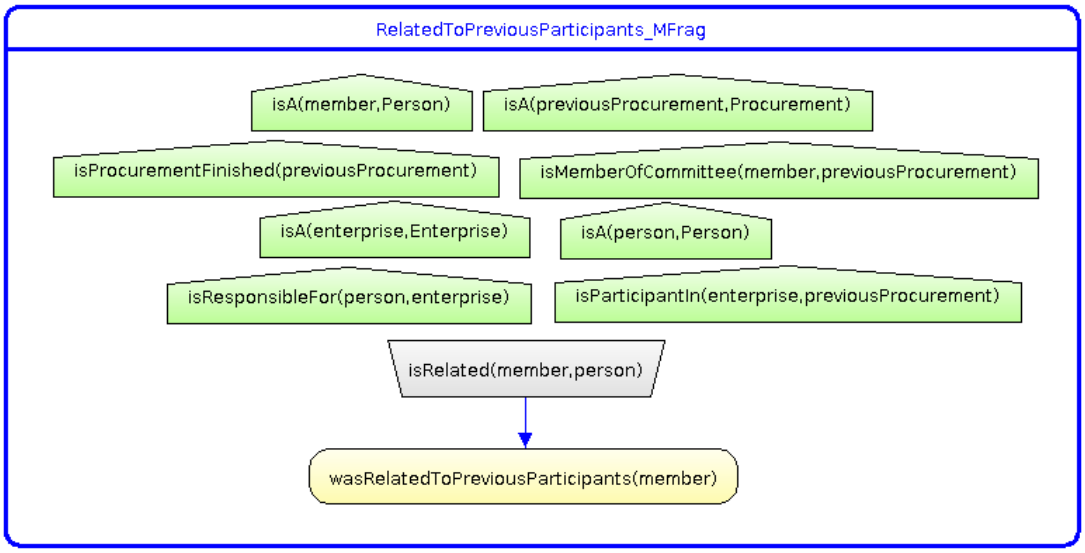

Figura 5.4: MFrag Gerada a partir do grupo Related To Previous Participant

O modelo gerado $\mathcal{T}$ será capaz de ser lido e editado pelo *plug-in* MEBN. Para que o usuário instancie as variáveis aleatórias do modelo e gere uma SSBN, é necessário que seja definido a distribuição probabilística de cada variável aleatória mapeada. Por isso, a geração do modelo PR-OWL/MEBN não substitui a disciplina de Implementação. É possível também que o usuário modifique as MFrags da ontologia probabilística gerada durante a sua edição no *plug-in* MEBN. Caso isso ocorra, será necessário que o usuário modifique o modelo UMP-ST.

Capítulo 6

Conclusões

A formalização das regras e do mapeamento dão ao UMP-ST mecanismos para geração automática de uma ontologia probabilística. Entretanto, a geração de PO em MEBN/PR-OWL não substitui a disciplina de Implementação. O *plug-in* para geração do modelo MEBN/PR-OWL agiliza e facilita a implementação das ontologias em PR-OWL. Isso ocorre, pois não é preciso criar no MEBN os elementos mapeados a partir do UMP-ST.

6.1 Limitações e Trabalhos Futuros

Durante a formalização, não é claro como é feito o mapeamento dos atributos e qual a sua definição nas relações de dependência. No UMP-ST, os atributos denotam um conjunto de características pertencentes a uma Entidade [3, 2]. No MEBN e PR-OWL, os atributos podem ser definidos como variáveis aleatórias ou variáveis ordinárias.

O mapeamento de atributos para o modelo MEBN/PR-OWL é feito manualmente pelo engenheiro de ontologias. É possível, como trabalho futuro, expandir a formalização das regras para a representação dos atributos. Isso pode ser feito a partir da definição de possíveis estados relacionados ao atributo durante a definição de um relacionamento de identificadores mapeado como variável aleatória residente.

O *plug-in* desenvolvido não é capaz de identificar de forma automática as modificações feitas pelo usuário após a geração do modelo MEBN/PR-OWL. Isso pode ser feito por meio da comparação com o modelo atual (modificado após a geração) e o modelo antigo (gerado pelo *plug-in*). Ao verificar as mudanças, o *plug-in* poderia indicar ao engenheiro de ontologias as modificações que devem ser feitas no modelo UMP-ST. Isso facilitaria e agilizaria a atualização do modelo UMP-ST e, consequentemente, a geração de uma nova ontologia.

Referências

[1] Paulo, CG Costa: *Bayesian semantics for the semantic web.* PhD, George Mason University, 2005. 1, 2, 14, 15, 26, 27

[2] Laskey, Kathryn Blackmond: *Mebn: A language for first-order bayesian knowledge bases.* Artificial intelligence, 172(2):140–178, 2008. 1, 3, 7, 10, 11, 12, 26, 27, 60

[3] Carvalho, Rommel Novaes: *Probabilistic Ontology: Representation and Modeling Methodology.* Tese de Doutoramento, George Mason University, 2011. 1, 13, 14, 15, 16, 17, 18, 19, 20, 21, 24, 26, 27, 28, 29, 32, 50, 51, 52, 60

[4] Matsumoto, Shou, Rommel Novaes Carvalho, Marcelo Ladeira, Paulo Cesar G da Costa, Laecio Lima Santos, Danilo Silva, Michael Onishi, Emerson Machado e Ke Cai: *Unbbayes: a java framework for probabilistic models in ai.* Java in Academia and Research, página 34, 2011. 2, 13, 14, 20

[5] Carvalho, Rommel N, Marcelo Ladeira, Rafael Mezzomo De Souza, Shou Matsumoto, Henrique A Da Rocha e Gilson Libório Mendes: *Ump-st plug-in: a tool for documenting, maintaining, and evolving probabilistic ontologies.* Em *Proceedings of the 9th International Conference on Uncertainty Reasoning for the Semantic Web-Volume 1073*, páginas 15–26. CEUR-WS. org, 2013. 2, 8, 17, 18, 20, 21, 24, 25, 28, 29, 32

[6] Da Costa, Paulo Cesar G, 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*, páginas 88–107. Springer, 2008. 2, 14

[7] Costa, Paulo CG e Kathryn B Laskey: *Pr-owl: A framework for probabilistic ontologies.* Frontiers in Artificial Intelligence and Applications, 150:237, 2006. 2

[8] Carvalho, Rommel N, Marcelo Ladeira, Laécio L Santos, Shou Matsumoto e Paulo Cesar G Costa: *A gui tool for plausible reasoning in the semantic web using mebn.* Em *Innovative Applications in Data Mining*, páginas 17–45. Springer, 2009. 2

[9] Carvalho, Rommel N, Marcelo Ladeira, Laécio L Santos, Shou Matsumoto e Paulo CG Costa: *Unbbayes-mebn: Comments on implementing a probabilistic ontology tool.* Em *Proceedings of the IADIS International Conference on Applied Computing*, páginas 211–218, 2008. 2

[10] Costa, Paulo CG, Marcelo Ladeira, Rommel N Carvalho, Kathryn B Laskey, Laécio L Santos e Shou Matsumoto: *A first-order bayesian tool for probabilistic ontologies.* Em *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference*, páginas 631–636, 2008. 2, 20

[11] Da Costa, Paulo Cesar G, Kathryn B Laskey e Kenneth J Laskey: *Probabilistic ontologies for efficient resource sharing in semantic web services*. Em *Proceedings of the Second International Conference on Uncertainty Reasoning for the Semantic Web-Volume 218*, páginas 21–30. CEUR-WS. org, 2006. 2

[12] Chang, KC, Kathryn Blackmond Laskey e Paulo Cesar G da Costa: *Prognos: applying probabilistic ontologies to distributed predictive situation assessment in naval operations*. 2009. 2

[13] Costa, Paulo Cesar G, Kuo Chu Chang, Kathryn B Laskey e Rommel N Carvalho: *A multi-disciplinary approach to high level fusion in predictive situational awareness*. Em *Information Fusion, 2009. FUSION'09. 12th International Conference on*, páginas 248–255. IEEE, 2009. 2

[14] Laskey, Kathryn Blackmond, Paulo CG da Costa, Edward J Wright e Kenneth J Laskey: *Probabilistic ontology for net-centric fusion*. Em *Information Fusion, 2007 10th International Conference on*, páginas 1–8. IEEE, 2007. 2

[15] Laskey, Kathryn Blackmond, Paulo CG Costa e Terry Janssen: *Probabilistic ontologies for knowledge fusion*. Em *Information Fusion, 2008 11th International Conference on*, páginas 1–8. IEEE, 2008. 2

[16] Laskey, Kathryn Blackmond, Paulo CG Costa e Terry Janssen: *Probabilistic ontologies for multi-int fusion*. Ontologies and Semantic Technologies for Intelligence, 213:147, 2010. 2

[17] Carvalho, Rommel, Kathryn Laskey, Laecio Santos, Marcelo Ladeira, Paulo Costa e Shou Matsumoto: *UnBBayes: modeling uncertainty for plausible reasoning in the semantic web*. Citeseer, 2010. 2

[18] *UMP-ST Plugin: Uma ferramenta de modelagem de ontologias probabilísticas para o UnBBayes, author=de Souza, Rafael Mezzomo, year=2011, school=Universidade de Brasília*. Tese de Doutoramento. 2, 28

[19] Pearl, Judea: *Fusion, propagation, and structuring in belief networks*. Artificial intelligence, 29(3):241–288, 1986. 5

[20] Russell, Stuart Jonathan, Peter Norvig, John F Canny, Jitendra M Malik e Douglas D Edwards: *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003. 5

[21] Charniak, Eugene: *Bayesian networks without tears.* AI magazine, 12(4):50, 1991. 6

[22] Matsumoto, Shou: *Um Framework Baseado em Plug-ins para Raciocínio em Ontologias PR-OWL 2*. Tese de Doutoramento, Universidade de Brasília, 2011. 7, 11, 12, 14, 21, 65

[23] Costa, Paulo CG e Kathryn B Laskey: *Multi-entity bayesian networks without multi-tears*. 2006. 7, 25

[24] Laskey, Kathryn Blackmond e Paulo da Costa: *Of starships and klingons: Bayesian logic for the 23rd century*. arXiv preprint arXiv:1207.1354, 2012. 11, 12

[25] Mahoney, Suzanne M e Kathryn Blackmond Laskey: *Constructing situation specific belief networks*. Em *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, páginas 370–378. Morgan Kaufmann Publishers Inc., 1998. 12

[26] Tao, Jia, Zhao Wen, Wang Hanpin e Wang Lifu: *Prdls: a new kind of probabilistic description logics about belief*. Em *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, páginas 644–654. Springer, 2007. 12

[27] Gomez-Perez, Asuncion, Mariano Fernández-López e Oscar Corcho: *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science & Business Media, 2006. 13

[28] Berners-Lee, Tim, James Hendler, Ora Lassila *et al.*: *The semantic web*. Scientific american, 284(5):28–37, 2001. 13

[29] Tanenbaum, Andrew S *et al.*: *Computer networks, 4-th edition*. ed: Prentice Hall, 2003. 13

[30] Yang, Yi e Jacques Calmet: *Ontobayes: An ontology-driven uncertainty model*. Em *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 1, páginas 457–463. IEEE, 2005. 14

[31] Ding, Zhongli, Yun Peng e Rong Pan: *Bayesowl: Uncertainty modeling in semantic web ontologies*. Em *Soft Computing in Ontologies and Semantic Web*, páginas 3–29. Springer, 2006. 14

[32] Koller, Daphne, Alon Levy e Avi Pfeffer: *P-classic: a tractable probablistic description logic*. AAAI/IAAI, 1997:390–397, 1997. 14

[33] Lukasiewicz, Thomas: *Expressive probabilistic description logics*. Artificial Intelligence, 172(6):852–883, 2008. 14

[34] Laskey, Kenneth J e Kathryn Blackmond Laskey: *Uncertainty reasoning for the world wide web: Report on the urw3-xg incubator group*. Em *Proceedings of the Fourth International Conference on Uncertainty Reasoning for the Semantic Web-Volume 423*, páginas 108–116. CEUR-WS. org, 2008. 15

[35] Carvalho, Rommel N, Kathryn B Laskey e Paulo CG Costa: *Pr-owl 2.0–bridging the gap to owl semantics*. Em *Uncertainty Reasoning for the Semantic Web II*, páginas 1–18. Springer, 2013. 15

[36] Carvalho, Rommel N, Kathryn Blackmond Laskey e Paulo Cesar G da Costa: *Compatibility formalization between pr-owl and owl*. Em *UniDL*, 2010. 15

[37] Carvalho, Rommel N, Kathryn B Laskey e Paulo CG Da Costa: *Uncertainty modeling process for semantic technology*. Relatório Técnico, PeerJ Preprints, 2016. 16, 17, 18, 19, 20, 25, 32, 50

[38] Jacobson, Ivar, Grady Booch, James Rumbaugh, James Rumbaugh e Grady Booch: *The unified software development process*, volume 1. Addison-wesley Reading, 1999. 16

[39] Kruchten, Philippe: *The rational unified process: an introduction*. Addison-Wesley Professional, 2004. 16

[40] Balduino, Ricardo: *Introduction to openup (open unified process)*. Eclipse site, 2007. 16

[41] Wazlawick, Raul Sidnei: *Engenharia de Software: conceitos e práticas*. 2013. 16

[42] Royce, Winston W: *Managing the development of large software systems*. Em *proceedings of IEEE WESCON*, volume 26, páginas 328–338. Los Angeles, 1970. 19

[43] Cozman, Fabio G e Denis D Mauá: *Specifying probabilistic relational models with description logics*. Proceedings of the XII Encontro Nacional de Inteligência Artificial e Computacional (ENIAC), Natal, RN, Brazil, 2015. 20

[44] Noy, Natalya Fridman, Ray W Fergerson e Mark A Musen: *The knowledge model of protege-2000: Combining interoperability and flexibility*. Em *International Conference on Knowledge Engineering and Knowledge Management*, páginas 17–32. Springer, 2000. 21

[45] Azevedo, Diego M, Marcelo Ladeira, Laécio L Santos e Rommel N Carvalho: *Automatic generation of probabilistic ontologies from ump-st model*. 25

Anexo I

Implementação da Geração Automática no UnBBayes

Este Anexo descreve a implementação do *plug-in* para geração automática de ontologias probabilísticas em MEBN/PR-OWL, apresentado na Seção 4.5. Para isso, é apresentado na Seção I.1 a arquitetura em *plug-ins* do UnBBayes, na Seção I.2, a arquitetura do *plug-in* UMP-ST. Na Seção I.2, será apresentada a arquitetura do *plug-in* para geração automática de ontologias probabilísticas.

I.1 Arquitetura do UnBBayes

A arquitetura do UnBBayes é voltada para *plug-ins* e, por isso, é baseada no *Java Plugin Framework* (JPF)[3]. Ao utilizar o JPF, o UnBBayes permite que novas implementações sejam desenvolvidas de forma independente. Para isso, o UnBBayes é composto por um *plug-in* `core` que contém funcionalidades básicas. Uma delas é a implementação dos algoritmos para inferência em redes bayesianas [22].

O acoplamento do UnBBayes com os *plug-in* é feito por meio pontos de extensão. Os pontos de extensão são métodos bem definidos que pode ser estendidos por outros *plug-ins* para adicionar novas características e funcionalidades, tal como ilustrado na Figura I.1.

Na Figura I.1, o ponto de extensão `Module` do plug-in core é estendido pelo *plug-in* `MEBN`. O *plug-in* deve ser invocado a partir da barra de ferramentas principal do UnBBayes por janelas internas. Além disso, ele deve ser relativamente auto-suficiente [22].

Os pontos de extensão definidos pelo *plug-in* MEBN são:

[3] `http://jpf.sourceforge.net/`

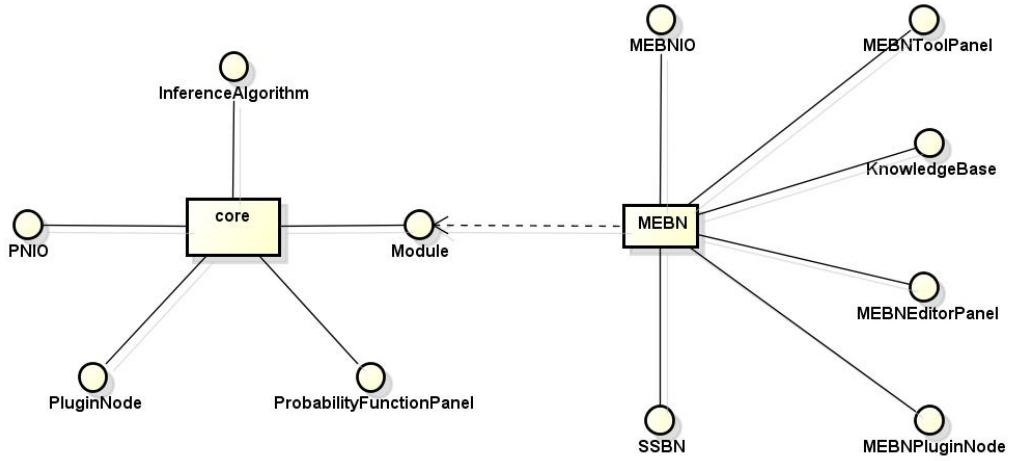

Figura I.1: Pontos de Extensão do plug-in `core` e MEBN no UnBBayes

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

I.2 Arquitetura do Plug-in UMP-ST

O *plug-in* `UMP-ST` é implementado por meio do padrão de projeto MVC (`Model, View, Controller`). Tal como mostra a Figura I.2, o `View` esta sendo representado pelo pacote `umpst.GUI`, o `Controller` pelo pacote `umpst.Controller` e o `Model` pelos pacotes `umpst.IO` e `umpst.Model`.

Dentro de cada pacote, as classes principais utilizadas são:

`umpst.GUI`:

- `UmpstModule`: utilizada para estender um ponto de extensão do UnBBayes;

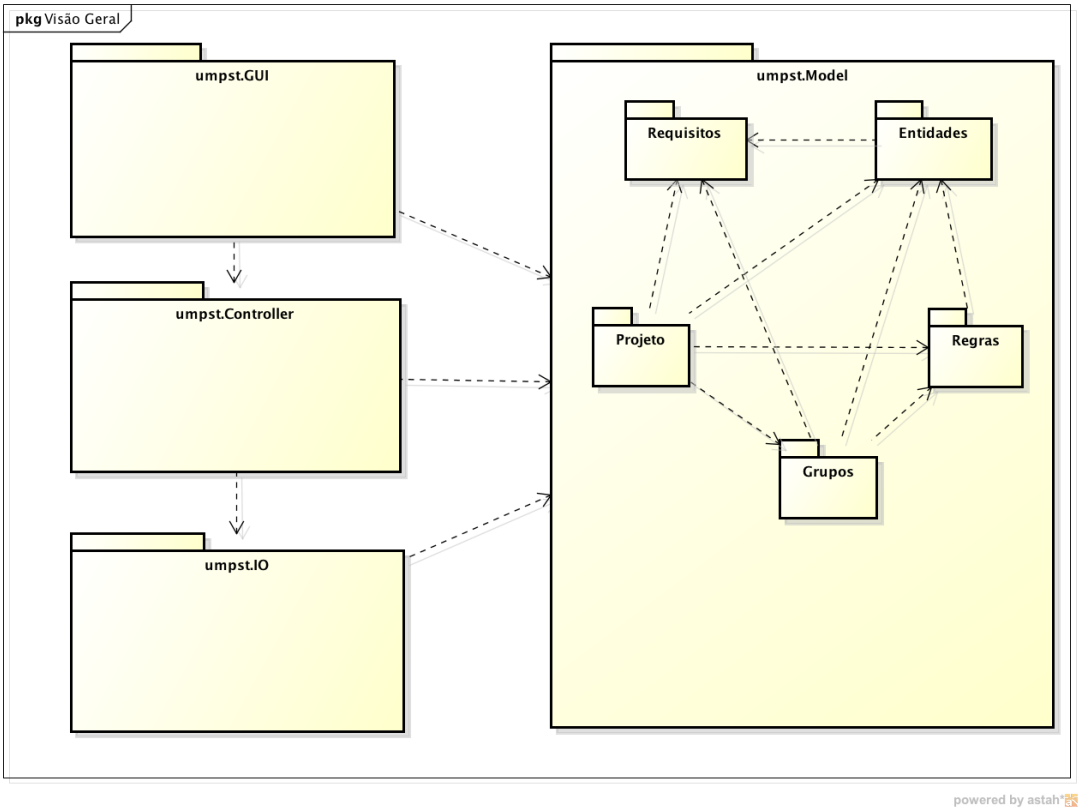

Figura I.2: Visão Geral do plug-in `UMP-ST`

- `MainPanel`: utilizada como classe principal para interface com usuário;
- `IUMPSTPanel`: interface utilizada como ponto de extensão para as classes que implementam as fases do POMC;

`umpst.Controller`:

- `Controller`: utilizada como controlador geral do *plug-in*.

`umpst.Model`:

- `GoalModel:` armazena informações relacionadas aos objetivos;
- `HypothesisModel:` armazena informações relacionadas às hipóteses;
- `EntityModel:` armazena informações relacionadas às entidades;
- `AttributeModel:` armazena informações relacionadas aos atributos das entidades;
- `RelationshipModel:` armazena informações relacionadas aos relacionamentos entre entidades;

`umpst.IO`:

- `FileSaveObject:` salva arquivo como objeto;
- `FileLoadObject:` carrega arquivo como objeto;

I.3 Arquitetura do Plug-in para Geração Automática de ontologias probabilísticas

O *plug-in* implementado segue os padrões de projeto definidos pelo MVC, ao criar uma classe controladora, classes para `View` definidas na GUI e classes para `Model`. O Diagrama ilustrado na Figura I.3 mostra, basicamente, os pontos de extensão entre o as classes da `Model` implementadas no *plug-in* que estendem classes do *plug-in* `MEBN`.

Dentre as classes apresentadas na Figura I.3, as principais classes implementadas são:

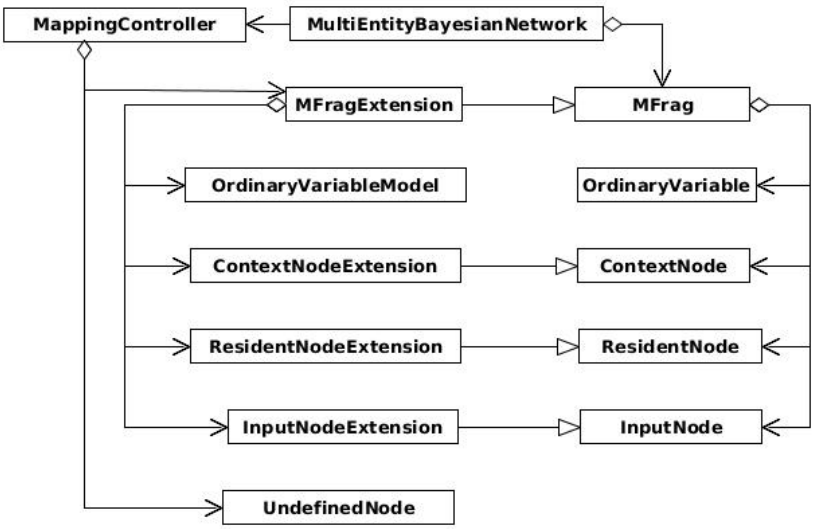

Figura I.3: Diagrama de Classes Extensão plug-in UnBBayesMEBN

- `MappingController:` controladora que gerencia os critérios de seleção e contém métodos para mapeamento de identificadores e fórmulas para nós de contexto.
- As classes `Extension:` estendem as classes do *plug-in* `MEBN`;
- `OrdinaryVariableModel`: implementa os identificadores na formalização do mapeamento e da regra. Internamente, é composto por um objeto da classe `EntityModel` do *plug-in* `UMP-ST`.
- `UndefinedNode`: implementa nós que ainda não foram mapeados (nós em dúvida).

Outras classes implementadas pelo *plug-in* são ilustradas pelo Diagrama na Figura I.4. As classes apresentadas serão utilizadas para implementação das relações de dependência.

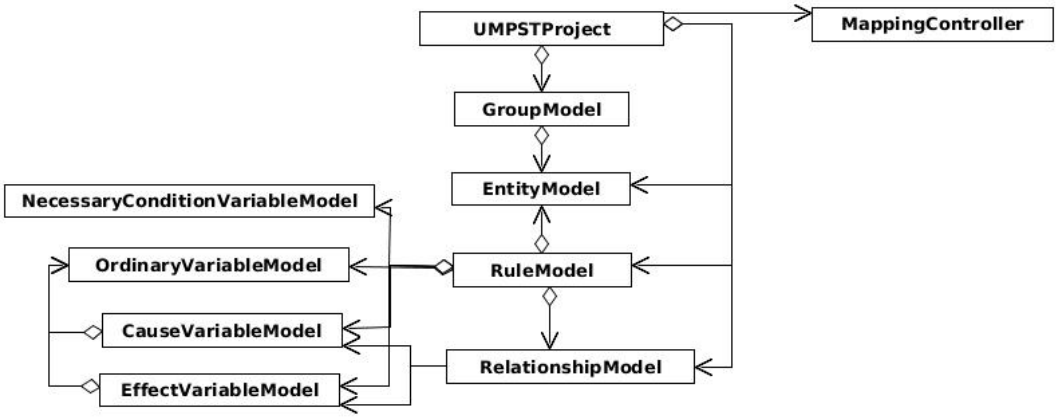

Figura I.4: Diagrama de Classes, Extensão do plug-in UMP-ST

- `CauseVariableModel`: implementa o relacionamento de identificador relativo a causa na relação causal. Internamente, é composto por uma lista de objetos da classe `OrdinaryVariableModel` e um objeto da classe `RelationshipModel`.
- `EffectVariableModel`: implementa o relacionamento de identificador relativo a causa na relação causal. Internamente, é composto por uma lista de objetos da classe `OrdinaryVariableModel` e um objeto da classe `RelationshipModel`.
- `NecessaryConditionVariableModel`: implementa as fórmulas da relação de dependência.

A implementação da GUI é ilustrada na Figura I.5. A GUI é composta pelos painéis para edição das relações de dependência e listagem das regras, além do painel para classificação dos nós em dúvida pelo usuário. A implementação é feita por meio da extensão da interface `IUMPSTPanel`.

Dentre as classes implementadas estão:

- `OrdinaryVariableEditPanel:` responsável pela implementação dos identificadores;
- `NecessaryConditionEditPanel:` responsável pela implementação das fórmulas. As fórmulas serão definidas a partir de um conjunto de operadores implementados de acordo com a Definição 9;
- `CauseVariableEditPanel:` responsável pela implementação das causas;
- `EffectVariableEditPanel:` responsável pela implementação dos efeitos;

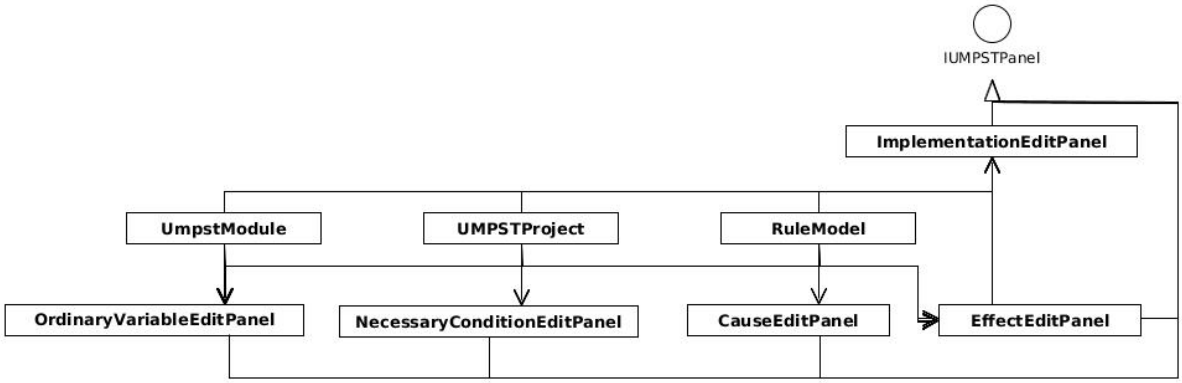

Figura I.5: Diagrama de Classes para GUI da Relação de Dependência

- `ImplementationEditPanel;` classe principal que implementa o painel para edição das outras classes da GUI.

Anexo A

Fichamento de Artigo Científico

Automatic generation of Probabilistic Ontologies from UMP-ST model

Diego M. Azevedo, Marcelo Ladeira, Lacio L. Santos, and Rommel N. Carvalho

Department of Computer Science
University of Braslia
Campus Universitrio Darcy Ribeiro
Braslia, Distrito Federal, Brazil
diegomarques.azevedo@gmail.com,mladeira@unb.br,
laecio@cic.unb.br
http://www.cic.unb.br
Department of Strategic Information
Brazilian Office of the Comptroller General
SAS, Quadra 01, Bloco A, Edifcio Darcy Ribeiro
Braslia, Distrito Federal, Brazil
rommel.carvalho@cgu.gov.br
http://www.cgu.gov.br

Abstract. The Uncertainty Reasoning Process for Semantic Technologies (URP-ST) is a general methodology based on the Unified Process (UP) to guide the modeler in how to design Probabilistic Ontologies (POs). This is a methodology to deal with the difficulty and little guidance for representing uncertainty in ontologies. The Uncertainty Modeling Process for Semantic Technologies (UMP-ST) is an incremental and iterative approach that covers the modeling step related to the URP-ST. It is a general methodology for the majority of the existing semantic technologies which support uncertainty. One of them is the Probabilistic OWL (PR-OWL), which is a language for representing Multi-Entity Bayesian Networks (MEBN). The modeling of a PO using UMP-ST methodology and MEBN/PR-OWL representation is supported by UnBBayes, a framework for building probabilistic graphical models and performing plausible reasoning. Although there is a guidance described by UMP-ST to model a PO, the implementation of a PO is painful and repetitive. Nowadays, the user needs to build the ontology from the zero in a specific technology, even if the user models the PO in UMP-ST. A proper integration that helps the user to implement the PO such as an intermediate structure makes implementation easier than build the PO from the zero and speeds up the modeling process. This paper presents an automatic way to generate POs using MEBN representation from the UMP-ST model by mapping the elements of both sides. This is basically an extension of the UMP-ST to generate POs to an specific formalism and it is developed as a plug-in extension to the UMP-ST plug-in which is supported by UnBBayes. *abstract* environment.

Keywords: Automatic Generation, Probabilistic Ontology, Uncertainty Modeling Process, UMP-ST, POMC, Fraud Detection, MEBN, UnBBayes.

1 Introduction

The limitation of OWL in representing uncertainty are covered by many approaches based on different formalisms, one of them is the Probabilistic Web Ontology Language (PR-OWL) [2]. PR-OWL is based on Multi-Entity Bayesian Networks (MEBN), a formalism that joins the expressiveness power of First Order Logic to represent a domain and the ability to perform plausible reasoning of Bayesian Networks.

Although probabilistic reasoning is one of the most promising approaches to deal with uncertainty in ontologies, little support was giving because of the difficulty level to execute and replicate probabilistic ontologies. Carvalho [1], focusing on how to design this kind of probabilistic ontologies, proposes a methodology based on the Unified Process (UP) to make the working related to build this models less complex. The Uncertainty Reasoning Process for Semantic Technologies (URP-ST) is a framework that guides this working process though a series of steps as modeling, populating the knowledge base, and making inferences on the model.

The Uncertainty Modeling Process for Semantic Technologies (UMP-ST) is an incremental and iterative approach to model probabilistic ontologies though a set of disciplines. Carvalho defines Probabilistic Ontology Modeling Cycle (POMC) as a project lifecycle composed by four disciplines to handle the uncertainty modeling and reasoning processes. These disciplines are composed by Requirements, Analysis & Design, Implementation, and Test [1]. The URP-ST methodology is supported by UnBBayes [1, 6].

UnBBayes [1] is an open source framework with support to plug-ins that offers support to many methods with different formalisms based on Bayesian Networks (BN) [7]. One of them gives support to UMP-ST modeling process. Basically, the UMP-ST plug-in supports the Requirements and Analysis & Design disciplines. [5]. The Implementation and Test disciplines as well as others steps of the URP-ST, such as populating the knowledge base and making inferences, can be done in UnBBayes using the MEBN/PR-OWL plug-in.

Although UnBBayes gives support to URP-ST, it does not have a link between modeling and implementing disciplines. This makes the developing process more painful because the effort needed to build the same model switching between the UMP-ST and in PR-OWL/MEBN plug-in, without a proper integration. This paper presents in detail how elements defined in UMP-ST are mapped to PR-OWL/MEBN elements and how is made the automatic generation of probabilistic ontology in PR-OWL from the UMP-ST plug-in extension.

This paper is organized as follows: Section 2 presents fundamental knowledge of URP-ST, UMP-ST, PR-OWL and MEBN. Section 3 describes UnBBayes and the plug-ins developed for PR-OWL and UMP-ST. Section 4 defines how it is possible to generate PR-OWL 2 ontologies with UMP-ST models and uses the Fraud Detection proposed by Carvalho [1] as a case study. Section 5 presents some concluding remarks.

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

2 Fundamental Knowledge

This section presents basic concepts related to URP-ST, UMP-ST, PR-OWL/PR-OWL 2 and MEBN, providing an approach about these formalisms and their application in the PO context.

2.1 URP-ST

Uncertainty Reasoning Process for Semantic Technologies was proposed by Carvalho [1] as an approach for modeling a probabilistic ontology and using it for plausible reasoning in applications that use Semantic Technologies. It is divided into three steps: modeling the domain, populating the model with data, and using both the model and the data available, i.e Knowledge base, for reasoning. The model describes how different concepts in our ontology interact under uncertainty by knowing facts that will guide the construction process. Once the model is available, reasoning can be done when the model is populated with data, i.e entities and evidence. Then it is possible make inferences to answer queries from the users.

2.2 UMP-ST

Uncertainty Modeling Process for Semantics Technologies was proposed by Carvalho [1] as an iterative and incremental methodology for modeling POs. It describes the first phase of the URP-ST and it is compounded by four major disciplines: Requirements (functional and non-functional), Analysis & Design, Implementation, and Test. The modeling process described by UMP-ST is also consistent with the bayesian network modeling methodology [9][10].

Like in the UP, all disciplines in each iteration of the UMP-ST cover a set of requirements, building deliverable versions of the model. The idea behind every iteration is to take advantage of what has been learned during the modeling of earlier versions. It comes from discovering new requirements, rules, entities, and relations from previous versions.

Following the steps described by the UMP-ST, it is possible to build a Requirements Traceability Matrix (RTM) defined by traceability of work products of different disciplines. It can describe which requirement item is being implemented in each design object or implementation object. I.e, an entity and relationship defined in Analysis & Design discipline can be linked to goal described in the first discipline. Traceability promotes an easy identification of objects that should be analyzed or modified when implementing changes and also ensures that all requirements are covered.

The interaction process covered by the UMP-ST is depicted in Probabilistic Ontology Modeling Cycle (POMC), illustrated in Figure 1. This cycle defines the major outputs from each discipline and the natural order in which outputs are produced. Although there is a modeling order, it is not necessary to occur in a rigid and sequential way like the waterfall model. The interactions between the disciplines are not restricted to the arrows presented.

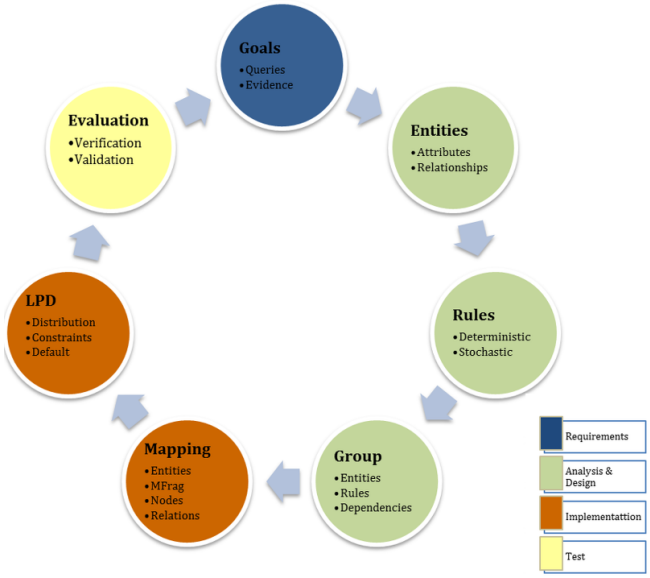

Fig. 1. POMC flow.

The Requirements discipline defines objectives that must be achieved by reasoning in PO. For each goal, it is necessary to define queries that will be answered with the aid of evidence.

Analysis & Design discipline defines classes of entities, their attributes, relationships, and rules. It will be grouped to isolate characteristics of the domain and to facilitate the visualization of dependencies that will guide implementation of the PO model in the next step.

In the Implementation discipline, entities, relationships, attributes, rules, and groups will be mapped to their corresponding concepts in a specific formalism that allows uncertainty representation. In our case study, the mapping is done to PR-OWL 2 format.

In the Test discipline, the user will validate and verify the model, in which it is observed whether the resulting ontology has an expected behavior and whether all requirements have been implemented.

2.3 MEBN and PR-OWL

Multi-Entity Bayesian Networks (MEBN) was proposed by Laskey [4] as a language for representing probabilistic knowledge. It extends Bayesian networks by incorporating the expressive power of first-order logic, and expands first-order logic by adding a way to specify probability distribution through its random variables. Thus, MEBN expands Bayesian network approach to a domain where the number of random variables (RVs) involved is unknown.

MEBN describes knowledge as a collection of MEBN Fragments (MFrags), organized into MEBN Theories (MTheories). Each MFrag represents a repeatable pattern of knowledge that can be instantiated as many times as needed to form a BN addressing a specific situation. Thus, it can be seen as a template for building fragments of a Bayesian network. MFrags are instantiated by binding their arguments to domain entity identifiers to create instances of RVs. Each MEBN fragment has three kinds of RV: context, resident, and input. Context RVs are conditions that guarantee the consistency of the distributions defined in a MFrag. Input RVs influence the distributions of other RVs in a MFrag, but their distribution are defined in their home MFrag. Resident RVs are defined by specifying their Local Probability Distribution (LPD) in their home MFrag. Each LDP of Resident RV is conditioned on the values of instances of its parents.

An MTheory is a set of MFrags that collectively satisfy conditions of consistency ensuring the existence of a unique joint probability distribution over instances of its random variables. The instantiation of its RVs by biding their arguments to domain entity identifiers results in a Situation-Specific Bayesian Network (SSBN), where the RVs of MFrags will become RVs in an ordinary Bayesian network. Then, these SSBNs can use regular BN inference engines to answer the query.

PR-OWL is a language for representing uncertainty knowledge in a principled, structured, and sharable way proposed by Costa [2]. It was developed as an extension of OWL, using MEBN to allow applications to perform plausible reasoning and requiring MEBN inference engine to process additional syntax based on Bayesian network probabilistic inference. This language consists of a set of classes and subclasses that collectively form a framework for building probabilistic ontologies [6].

PR-OWL 2 is an extension of PR-OWL proposed by Carvalho [1], providing a consistent mapping between OWL concepts and data types and PR-OWL. It facilitates the construction of hybrid ontologies with deterministic and probabilistic statements.

3 UnBBayes and the plug-in for UMP-ST

UnBBayes is an open-source Java^{TM} application distributed in GPL license. It has been developed since 2000 and was created for building probabilistic graphical models and performing plausible reasoning [7]. In mid-2010, UnBBayes had its architecture modified to give support to plug-in extension form. This facilitates its expansion into new formalisms. In core version, UnBBayes implements Bayesian network (BN) and Inference Diagrams (ID). Multiply-Sectioned Bayesian Network (MSBN), Object-Oriented Bayesian network (OOBN), Hybrid Bayesian Network (HBN), MEBN, PR-OWL, and PR-OWL 2 as others formalisms are supported by adding plug-ins extension.

MEBN implementation in UnBBayes is saved using the PR-OWL format. It was the first implementation of MEBN formalism in the world. In order to popularize PR-OWL language and MEBN formalism, UnBBayes provides a GUI for

editing ontologies making inference through the creation of SSBNs. PR-OWL 2 was implemented to allow the creation of probabilistic ontologies from deterministic ontologies in OWL. PR-OWL 2 plug-in was developed to allow the user to edit both probabilistic and deterministic parts of ontologies. It is possible because Proteg has been integrated into UnBBayes to design the deterministic part.

UMP-ST plug-in was implemented to solve three problems: the complexity in creating POs; the difficulty in maintenance and evolution of existing POs; and the lack of a centralized tool for documenting POs [5]. One of the key features in this plug-in is the implicitly creation of the traceability matrix by linking the objects created during the disciplines to the requirements they relate to. The plug-in allows the user to work with the Requirements and Analysis & Design disciplines. However, to allow the automatic generation of a PO from the UMP-ST model, it is necessary select a specific language and formalism that will be mapped in the Implementation discipline. MEBN and PR-OWL 2 were chosen because UnBBayes has support to both formalisms as well a plug-in for UMP-ST.

4 Automatic generation of probabilistic ontologies from UMP-ST models

After the modeler defines the elements in the Analysis & Design discipline, it is necessary to choose a specific language that allows uncertainty representation in semantic technologies to implement the model designed. This work describes how to generate a PO in PR-OWL 2 using UnBBayes from the model designed in UMP-ST plug-in. One of the main advantages is the traceability between the elements defined in the Requirements and Analysis & Design disciplines and the PR-OWL elements in the Implementation discipline.

To automatic generate a model in PR-OWL 2, it is necessary to start by mapping the entities, their attributes, and relations to the chosen language. Figure 2 presents how the elements created in UMP-ST are mapped to PR-OWL 2, which essentially uses MEBN terms [1]. The designed model needs to be simplified, since UnBBayes has some limitations. E.g., the lack of support for a type hierarchy.

As represented in Figure 2, the elements of UMP-ST (in blue) at the left side are mapped to MEBN terms present at the right side by considering the limitations of UnBBayes. Thus, the entities in the model will be mapped to an entity in PR-OWL 2. The attributes and relationships of the model are characteristics that may be uncertain. In MEBN, it is represented by defining random variables (RVs), which can be a kind of context RV (in green), input RV (in gray) or resident RV (in yellow).

To define a RV in UnBBayes PR-OWL, first we need to define its home MFrag. However, it is not necessary with the PR-OWL 2 plug-in, because RVs are independent of MFrag and are defined globally. This makes the RVs definition easier compared with UnBBayes PR-OWL definition process [1].

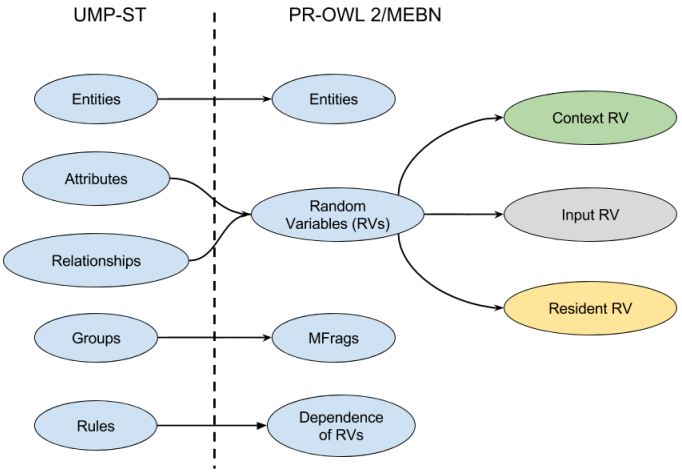

Fig. 2. Mapping of UMP-ST elements to PR-OWL 2.

In UMP-ST, the attributes and relationships as others elements defined in the Analysis & Design discipline are separated in different groups. Also in MEBN, each RV has a home MFrag into which it is defined the local probability distribution. So on, in Figure 2 the groups (on the left side) will be mapped as MFrags (on the right side).

As described in Figure 3, relationships in UMP-ST (on top) can be defined as a relation of one or more entities, being a predicate or a function, depending of their context, e.g, `livesAtSameAddress(Person, Person)`. In MEBN theories, a RV (in yellow) takes arguments that refer to entities of the domain of application [4]. These arguments in UnBBayes have `OrdinaryVariable` type and they are represented by a Context RV (in green) with Boolean range as the built-in RV `isA(resource, class)`. `resource`, in this type of RV, represents individuals of `OrdinaryVariable` and `class` represents entities related to the domain of application [8, 1].

Relationships defined in the UMP-ST will be mapped to (input or resident) RVs (in yellow) and for each entity present in relationship, context RVs (in green) with Boolean range will be created as the built-in RV `isA(resource, class)`. This will be necessary to guarantee the unique joint probability distribution over the RVs when the model is instantiated.

Even though UMP-ST is a generic modeling process for the majority of existing semantic technologies which support uncertainty representation, the chosen language does not offer a formal link between these terms. It is impossible for reasoners to identify that these terms are even linked. At best, it could only "guess" they are the same, since they have similar syntax [1].

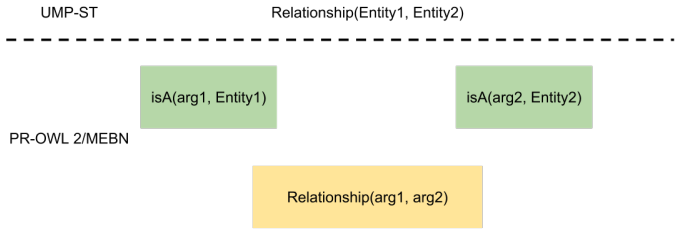

Fig. 3. Mapping a relationship in UMP-ST to a RV in PR-OWL 2.

Once all RVs are mapped, their relations and constraints can be defined by analyzing RVs dependencies described by the rules in the UMP-ST model [3, 1]. It is important to capture local consistence of the model by testing these rules even before implementing the model in a specific language by creating simple probabilistic model to evaluate whether it will behave as expected. However, rules in UMP-ST are basically described in textual form [3], which makes an automatic identification of the rule structure more complexity and subject to errors.

To make the rule structure understandable by computer and less subject to errors, an extension of the rule definition is proposed during the Implementation discipline. It is based on four steps. In the first and the second steps, the rule context (ordinary variables and necessary conditions) is specified. In the third and last steps, a casual relation to describe the dependence of RVs is defined. Figure 4 presents an example to illustrate this form of extension.

Considering the rule and the group description in the Analysis & Design discipline, each rule is related to only one group and each group can have one or more rules. Therefore, the relation of dependence over the RVs described by the rule is restricted to the group in which it is declared. This relation of dependence defined in a specific group will contribute to set the local probability distribution over the RVs by describing who it is cause and who it is effect.

The new rule definition extension allows mapping the ordinary variables and the necessary conditions to context RVs and the cause and the effect to (input or resident) RVs. For instance, in Figure 4 the ordinary variables `person` and `enterprise` are both related to the class `Person` and `Enterprise`. Thus, they are mapped in an MFrag to the `isA(person, Person)` and the `isA(enterprise, Enterprise)` context RVs. The necessary condition can be described by a first-order logic and also can be a RV as `isResponsibleFor(person, enterprise)` which is mapped to a context RV. This condition, in the MEBN fragment, will restrict the application of the probability distribution over its RVs to instances of `Person` that are responsible for the instances of `Enterprise`.

With all elements defined in the Analysis & Design already mapped to the PR-OWL 2 language, now it is necessary to distinguish input and resident RVs. An algorithm to select RVs based on their home MFrag and their relation of dependence is proposed. This algorithm is composed by three criterion of selections.

The **First criteria of selection** defines as resident RV the RV present only in its home MFrag and does not influence a probability distribution of other RVs as a possible input RV. E.g., `ownsSuspendedEnterprise(person)` in Owns_Suspended_Enterprise MFrag illustrated by Figure 5. This is done by the algorithm, by identifying the relationships that there are in just one group.

The **Second criteria of selection** defines as input RV in a MFrag the RV that also is defined as resident RV in another MFrag. E.g, in Figure 5, `isSuspended(enterprise)` has input type in Owns_Suspended_Enterprise MFrag, but first it is defined as resident RV in Enterprise_Information MFrag.

In the Second criteria of selection, the algorithm analyses the rules of each group and identify where the RVs are the cause and where they are the effect. If there is a RV that is cause and in other group it is defined as effect by a rule, then this RV which is cause will have input type and the other which is effect will have resident type. This is possible because in MEBN an input RV influences the local probability distribution of resident RV like a cause and effect relation, but first the RV which has input type needs to be a resident RV in its home MEBN fragment.

After analyzing the causal relation defined by the rules, it is possible to exist some RVs that still have undefined type. This kind of RVs can be identified by analyzing the RVs which are not defined by a rule but they still participate in a MFrag. E.g, in Figure 5 in Enterprise_Information MFrag, the RV `isResponsibleFor(person, enterprise)` is not defined by some rule because in the group related to its MFrag, it is not a cause or effect. In this case, the algorithm considers that this kind of MFrag is the home MFrag of the RV analyzed and it will be defined as resident RV. Nevertheless, if there are RVs that still do not have a type, they will be added in a list of doubts.

The **Third criteria of selection** analyzes whether after first and second criterion of selection may exist some RVs that still have undefined type. So on, if there are RVs in the list of doubts, then each RV in doubt will be presented to the user decides manually whether this RV has input or resident type. The type of RV decided by the user will be added to a list of hypothesis and it will be saved in UMP-ST plug-in file.

Carvalho [1] proposes the probabilistic ontology for procurement fraud and detection and prevention in Brazil. This case study is modeled in PR-OWL 2 with URP-ST and it will be used to illustrate the generation process using the algorithm proposed.

To explain the problem described by Carvalho in the fraud and detection model, it is necessary to have in mind the principles established by Brazil's Law among equality of bidders. This Law prohibits the procurement agent from discrimination among potential suppliers. Therefore, the procurement agent cannot

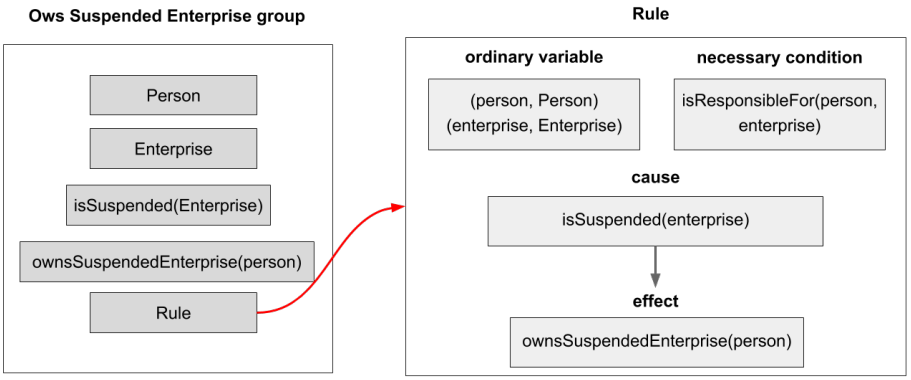

Fig. 4. Group and rule representation.

be related to the bidder or might feed information in order to favor the bidder's enterprise. Also, the committee for a procurement must be well prepared and must has a clean history with no criminal or administrative convictions. These constraints give rise to a set of goals, queries and evidences which will guide the modeling process.

During the Analysis & Design discipline, one of the goals covered by Carvalho identifies whether some person `ownsSuspendedEnterprise`. This relationship is defined as a relation between `Person` and `Enterprise` entities. However, to know whether some person `ownsSuspendedEnterprise`, it is necessary to known also the probability of the enterprise analyzed `isSuspended`. This rule only will be valid whether the person `isResponsibleFor` the enterprise. Figure 4 illustrates this rule in extension form.

This goal can be analyzed and designed in two groups. `isSuspended` and `isResponsibleFor` can be grouped as `Enterprise Information` which also will contain the entities that they relate to. The other group will contain the elements necessary to describe the rule defined before. So on, it needs to have `isResponsibleFor`, `isSuspended` and `ownsSuspendedEnterprise` as relationships and `Person` and `Enterprise` as entities.

After modeling the elements during Analysis & Design discipline, to generate the PO, first each element will be mapped as described in Figure 2. Then, the algorithm will run the sequence of steps presented below to define the RVs type based on rules and groups distribution.

i It is identified whether the model had already been generated by analyzing the list of hypothesis.
ii Case the list of hypothesis is not empty, then each RV listed will have the type defined by the user hypothesis.
iii First criterion of selection is executed and the RVs present in just one MFrag will be defined as resident.

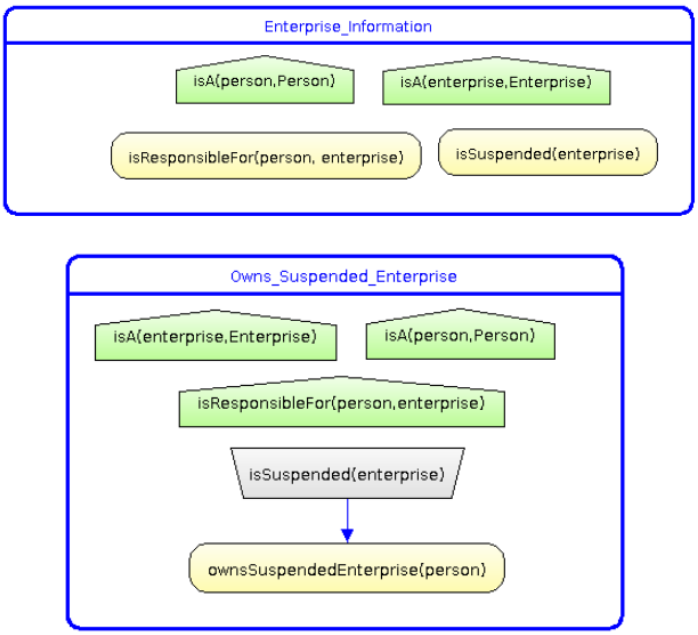

Fig. 5. MFrag generated after mapping elements of Analysis & Design stage.

iv Second criterion of selection is executed. If the RV is defined as cause in a rule and in other group it is defined as effect, then it will have input type in MFrag that it was cause and in the other that it was effect, it will have resident type.

v Case there are RVs that were not declared in any rule, but they were defined in a group, then they will have resident type.

vi Third criterion of selection will define the type of the RVs that still are in doubt by presenting them to the user decides.

The PR-OWL 2 format generated by the algorithm does not define the local probability distribution presents in each resident RVs. Therefore, the model generated is not complete. The user needs to define the local probability distribution by editing the code generated in MEBN plug-in supported by UnBBayes. After that, the PO is validated and verified by analyzing whether the ontology has a expected behavior and whether all requirements have been implemented. Case one requirement has not been implemented, the user will edit the PO in the MEBN plug-in, but if it is necessary to change some rule, then it is possible to generate other PO in PR-OWL 2 format from the UMP-ST plug-in extension.

5 Conclusion

It is possible to generate automatically a PO in PR-OWL 2 format able to be read and edited by UnBBayes. Although there is not a formal link between the UMP-ST elements and the output format, the mapping of elements defined in Analysis & Design discipline to PR-OWL 2 format was based on the related concepts of each element of both representations.

The new definition of the rule during the generation process is necessary to make its format understandable by a computer. This is possible by defining the ordinary variables, the necessary conditions and the causal relation over its RVs. These elements will be mapped to MEBN/ PR-OWL 2 and, by executing the algorithm, the model defined by UMP-ST plug-in will generate a MEBN theory.

The generation of the PR-OWL 2 code by the extension of the UMP-ST plug-in speeds up and facilitates the creation of a MEBN model (PR-OWL 2 PO). The development of this extension is on implementing phase. It is developed in $Java^{TM}$ and riles on Java Plugin Framework[2] (JPF) version 1.5.1.

References

1. R. N. Carvalho, "Probablistic ontology: Representation and modeling methodology," PhD, George Mason University, Fairfax, VA, USA, 2011.
2. P. C. G. Costa, "Bayesian semantics for the semantic web," PhD, George Mason University, Fairfax, VA, USA, Jul. 2005.
3. R. M. de Souza, "UnBBayes plug-in for documenting probabilistic ontologies using UMP-ST," B.S., University of Braslia, Braslia, Brazil, 2011.
4. K. B. Laskey, "MEBN: a language for First-Order bayesian knowledge bases," *Artificial Intelligence*, vol. 172, no. 2-3, pp 140-178, 2008.
5. R. N. Carvalho, M. Ladeira, R. M. de Souza, S. Matsumoto, H. A. D. Rocha, and G. L. Mendes, UMP-ST plug-in: A tool for documenting, maintaining, and evolving probabilistic ontologies. in URSW, ser. CEUR Workshop Proceedings, F. Bobillo, R. N. Carvalho, P. C. G. da Costa, C. dAmato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz, T. Martin, M. Nickles, and M. Pool, Eds., vol. 1073. CEUR-WS.org, 2013, pp. 1526.
6. R. N. Carvalho, K. B. Laskey, P. C. G. da Costa, M. Ladeira, L. L. Santos, and S. Matsumoto, UnBBayes: modeling uncertainty for plausible reasoning in the semantic web, in Semantic Web, gang wu ed. INTECH, Jan. 2010, pp. 128.
7. S. Matsumoto, R. N. Carvalho, M. Ladeira, P. C. G. da Costa, L. L. Santos, D. Silva, M. Onishi, E. Machado, and K. Cai, UnBBayes: a java framework for probabilistic models in AI, in Java in Academia and Research. iConcept Press, 2011.
8. S. Matsumoto, "Framework based in plug-ins for reasoning with probabilistic ontologies," M.Sc., University of Braslia, Braslia, Brazil, 2011.
9. K. B. Laskey and S. M. Mahoney, Network engineering for agile belief network models, IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 4, pp. 487498, 2000. [Online]. Available: `http://portal.acm.org/citation.cfm?id=628073`
10. K. B. Korb and A. E. Nicholson, Bayesian Artificial Intelligence, 1st ed. & Hall/CRC, Sep. 2003. Chapman

[2] JPF homepage: http://jpf.sourceforge.net/