Rodrigo Bellizia Polastro

Lógica Probabilística Baseada em Redes Bayesianas Relacionais com Inferência em Primeira Ordem

São Paulo
2012

Rodrigo Bellizia Polastro

Lógica Probabilística Baseada em Redes Bayesianas Relacionais com Inferência em Primeira Ordem

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia Mecânica.

São Paulo
2012

Rodrigo Bellizia Polastro

Lógica Probabilística Baseada em Redes Bayesianas Relacionais com Inferência em Primeira Ordem

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia Mecânica.

Área de concentração: Engenharia de Controle e Automação Mecânica

Orientador:
Prof. Dr. Fabio Gagliardi Cozman

São Paulo
2012

Agradecimentos

Agradeço a todos que direta ou indiretamente contribuíram para a conclusão desse trabalho, especialmente:

- Aos meus pais pelo incentivo, suporte e minha formação.
- Aos meus amigos Paulo, Fausto, Mari e meu primo Fernando por todo o apoio e conselhos em todos esses anos.
- Ao professor Fabio Cozman pela oportunidade de pesquisa, orientação, compreensão e paciência durante todos esses anos.
- Aos amigos Guga, Bruna, minha prima Daniele, Rodrigo, e muitos outros, por todos os fins de semana "perdidos" de trabalho e pelas extensas madrugadas de conversa.
- Ao pessoal da Poli: Erick, Fabio, Vitor, Sandro, Ronny, Marco, outro Marco, e todos os outros que são muito extensos para citar aqui, por todos os churrascos no estacionamento, pizzas, gargalhadas e jogos durante essa longa jornada.
- Aos membros do Laboratório de Tomada de Decisão, particularmente ao Denis e ao Victor que iniciaram esse projeto junto comigo, ao Daniel, Ricardo e José pelas sugestões, e aos que vieram depois, atuais membros do laboratório.
- Aos professores membros da banca.
- A Deus por todos os dias.
- À FAPESP pela bolsa de Doutorado Direto que permitiu a conclusão desse projeto.
- À empresa Hewlett-Pakkard (HP) pela concessão da bolsa de Mestrado que permitiu o inicio desse projeto.
- E por ultimo, mas não menos importante, a você leitor pelo interesse e pela coragem de se aventurar por essas páginas...

"Quem deseja um dia aprender a voar, deve primeiro aprender a se levantar
e caminhar, correr, saltar e dançar;
Não se aprende a voar voando."
– *Friedrich Nietzsche*

Resumo

Este trabalho apresenta três principais contribuições: i. a proposta de uma nova lógica de descrição probabilística; ii. um novo algoritmo de inferência em primeira ordem a ser utilizado em terminologias representadas nessa lógica; e iii. aplicações práticas em problemas reais.

A lógica aqui proposta, CR$\mathcal{ALC}$ ("credal" $\mathcal{ALC}$), adiciona inclusões probabilísticas na popular lógica $\mathcal{ALC}$ combinando as terminologias com condições de aciclicidade, de Markov, e adotando uma semântica baseada em interpretações.

Como os métodos de inferência exata tradicionalmente apresentam problemas de escalabilidade devido à presença de quantificadores (restrições universal e existencial), apresentamos um algoritmo de *loopy propagation* em primeira-ordem que se comporta bem para terminologias com domínios não triviais. Uma série de testes foi feita com o algoritmo proposto em comparação com algoritmos tradicionais da literatura; os resultados apresentados mostram uma clara vantagem em relação aos outros algoritmos.

São apresentadas ainda duas aplicações da lógica e do algoritmo para resolver problemas reais da área de robótica móvel. Embora os problemas tratados sejam relativamente simples, eles constituem a base de muitos outros problemas da área, sendo um passo importante na representação de conhecimento de agentes/robôs autônomos e no raciocínio sobre esse conhecimento.

Abstract

This work presents three major contributions: i. a new probabilistic description logic; ii. a new algorithm for inference in terminologies expressed in this logic; iii. practical applications in real tasks.

The proposed logic, referred to as CR$\mathcal{ALC}$ ("credal" $\mathcal{ALC}$), adds probabilistic inclusions to the popular logic $\mathcal{ALC}$, combining the usual acyclicity and Markov conditions, and adopting interpretation-based semantics.

As exact inference does not seem scalable due to the presence of quantifiers (existential and universal), we present a first-order loopy propagation algorithm that behaves appropriately for non-trivial domain sizes. A series of tests were done comparing the performance of the proposed algorithm against traditional ones; the presented results are favorable to the first-order algorithm.

Two applications in the field of mobile robotics are presented, using the new probabilistic logic and the inference algorithm. Though the problems can be considered simple, they constitute the basis for many other tasks in mobile robotics, being an important step in knowledge representation and in reasoning about it.

Lista de Figuras

2.1 Exemplo de rede Bayesiana proposto por Charniak (1991). . . . . p. 19

2.2 Estrutura de rede Bayesiana (a) e rede Bayesiana relacional (b). . . . . p. 23

3.1 Grafo com a rede gerada por inclusões probabilísticas, e a instanciação para um domínio com 2 indivíduos. . . . . p. 33

3.2 Redes geradas pela definição e inclusão de conceitos. . . . . p. 33

3.3 Tabelas de probabilidades condicionais geradas pelos construtores lógicos. p. 33

3.4 Representação de restrições universais e existenciais instanciadas para um domínio com dois indivíduos. . . . . p. 34

3.5 Grafo para a terminologia $\mathcal{T}_1$ do Exemplo 3 e a instanciação do mesmo para o domínio $\mathcal{D} = \{a, b\}$. . . . . p. 35

4.1 Rede multiconexa e poliárvore equivalente obtida por agrupamento de variáveis. . . . . p. 38

4.2 Versão shattered da terminologia $\mathcal{T}_1$ para a consulta $P(C\,(a_0))$. . . . . p. 40

4.3 Fatias de uma rede instanciada para um domínio com 3 indivíduos. . . p. 42

4.4 Inferência exata é utilizada dentro de fatia e os resultados são propagados para as demais. . . . . p. 43

5.1 Interface gráfica do pacote de software. . . . . p. 46

5.2 Resultados na execução em modo de linha de comando. . . . . p. 46

5.3 Especificação da terminologia. . . . . p. 48

5.4 Consulta, evidência e resultado. . . . . p. 49

7.1 Mapas criados para as três áreas identificadas. . . . . p. 72

7.2 Vista tridimensional da área 1. . . . . p. 72

Lista de Tabelas

6.1 Resultado da inferência $P(C\,(0))$ em $\mathcal{T}_1$ para um domínio com N indivíduos. p. 52

6.2 Variação das probabilidade dos quantificadores com a mudança do domínio. p. 53

6.3 Tempo (ms) demandado para realizar a inferência em $\mathcal{T}_1$ para um domínio com N indivíduos. . . . . p. 53

6.4 Tempo (ms) e número de iterações demandado para realizar a inferência em $\mathcal{T}_1$ para um domínio com N indivíduos, com uma observação para cada indivíduo. . . . . p. 54

6.5 Resultado da inferência $P(Parent\,(0))$ na Kangaroo para um domínio com N indivíduos. . . . . p. 55

6.6 Tempo (ms) demandado para realizar a inferência na Kangaroo para um domínio com N indivíduos. . . . . p. 56

6.7 Tempo (ms) e número de iterações demandado para realizar a inferência na Kangaroo para um domínio com N indivíduos, com uma observação para cada indivíduo. . . . . p. 56

6.8 Inferência realizada para 3 indivíduos distintos. . . . . p. 62

7.1 Ambientes identificados para 6 pontos de coleta de dados. . . . . p. 70

7.2 Comparação de tempos (em milissegundos) para criação dos mapas. . . p. 71

Sumário

1 Introdução p. 12

1.1 Objetivos . . . . . p. 13

1.2 Organização do Texto . . . . . p. 14

2 Revisão Bibliográfica e Teoria Básica p. 15

2.1 Probabilidades e Regra de Bayes . . . . . p. 15

2.2 Redes Bayesianas . . . . . p. 18

2.3 Lógica de Primeira Ordem . . . . . p. 23

2.4 Lógicas de Descrição Probabilísticas . . . . . p. 26

3 CR$\mathcal{ALC}$ p. 29

4 Algoritmos de Inferência p. 36

4.1 Inferência Exata . . . . . p. 36

4.2 Inferência Aproximada . . . . . p. 37

4.3 Inferência em Primeira Ordem . . . . . p. 39

5 Implementação p. 45

6 Experimentos p. 51

6.1 Terminologia de Teste . . . . . p. 51

6.2 Ontologia Kangaroo . . . . . p. 54

6.3 Ontologia Wine e Utilização de Nominais . . . . . p. 56

7 Aplicações p. 63

7.1 Extração de Contexto em Robótica Móvel . . . . . p. 63

7.2 Rótulos e Mapeamento Semântico . . . . . p. 67

8 Publicações p. 73

9 Considerações Finais p. 75

Referências Bibliográficas p. 77

Apêndice A – Ontologia Wine. p. 82

Apêndice B – Ontologia utilizada para extração de contexto em robótica móvel. p. 94

Apêndice C – Ontologia utilizada para criação de mapas semânticos. p. 101

1 Introdução

O campo da inteligência artificial *(I.A.)* pode ser considerado como uma ciência relativamente nova. Este termo foi primeiramente usado em 1956, mas apesar de haver muito desenvolvimento desde então, resultados práticos significativos só começaram a aparecer substancialmente a partir das décadas de 80 e 90. I.A. é uma ciência muito abrangente que engloba a percepção e interpretação do ambiente, a atuação no mesmo, a representação de conhecimento, raciocínio e aprendizado. Todas essas tarefas são igualmente importantes para um agente (ou robô) autônomo mas o cerne de um sistema inteligente está nos últimos três itens citados.

Este trabalho propõe uma lógica de descrição probabilística assim como um algoritmo de inferência para a mesma. Essa lógica pode ser usada para representar o conhecimento através de ontologias, e o algoritmo de inferência nos permite fazer o raciocínio e o aprendizado. A não ser quando explicitado de outra forma, o termo inferência corresponderá à inferência probabilística, ou seja, ao cálculo das probabilidades de valores de uma variável. O algoritmo proposto foi desenvolvido a partir de modelos probabilísticos chamados de redes Bayesianas *(BN - Bayesian Networks)*, mais especificamente, de uma extensão denominada rede Bayesiana relacional *(RBN - Relational Bayesian Network)* apresentada por Jaeger (1997), que combina as redes Bayesianas com relações lógicas.

São apresentadas então as três principais contribuições deste trabalho para o campo de inteligência artificial, sendo:

i. uma nova lógica de descrição probabilística (uma linguagem formal para a especificação de modelos lógico-probabilísticos);

ii. um algoritmo de inferência em primeira ordem;

iii. aplicações práticas em problemas reais.

A primeira contribuição tem como intuito facilitar a representação de conhecimento, enquanto que a segunda nos dá uma ferramenta para melhor raciocinar sobre o conheci-

mento representado; a terceira contribuição aborda algumas tarefas elementares que são base para outras tarefas.

Antes de prosseguir com a descrição detalhada dos objetivos deste trabalho, vale resumir brevemente a evolução do trabalho que permitiu a percepção das necessidades aqui tratadas. O tema de pesquisa inicial foi o aprendizado de modelos estatísticos baseados em redes Bayesianas e de Markov e também relações lógicas. Foi um trabalho voltado ao diagnóstico de problemas em sistemas computacionais distribuídos, aplicação de interesse da Hewlett Packard do Brasil, com foco na utilização de Redes Lógicas de Markov *(MLN - Markov Logic Network)*, um dos modelos mais promissores até então. Porém foi constatada uma grande defasagem entre a idealização das MLN e o que de fato era possível se fazer nas aplicações. Isso levou à decisão de desenvolver novos algoritmos de inferência, juntamente com um novo modelo lógico-probabilístico que melhor se adeque a eles. Desta forma foi possível estabelecer os objetivos tratados a seguir.

1.1 Objetivos

O principal objetivo deste trabalho é o desenvolvimento de uma nova lógica probabilística e de um algoritmo de inferência para raciocinar sobre a mesma. Para atingir esse objetivo, foram realizados os seguintes passos:

- Desenvolvimento de uma linguagem formal para a especificação de modelos lógico-probabilísticos.
- Desenvolvimento de um algoritmo de inferência probabilística (CHAVIRA; DARWICHE; JAEGER, 2006).
- Teste dos algoritmos.
- Aplicação em problemas reais.

Os testes do algoritmo de inferência devem ser feitos com dados simulados e com dados reais. A literatura sobre modelos probabilísticos relacionais hoje lista uma série de aplicações importantes, desde interpretação de imagens e linguagem, até o raciocínio em redes sociais e problemas de localização robótica (GETOOR; TASKAR, 2007). Dentre essas, uma aplicação de grande interesse que foi utilizada para os testes do algoritmo é a que aborda o problema de mapeamento e localização de robôs móveis. Uma das atividades em robôs móveis é a determinação da localização, da posição e de atitude a serem tomadas a

partir de informações sensoriais. A estratégia dominante para localização de robôs móveis é hoje a modelagem estatística do robô (incluindo sensores e atuadores) e de seu entorno, usando modelos frequentemente baseados em campos de Markov e redes Bayesianas. Este enfoque probabilístico é usualmente referido como *Probabilistic Robotics* (THRUN; BURGARD; FOX, 2005). Problemas de robótica probabilística se caracterizam pelo interesse em relações entre objetos e locais e pela existência de conhecimento determinístico sobre variáveis. Estas características podem ser abordadas através de modelos probabilísticos relacionais, como já explorado na literatura (LIAO; FOX; KAUTZ, 2005). Estas idéias foram aqui exploradas com o uso de redes Bayesianas relacionais, mais especificamente em problemas de mapeamento, como detalhado nos Capítulos 6 e 7.

1.2 Organização do Texto

O Capítulo 2 apresenta a revisão sobre a literatura relevante a este trabalho, além de apresentar os conceitos básicos e teoria necessária para o entendimento do texto. Nele são apresentados os conceitos referentes à teoria de probabilidades, inferência, redes Bayesianas e redes Bayesianas relacionais, lógicas de primeira ordem, lógicas de descrição e lógicas de descrição probabilísticas.

No Capítulo 3 é definida a lógica de descrição probabilística proposta nesse trabalho, sobre a qual o algoritmo de inferência foi desenvolvido e testado, enquanto que no Capítulo 4 é apresentado o algoritmo propriamente dito.

O Capítulo 5 apresenta a implementação do algoritmo proposto em um pacote de software, assim como a descrição do funcionamento do mesmo.

No Capítulo 6 mostram-se os experimentos realizados para testar os algoritmos, enquanto que no Capítulo 7 apresentam-se exemplos reais onde os algoritmos foram aplicados. O Capítulo 8 contem uma breve descrição dos artigos que foram publicados a partir dos resultados obtidos.

Por fim, no Capítulo 9 estão as considerações finais e notas sobre trabalhos futuros.

2 *Revisão Bibliográfica e Teoria Básica*

Neste capítulo será apresentada a teoria básica de probabilidades (Seção 2.1), redes Bayesianas (Seção 2.2), lógicas de primeira ordem (Seção 2.3) e lógicas probabilísticas relevantes a este texto (Seção 2.4) junto com uma revisão geral da literatura pertinente. O intuito desse capítulo é revisar os assuntos relevantes para o bom entendimento deste trabalho, assim como as referências necessárias para uma consulta mais detalhada.

2.1 Probabilidades e Regra de Bayes

A probabilidade é a representação das chances de que um dado evento ocorra. Existem diferentes definições formais para probabilidades mas independente da definição, os *Axiomas de Kolmogorov* são sempre adotados (DEGROOT, 1986).

Axiomas:

i. Para todo evento A, $P(A) \geq 0$;

ii. $P(\Omega) = 1$;

iii. Para eventos $E_1, E_2, \ldots$ disjuntos, $P(E_1 \cup E_2 \cup \ldots) = \sum_i P(E_i)$.

Denomina-se o conjunto de todos os estados possíveis do mundo como *espaço amostral* (Ω), sendo que subconjuntos do espaço amostral são chamados *eventos*.

Consequências:

– Para dois conjuntos A e B, $P(A) \leq P(B)$ se $A \subseteq B$;

– $P(\emptyset) = 0$;

– Para todo evento E, $0 \leq P(E) \leq 1$;

– $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

Embora os axiomas estabeleçam propriedades importantes, eles não definem como obter a probabilidade de um dado evento. As definições formais mais aceitas para tal (NEAPOLITAN, 2003) são dadas segundo as teorias a seguir:

Teoria Clássica: A teoria clássica assume que todos os estados de Ω são equiprováveis na falta de informação contrária. Para o caso discreto, define-se a probabilidade de um dado evento como a razão entre o número de elementos no evento (n_{evento}) e o número de elementos no espaço amostral (n_Ω). Considere como exemplo o lançamento de um dado de 6 faces. Deseja-se a probabilidade de que saia um número par.

$\Omega = \{1, 2, 3, 4, 5, 6\}$, e $Par = \{2, 4, 6\}$,

assim, temos que

$P(Par) = \frac{n_{Par}}{n_\Omega} = \frac{3}{6} = 0.5.$

Teoria Frequentista: Considere o caso de uma moeda viciada, ou qualquer variável que não siga uma distribuição uniforme (como por exemplo a variável que indica a altura das pessoas de uma população) no qual a utilização da teoria clássica levaria a um resultado não condizente com a realidade. Em casos como esses, a teoria frequentista se utiliza da *Lei dos Grandes Números* e define a probabilidade de um evento como sendo a frequência relativa de ocorrência do mesmo para um conjunto de n amostras ou repetições, com $n \to \infty$. Usualmente aproxima-se o resultado para um número finito de repetições. Tomando como exemplo o lançamento de uma moeda; a moeda é lançada 100 vezes, observando-se que em 61 dos lançamentos o resultado foi *cara*. Ou seja,

$n = 100$, e $n_{cara} = 61$

assim, temos que para a moeda em questão

$P(cara) = \lim_{n\to\infty} \frac{n_{cara}}{n} \approx \frac{61}{100} = 0.61.$

A teoria frequentista, embora mais flexível que a clássica, está sujeita a uma restrição importante. Se por um lado pode-se facilmente repetir experimentos como o lançamento de moedas ou dados ou ainda pode-se amostrar peças que saem de uma linha de produção para a obtenção de uma freqüência relativa, por outro lado em atividades únicas ou situações que não podem ser replicadas é impossível a obtenção da frequência relativa.

Teoria Subjetivista: A teoria subjetivista define a probabilidade de um evento como sendo um número no intervalo $[0, 1]$ que representa o grau de crença que uma dada pessoa tem sobre o evento em questão, sendo 1 o indicativo da crença que o evento certamente ocorrerá e 0 indica a crença de que o evento seja impossível. Segundo essa teoria, a probabilidade reflete uma opinião e seu valor pode mudar de pessoa para pessoa. Como exemplo, considere as chances de um determinado presidente se reeleger. Obviamente não vale a hipótese de estados equiprováveis do mundo, e como é um acontecimento único não se pode obter uma frequência relativa. Nesse caso, uma pessoa de um certo contexto dirá que $P(reeleito) = 0.75$ enquanto que outra pessoa com outro contexto dirá que $P(reeleito) = 0.43$.

Além dos Axiomas descritos no inicio da seção, existem outros conceitos e propriedades que podem ser definidos como segue:

Probabilidade condicional: Em alguns casos, a ocorrência de um certo evento tem influência direta na probabilidade de outro evento, ou seja, está condicionada a outro evento. Por exemplo, a probabilidade de ter chovido em um local deve mudar se sabemos que o chão está seco ou molhado. Para dois eventos A e B, define-se a probabilidade condicional de A dado B (DEGROOT, 1986) como

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \tag{2.1}$$

com $P(B) > 0$.

Regra de Bayes: A interseção de dois conjuntos é simétrica, ou seja, $A \cap B = B \cap A$ e consequentemente $P(A \cap B) = P(B \cap A)$. Com isso, partindo da definição de probabilidades condicionais, pode-se facilmente se chegar em

$$\begin{aligned} P(A|B) &= \frac{P(B|A)\,P(A)}{P(B)} \\ &= \frac{P(B|A)\,P(A)}{P(B|A)\,P(A) + P(B|A^C)\,P(A^C)} \quad , \end{aligned} \tag{2.2}$$

chamada Regra de Bayes, com A^C sendo o complemento de A.

A regra de Bayes é bastante útil para testar hipóteses condicionadas a evidências, em

casos onde seja mais fácil se obter as probabilidades das evidências dada a hipótese do que a probabilidade da hipótese dadas as evidências.

Independência: Dois eventos A e B são ditos independentes quando o conhecimento de um deles não altera as probabilidades do outro. Ou seja, para A e B independentes, $P(A|B) = P(A)$ para $P(B) > 0$; consequentemente $P(A \cap B) = P(A)\,P(B)$.

O uso de probabilidades condicionais e a representação da independência entre duas variáveis nos permite fazer modelos matemáticos complexos que representem a relação entre diversas variáveis. Dentre esses modelos, um dos mais tradicionais é a rede Bayesiana, que será tratada na seção a seguir.

2.2 Redes Bayesianas

O termo *rede Bayesiana* (PEARL, 1988; SHACHTER, 1986) surgiu na década de 80, originalmente nas áreas de inteligência artificial e pesquisa operacional, sendo usado para denominar um tipo específico de modelo probabilístico que representa relações de dependência entre um conjunto de *variáveis aleatórias* (JENSEN, 1996; PEARL, 1988). Ao longo dos últimos anos a importância deste tipo de modelo transcendeu suas áreas de origem, adquirindo importância em várias outras áreas como, por exemplo, estatística Bayesiana (GILKS; RICHARDSON; SPIEGELHALTER, 1996) e bioinformática (BALDI; BRUNAK, 2001).

Uma rede Bayesiana representa um conjunto de variáveis $\mathbf{X} = \{X_1, \ldots, X_n\}$ onde cada variável constitui um nó de um grafo. Os nós são ligados através de arcos direcionados representando dependência entre os mesmos. Se existe um arco $X_i \rightarrow X_j$, diz-se que X_i é *pai* de X_j. Cada nó do grafo contém uma *Distribuição de Probabilidades Condicionais* $P(X_i|\text{pais}(X_i))$ que representa a probabilidade de X_i condicionada a seus pais. Adicionalmente, o grafo de uma rede Bayesiana não pode conter ciclos direcionados.

Exemplo 1 *Na rede da Figura 2.1 (CHARNIAK, 1991) temos 5 variáveis: F indica que uma dada família está fora de casa; L indica que a luz da casa dessa família está acesa, e temos que L depende de F; D representa que o cachorro da família encontra-se no quintal, e depende de F e da variável B que por sua vez indica se o cachorro está com problemas intestinais; por fim, H mostra que se pode ouvir latidos de cachorro, e isso depende apenas do cachorro estar no quintal. Ao lado do grafo, temos a especificação completa das probabilidades dessa rede Bayesiana. Pode-se ver, por exemplo, que a probabilidade*

da luz estar acesa é 0.60 *se a família estiver fora de casa; mas essa mesma probabilidade vale* 0.05 *se a família não estiver fora.*

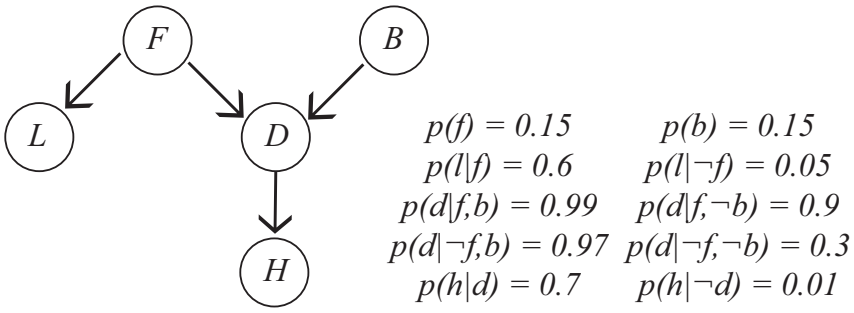


Figura 2.1: Exemplo de rede Bayesiana proposto por Charniak (1991).

Essa representação se mostra muito eficiente por ser compacta, geralmente precisando de poucos valores para especificar a rede inteira. Considere a rede Bayesiana do Exemplo 1, que contém 5 variáveis binárias. Para especificar completamente a rede foram necessários apenas 10 valores, enquanto que para especificar a probabilidade para todas as combinações sem a noção de dependência entre variáveis seriam necessários $2^5 - 1 = 31$ valores.

Em uma rede Bayesiana as relações de dependência entre as variáveis são codificadas através da seguinte *condição de Markov*: uma variável X_i é independente de seus não-descendentes no grafo dados os pais de X_i. Essa condição permite que a distribuição de probabilidades sobre todas as variáveis $X_1, \ldots, X_n$ seja fatorada pela expressão

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\text{pais}(X_i))\,. \tag{2.3}$$

Ainda considerando o Exemplo 1, pela condição de Markov, podemos calcular a probabilidade para a seguinte situação:

- a família não se encontra fora de casa ($\neg f$);
- a luz não está acesa ($\neg l$);
- o cachorro não está com problemas ($\neg b$);
- o cachorro está no quintal (d);
- pode-se ouvir latidos (h).

Essa probabilidade é dada pelo produto

$$\begin{aligned} P(\neg f, \neg l, \neg b, d, h) &= P(\neg f)\, P(\neg l|\neg f)\, P(\neg b)\, P(d|\neg b \neg f)\, P(h|d) \\ &= 0.85 \cdot 0.95 \cdot 0.85 \cdot 0.30 \cdot 0.70 \\ &= 0.14. \end{aligned}$$

O cálculo de uma probabilidade sobre todas as variáveis, apesar de fácil de se realizar, não é muito útil. Normalmente se está interessado em descobrir a probabilidade de uma variável específica (ou um conjunto de variáveis) ao invés de se descobrir a probabilidade para um dos inúmeros estados de um modelo. Ou seja, geralmente deseja-se calcular $P(X_i|evidências)$ ao invés de $P(X_1, X_2, \ldots, X_n)$. Para isso, primeiramente vamos definir a notação necessária:

- $\mathbf{X} = X_1, \ldots, X_n$ é o conjunto de todas as variáveis do modelo.
- $\mathbf{X}_q$ é o conjunto de variáveis cuja distribuição de probabilidades é de interesse.
- $\mathbf{X}_E$ são o conjunto de variáveis cujo valor é observado.
- E é a evidência propriamente dita (por exemplo, $E = \{x_1, \neg x_3\}$).
- $\mathbf{Y} \setminus \mathbf{Z}$ são as variáveis pertencentes a um conjunto $\mathbf{Y}$ mas não pertencentes a um conjunto $\mathbf{Z}$.

O cálculo de $P(\mathbf{X}_q|\mathbf{X}_E)$ é chamado de inferência, e segundo a regra de Bayes pode ser feito por

$$P(\mathbf{X}_q|\mathbf{X}_E) = \frac{P(\mathbf{X}_q, E)}{P(E)} \ ,$$

onde $P(\mathbf{X}_q, E)$ e $P(E)$ podem ser obtidas pela marginalização da Equação (2.3), ou seja,

$$P(\mathbf{X}_q|\mathbf{X}_E) = \frac{P(\mathbf{X}_q, E)}{P(E)} = \frac{\sum_{\mathbf{X}\setminus\mathbf{X}_q,\mathbf{X}_E} P(\mathbf{X})}{\sum_{\mathbf{X}\setminus\mathbf{X}_E} P(\mathbf{X})} \ . \tag{2.4}$$

Exemplo 2 *Ainda considerando a rede Bayesiana da Figura 2.1, deseja-se inferir a probabilidade de D dado $H = verdadeiro$. Temos:*

$$P(D|h) = \frac{\sum_{F,B,L} P(F)\times P(B)\times P(L|F)\times P(D|F,B)\times P(h|D)}{\sum_{F,B,L,D} P(F)\times P(B)\times P(L|F)\times P(D|F,B)\times P(h|D)} \quad .$$

Como pode-se ver no Exemplo 2, o cálculo da inferência é bastante extenso, sofrendo com problemas de escalabilidade conforme aumenta o número de variáveis da rede. Existe uma série de algoritmos feitos para lidar com esse problema, mas embora os algoritmos consigam melhorar a escalabilidade, a inferência em redes complexas muito grandes (ordem de 10^4 nós) ainda é inviável. Esses algoritmos serão apresentados no Capítulo 4, junto com o algoritmo proposto neste trabalho.

Apesar dessa questão de escalabilidade, redes Bayesianas são bastante eficientes a ponto de serem tradicionalmente utilizadas em problemas de inteligência artificial. Isso levou ao desenvolvimento de diversas extensões para o modelo de redes Bayesianas, muitas delas misturando conceitos de lógica aos conceitos estatísticos. Este desejo de combinar probabilidades e sentenças lógicas é antigo, já estando presente no trabalho de Boole (BOOLE, 1958), mas foi ao longo do último século que essa união entre disciplinas foi redescoberto tanto em filosofia (HOWSON; URBACH, 1993; Kyburg Jr.; TENG, 2001) quanto em estatística (FINETTI, 1974) e em computação (HAILPERIN, 1965; NILSSON, 1986).

Grosso modo, existem hoje dois tipos de linguagens que recebem a denominação de *lógicas probabilísticas*:

- Lógicas probabilísticas "clássicas", que permitem asserções probabilísticas sobre sentenças genéricas com o objetivo de obter intervalos para probabilidades de interesse (BACCHUS, 1990; GEORGAKOPOULOS; KAVVADIAS; PAPADIMITRIOU, 1988; HALPERN, 1990; JAUMARD; HANSEN; AO, 1991; LUKASIEWICZ, 2001). Por exemplo, se A, B e C são proposições, as asserções $P(B) = P(A \rightarrow \neg C) = P(B \rightarrow C) = 2/3$ implicam $P(A \wedge B) \in [0, 2/3]$.

- Lógicas probabilísticas que permitem relações de independência expressas por grafos (GETOOR et al., 2001; GETOOR; TASKAR, 2007; JAEGER, 1997; KERSTING; DICK, 2004; MILCH et al., 2005; NGO; HADDAWY, 1997; POOLE, 1993; SATO; KAMEYA, 2001). Estas lógicas são mais intuitivas que as clássicas e são freqüentemente chamadas de *modelos probabilísticos relacionais*. Em geral, nestas lógicas a semântica é tal que qualquer conjunto de sentenças pode ser transformado em uma

rede Bayesiana ou um campo de Markov[1]. Isso é obtido restringindo-se a linguagem, em alguns casos, drasticamente.

Apesar das limitações dos modelos probabilísticos relacionais, eles tem encontrado grande aplicação principalmente como substrato para aprendizado (RICHARDSON; DOMINGOS, 2006). Várias destas lógicas têm sido aplicadas em "indução de programas lógicos" em uma área geralmente referida como ILP (de *Inductive Logic Programming*) (MUGGLETON; RAEDT, 1994). A maioria das propostas hoje na literatura ainda está em fase inicial. Dentre aquelas que apresentam razoável maturidade e já levaram a implementações, identificam-se as linguagens PRISM (SATO; KAMEYA, 2001), *redes Bayesianas relacionais* (JAEGER, 2001) e *redes lógicas de Markov* (RICHARDSON; DOMINGOS, 2006). Essas linguagens são analisadas a seguir.

PRISM é uma linguagem implementada em pacote de mesmo nome (`http://sato-www.cs.titech.ac.jp/prism/`) e tem como foco a criação de uma versão probabilística da linguagem Prolog.

Rede Lógica de Markov é uma linguagem proposta por Richardson e Domingos (2006) que usa sentenças lógicas para especificar campos de Markov (referidos como *redes de Markov*). Uma *rede de lógica de Markov* é definida por um conjunto de pares (F_i, w_i), onde cada F_i é uma sentença de lógica de primeira-ordem (sem funções) e w_i é um *peso*, e por um domínio conhecido finito. Esse conjunto de pares e domínio define uma rede de Markov, possivelmente grande, onde cada nó é uma relação instanciada por elementos do domínio, e existe um arco entre dois nós se estes nós aparecem em uma fórmula instanciada. Cada uma destas relações instanciadas é uma variável X_i na rede de Markov. A Expressão (2.3) é reescrita como

$$P(X_1, \ldots, X_n) = \frac{1}{Z} \exp(\sum_k w_k F_k(X_k)), \tag{2.5}$$

onde w_k é justamente o peso de F_k (agora instanciada por X_k). Redes Lógicas de Markov já foram implementadas no pacote Alchemy (`http://alchemy.cs.washington.edu/`) que permite inferência e aprendizado. Todavia, existem algumas dificuldades relacionadas à interpretação do pesos associados às fórmulas, já que um mesmo peso pode gerar diferentes probabilidades dependendo da cardinalidade do domínio de interesse.

[1]Campos de Markov são modelos semelhantes a redes Bayesianas, porém utilizam grafos não-direcionados, substituindo as probabilidades condicionais por funções potenciais arbitrárias.

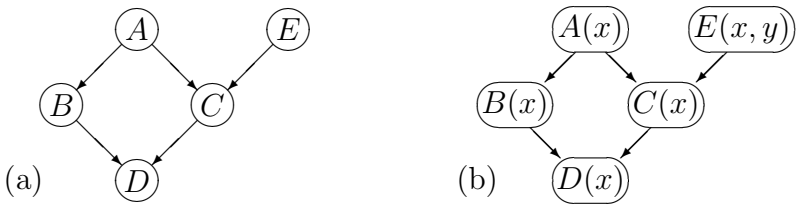


Figura 2.2: Estrutura de rede Bayesiana (a) e rede Bayesiana relacional (b).

Rede Bayesiana Relacional é um modelo que estende redes Bayesianas ao permitir que um nó do grafo seja associado a uma relação parametrizada por variáveis lógicas. Por exemplo, a rede Bayesiana da Figura 2.2 (a) pode ser transformada em uma rede Bayesiana relacional se associarmos a relação $A(x)$ ao nó rotulado como A ou $E(x, y)$ ao nó rotulado como E. Aqui as variáveis lógicas x e y indicam elementos em um domínio $\mathcal{D}$, que é suposto conhecido e finito (embora também hajam extensões para domínios infinitos (JAEGER, 1998a, 1998b)). São propostas sintaxe e semântica para *funções de combinação* (quantificadores $\exists$ e $\forall$, média, máximo, "Noisy-OR", etc) que podem ser utilizadas para especificar a probabilidade de uma variável condicionada a todas as instanciações de outra variável. O modelo de rede Bayesiana relacional está implementado no pacote Primula (`http://www.cs.aau.dk/~jaeger/Primula/`), que permite inferências com redes Bayesianas relacionais e permite o aprendizado dos parâmetros das mesmas.

Assim como as redes Bayesianas, as redes Bayesianas relacionais e outras lógicas probabilísticas sofrem de problemas computacionais para inferência, enfrentando dificuldades numéricas que claramente exigem algoritmos mais sofisticados do que os atualmente disponíveis. A partir dessa constatação, fica evidente que existe uma oportunidade de pesquisa no desenvolvimento, implementação e aplicação de novos algoritmos de inferência. Porém antes de dar início ao desenvolvimento dos algoritmos, vale aqui fazer uma revisão do ponto de vista de lógica.

2.3 Lógica de Primeira Ordem

A lógica de primeira ordem é uma lógica relativamente simples, mas com grande expressividade. O vocabulário das lógicas de primeira ordem conta com *constantes*, *variáveis*, *relações* e *funções*.

– Constantes são os objetos que populam o domínio: `João`, `José` e `Maria`;

– Variáveis podem assumir valores de objetos do domínio: a variável `pessoa` pode

assumir o valor `José`;

- Relações são predicados binários que expressam conexões entre objetos do domínio: `pai(pessoa`$_1$`,pessoa`$_2$`)` nos diz que `pessoa`$_1$ é pai de `pessoa`$_2$;
- Funções mapeiam objetos do domínio a outros objetos do domínio: `paiDe(João)` indica um objeto referenciando-o por meio do objeto `João`).

A lógica de primeira ordem ainda conta com *operadores booleanos* ($\cup$, $\cap$, $\neg$,...) e *quantificadores* ($\exists$, $\forall$, ...). As lógicas de primeira ordem se caracterizam por permitir que as variáveis de uma fórmula sejam quantificadas, enquanto que *lógicas de segunda ordem* permitem que, além das variáveis, funções sejam quantificadas, enquanto que *lógicas proposicionais* não permitem quantificadores.

Fragmentos de lógica de primeira-ordem com excelente balanço entre expressividade e complexidade computacional são as chamadas *Lógicas de Descrição* (BAADER et al., 2002). Lógicas de descrição têm sido bem sucedidas na representação de ontologias (sobretudo ontologias médicas). Propostas para normatização de linguagens de representação de conhecimento têm sido baseadas nestas lógicas devido à sua boa expressividade. Em particular, o *World-Wide-Web Consortium* tem usado lógicas de descrição na "rede semântica" (ou seja, na rede de conteúdos semânticos formada a partir da internet) (HORROCKS; PATEL-SCHNEIDER; HARMELEN, 2003).

Uma lógica de descrição tradicionalmente contém *indivíduos*, *conceitos* e *papéis* (BAADER et al., 2002), sendo que indivíduos são os objetos que populam o domínio de interesse, os conceitos são relações unárias utilizados para descrever os indivíduos, e os papéis são relações binárias que expressam conexão entre os indivíduos. Por exemplo, considere:

- o indivíduo `Maria`;
- `Mulher(x)` é um conceito que diz que um dado indivíduo é uma mulher, assim, `Mulher(Maria)` nos diz que o indivíduo `Maria` é uma mulher;
- um papel `mãe(x,y)` nos diz que um indivíduo `x` é mãe de `y`, assim sendo, `mãe(Maria,José)` representa que `Maria` é mãe de `José`.

Conceitos e papéis podem ser combinados para formar novos conceitos pela utilização de um conjunto de *construtores*. A tradicional lógica $\mathcal{ALC}$ (SCHMIDT-SCHAUSS; SMOLKA, 1991) conta com os seguintes construtores:

- *conjunção* ($C \sqcap D$);
- *disjunção* ($C \sqcup D$);
- *negação* ($\neg C$);
- restrição *existencial* ($\exists r.C$);
- restrição *universal* ($\forall r.C$);

e ainda contam com

- *definição* ($C \equiv D$);
- *inclusão* ($C \sqsubseteq D$).

A semântica desses elementos é bastante direta, dada pela função de interpretação $\mathcal{I}$ definida a seguir. Seja Δ o conjunto não vazio de interpretações do domínio. Cada indivíduo a é mapeado a um elemento $\mathcal{I}(a) \in \Delta$, cada conceito C é mapeado a um conjunto $\mathcal{I}(C) \subseteq \Delta$, cada papel r é mapeado a uma relação binária $\mathcal{I}(r) \subseteq \Delta \times \Delta$. Assim:

- $\mathcal{I}(\top) = \Delta$;
- $\mathcal{I}(\bot) = \emptyset$;
- $\mathcal{I}(C \sqcap D) = \mathcal{I}(C) \cap \mathcal{I}(D)$ (conjunção é interpretada como interseção);
- $\mathcal{I}(C \sqcup D) = \mathcal{I}(C) \cup \mathcal{I}(D)$ (disjunção é interpretada como união);
- $\mathcal{I}(\neg C) = \mathcal{D} \backslash \mathcal{I}(C)$ (complemento é interpretado como negação);
- $\mathcal{I}(\exists r.C) = \{x \in \mathcal{D} | \exists y : (x, y) \in \mathcal{I}(r) \wedge y \in \mathcal{I}(C)\}$;
- $\mathcal{I}(\forall r.C) = \{x \in \mathcal{D} | \forall y : (x, y) \in \mathcal{I}(r) \rightarrow y \in \mathcal{I}(C)\}$.

por fim, Diz-se $C \sqsubseteq D$ se e somente se $\mathcal{I}(C) \subseteq \mathcal{I}(D)$, enquanto que $C \equiv D$ se e somente se $\mathcal{I}(C) = \mathcal{I}(D)$. Todos os construtores acima têm tradução nas lógicas modal e de primeira-ordem (BAADER et al., 2002); que geralmente tratam um conceito C como um predicado unário $C(x)$, um papel r como um predicado binário $r(x, y)$, e restrições como quantificadores. Assim, uma restrição existencial $\exists r.C$ indica o conjunto de elementos x do domínio para os quais existe pelo menos um y tal que $r(x, y)$ satisfaz $C(y)$.

Todos os construtores podem ser combinados; por exemplo,

$$D \sqsubseteq (\exists\, \texttt{temFilho.Homem}) \sqcap (\forall \texttt{compra.}(\texttt{Orgânico} \sqcup \texttt{Fruta}))$$

indica que D contém indivíduos que tem pelo menos um filho e que somente compra ou frutas ou comida orgânica. Um grande número de operadores e variantes pode ser encontrado na literatura (BAADER et al., 2002), sendo que a presença ou ausência dos operadores definem a expressividade e complexidade da lógica.

2.4 Lógicas de Descrição Probabilísticas

Anteriormente foi vista a importância dos modelos probabilísticos como redes Bayesianas e campos de Markov, assim como a tendência a reunir em um só modelo conceitos lógicos e conceitos estatísticos. Por sua vez, as lógicas de descrição são fragmentos de lógica de primeira-ordem com excelente balanço entre expressividade e complexidade computacional. Tendo isso em vista, têm surgido na literatura muitas tentativas para a criação de lógicas de descrição probabilísticas. Essa seção é destinada a revisão da literatura relevante.

Heinsohn (1994), Jaeger (1994) e Sebastiani (1994) consideram, em seus respectivos trabalhos, axiomas de inclusão probabilísticos como $P_{\mathcal{D}}(\texttt{Planta}) = \alpha$, tendo como significado que ao selecionar aleatoriamente um indivíduo, tem-se a probabilidade α de que o indivíduo seja uma `Planta`. Ou seja, as probabilidades são atribuídas a subconjuntos do domínio $\mathcal{D}$, caracterizando assim o que é chamada de *semântica baseada em domínio*. Sebastiani também permite que sejam feitas asserções como $P(\texttt{Planta}(\texttt{Maria})) = \alpha$, especificando assim, probabilidades sobre as interpretações propriamente ditas. Por exemplo, a asserção $P(\texttt{Pássaro}(\texttt{Maria})) = 0.001$ é entendida como a atribuição da probabilidade 0.001 para todas as interpretações onde `Maria` é um pássaro. Isso caracteriza a chamada *semântica baseada em interpretações*. De maneira geral, a maioria das propostas abordam semânticas baseadas em domínio (DING; PENG; PAN, 2006; DÜRIG; STUDER, 2005; GIUGNO; LUKASIEWICZ, 2002; HEINSOHN, 1994; JAEGER, 1994; KOLLER; PFEFFER, 1997; LUKASIEWICZ, 2008; SEBASTIANI, 1994; YELLAND, 1999), enquanto relativamente poucas propostas abordaram semânticas baseadas em interpretações (COSTA; LASKEY, 2006; SEBASTIANI, 1994; D'AMATO; FANIZZI; LUKASIEWICZ, 2008).

A dificuldade de se trabalhar com semântica baseada em domínio está no problema

chamado de *inferência direta* (BACCHUS, 1990; Kyburg Jr.; TENG, 2001) no qual as informações estatísticas sobre o domínio não se transmitem aos indivíduos. Por exemplo, supondo a atribuição $P(\texttt{Pássaro}) = 0.3$; em uma semântica baseada em domínio o domínio é definido, e as interpretações são fixadas segundo a probabilidade 0.3 de que um elemento qualquer do domínio seja um pássaro. Entretanto, ao olhar para um indivíduo específico do domínio, `Maria`, $P(\texttt{Pássaro}(\mathsf{Maria})) = 1$ ou 0 uma vez que as interpretações estão fixas e `Maria` ou é um pássaro ou não é. Por esse motivo, a maioria das lógicas de descrição probabilísticas simplesmente não aceita essas atribuições de probabilidade. Vale notar que Dürig e Studer (2005) evitam o problema de inferência direta ao permitir o uso de probabilidades somente sobre as atribuições. Já Giugno e Lukasiewicz (2002) e Lukasiewicz (2008) propõem um modelo onde probabilidades sobre terminologias e atribuições se misturam pela utilização de relações não monotônicas, tendo sua lógica P-$\mathcal{SHOIN}$(D) como sendo a mais expressiva da literatura.

As lógicas de descrição probabilísticas citadas até agora têm em comum que seus conjuntos de fórmulas podem ser satisfeitos por uma ou mais distribuições de probabilidades, característica frequentemente adotada por lógicas probabilísticas tanto de primeira ordem quanto proposicionais (HALPERN, 2003). Outra característica comum é que elas não expressam condições de independência. Entretanto há um esforço recente para combinar construtores lógicos com redes Bayesianas e campos de Markov de forma a utilizar as condições de independência para facilitar os cálculos (GETOOR; TASKAR, 2007; MILCH; RUSSELL, 2007; RAEDT; KERSTING, 2008). De fato, muitas das propostas recentes adotam semânticas baseadas em redes Bayesianas. A primeira delas, chamada P-CLASSIC (KOLLER; PFEFFER, 1997), estende a lógica CLASSIC com um conjunto de redes Bayesianas e condições de independência de forma a especificar uma única distribuição de probabilidades sobre o domínio, porém tendo a limitação de não trabalhar com atribuições.

Outras lógicas que combinam terminologias com redes Bayesianas são a *Tiny Description Logic* (YELLAND, 1999), *BayesOWL* (DING; PENG; PAN, 2006), além da lógica proposta por Staker (2002) porém nenhuma delas permite atribuições. Ainda existe a linguagem *PROWL* (COSTA; LASKEY, 2006) com semântica baseada em interpretações baseada nas redes Bayesianas Multi-entidades *(Multi-entity Bayesian networks – MEBNs)* (COSTA; LASKEY, 2005). Por fim, ainda existe uma versão probabilística da OWL proposta por Nottelmann e Fuhr (2006)

Além da literatura aqui citada, ainda há uma grande quantidade de propostas para a

utilização de probabilidades em bases de conhecimento (HUNG; GETOOR; SUBRAHMANIAN, 2007) ou ainda *lógicas-fuzzy de descrição* (LUKASIEWICZ; STRACCIA, 2008), que não serão discutidas aqui.

3 CR$\mathcal{ALC}$

Como visto no Capítulo 2, as lógicas de descrição probabilísticas são modelos com uma ótima relação entre expressividade e complexidade. E embora haja um esforço na literatura, ainda não foi possível achar uma solução definitiva para o problema de escalabilidade. Tendo isso em mente, decidiu-se propor uma nova lógica probabilística e um algoritmo para inferência para ela. Assim, a primeira contribuição desse trabalho é a lógica aqui proposta, intitulada CR$\mathcal{ALC}$ – *Credal* $\mathcal{ALC}$ (COZMAN; POLASTRO, 2008), que se baseia em uma lógica de descrição simples denominada $\mathcal{ALC}$ e se foca no desenvolvimento de métodos de inferência escaláveis.

Primeiramente, assume-se que o domínio é finito e conhecido. Assume-se também um vocabulário contendo indivíduos, conceitos e papéis. Assim como definidos na Seção 2.3, conceitos são utilizados para descrever os indivíduos, enquanto papéis expressam relações entre os indivíduos. Conceitos e papéis podem ser combinados para formar novos conceitos pela utilização dos seguintes construtores da lógica $\mathcal{ALC}$: conjunção ($C \sqcap D$), disjunção ($C \sqcup D$) e negação ($\neg C$) juntamente com restrição existencial ($\exists r.C$) e restrição universal ($\forall r.C$). O conceito ($C \sqcup \neg C$) é representado por $\top$ enquanto que o conceito ($C \sqcap \neg C$) é representado por $\bot$; para facilitar a leitura, $r.\top$ será simplesmente abreviada por r (por exemplo, em restrições como $\exists r$). Inclusão e definição de conceitos são expressas respectivamente por $C \sqsubseteq D$ e $C \equiv D$, onde C e D são conceitos. A semântica desses elementos segue o que foi apresentado na Seção 2.3.

São permitidas inclusões probabilísticas como $P(C|D) = \alpha$, onde C é o nome de um conceito e D é um conceito. Essa inclusão é interpretada de acordo com a seguinte semântica:

$$\forall x : P(C(x)|D(x)) = \alpha;$$

ou seja: todo elemento x do domínio satisfaz a asserção probabilística $P(C(x)|D(x)) = \alpha$. No caso onde D é $\top$, escreve-se simplesmente $P(C) = \alpha$. Não são permitidas inclusões probabilísticas do tipo $P(\forall r.C|D)$, sendo permitidas somente inclusões que tenham apenas nomes de conceitos em seu condicionante. Também são permitidas inclusões pro-

babilísticas como $P(r) = \beta$, onde r é um papel. De forma análoga, essa inclusão é interpretada de acordo com a seguinte semântica:

$$\forall x, y : P(r(x, y)) = \beta.$$

Vale notar aqui que a asserção $P(C|A \sqcup B) = \alpha_1$ e $P(D|A \sqcap B) = \alpha_2$ são interpretadas como:

$$\begin{aligned} P(C|A \sqcup B) &= \alpha_1, \\ P(C|A \sqcup \neg B) &= \alpha_1, \\ P(C|\neg A \sqcup B) &= \alpha_1, \\ P(D|A \sqcap B) &= \alpha_2. \end{aligned}$$

e não como

$$\frac{P(C \sqcap (A \sqcup B))}{P(A \sqcup B)} = \alpha_1, \qquad \frac{P(D \sqcap (A \sqcap B))}{P(A \sqcup B)} = \alpha_2.$$

Ambas as interpretações são bastante naturais, mas por facilidade, escolheu-se o primeiro caso.

Por fim, adotam-se as seguintes suposições:

Aciclicidade: Considere as inclusões $P(C|B) = \alpha$ e $D \sqsubseteq C$. Diz se que C "utiliza" B e que D "utiliza" C; a propriedade é transitiva e portanto D "utiliza" B. Um conjunto de definições e inclusões de conceitos é chamado de *terminologia*; uma terminologia é dita *acíclica* se nenhum de seus conceitos utilizar a si mesmo. Essa propriedade é comumente imposta em lógicas de descrição (BAADER et al., 2002). Assume-se que toda terminologia em CR$\mathcal{ALC}$ é acíclica, permitindo assim que seja formado um grafo acíclico direcionado para as terminologias, possibilitando que as mesmas sejam representadas em redes Bayesianas relacionais.

Semântica baseada em interpretações: Conforme foi visto na Seção 2.4, semânticas baseadas em domínio consideram inclusões probabilísticas que são naturalmente entendidas como

$$P(\text{conjunto de } C\text{s} \mid \text{conjunto de } D\text{s}) = \alpha;$$

enquanto que as baseadas em interpretação consideram inclusões probabilísticas naturalmente entendidas como

$$\forall x : P(C(x)|D(x)) = \alpha.$$

Adota-se esta última semântica em CR$\mathcal{ALC}$ pois ela nos permite fazer inferências sobre $P(A(i)|B(j))$ para os conceitos A e B respectivamente instanciados para os indivíduos i e j enquanto que uma semântica baseada em domínio atribuiria o valor 0 ou 1 para a interpretação, ficando presa no problema de inferência direta. Note que não há contradição entre $\forall x : P(C(x)) = \alpha$ e uma observação $C(a) = verdadeiro$ já que $P(C(a)|C(a)) = 1$ enquanto ainda temos $P(C(a)) = \alpha$. Assume-se que os indivíduos são designadores rígidos em relação às interpretações, ou seja, um indivíduo corresponde ao mesmo elemento do domínio para todas as interpretações (BACCHUS, 1990), como será visto nos parágrafos a seguir.

Independência e Condição de Markov: Lógicas de descrição probabilísticas como P-CLASSIC, BayesOWL e PROWL têm uma propriedade de independência definida por uma condição de Markov que restringe os valores de probabilidade e decompõe os modelos em pequenas partes isoladas. Nessas lógicas não há uma sintaxe que expresse independência; independência é extraída diretamente da estrutura das fórmulas, fato comum a muitas lógicas (GETOOR et al., 2001; MILCH; RUSSELL, 2007). A lógica CR$\mathcal{ALC}$ também assume essa posição, sendo que a independência é representada pela seguinte condição de Markov:

> *Para todo conceito C em uma terminologia e para cada x no domínio $\mathcal{D}$, $C(x)$ é independente de todas as afirmações que não utilizem $C(x)$, dadas as afirmações que utilizem diretamente C.*
> *Para todo papel r e para todo (x, y) em $\mathcal{D} \times \mathcal{D}$, $r(x, y)$ é independente de todas as afirmações que não utilizem $r(x, y)$, dadas as afirmações que utilizem diretamente r.*

Homogeneidade: Uma terminologia pode não especificar uma única distribuição de probabilidades sobre as interpretações. Por exemplo: $P(C|B \sqcup \exists r.D) = \alpha_1$ não garante que $P(C|B \sqcap \exists r.D)$ seja restrito a um único valor. Por isso, assume-se a seguinte condição de homogeneidade:

> *Considere um conceito C com pais $D_1, \ldots, D_m$. Para cada conjunção de m conceitos $\pm D_i$ [1], tem-se $P(C| \pm D_1 \sqcap \pm D_2 \sqcap \cdots \sqcap \pm D_m) = \gamma$.*

[1] $\pm$ indica que os conceitos podem aparecer negados ou não.

Com essa condição, cada terminologia em CR$\mathcal{ALC}$ pode ser representada como um grafo acíclico direcionado onde cada nó está associado a uma relação. Se cada probabilidade for especificada por um valor preciso, então a terminologia pode ser representada por uma rede Bayesiana relacional, sendo que a proposicionalização da mesma gera uma rede Bayesiana. Para casos onde nem todas as probabilidades são precisamente especificadas, a proposicionalização gera uma chamada *rede credal* (COZMAN, 2000).

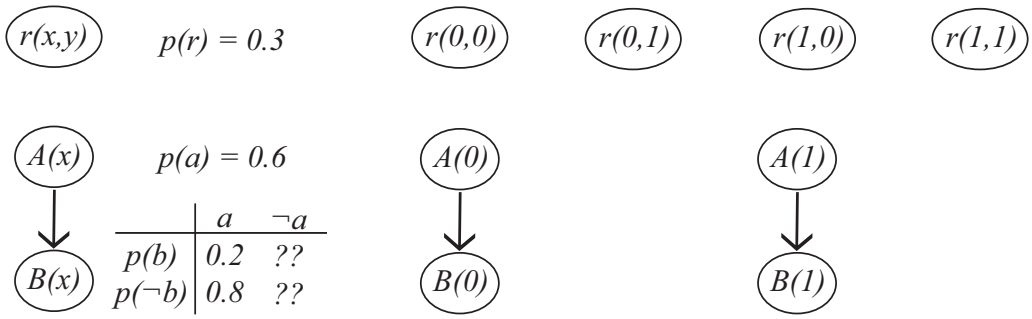
Unicidade: Adota-se a suposição de *nomes únicos*: nomes distintos de indivíduos correspondem a elementos distintos do domínio. Adota-se também a seguinte condição de unicidade:

- Cada conceito C pertence a uma de duas categorias: ou C é especificado por uma definição, ou C tem um pai D e são feitas inclusões a respeito de D e $\neg D$. Para o último caso há três opções:
  i. $C \sqsubseteq D$ e $P(C|D) = \alpha$;
  ii. $C \sqsubseteq \neg D$ e $P(C|\neg D) = \alpha$;
  iii. $P(C|D) = \alpha'$ e $P(C|\neg D) = \alpha''$.
- Para cada papel r é feita uma atribuição de probabilidade $P(r) = \alpha$, cuja semântica é $\forall x, y : P(r(x, y)) = \alpha$.

Com a condição de unicidade e os nomes únicos garante-se que ao se proposicionalizar a terminologia, se obtém uma rede Bayesiana. Suposições similares são usadas para garantir unicidade em outras lógicas como P-CLASSIC e PROWL.

Representando CR$\mathcal{ALC}$ com redes Bayesianas relacionais: Em consequência da condição de Markov, CR$\mathcal{ALC}$ pode ser usada para se especificar redes Bayesianas relacionais de forma simples com elementos de lógica. A tradução direta de CR$\mathcal{ALC}$ para redes Bayesianas relacionais se dá de acordo com os seguintes itens:

Atribuições de probabilidades já são naturalmente representadas nas redes Bayesianas relacionais. As asserções $P(A) = 0.6$, $P(B|A) = 0.2$ e $P(r) = 0.3$ geram a rede que pode ser vista na Figura 3.1(a); a a proposicionalização da mesma pode ser vista na Figura 3.1(b). Note que a rede não está totalmente especificada, já que não se conhece $P(B|\neg A)$.

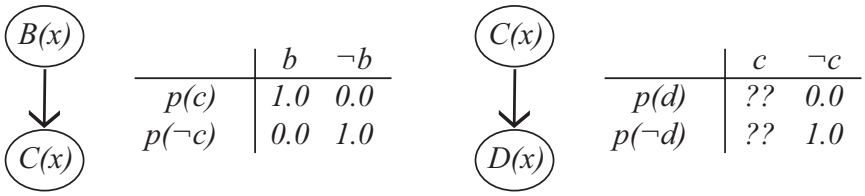


(a) Rede Bayesiana relacional.

(b) Rede instanciada para 2 indivíduos.

Figura 3.1: Grafo com a rede gerada por inclusões probabilísticas, e a instanciação para um domínio com 2 indivíduos.

Uma definição ou inclusão de conceito expressa uma relação determinística entre os termos envolvidos. $C \equiv B$ é equivalente a $P(C|B) = 1$ e $P(C|\neg B) = 0$. $D \sqsubseteq C$ é equivalente a $P(D|\neg C) = 0$; note que a inclusão de conceito nada diz sobre $P(D|C)$, que deverá ser posteriormente especificado para completar a rede. A Figura 3.2 ilustra estes casos.

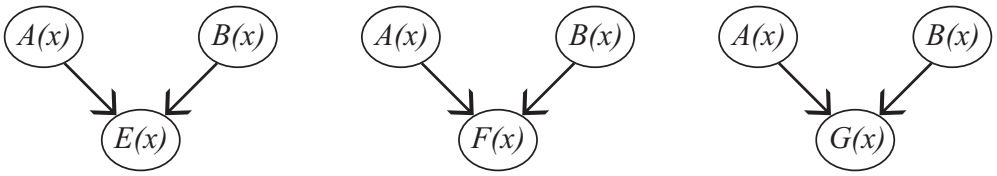


(a) $C \equiv B$

(b) $D \sqsubseteq C$

Figura 3.2: Redes geradas pela definição e inclusão de conceitos.

De forma análoga, a Figura 3.3 apresenta as tabelas de probabilidades condicionais para $P(E|A \sqcup \neg B) = 0.8$, $F \equiv (A \sqcup \neg B)$ e $G \sqsubseteq (A \sqcup \neg B)$.



	a	$\neg a$
b	0.8	??
$\neg b$	0.8	0.8

(a) $P(E|A \sqcup \neg B) = 0.8$

	a	$\neg a$
b	1.0	0.0
$\neg b$	1.0	1.0

(b) $F \equiv (A \sqcup \neg B)$

	a	$\neg a$
b	??	0.0
$\neg b$	??	??

(c) $G \sqsubseteq (A \sqcup \neg B)$

Figura 3.3: Tabelas de probabilidades condicionais geradas pelos construtores lógicos.

Por fim, temos que as restrições universais $\forall r.C$ são escritas na forma $\bigcap_{i=1}^{n}(r_i \rightarrow C_i)$, ou seja $\forall r(x,y).C(y)$ pode ser escrito como

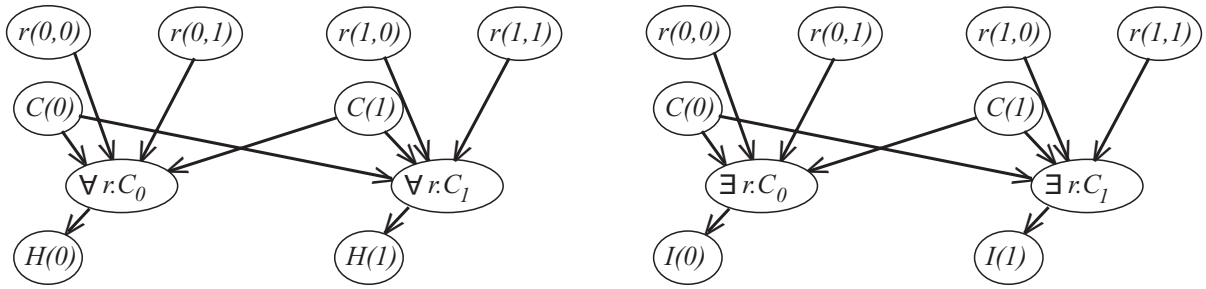
$$\big(r(x,0) \rightarrow C(0)\big) \ \sqcap \ \big(r(x,1) \rightarrow C(1)\big) \ \sqcap \ \ldots \ \sqcap \ \big(r(x,n) \rightarrow C(n)\big),$$

onde $r \rightarrow C$ é representado por $\neg r \sqcup C$.

As restrições existenciais $\exists r.C$ por sua vez são escritas na forma $\bigcup_{i=1}^{n}(r_i \cap C_i)$, ou seja $\exists r(x,y).C(y)$ pode ser escrito como

$$\big(r(x,0) \sqcap C(0)\big) \ \sqcup \ \big(r(x,1) \sqcap C(1)\big) \ \sqcup \ \ldots \ \sqcup \ \big(r(x,n) \sqcap C(n)\big).$$

A Figura 3.4 ilustra a representação das restrições mostrando a rede gerada para $H \equiv \forall r.C$ e $I \equiv \exists r.C$.



(a) $\forall r.C$ (b) $\exists r.C$

Figura 3.4: Representação de restrições universais e existenciais instanciadas para um domínio com dois indivíduos.

Exemplo 3 *Considere uma terminologia $\mathcal{T}_1$ com os conceitos A, B, C, D. Suponha $P(A) = 0.9$, $B \sqsubseteq A$, $P(B|A) = 0,45$, $D \equiv \forall r.A$, $C \equiv B \sqcup \exists r.D$ e $P(r) = 0,3$. O grafo da terminologia está representado na Figura 3.5, assim como a instanciação da mesma para o domínio $\mathcal{D} = \{a, b\}$.*

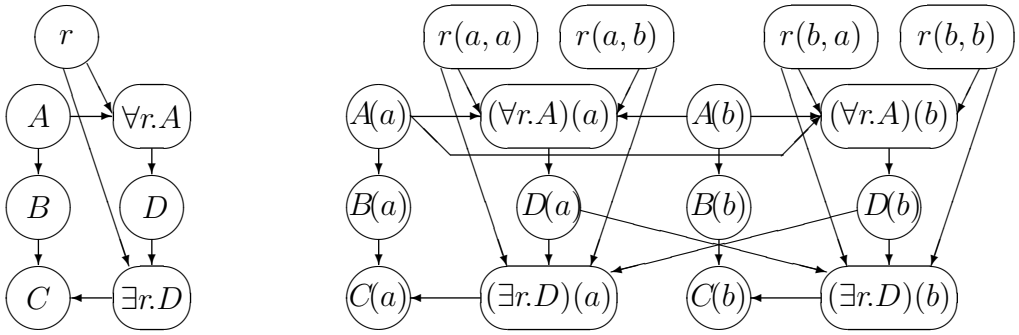


Figura 3.5: Grafo para a terminologia $\mathcal{T}_1$ do Exemplo 3 e a instanciação do mesmo para o domínio $\mathcal{D} = \{a, b\}$.

4 Algoritmos de Inferência

Um dos pontos fortes da lógica CR$\mathcal{ALC}$, proposta na seção anterior, é o fato de ela ter tradução direta para redes Bayesianas relacionais, podendo assim aproveitar algoritmos robustos de inferência. Esses algoritmos, mesmo sendo eficientes, ainda sofrem com problemas de escalabilidade, uma vez que as redes geradas pela CR$\mathcal{ALC}$ crescem com o quadrado do número de indivíduos de seu domínio. Assim, a segunda contribuição deste trabalho é um algoritmo de inferência em primeira ordem, para tratar a inferência nas redes criadas pela CR$\mathcal{ALC}$. A seguir serão apresentados alguns algoritmos tradicionais de inferência para redes bayesianas que podem ser utilizados com a CR$\mathcal{ALC}$, seguidos do novo algoritmo proposto.

4.1 Inferência Exata

O problema de inferência exata é o cálculo do valor da Equação (2.4). Observando-se um exemplo pequeno como o Exemplo 2, pode-se ver que há problemas de viabilidade de marginalização para o cálculo da inferência em redes não triviais.

Eliminação de Variáveis: Aproveitando-se da propriedade associativa da adição e multiplicação, o método de eliminação de variáveis (RUSSELL; NORVIG, 1995) tem como o intuito reordenar os fatores presentes na equação da inferência de forma a retirar o máximo de variáveis de dentro dos termos da somatória, fazendo o cálculo do valor marginal de uma variável por vez. Considerando-se o numerador na equação do Exemplo 2 temos:

$$
\begin{aligned}
P(D|h) &= \frac{\sum_{F,B,L} P(F) \times P(B) \times P(L|F) \times P(D|F,B) \times P(h|D)}{\underbrace{\sum_{F,B,L,D} P(F) \times P(B) \times P(L|F) \times P(D|F,B) \times P(h|D)}_{\mathcal{Z}}} \\
&= \frac{1}{\mathcal{Z}} \sum_{F,B,L} P(F) \times P(B) \times P(L|F) \times P(D|F,B) \times P(h|D) \\
&= \frac{1}{\mathcal{Z}} P(h|D) \sum_{F,B,L} P(F) \times P(B) \times P(L|F) \times P(D|F,B) \\
&= \frac{1}{\mathcal{Z}} P(h|D) \sum_{F,B} P(F) \times P(B) \times P(D|F,B) \times \underbrace{\sum_{L} P(L|F)}_{=1} \\
&= \frac{1}{\mathcal{Z}} P(h|D) \sum_{B} P(B) \times \sum_{F} P(D|F,B)\, P(F)\,. \qquad (4.1)
\end{aligned}
$$

A ordem das variáveis eliminadas não faz diferença quanto ao resultado final, porém a escolha da ordem correta tem grande influência na eficiência do algoritmo. É um algoritmo eficiente para tratar de redes *unicamente conexas* (também chamadas de *poliárvores*), que são redes nas quais existe no máximo um caminho não-orientado entre dois nós quaisquer.

Agrupamento de Variáveis: Em redes multiconexas uma técnica que pode ser utilizada para facilitar o cálculo é o agrupamento de variáveis, no qual se substitui alguns nós individuais por um nó que represente a combinação dos mesmos, de forma que a rede possa ser representada por uma poliárvore.

Exemplo 4 *Considere uma rede multiconexa composta de quatro nós, como pode ser visto na Figura 4.1, onde N representa que o céu está nublado, C representa a probabilidade de chuva, I representa que o irrigador está ligado e M que a grama está molhada. Na mesma figura, à direita pode-se ver a poliárvore formada ao se agrupar I e C.*

4.2 Inferência Aproximada

Mesmo utilizando-se a eliminação e agrupamento de variáveis, em redes grandes com muitas conexões e poucas evidencias é inviável realizar o cálculo de inferência exata, e frequentemente são utilizados métodos de inferência aproximados.

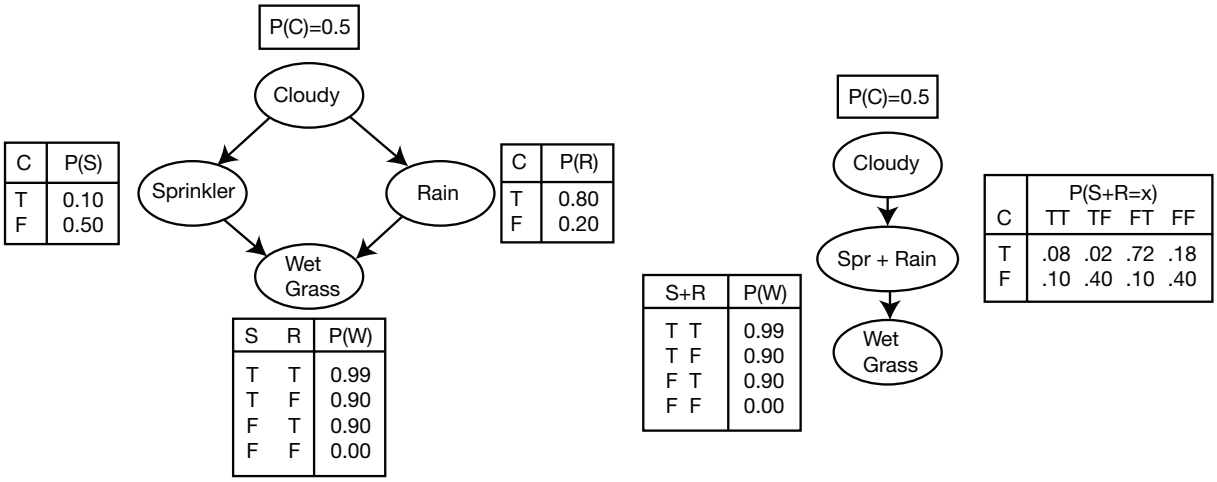


Figura 4.1: Rede multiconexa e poliárvore equivalente obtida por agrupamento de variáveis.

Métodos de amostragem: Um grupo de algoritmos para inferência aproximada bastante usado é o de inferência por amostragem, onde são geradas amostras aleatoriamente a partir de uma distribuição conhecida, e a probabilidade é obtida de acordo com a quantidade de amostras relevantes em relação ao total de amostras. Nem sempre é fácil se amostrar essas distribuições: se por um lado uma variável que representa o lançamento de uma moeda pode ser facilmente amostrada, por outro lado, modelos mais complexos precisam de outras distribuições mais elaboradas. Algumas variações podem ser usadas para melhorar o desempenho do algoritmo, como a rejeição ou ponderação das amostras, de acordo com as distribuições das quais se deseja amostrar. Uma classe de algoritmos de amostragem bastante utilizada é a chamada de *MCMC — Markov Chain Monte Carlo* (GILKS; RICHARDSON; SPIEGELHALTER, 1996) na qual as amostras são geradas a partir de possíveis configurações da rede.

Propagação de crença: Outro grupo importante de algoritmos para inferência aproximada é o de propagação de crenças. É um algoritmo que calcula para cada nó sua probabilidade de acordo com mensagens recebidas de seus nós vizinhos, e então repassa novas mensagens aos outros nós. São feitas iterações até a convergência dos resultados, e embora não haja garantias de convergência dos algoritmos, nem da precisão dos mesmos, os resultados empíricos são bastante sólidos. O algoritmo *LBP — Loopy Belief Propagation* (PEARL, 1988) é um dos mais empregados na literatura já que ele se mostra muito preciso e rápido para redes grandes e altamente conectadas. Uma particular variação do LBP é o algoritmo *L2U* (IDE; COZMAN, 2008), que faz a propagação de crenças em redes credais (redes bayesianas especificadas por probabilidades imprecisas). Outra

variação interessante é generalização proposta por Yedidia, Freeman e Weiss (2005), que melhora significativamente a precisão da aproximação.

4.3 Inferência em Primeira Ordem

Nesta seção será apresentado o algoritmo de inferência proposto nesse trabalho. Para o melhor entendimento do mesmo, considere uma terminologia $\mathcal{T}$, em CR$\mathcal{ALC}$, com domínio $\mathcal{D}$ de cardinalidade n e considere uma consulta sobre a probabilidade $Q = P\big(A_0(a_0)|\sqcap_{j=1}^{M} A_j(a_k)\big)$ onde A_j são nomes de conceitos e a_j são indivíduos, com $|\mathcal{D}| > M$.

Primeiramente precisamos definir a distribuição de probabilidade conjunta sobre um conjunto $V_{\mathcal{T}}^n$ que contenha todas as asserções geradas por $\mathcal{T}$ sobre $\mathcal{D}$. Seja $\mathcal{C}$ o conjunto dos conceitos e $\mathcal{R}$ o conjunto de papéis em $\mathcal{T}$. Para um conceito C_i, denominamos seus pais por pais$(C_i(x))$. Define-se

$\mathcal{D}' = \{a_0, \ldots, a_M\}$ e $\mathcal{D}'' = \mathcal{D} \backslash \mathcal{D}'$. Assim,

$$P(V_{\mathcal{T}}^n) = \prod_{C_i \in \mathcal{C}; x \in \mathcal{D}} P(C_i(x)|\text{pais}(C_i(x))) \times \prod_{r \in \mathcal{R}; x,y \in \mathcal{D}} P(r(x,y))\,. \tag{4.2}$$

Utilizando-se de *eliminação de variáveis em primeira ordem* (BRAZ; AMIR; ROTH, 2006a, 2006b; POOLE, 2003), reescreve-se a distribuição segundo a forma chamada *shattered* (BRAZ; AMIR; ROTH, 2006a) na qual cada átomo é instanciado ou em um conjunto de átomos idênticos ou em um conjunto de átomos distintos, tomando em consideração as restrições sobre as possíveis asserções.

$$\begin{aligned} P(V_{\mathcal{T}}^n) \;=\; & \prod_{C_i \in \mathcal{C}; a' \in \mathcal{D}'} P(C_i(a')|\text{pais}(C_i(a'))) \;\times \prod_{r \in \mathcal{R}; a', a'' \in \mathcal{D}'} P(r(a', a'')) \\ & \times \prod_{r \in \mathcal{R}; a' \in \mathcal{D}'; x_0 \in \mathcal{D}''} P(r(a', x_0))\, P(r(x_0, a')) \\ & \times \prod_{C_i \in \mathcal{C}; x_0 \in \mathcal{D}''} P(C_i(x_0)|\text{pais}(C_i(x_0))) \;\times \prod_{r \in \mathcal{R}; x_0, x_1 \in \mathcal{D}''} P(r(x_0, x_1))\,, \end{aligned} \tag{4.3}$$

sendo que:

- caso C_i seja o nome de um conceito:
  - pais$(C_i(x))$ denomina os pais de C_i;

– caso C_i seja uma restrição $\exists r.C$ ou $\forall r.C$:

  – pais$(C_i(a'))$ denomina $\{r(a', a''), C(a'') : a'' \in \mathcal{D}'\} \cup \{r(a', x_0), C(x_0) : x_0 \in \mathcal{D}''\}$;

  – pais$(C_i(x_0))$ denomina $\{r(x_0, a'), C(a') : a' \in \mathcal{D}'\} \cup \{r(x_0, x_1), C(x_1) : x_1 \in \mathcal{D}''\}$.

A primeira linha da Equação (4.3) é referente à parte da rede Bayesiana proposicionalizada sobre os indivíduos em Q. Esses indivíduos se conectam aos outros indivíduos do domínio através dos papéis $P(r(a', x_0))$ e $P(r(x_0, a'))$: segunda linha da equação. A terceira linha da equação refere-se aos indivíduos parametrizados, i.e. que estão fora de Q (indivíduos "genéricos" x_0 e x_1) e suas ligações. A Figura 4.2 apresenta a versão shattered da terminologia $\mathcal{T}_1$ do Exemplo 3 para a consulta $P(C\,(a_0))$.

O uso da rede shattered facilita bastante os cálculos, uma vez que ela "agrupa"todos os indivíduos do domínio que não estão em Q. Como a CR$\mathcal{ALC}$ (ou uma rede Bayesiana relacional) é estruturada por relações paramétricas, na instanciação dos nós da rede pode-se separar os nós em *fatias*, onde cada fatia contém as instanciações referentes a um indivíduo. Na Figura 4.3 pode-se ver uma rede genérica instanciada para um domínio com 3 indivíduos, com a separação de suas fatias (uma para cada indivíduo). Como a rede shattered agrupa os indivíduos parametrizados, a divisão da rede gera apenas $M + 1$ fatias: M fatias referentes aos indivíduos pertencentes a $\mathcal{D}'$ (a fatia de a_j contém todos os nós instanciados para $C_i(a_j) : C_i \in \mathcal{C}$ e para $r_i(a_j, x) : r_i \in \mathcal{R}$) e uma fatia é referente aos indivíduos parametrizados (nós instanciados para os elementos genéricos $x \in \mathcal{D}''$) da rede shattered. Para a consulta de $P(C\,(a_0))$ a rede shattered contém apenas duas fatias (Figura 4.2), uma para a_0 (único indivíduo presente na consulta) e uma para os elementos parametrizados.

Apesar da rede shattered agrupar os indivíduos parametrizados, o cálculo de nós que contém restrições ainda precisa considerar todo o domínio. Nas Equações (4.4) e

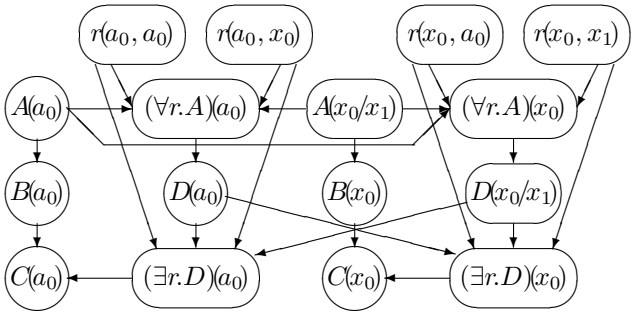


Figura 4.2: Versão shattered da terminologia $\mathcal{T}_1$ para a consulta $P(C\,(a_0))$.

(4.5) a seguir, vemos a forma algébrica da distribuição de probabilidades para restrições universais e existenciais.

$$
\begin{aligned}
P(\forall r(x,y)C(y)|\text{pais}(\forall r(x,y)C(y))) &= \Big(\forall r(x,y)C(y)\Big)\left(\prod_{y\in\mathcal{D}}\Big(1-r(x,y)+r(x,y)C(y)\Big)\right) \\
&+ \Big(1-\forall r(x,y)C(y)\Big) \\
&\left(1-\prod_{y\in\mathcal{D}}\Big(1-r(x,y)+r(x,y)C(y)\Big)\right). \qquad (4.4)\\
P(\exists r(x,y)C(y)|\text{pais}(\exists r(x,y)C(y))) &= \Big(\exists r(x,y)C(y)\Big)\left(1-\prod_{y\in\mathcal{D}}\Big(1-r(x,y)C(y)\Big)\right) \\
&\Big(1-\exists r(x,y)C(y)\Big)\left(\prod_{y\in\mathcal{D}}\Big(1-r(x,y)C(y)\Big)\right). (4.5)
\end{aligned}
$$

Esse cálculo pode proporcionar problemas de escalabilidade. No pior caso, onde todos os indivíduos fazem parte da consulta, a rede deve ser proposicionalizada por inteiro se tornando um processo inviável e requerendo métodos aproximados.

Assim, optou-se por aplicar a versão generalizada de Loopy Propagation de Yedidia, Freeman e Weiss (2005), ao invés de utilizar a propagação de crença na rede proposicionalizada, implementou-se o algoritmo para propagar mensagens na rede shattered. Os nós de restrições utilizam-se das Expressões (4.4) e (4.5) para combinar mensagens vindas das fatias de indivíduos instanciados e mensagens vindas da fatia que representa os indivíduos parametrizados, criando assim uma versão em *primeira-ordem* do Loopy Propagation. Se uma mensagem passa de um nó parametrizado para um nó com uma restrição, eleva-se a mensagem à potência igual ao número de indivíduos representados pelo nó parametrizado, no cálculo das expressões acima citadas. Esse é o principal ganho do algoritmo proposto; note que não é simplesmente o uso de Loopy Propagation em um grafo convenientemente modificado, mas sim a propagação de crenças em primeira ordem uma vez que as mensagens enviadas da parte parametrizada da rede são replicadas não sendo necessária a instanciação de todos os indivíduos da terminologia.

As únicas mensagens enviadas entre fatias são decorrentes dos nós de restrição, que são a única conexão entre fatias distintas, sendo que $P((\forall r.C)(a_j))$ recebe uma mensagem de cada $C(a_k) : a_k \in \mathcal{D}'$ e recebe $N - M$ mensagens idênticas vindas de $C(x)$.

Exemplo 5 *Considere novamente $\mathcal{T}_1$ do Exemplo 3, com a consulta $P(C(a_0))$. O nó $(\forall r.A)(a_0)$ recebe mensagens das instanciações de A e de r, as propagando para o nó $D(a_0)$. O nó $A(a_0)$ envia uma* única *mensagem para $(\forall r.A)(a_0)$ enquanto o nó $A(x)$ envia $N-1$ mensagens* idênticas *para $(\forall r.A)(a_0)$. Assim, o cálculo da Equação 4.4 fica*

$$P((\forall r.A)(a_0)) = (1-P(r)(1-P(A)))^{N-1}(1-P(r)(1-P(A(a_0)|B(a_0))).$$

Como resultado dessa divisão do grafo, dadas as mensagens recebidas em uma fatia, a inferência de probabilidades dentro dela fica independente de todos os outros nós da rede. Assim, pode se realizar o cálculo de inferência exata dentro de cada fatia, restando apenas a propagação de mensagens entre fatias. A Figura 4.4 ilustra a troca de mensagens entre fatias.

Considere a fatia do indivíduo a. O produto da multiplicação de todas as distribuições de probabilidades de seus nós, com exceção dos indicadores das restrições universal e exis-

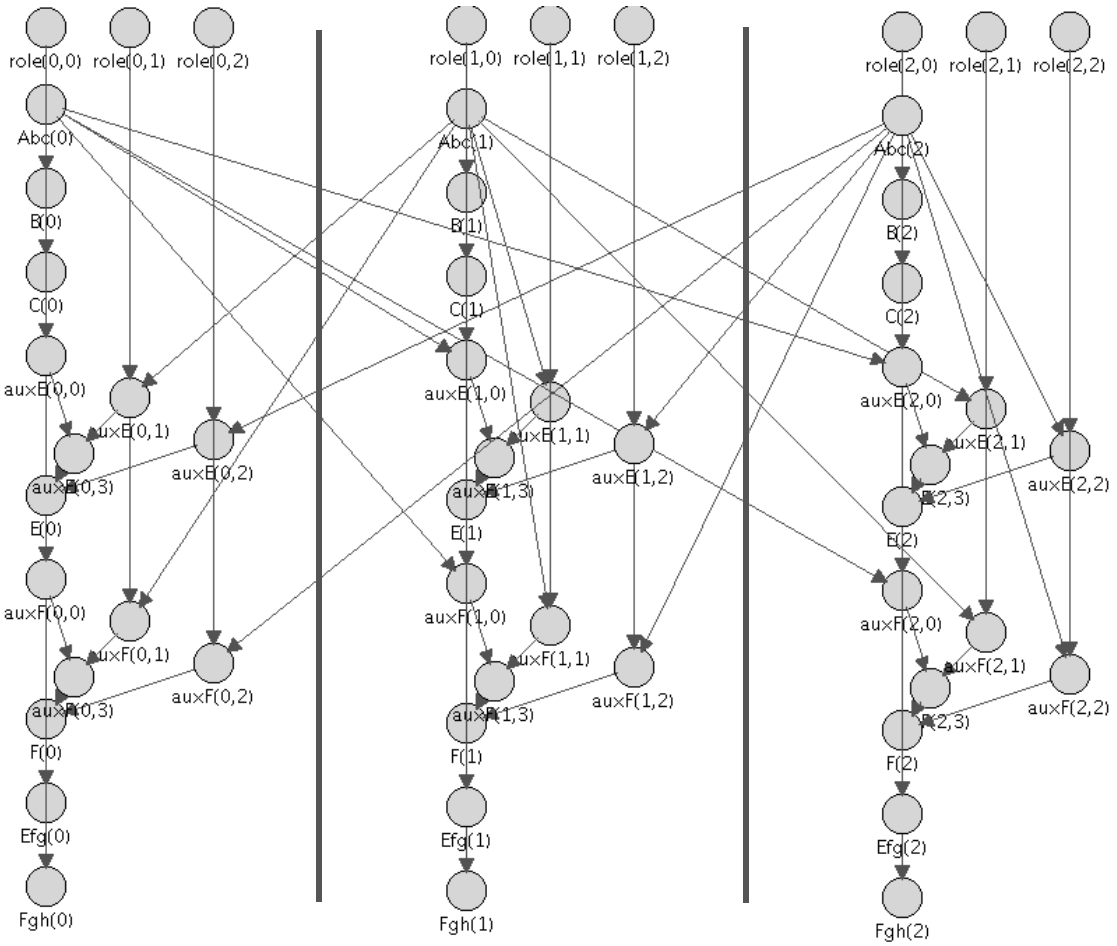


Figura 4.3: Fatias de uma rede instanciada para um domínio com 3 indivíduos.

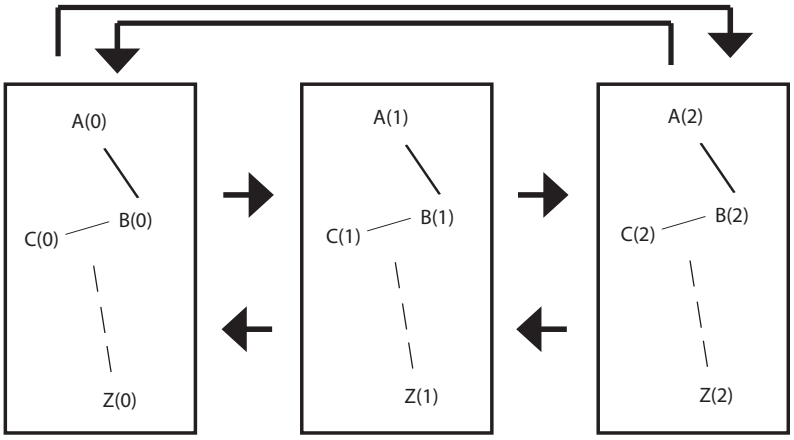


Figura 4.4: Inferência exata é utilizada dentro de fatia e os resultados são propagados para as demais.

tencial, é chamado de *fator de a* (f_a). O grafo para a fatia de a é composto então pelo fator f_a e pelos fatores correspondentes às restrições, junto com as variáveis associadas a esses fatores. Denomina-se $\mathbb{B}_a$ a rede Bayesiana com todos os nós usados em f_a; os nós de $\mathbb{B}_a$ que não são instanciações do indivíduo a são referenciados como nós *exteriores*. Por exemplo, para uma restrição $(\forall r.B)(a)$ em $\mathbb{B}_a$, os nós $B(y) : y \neq a$ são nós exteriores. Denomina-se por $\mathbf{E}_a$ as variáveis em $\mathbb{B}_a$ correspondentes às evidencias da consulta (também chamadas de *Abox*). Adiciona-se ainda ao grafo instanciações para um indivíduo auxiliar $b \notin \mathcal{D}'$ que é usado para o cálculo das mensagens enviadas aos indivíduos não pertencentes a $\mathcal{D}'$.

Com o grafo pronto, basta então propagar as mensagens propostas por Yedidia, Freeman e Weiss (2005, Eqs. (4), (5)) até a convergência dos resultados. Essas mensagens são uma generalização do loop propagation. No loop propagation tradicional, cada nó recebe uma mensagem e a repassa para os nós vizinhos, sendo que o valor da mensagem é igual à probabilidade do nó dadas as evidências que estão antes do próprio nó no sentido de propagação da mensagem. A generalização do loop propagation utiliza, ao invés da própria probabilidade, uma função potencial chamada de *energia livre*, que permite assim não só a propagação de mensagens entre variáveis, mas também entre conjuntos de variáveis.

Uma forma simples e eficiente de se calcular essas mensagens é com a utilização de *grafos de fatores* (*factor graphs*), que são grafos unidirecionais contendo nós circulares que representam variáveis e nós quadrados que representam fatores de distribuição conjunta de probabilidades; sempre que um arco liga um nó circular a um nó quadrado temos que a variável do primeiro faz parte do domínio do segundo. Cada fator $f_j(\mathbf{X}_j)$ é uma distribuição de probabilidade condicionais, onde $\mathbf{X}_j$ é o conjunto de funções de indicação

das variáveis instanciadas. Considerando novamente as fatias da rede shattered, para um conceito A e com $a \in \mathcal{D}' \cup \{b\}$, as mensagens propagadas ficam na seguinte forma:

$$n_{A(a)\to f_{a'}}(A(a)) = \prod_{f \in G(A(a))\setminus f_{a'}} m_{f\to A(a)}(A(a)), \tag{4.6}$$

$$m_{f_{a'}\to A(a)}(A(a)) = \sum_{\mathbf{X}_{a'}\setminus A(a)} f_{a'}(\mathbf{X}_{a'}) \prod_{\hat{A}(\hat{a}) \in G(f_{a'})\setminus A(a)} n_{\hat{A}(\hat{a})\to f_{a'}}(\hat{A}(\hat{a})), \tag{4.7}$$

onde $G(\cdot)$ denomina seus vizinhos no grafo.

Considere o cálculo da mensagem $m_{f_a\to A(a)}$, enviada do fator f_a para o nó $A(a)$. As distribuições provenientes de nós exteriores são: mensagem $n_{B(a_i)\to f_a}$ de um indivíduo a_i, ou mensagem $n_{B(b)\to f_a}$ do indivíduo genérico b, ou uma mensagem parametrizada $n_{B(x)\to f_a}$. A única questão que fica é como computar as $N - M$ mensagens $n_{B(x)\to f_a}$ de uma só vez, sem ter que computar cada uma delas. mas note que $n_{B(x)\to f_a}$ é exatamente igual a $n_{B(b)\to f_a}$, uma vez que qualquer x se comporta como b. Então basta calcular a mensagem de b e utilizar o resultado para os elementos restantes representados por x (esse é o motivo da instanciação do indivíduo b separadamente dos indivíduos em x). Assim, para uma restrição como $(\forall r.B)(a)$; precisa-se $P(\bigwedge_x r(a,x) \to B(x))$, aproximado por $(1 - P(r(a,x))\,(1 - n_{B(b)\to f_a}))^{N-M}$. Em resumo, a mensagem $m_{f_a\to A(a)}$ é o resultado de inferência probabilística em $\mathbb{B}_a$ utilizando-se de valores específicos de probabilidades para os nós exteriores; de fato, temos que $m_{f_a\to A(a)}$ é proporcional a $P(A(a), \mathbf{E}_a)$ em $\mathbb{B}_a$. Similarmente, mensagens $m_{f_b\to A(b)}$ e $m_{f_a\to A(b)}$ podem ser computadas pelo mesmo princípio. Da mesma forma, as mensagens $n_{A(a)\to f_a}$ podem ser interpretadas $P(A(a), \mathbf{E}_a)\times \prod_{f \in N(A(a)\setminus f_a} P(\mathbf{E}_b|A(a))$, com seus termos computados em $\mathbb{B}_a$. Ou seja, as mensagens podem ser computadas como inferência em redes Bayesianas apropriadas.

O algoritmo acima proposto foi implementado em um pacote de software, e uma série de experimentos foi realizada com finalidade de testar seu desempenho em comparação com algoritmos consagrados na literatura. Detalhes sobre as características e funcionalidades do software encontram-se no Capítulo 5 deste texto, enquanto que os resultados dos testes são apresentados no Capítulo 6.

5 *Implementação*

Foi implementado um pacote de software que, a partir de uma terminologia especificada em CR$\mathcal{ALC}$, cria a rede shattered, instancia as fatias relevantes e executa o algoritmo de Loopy Propagation em primeira ordem proposto no capítulo anterior.

A linguagem de programação utilizada para tal foi a linguagem Java versão 6 da Oracle[1]. A escolha da Java foi feita por ser uma tecnologia multi-plataforma, não requerendo compilações especificas para os diferentes sistemas operacionais existentes, além de ser uma tecnologia gratuita e bastante difundida. Como trata-se de um pacote com um algoritmo de inferência probabilística, poderia ser desejável a utilização de uma linguagem de melhor desempenho computacional como C/C++ ou ainda Matlab. Porém o desempenho do algoritmo de primeira ordem é computacionalmente barato o suficiente para adotar-se o Java, que da uma flexibilidade maior para o usuário em detrimento da performance.

O software dispõe de dois modos distintos de utilização, um através de uma interface gráfica, e outro através de prompt de linha de comando para execução em lote. No modo com interface gráfica é exibida a tela do programa, que pode ser vista na Figura 5.1 enquanto que no modo de linha de comando, os resultados são exibidos em texto na tela, como visto na Figura 5.2, e salvos em um arquivo de saída. A escolha entre os dois modos é realizada na chamada do programa, de acordo com parâmetros opcionais que podem ser fornecidos. Por default, na ausência de parâmetros, é carregada a interface gráfica, mas a escolha pode ser feita utilizando-se o parâmetro `-gui`:

```
[-gui {on|off}] ativa ou desativa a interface gráfica.
```

No caso de escolha do modo de linha de comando, mais parâmetros devem ser informados:

[1]Disponível em `http://www.java.com`.

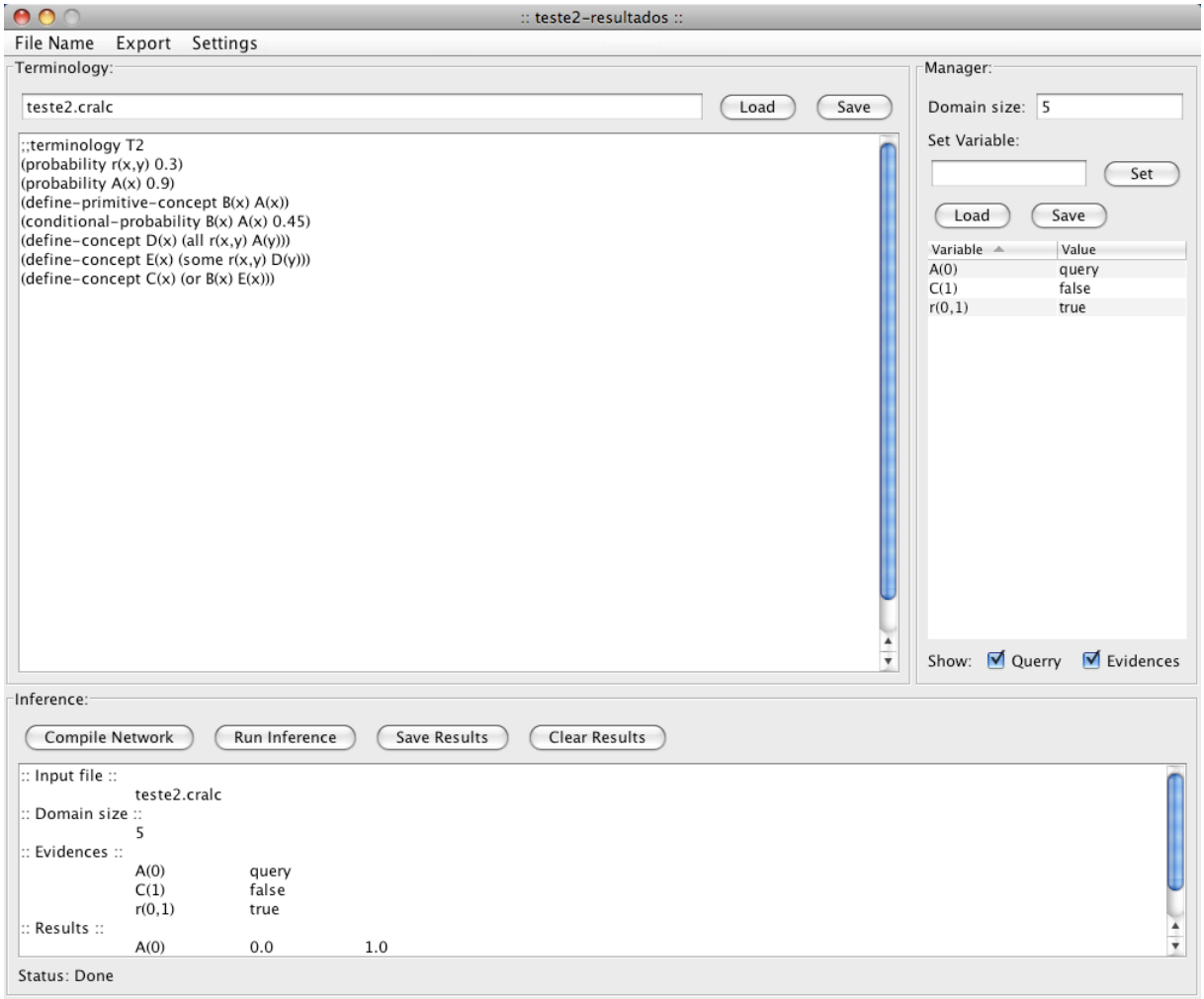


Figura 5.1: Interface gráfica do pacote de software.

```
Reading input file "teste2.cralc"...
File successfully red
Compiling network
Reading evidence file...
    A(0)    query
    C(1)    false
    r(0,1)  true
Evidences succefully red
Generating slices
Running inference... please wait...
Inference took 32ms (17 iterations).
:: Results ::
    A(0) 0.0 1.0
Exporting network: propositionalizing network to teste2-prop.bif
File successfully written
Done
```

Figura 5.2: Resultados na execução em modo de linha de comando.

```
[-i input_file N] carrega a terminologia especificada num
arquivo de entrada e estabelece um domínio de tamanho 'N'.
```

Alem de informar a terminologia e domínio, no modo de linha de comando, deve utilizar ao menos uma das duas opções a saber: i. exportar uma rede Bayesiana com a proposicionalização da terminologia

```
[-p output_file] proposicionaliza a terminologia e a exporta
como uma rede Bayesiana.
```

ou ii. informar as variáveis da consulta, as variáveis observadas, e um nome para os arquivos dos resultados.

```
[-e query_file] carrega a lista com as variáveis da con-
sulta e com as variáveis observadas.
[-r name] salva os resultados da inferência no arquivo
'name.txt' e salva cada fatia em um arquivo com nome se-
gundo o padrão name_SliceN.bif.
```

Os parâmetros apresentados podem ser utilizados em qualquer ordem, e não só no modo de linha de comando mas também na utilização com interface gráfica. Na interface gráfica, caso os parâmetros não sejam informados na chamada do programa, os campos da interface são exibidos sem nenhum preenchimento; caso contrário os campos são exibidos já preenchidos; em ambos os casos o usuário fica livre para editar e alterar o conteúdo dos campos conforme desejado. A janela do programa é dividida em três regiões principais: superior esquerda, superior direita e inferior; organizadas como visto a seguir.

A região superior esquerda contém um campo para a especificação da terminologia em CR$\mathcal{ALC}$ alem de botões para salvar ou carregar a especificação em um arquivo de texto, como pode-se ver na Figura 5.3. Detalhes sobre o formato desse arquivo serão informados mais a diante, no fim dessa seção.

A região superior direita contém um campo para definir o tamanho do domínio a ser utilizado, um campo para adicionar uma variável à lista de observações e consultas, uma tabela contendo a lista de todas as variáveis observadas e todas as consultas, botões para salvar ou carregar a lista da tabela, e por fim duas caixas de seleção que permitem filtrar a exibição tanto das observações quanto das consultas. Para salvar/carregar a lista,

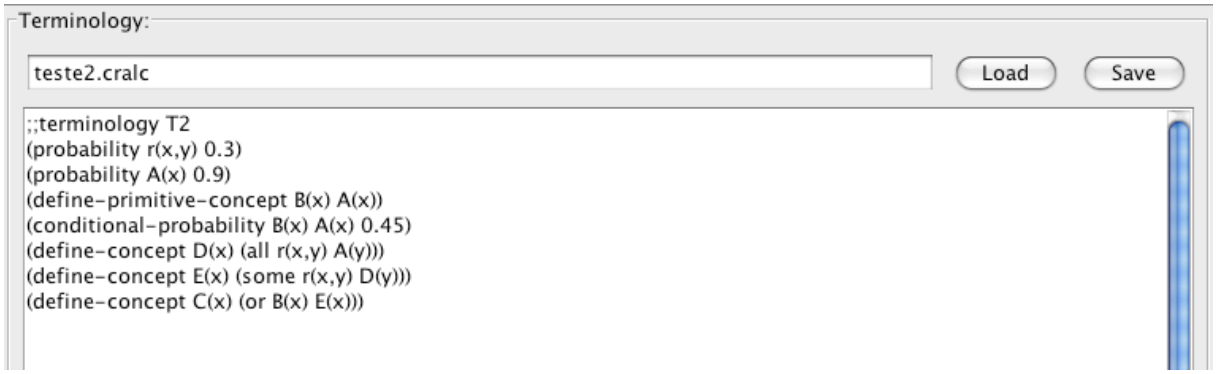
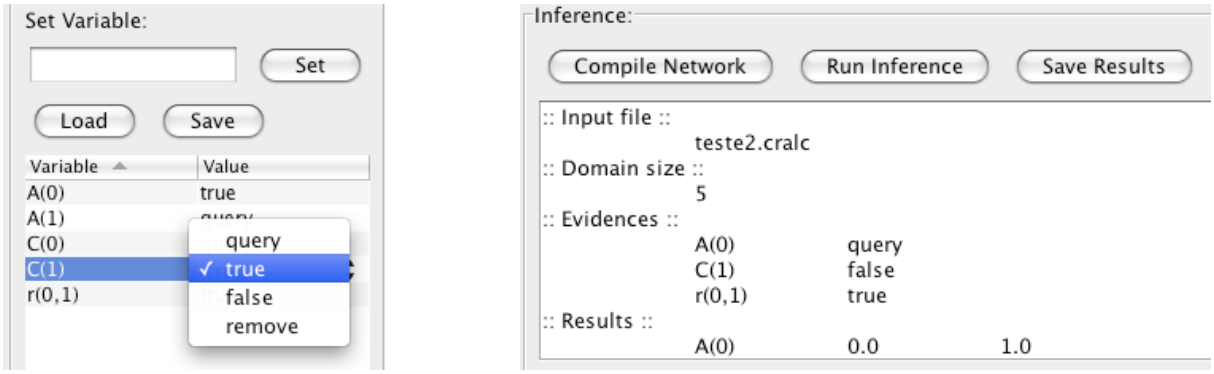


Figura 5.3: Especificação da terminologia.

caso um arquivo não tenha sido anteriormente indicado, aparecerá uma caixa de diálogo para a seleção do arquivo. Para incluir manualmente uma variável à lista, o nome da variável (conceitos ou papeis instanciados para indivíduos específicos do domínio, ou seja, $C(1)$, $r(0,1)$, etc) deve ser adicionado pelo campo `Set Variable`; a variável aparecerá na lista porém não estará atribuído a ela nenhum valor. Para atribuir um valor à variável (ou alterar o valor da mesma) deve-se clicar na linha da tabela correspondente a ela e selecionar um valor dentre as opções que aparecerão: `query` a estabelece como variável de consulta, `true` observa a variável como *verdadeiro*, `false` observa a variável como *falso* e `remove` exclui a variável da lista. Na Figura 5.4(a) pode-se ver um exemplo onde está sendo associado o valor `true` para a variável $C(1)$.

A região inferior contém os botões a serem utilizados para executar a inferência e dispõe de um campo para a exibição dos resultados. O botão `Compile Network` gera as fatias do modelo de acordo com as observações e consultas, exportando-as automaticamente como redes Bayesianas em arquivos `.bif`, enquanto que o botão `Run Inference` executa a inferência e exibe os resultados, como pode ser visto na Figura 5.4(b). Independentemente dos parâmetros utilizados na chamada do programa, no modo com interface gráfica os resultados não são automaticamente salvos (diferentemente do modo para linha de comando). Para salvar os resultados, é necessário clicar no botão `Save Results`, que os escreverá num arquivo com o nome fornecido para os resultados (parâmetro `[-r name]`). Na ausência de um nome, será utilizado o nome *Untitled*; para fácil referência, o nome dos resultados aparece no título da janela da interface.

Por fim, a janela da interface gráfica ainda apresenta um campo de status na borda inferior da janela, onde é exibido feedback das atividades em andamento. A janela conta também com três menus na borda superior da janela. O primeiro menu, `File Name`, contém duas opções: `Results` permite definir e alterar o nome do arquivo onde serão salvos



(a) Inclusão de evidência e consulta.

(b) Exibição dos resultados.

Figura 5.4: Consulta, evidência e resultado.

os resultados (assim como o nome dos arquivos com as fatias e o titulo da janela), enquanto que `Observations` permite definir e alterar o nome do arquivo com a lista de observações e consultas. O segundo menu, `Export`, contém apenas uma opção, para proposicionalizar a terminologia carregada e salvá-la como uma rede Bayesiana. E finalmente o terceiro menu, `Settings`, apresenta opções de configuração do programa: `Verbose` ativa a exibição de informações adicionais sobre o andamento das tarefas no campo de resultados da janela; `File type` permite escolher o formato a ser utilizado para exportar as fatias e para exportar a rede proposicionalizada; e `Inference` permite escolher o método de inferência a ser utilizado dentro das fatias.

Arquivos de entrada: A linguagem escolhida para representar uma terminologia em CR$\mathcal{ALC}$ segue a *Description-Logic Knowledge Representation System Specification* (KRSS). A especificação completa da KRSS encontra-se disponível em `http://dl.kr.org/krss-spec.ps` e embora seja uma especificação completa para lógicas de descrição, a CR$\mathcal{ALC}$ utiliza apenas um pequeno subconjunto da especificação, listado abaixo.

`(and C`$_1$`...C`$_n$`)`	$C_1 \sqcap \ldots \sqcap C_n$
`(or C`$_1$`...C`$_n$`)`	$C_1 \sqcup \ldots \sqcup C_n$
`(not C)`	$\neg C$
`(all r C)`	$\forall r.C$
`(some r C)`	$\exists r.C$
`(define-concept C D)`	$C \equiv D$
`(define-primitive-concept C D)`	$C \sqsubseteq D$

Note que em `(define-concept C D)` e em `(define-primitive-concept C D)`, `D` pode ser substituído por qualquer expressão tratável pela CR$\mathcal{ALC}$, com uma exceção: quantificadores devem ser definidos separadamente. Por exemplo, $A \sqsubseteq B \sqcup \forall r.C$ deve ser escrito como $A \sqsubseteq B \sqcup D$ com $D \equiv \forall r.C$

Inclusões de probabilidade são representadas por:

`(probability B` α`)`	$P(B) = \alpha$
`(conditional-probability B A` α`)`	$P(B\|A) = \alpha$

e assim como no caso anterior, onde temos `(conditional-probability B A` α`)`, `A` pode ser substituído por qualquer expressão tratável pela CR$\mathcal{ALC}$, salvo a exceção já descrita.

Quebras de linha, espaços e tabulações podem ser utilizados para formatar o arquivo e comentários podem ser adicionados caracterizados por '`;;`'. Abaixo pode-se ver um exemplo de um arquivo válido.

```
;; This is a valid input file
(probability A(x) 0.7)
(probability B(x) 0.4)
(define-concept C(x)
    (and
         A(x)           ;; note that comments can be
         (not B(x) )    ;; used in the middle of sentences.
    )
)
```

Os arquivos contendo as variáveis da consulta e as observações são ainda mais simples. Cada linha deve conter o nome da variável seguido do valor atribuído ao mesmo (`query`, `true` e `false`). Comentários são permitidos, no mesmo formato já descrito.

```
;; This is a valid file
A(1)    query
B(0)    true
r(0,1)  false
```

6 Experimentos

Neste capítulo serão mostrados alguns experimentos realizados para testar o desempenho do algoritmo proposto, testar a expressividade da CR$\mathcal{ALC}$ assim como a facilidade de se adicionar extensões à mesma. Todos os testes foram realizados em um computador com processador Intel Core 2 Duo 2.0GHz e 2GB de memória RAM rodando sistema operacional Linux Ubuntu.

Sempre que relevantes, foram realizados testes comparativos com outros algoritmos de inferência já consagrados como os disponíveis nos pacotes SAMIAM[1] e ACE[2] (para realizar inferência exata), e no pacote 2UBayes[3] (para realizar inferência aproximada com Loop Belief Propagation). Nestes testes foram observados não só o valor inferido, como também o tempo gasto no cálculo da inferência. Foram considerados apenas os tempos da inferência propriamente dita, sendo desprezados os tempos necessários para o carregamento dos modelos na memória do computador.

As sessões abaixo apresentam detalhes em específico dos testes realizados.

6.1 Terminologia de Teste

Primeiramente considere a terminologia apresentada no Exemplo 3. Apesar de simples, ela contém todos os elementos considerados pela CR$\mathcal{ALC}$ e vale como um primeiro teste embora a terminologia em si não tenha nenhum significado prático.

Novamente, temos $P(A) = 0.9$, $B \sqsubseteq A$, $P(B|A) = 0.45$, $D \equiv \forall r.A$, $C \equiv B \sqcup \exists r.D$ e $P(r) = 0.3$. Como teste, consideremos a inferência sobre $P(C\,(0))$. A terminologia foi instanciada para diversos domínios de tamanhos crescentes e as inferências foram realizadas até onde foi possível com os recursos computacionais.

[1] disponível em `http://reasoning.cs.ucla.edu/samiam/index.php`
[2] disponível `http://reasoning.cs.ucla.edu/ace/`
[3] disponível em `http://www.pmr.poli.usp.br/ltd/Software/2UBayes/2UBayes.html`

Os domínios foram incrementados de 1 elemento até se chegar em 10; de 10 a 100, os incrementos foram de 10 elementos; e de 100 a 500 foram utilizados incrementos de 100. Foi realizada então a inferência de $P(C\,(0))$. A Tabela 6.1 mostra os resultados obtidos com o algoritmo proposto em comparação com os resultados de inferência exata do ACE e inferência de Loop Belief Propagation do 2UBayes.

Tabela 6.1: Resultado da inferência $P(C\,(0))$ em $\mathcal{T}_1$ para um domínio com N indivíduos.

N	1	5	9	20	50	200	500
Exata	0.5535	0.8445	0.9210	—	—	—	—
LBP	0.5781	0.8558	0.9421	0.9831	0.9798	—	—
Proposta	0.5335	0.8445	0.9285	0.9774	0.9739	0.4723	0.4050

A inferência exata foi possível até a rede instanciada para o domínio com 9 elementos. A inferência aproximada por LBP foi possível até a rede instanciada para 50 elementos. A versão em primeira ordem, foi realizada para todas as redes instanciadas, sendo que de 300 em diante, foi obtido sempre o mesmo resultado, $P(C\,(0)) = 0.4050$, que é o resultado teórico previsto para para $N = \infty$ (COZMAN; POLASTRO, 2008).

Note que seria um engano tomar o resultado obtido com 50 indivíduos (máximo possível com o LBP) como aproximação de um domínio infinito. A variação do resultado pode parecer suspeita já que primeiramente o valor inferido aumenta junto com o tamanho do domínio para depois cair abruptamente. Essa variação acontece uma vez que C é definido a partir do quantificador $\exists r.D$ e D é definido a partir do quantificador $\forall r.A$. A probabilidade do $\exists$ tende a 1 para termos de probabilidade não nula, enquanto que a probabilidade do $\forall$ tende a zero a não ser que todos os termos tenham probabilidade 1. Assim, para a terminologia em questão, com o aumento do número de indivíduos no domínio $P(\exists r.D)$ se aproxima de 1, elevando $P(C\,(0))$ até que $P(\forall r.A)$ caia para zero (ou muito próximo de zero), levando consigo a probabilidade do $\exists r.D$ para zero também, fazendo com que $P(C\,(0))$ diminua. Na Tabela 6.2 pode-se ver a variação de $P(\exists r\,(0,y)\,.D\,(y))$ e $P(\forall r\,(0,y)\,.A\,(y))$ de acordo com o tamanho do domínio. A partir do domínio de tamanho 327, a probabilidade dos quantificadores chegou a zero, fazendo com que a inferência tivesse o mesmo valor esperado para um domínio infinito.

Para encerrar, temos na Tabela 6.3 o tempo dispendido pelos algoritmos para se chegar nas soluções apresentadas na Tabela 6.1. Note que os resultados de inferência exata e de inferência aproximada crescem exponencialmente com o tamanho do domínio, uma vez que

Tabela 6.2: Variação das probabilidade dos quantificadores com a mudança do domínio.

N		01	05	09	15	20	30	40	50
$P(\forall r.A)$	Exata	0.9700	0.8587	0.7602	—	—	—	—	—
	Proposta	0.9700	0.8587	0.7602	0.6332	0.5438	0.4010	0.2958	0.2181
$P(\exists r.D)$	Exata	0.2700	0.7610	0.8774	—	—	—	—	—
	Proposta	0.2700	0.7448	0.8823	0.9455	0.9625	0.9713	0.9678	0.9563
N		60	70	80	90	100	200	327	500
$P(\forall r.A)$	Exata	—	—	—	—	—	—	—	—
	Proposta	0.1668	0.1186	0.0875	0.0645	0.0475	0.0022	0.0000	0.0000
$P(\exists r.D)$	Exata	—	—	—	—	—	—	—	—
	Proposta	0.9361	0.9046	0.8614	0.8050	0.7352	0.1132	0.0000	0.0000

a rede instanciada também cresce exponencialmente com o mesmo: um papel $r(x, y)$ acaba gerando N^2 nós na rede e os quantificadores $\forall r(x, y).A(y)$ e $\exists r(x, y).A(y)$ também geram N^2 nós, além de alguns nós auxiliares, para combinar todos os indivíduos do domínio. Já o algoritmo aqui proposto mostrou um desempenho invariante com o tamanho do domínio; como a rede é dividida em *fatias* e as fatias não observadas são computados apenas uma vez tendo suas informações replicadas, o desempenho do algoritmo proposto depende apenas do número de indivíduos observados, e não do número total de indivíduos do domínio.

Tabela 6.3: Tempo (ms) demandado para realizar a inferência em $\mathcal{T}_1$ para um domínio com N indivíduos.

N	1	5	9	20	50	200	500
Exata	1	32	1.361	—	—	—	—
LBP	25	1.161	3.518	48.635	5.420.497	—	—
Proposta	32	30	30	31	29	29	30

Considerando-se o pior caso, onde todos os elementos do domínio são observados, o algoritmo só propaga mensagem entre as fatias, e não entre todos os nós da rede, como o LBP. Por exemplo, para um domínio com 10 indivíduos, com todos eles observados, o algoritmo propaga informação entre 10 fatias, enquanto que o LBP propaga informação entre todos os nós da rede, no caso 539 nós. Na Tabela 6.4 podemos ver um segundo teste, no mesmo padrão do primeiro, porém com uma observação para cada indivíduo do domínio. A tabela mostra o tempo necessário para realizar o calculo da inferência, assim como numero de iterações que foram necessárias.

Tabela 6.4: Tempo (ms) e número de iterações demandado para realizar a inferência em $\mathcal{T}_1$ para um domínio com N indivíduos, com uma observação para cada indivíduo.

N		1	5	9	20	50	200	500
LBP	tempo	29	1.022	3.618	48.981	5.392.119	—	—
	iterações	4	13	17	31	91	—	—
Proposta	tempo	28	438	709	3.025	24.404	1.086.459	18.493.955
	iterações	1	3	3	3	3	3	3

6.2 Ontologia Kangaroo

Um segundo experimento foi realizado, nos mesmos moldes do primeiro experimento. Dessa vez a terminologia foi adaptada de uma ontologia de um arquivo de exemplo do sistema *CEL*[4] disponível em `http://lat.inf.tu-dresden.de/~meng/ontologies/kangaroo.cl` no qual são representados humanos e cangurus, assim como relações pai-filho diferenciadas para ambos casos.

Novamente, a preocupação maior é devida a definições como:

$$\mathsf{Father} \equiv \mathsf{Human} \sqcap \forall \mathsf{hasChild}.\mathsf{Human},$$

que, ao se proposicionalizar nos algoritmos tradicionais, gera na melhor hipótese uma rede proporcional a $|\mathcal{D}|^2$; ou seja, quadrática em relação ao tamanho do domínio. Esse tamanho depende da representação das distribuições de probabilidade. Em teoria, um nó com um quantificador tem tamanho exponencial $2^{|\mathcal{D}|}$, tratando-se de variáveis binárias, já que todos os indivíduos do domínio aparecem como pai do quantificador. Na prática, pode-se quebrar um quantificador segundo uma série de nós auxiliares combinados dois a dois, assim cada quantificador é contado como complexidade linear, porém, pela quantidade de nós da rede, mantém-se o problema de ordem quadrática.

A terminologia completa é definida por:

$$\begin{aligned} P(\mathsf{Animal}) &= 0.9, \\ P(\mathsf{Rational}) &= 0.6, \\ P(\mathsf{hasChild}) &= 0.3, \end{aligned}$$

[4]CEL é um classificador para a lógica de descrição $\mathcal{EL}+$ com desempenho em tempo polinomial, disponível em `http://lat.inf.tu-dresden.de/systems/cel/`.

$$
\begin{aligned}
\mathsf{Human} &\equiv \mathsf{Animal} \sqcap \mathsf{Rational}, \\
\mathsf{Beast} &\equiv \mathsf{Animal} \sqcap \neg\mathsf{Rational}, \\
\mathsf{Kangaroo} &\sqsubseteq \mathsf{Beast}, \\
P(\mathsf{Kangaroo}|\mathsf{Beast}) &= 0.4, \\
\mathsf{Parent} &\equiv \mathsf{Human} \sqcap \exists\mathsf{hasChild}.\mathsf{Human}, \\
\mathsf{MaternityKangaroo} &\equiv \mathsf{Kangaroo} \sqcap \exists\mathsf{hasChild}.\mathsf{Kangaroo}.
\end{aligned}
$$

Assim como no primeiro experimento, a terminologia foi instanciada para diversos domínios de tamanhos crescentes e as inferências foram realizadas até onde foi possível com os recursos computacionais. Os domínios foram incrementados de 1 elemento até se chegar em 10; de 10 a 100, os incrementos foram de 10 elementos; e de 100 a 500 foram utilizados incrementos de 100. Foi realizada então a inferência de $P(Parent\,(0))$. A Tabela 6.5 mostra os resultados obtidos com o algoritmo proposto em comparação com os resultados de inferência exata do ACE e inferência de Loop Belief Propagation do 2UBayes.

Tabela 6.5: Resultado da inferência $P(Parent\,(0))$ na Kangaroo para um domínio com N indivíduos.

N	1	5	9	20	50	200	500
Exata	0.1620	0.3536	0.4481	—	—	—	—
LBP	0.0875	0.3196	0.4299	0.5243	0.5392	—	—
Proposta	0.1620	0.3536	0.4481	0.5268	0.5399	5400	0.5400

A inferência exata foi possível até a rede instanciada para o domínio com 9 elementos. A inferência aproximada por LBP foi possível até a rede instanciada para 40 elementos. A versão em primeira ordem, foi realizada para todas as redes instanciadas, sendo que de 200 em diante, foi obtido sempre o mesmo resultado, $P(Parent\,(0)) = 0.5400$. Na Tabela 6.6 encontra-se o tempo dispendido pelos algoritmos para se chegar nas soluções apresentadas na Tabela 6.5.

Novamente, note que os resultados de inferência exata e de inferência aproximada crescem exponencialmente com o tamanho do domínio, enquanto que o algoritmo aqui proposto mostrou um desempenho invariante com o tamanho do domínio, dependendo apenas da quantidade de indivíduos observados.

Assim como visto no experimento anterior, considerando o pior caso onde todos os

Tabela 6.6: Tempo (ms) demandado para realizar a inferência na Kangaroo para um domínio com N indivíduos.

N	1	5	9	20	50	200	500
Exata	1	29	271	—	—	—	—
LBP	100	976	2.831	24.787	2.952.332	—	—
Proposta	32	32	34	31	36	32	30

indivíduos são observados, as mensagens só são propagadas entre as fatias em vez de serem propagadas entre todos os nós. Para um domínio com 10 indivíduos a rede instanciada apresenta 569 nós. Na Tabela 6.7 podemos ver um segundo teste, no mesmo padrão do primeiro, porém com uma observação para cada indivíduo do domínio. A tabela mostra o tempo necessário para realizar o calculo da inferência, assim como numero de iterações que foram necessárias.

Tabela 6.7: Tempo (ms) e número de iterações demandado para realizar a inferência na Kangaroo para um domínio com N indivíduos, com uma observação para cada indivíduo.

N		1	5	9	20	50	200	500
LBP	tempo	95	961	2.797	24.824	2.921.355	—	—
	iterações	7	9	12	16	47	—	—
Proposta	tempo	27	178	377	2.374	18.202	695.853	11.284.516
	iterações	1	3	3	3	3	3	3

6.3 Ontologia Wine e Utilização de Nominais

O terceiro experimento realizado, diferente dos anteriores, não tem como intuito avaliar o desempenho do algoritmo de inferência, mas sim a expressividade da lógica CR$\mathcal{ALC}$.

Lógicas de descrição tradicionais e probabilísticas tem sido bastante empregadas na comunidade de *Web Semântica*, que tenta atribuir significado aos dados de uma rede de informações de forma a tornar o sentido perceptível tanto para humanos quanto para máquinas. Nessa comunidade é frequente a utilização de elementos chamados *nominais*, que podem ser usados para:

i. definição de conceitos (ou atributos) através da denominação dos indivíduos que constituem os mesmos, em sentenças como

$$\mathsf{WineColor} \equiv \{\mathsf{Red}, \mathsf{White}, \mathsf{Rose}\},$$

onde Red, White e Rose são os nominais que especificam o conceito WineColor.

ii. estabelecer o domínio de papéis em restrições lógicas, em expressões como

$$\exists \mathsf{hasColor}.\{\mathsf{Red}\},$$

onde o nominal Red especifica o que a restrição considera o papel hasColor apenas quando for instanciado para o indivíduo que representa a cor vermelha.

A utilização desses nominais aumenta a expressividade das lógicas, porém, a um custo de processamento muito elevado mesmo para lógicas de descrição consagradas, podendo impossibilitar a inferência nas mesmas.

Assim, nesta seção serão representados nominais de forma limitada, permitindo apenas o uso de nominais de acordo com o segundo caso citado acima, de forma a possibilitar a utilização de sentenças lógicas do tipo

$$P(\mathsf{Merlot(a)}|\exists \mathsf{hasColor}.\{\mathsf{Red}\}) = \alpha.$$

Essa sentença se refere à probabilidade de um determinado vinho ser um Merlot dado que sua cor é vermelha. Para um papel r e um nominal $\{a\}$ a semântica aqui utilizada para uma restrição existencial $\exists r.\{a\}$ é definida por

$$\exists y : r\,(x, y) \wedge (y = a),$$

e a semântica para a restrição universal $\forall r.\{a\}$ é definida por

$$\forall y : r\,(x, y) \rightarrow (y = a).$$

Em restrições que contenham mais de um nominal, como em $r.\{a, b, c\}$, a restrição considera a disjunção das atribuições para a, b e c.

Quando utilizados nos papéis, os nominais restringem subconjuntos do domínio: se um papel $\mathsf{temPropriedade}(x, y)$ indica que o indivíduo x tem uma propriedade específica de valor y, então temos que x deve ser um indivíduo que representa o objeto sendo descrito

e y deve ser o indivíduo que representa o nominal que descreve a propriedade indicada pelo papel. Por exemplo, $\exists \mathsf{hasColor}.\{\mathsf{Red}\}$ é interpretado por

$$\exists y \in D : \mathsf{hasColor}\,(x, y) \wedge (y = \mathsf{red}),$$

onde $\mathcal{D}$ é o domínio que contém os elementos sendo descritos e y mapeia todos os nominais que representam cores. Entretanto, a lógica CR$\mathcal{ALC}$ (ou ainda a lógica $\mathcal{ALC}$) não permite o uso de nominais, uma vez que não se pode ter indivíduos na definição de conceitos. CR$\mathcal{ALC}$ também não prevê o uso de variáveis tipadas, o que resultaria em uma rede instanciada para um domínio com todos os indivíduos e nominais. Porém, como já foi visto, domínios muito grandes levam a problemas de recursos computacionais.

É necessária uma forma de se instanciar apenas a parte relevante aos nominais e aos indivíduos descritos. Para tanto, foi criado um conceito para cada possível instanciação de um papel e seus respectivos nominais de forma a substituir a expressão $\mathsf{hasColor}.\{\mathsf{Red}\}$ por um novo conceito $\mathsf{HasColorRed}(x)$ instanciado normalmente com o domínio contendo apenas os indivíduos a serem descritos pela terminologia.

Frequentemente nominais são definidos como mutualmente exclusivos. Embora CR$\mathcal{ALC}$ não tenha nenhum construtor para isso, o problema pode ser facilmente resolvido pela inclusão de um nó probabilístico que tenha os nós exclusivos como pais, e uma tabela de probabilidades condicionais que represente o operador lógico *OU Exclusivo*. Este nó deve ser observado como `verdadeiro` de forma a estabelecer a dependência entre seus pais.

Para ilustrar a questão foi utilizado um trecho da ontologia *Wine* que faz uso extenso dos nominais para descrever diferentes tipos de vinho e suas propriedades. É uma ontologia bem conhecida da literatura de lógicas de descrição, extraída de um arquivo OWL[5] A ontologia é bastante extensa, portanto será apresentado somente uma parte da mesma.

Iniciamos estabelecendo a probabilidade à priori de um indivíduo do domínio ser um vinho. O valor dessa probabilidade não influencia no resultado já que representaremos apenas vinhos e consequentemente os nós referentes a esse conceito sempre estarão observados.

$$P(\mathsf{Wine}) = 0.9.$$

Segue-se com os conceitos criados para os nominais que representam o tipo de uva do qual o vinho é feito relacionado pelo papel $\mathsf{madeFrom}(x, y)$ que associa o vinho à uva.

[5]disponível no repositório do *Temporal Knowledge Base Group from Universitat Jaume I*, `http://krono.act.uji.es/Links/ontologies/wine.owl/view` .

Note que o intuito do experimento é sobre a lógica em questão e não sobre a representação de vinhos. Assim, adotou-se uma distribuição uniforme entre os possíveis tipos de uva, ao invés de se obter as probabilidades de acordo com a produção mundial de vinhos.

$$
\begin{aligned}
P(\mathsf{MadeFromCabernetFrancGrape}) &= 0.06,\\
P(\mathsf{MadeFromCabernetSauvignonGrape}) &= 0.06,\\
P(\mathsf{MadeFromChardonnayGrape}) &= 0.06,\\
P(\mathsf{MadeFromCheninBlancGrape}) &= 0.06,\\
P(\mathsf{MadeFromGamayGrape}) &= 0.06,\\
P(\mathsf{MadeFromMalbecGrape}) &= 0.06,\\
P(\mathsf{MadeFromMerlotGrape}) &= 0.06,\\
P(\mathsf{MadeFromPetiteSyrahGrape}) &= 0.06,\\
P(\mathsf{MadeFromPetiteVerdotGrape}) &= 0.06,\\
P(\mathsf{MadeFromPinotBlancGrape}) &= 0.06,\\
P(\mathsf{MadeFromPinotNoirGrape}) &= 0.06,\\
P(\mathsf{MadeFromRieslingGrape}) &= 0.06,\\
P(\mathsf{MadeFromSangioveseGrape}) &= 0.06,\\
P(\mathsf{MadeFromSauvignonBlancGrape}) &= 0.06,\\
P(\mathsf{MadeFromSemillonGrape}) &= 0.06,\\
P(\mathsf{MadeFromZinfandelGrape}) &= 0.06.
\end{aligned}
$$

De forma análoga foram criados conceitos para os nominais que representam a região onde o vinho foi produzido, e o papel $\mathsf{locatedIn}(x, y)$ que associa o vinho à região. Inclusão de conceitos e probabilidades condicionais foram utilizadas para melhor representar o agrupamento das regiões. Novamente, foram utilizadas distribuições uniformes de probabilidade sempre que possível.

$$
\begin{aligned}
P(\mathsf{LocatedInAustralianRegion}) &= 0.15,\\
P(\mathsf{LocatedInFrenchRegion}) &= 0.15,\\
P(\mathsf{LocatedInGermanyRegion}) &= 0.15,\\
P(\mathsf{LocatedInItalianRegion}) &= 0.15,\\
P(\mathsf{LocatedInNewZealandRegion}) &= 0.15,
\end{aligned}
$$

$$
\begin{aligned}
P(\mathsf{LocatedInPortugalRegion}) &= 0.15, \\
P(\mathsf{LocatedInUSRegion}) &= 0.15.
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{LocatedInAlsaceRegion} &\sqsubseteq \mathsf{LocatedInFrenchRegion}, \\
P(\mathsf{LocatedInAlsaceRegion} \mid \mathsf{LocatedInFrenchRegion}) &= 0.20, \\
\mathsf{LocatedInBordeauxRegion} &\sqsubseteq \mathsf{LocatedInFrenchRegion}, \\
P(\mathsf{LocatedInBordeauxRegion} \mid \mathsf{LocatedInFrenchRegion}) &= 0.20, \\
&\vdots \\
\mathsf{LocatedInStEmilionRegion} &\sqsubseteq \mathsf{LocatedInBordeauxRegion}, \\
P(\mathsf{LocatedInStEmilionRegion} \mid \mathsf{LocatedInBordeauxRegion}) &= 0.33.
\end{aligned}
$$

Também foram criados conceitos para representar os nominais referentes à cor, sabor, corpo e teor de açúcar do vinho:

$$
\begin{aligned}
P(\mathsf{HasWineColorRed}) &= 0.33, \\
P(\mathsf{HasWineColorRose}) &= 0.33, \\
P(\mathsf{HasWineColorWhite}) &= 0.33.
\end{aligned}
$$

$$
\begin{aligned}
P(\mathsf{HasWineBodyLight}) &= 0.33, \\
P(\mathsf{HasWineBodyMedium}) &= 0.33, \\
P(\mathsf{HasWineBodyFull}) &= 0.33.
\end{aligned}
$$

$$
\begin{aligned}
P(\mathsf{HasWineFlavourDelicate}) &= 0.33, \\
P(\mathsf{HasWineFlavourModerate}) &= 0.33, \\
P(\mathsf{HasWineFlavourStrong}) &= 0.33.
\end{aligned}
$$

$$
\begin{aligned}
P(\mathsf{HasWineSugarDry}) &= 0.33, \\
P(\mathsf{HasWineOffDry}) &= 0.33, \\
P(\mathsf{HasWineSweet}) &= 0.33.
\end{aligned}
$$

Por fim, são apresentados os construtores usados para definir os diversos tipos de vinho de acordo com suas características. É uma ontologia bastante extensa e, por isso, será

apresentada apenas uma fração da mesma. Para uma versão completa da ontologia, consulte o Apêndice A que apresenta o conteúdo do arquivo KRSS com a ontologia.

$$
\begin{aligned}
\mathsf{RedWine} &\equiv \mathsf{Wine} \sqcap \mathsf{HasWineColorRed};\\
\mathsf{LateHarvest} &\equiv \mathsf{Wine}\\
&\quad \sqcap (\mathsf{HasWineFlavourModerate} \sqcup \mathsf{HasWineFlavourStrong})\\
&\quad \sqcap \mathsf{HasWineSugarSweet};\\
\mathsf{DessertWine} &\equiv \mathsf{Wine}\\
&\quad \sqcap (\mathsf{HasWineSugarOffDry} \sqcup \mathsf{HasWineSugarSweet});\\
&\vdots\\
\mathsf{IceWine} &\equiv \mathsf{DessertWine}\\
&\quad \sqcap \mathsf{LateHarvest}\\
&\quad \sqcap \mathsf{HasWineColorWhite}\\
&\quad \sqcap (\mathsf{HasWineBodyMedium} \sqcup \mathsf{HasWineBodyFull})\\
&\quad \sqcap (\mathsf{HasWineFlavourModerate} \sqcup \mathsf{HasWineFlavourStrong});\\
\mathsf{Port} &\equiv \mathsf{RedWine}\\
&\quad \sqcap \mathsf{HasWineBodyFull}\\
&\quad \sqcap \mathsf{HasWineFlavourStrong}\\
&\quad \sqcap \mathsf{HasWineSugarSweet}\\
&\quad \sqcap \mathsf{LocatedInPortugalRegion};\\
&\vdots\\
\mathsf{Merlot} &\equiv \mathsf{Wine}\\
&\quad \sqcap \mathsf{MadeFromMerlotGrape}\\
&\quad \sqcap (\mathsf{HasWineBodyLight} \sqcup \mathsf{HasWineBodyMedium})\\
&\quad \sqcap \mathsf{HasWineColorRed}\\
&\quad \sqcap (\mathsf{HasWineFlavourModerate} \sqcup \mathsf{HasWineFlavourDelicate})\\
&\quad \sqcap \mathsf{HasWineSugarDry}.
\end{aligned}
$$

Essa representação dos nominais acarreta em uma consequência de grande vantagem prática: os nominais não criam ligações entre indivíduos diferentes do domínio, ou seja, todos os conceitos gerados a partir de nominais para o indivíduo a_i são pertencentes à fatia de a_i. E como a ontologia não apresenta nenhuma restrição ligando diferentes indivíduos

do domínio (já que a classificação do vinho depende apenas de seus atributos, e não dos atributos dos vinhos vizinhos), a instanciação da ontologia gera uma rede Bayesiana com vários pedaços desconexos, de tamanho fixo, onde cada pedaço representa a instanciação de um vinho. Do ponto de vista de inferência, uma rede com N vinhos é equivalente a N redes com 1 vinho cada, eliminando qualquer preocupação quanto a escalabilidade do modelo e desempenho do algoritmo de inferência.

Foram feitos alguns conjuntos de observações sobre as propriedades dos vinhos sobre os quais foi realizada inferência exata para se obter $P(\mathsf{Merlot} \mid evidencias)$. Alguns resultados podem ser vistos na Tabela 6.8. Pode-se ver que a solução utilizada é viável para a representação de nominais e para a realização de inferência sobre os mesmos, tarefa que geralmente tem um custo computacional muito elevado em outras lógicas de descrição.

Tabela 6.8: Inferência realizada para 3 indivíduos distintos.

Observações		
$\mathsf{HasWineBodyMedium}(a)$ $\mathsf{HasWineColorRed}(a)$ $\mathsf{HasWineFlavourModerate}(a)$ $\mathsf{HasWineSugarDry}(a)$ $\mathsf{MadeFromMerlotGrape}(a)$	$\mathsf{HasWineBodyMedium}(b)$ $\mathsf{HasWineColorRed}(b)$ $\mathsf{HasWineFlavourModerate}(b)$ $\mathsf{HasWineSugarDry}(b)$	$\mathsf{HasWineSugarSweet}(c)$ $\mathsf{MadeFromMerlotGrape}(c)$
Inferência		
$P(\mathsf{Merlot}(a) \mid E(a)) = 1.0$	$P(\mathsf{Merlot}(b) \mid E(b)) = 0.5$	$P(\mathsf{Merlot}(c) \mid E(c)) = 0.0$

7 *Aplicações*

Neste capítulo são apresentadas aplicações práticas da lógica proposta e do algoritmo de inferência. Foram feitas duas aplicações na área de robótica móvel: a primeira aplicação envolve a extração de contexto, no qual é feita a representação de possíveis ambientes onde um robô pode se encontrar, e então é inferido em qual ambiente ele está com base nos objetos contidos no mesmo; a segunda aplicação utiliza as informações do contexto do robô para a melhoria de técnicas de mapeamento e criação de mapas semânticos.

7.1 Extração de Contexto em Robótica Móvel

Como primeira aplicação prática aborda-se o problema de extração de contexto em robótica móvel onde um conjunto de objetos é detectado por sensores do robô e deseja-se agrupar os objetos em descrições conceituais de alto nível. Por exemplo, um local contendo "cadeiras", "refrigerador" e "forno" deve ser rotulado como "cozinha". Este rótulo é dito *contexto* do robô. Métodos atuais de robótica computacional são capazes de detectar e classificar objetos presentes em imagens capturadas por câmeras. O intuito aqui é: ao reconhecer um conjunto de objetos, deseja-se saber se o robô encontra-se em um laboratório, estacionamento ou outro ambiente definido.

O desafio aqui é decorrente da incerteza comumente associada à descrição dos contextos (ambientes): duas instâncias do mesmo rótulo, na maioria das vezes, são diferentes entre si, i.e. dois escritórios diferentes contêm cadeiras e mesas mas a quantidade desses objetos é única para cada um deles; além de que um deles pode conter uma máquina de café e o outro não. Em um caso mais extremo, dois parques podem ser completamente diferentes em questão de vegetação e podem ter atributos exclusivos, mas mesmo assim precisam ser rotulados por um conceito único "parque". Essas incertezas tornam natural o uso de uma lógica de descrição probabilística. Embora algumas probabilidades possam ser facilmente obtidas – a probabilidade de um parque conter um pinheiro pode ser estimada por sua frequência relativa – a maioria das probabilidades envolvidas não são simples de

se obter. Considere a probabilidade de um papel r que represente que dois objetos do ambiente são relacionados: pode parecer óbvio que uma cadeira está relacionada a uma mesa em um escritório, mas a mesma cadeira não é relacionada a outras cadeiras. Como não há nenhuma informação à priori sobre quais objetos serão identificados pelo robô, a probabilidade de r não pode ser definida e deve ser mantida como uma probabilidade imprecisa no intervalo [0,1]. Para lidar com essas probabilidades imprecisas, foi usado o algoritmo de inferência L2U, do pacote 2UBayes, que implementa loopy propagation para *redes credais*, i.e. redes Bayesianas com probabilidades imprecisas. Por motivos práticos o intervalo [0.25, 0.75] foi adotado ao invés de [0,1] pois como será visto a seguir, a ontologia modelada só utiliza definição e inclusão de conceitos e, portanto, caso o intervalo [0,1] tivesse sido utilizado todas as inferências realizadas retornariam o resultado [0,1].

Para caracterizar o contexto do robô com a informação dos objetos, foi montada uma ontologia como especificado a seguir.

Considere C_0 o conjunto de todos os conceitos que representam um tipo de objeto a ser identificado pelo robô, como $Chair$, $Table$, Car, $Tree$ e assim por diante. Como não há informação a priori sobre os objetos que serão encontrados, as probabilidade de todos esses conceitos são desconhecidas e mantidas livres, como $P(C) \in [0.25, 0.75]$ para $C \in C_0$.

Um papel $contains(x, y)$ expressa relação entre instâncias de conceitos, representando que dois objetos x e y estão relacionados. Mais uma vez não há informação a priori, $P(contains(x, y)) \in [0.25, 0.75]$.

Novos conceitos são definidos com base nos já apresentados, de forma a facilitar a descrição dos ambientes, como por exemplo

$$Desk \equiv Table \wedge \exists contains.Chair.$$

Mais conceitos são adicionados para agrupar tipos de objetos ($IndoorObjects$, $HomeObjects$, $Plants$) e inclusão de conceitos é utilizada para estabelecer uma hierarquia entre eles:

$$Bed \sqsubseteq HomeObjects,$$

$$HomeObjects \sqsubseteq IndoorObjects,$$

$$Tree \sqsubseteq Plants.$$

Note que para dois conceitos C e D com $C \sqsubseteq D$ temos que $P(C|\neg D) = 0$ e $P(C|D) \in [0, 1]$.

Por fim, conceitos são criados para cada possível ambiente como $Office$, $Kitchen$ e $Park$ e são descritos pela combinação de seus conceitos característicos, como pode ser visto abaixo:

$$\begin{aligned} Office \equiv\ & \exists contains.Desk \\ \sqcap\ & \exists contains.Computer \\ \sqcap\ & \neg\exists contains.HomeObjects \\ \sqcap\ & \neg\exists contains.OutdoorObjects. \end{aligned}$$

Note que os contextos foram descritos por definição de conceitos ($\equiv$). Em exemplos não determinísticos, ainda seria possível a representação do contexto pelo uso e probabilidades condicionais na forma

$$P(Rótulo \mid \exists contains.Objeto1 \sqcap \ldots \sqcap \neg\exists contains.ObjetoN) = 0.8.$$

A ontologia inteira é muito extensa e não será apresentada aqui. Para uma versão completa da mesma, consulte o Apêndice B que apresenta o conteúdo do arquivo KRSS com a ontologia.

Depois de definida a ontologia, uma série de imagens foram capturadas por um robô móvel em alguns ambientes distintos. Utilizando-se de uma algoritmo SIFT (LOWE, 2004) foram identificados os objetos presentes no ambiente. Apesar de haver algoritmos mais robustos que o SIFT, os resultados apresentados pelo mesmo foram satisfatórios, uma vez que o estudo aqui é sobre o raciocínio sobre os objetos identificados e não sobre a identificação destes. Foi então instanciada uma *rede credal* (rede Bayesiana com probabilidades imprecisas) onde foram observados os objetos identificados no SIFT. Para essas redes foram feitas a inferência das probabilidades de cada rótulo, ou seja $P(Office|evidências)$, $P(Dormitory|evidências)$, etc. Também foi instanciado um indivíduo extra (a_j) na rede, que não foi observado, com o intuito de representar objetos não reconhecidos pelo sistema de visão; foi calculada para este indivíduo, as probabilidades de todos os objetos representados pela terminologia, assim, $P(a_j = tree)$ representa a probabilidade de se encontrar uma arvore no ambiente. Abaixo vemos dois dos testes realizados.

Teste 1 – Em um escritório.

Evidências (Objetos identificados pela câmera):

$$E = \{3\,\text{mesas}, 3\,\text{cadeiras}, 2\,\text{computadores}\}$$

.

Algumas das probabilidades inferidas:

$$\begin{aligned} P(Office|E) &= [0.53, 0.87] \\ P(Kitchen|E) &= [0.07, 0.39] \\ P(Park|E) &= [0.03, 0.21] \\ P(a_j = chair|E) &= [0.36, 0.82] \\ P(a_j = tree|E) &= [0.01, 0.49] \end{aligned}$$

Teste 2 – Em um parque.

Evidências (Objetos identificados pela câmera):

$$E = \{3\,\text{arbustos}, 1\,\text{gramado}, 1\,\text{cadeira.}^{1}\}$$

Algumas das probabilidades inferidas:

$$\begin{aligned} P(Park|E) &= [0.61, 0.94] \\ P(Desert|E) &= [0.10, 0.43] \\ P(Office|E) &= [0.06, 0.29] \\ P(a_j = tree|E) &= [0.18, 0.73] \\ P(a_j = computer|E) &= [0.01, 0.47] \end{aligned}$$

Apesar da complexidade do problema, os resultados obtidos foram bastante satisfatórios. Em todos os testes, foi instanciada a rede com um objeto a mais do que os identificados. Esse objeto adicional representa os objetos que podem ter passado desper-

[1] Um banco classificado como cadeira, já que não há a distinção dos dois na ontologia.

cebido pelos sensores do robô ou pelo SIFT, e flexibiliza o resultado das inferências, uma vez que ele pode representar qualquer objeto possível de ser identificado. Um único objeto adicional é suficiente já que os conceitos que representam os tipos de objetos não são exclusivos, de forma que todos podem ser observados como verdadeiro simultaneamente, e não há nenhuma restrição de cardinalidade.

Esse nó adicional também permite que seja calculada a probabilidade de cada tipo de objeto para um novo objeto a ser eventualmente encontrado no domínio, bastando para isso realizar a inferência de que esse objeto desconhecido seja uma cadeira, árvore, e assim por diante.

Uma alternativa para melhorar ainda mais o desempenho dessa ontologia seria utilizar uma estimativa inicial mais precisa da probabilidade de cada objeto. Uma mudança mais promissora seria modificar a probabilidade do papel que diz que dois objetos são relacionados: se o intervalo da probabilidade fosse correlacionado com a distância entre eles, possivelmente os resultados seriam ainda mais precisos.

Os resultados desta aplicação serviram como preparação para a aplicação descrita na seção a seguir. Aqui apenas nos preocupamos em obter os rótulos para os ambientes, enquanto que na próxima seção, utilizamos esses rótulos para auxiliar outra tarefa.

7.2 Rótulos e Mapeamento Semântico

Inspirado na primeira aplicação foi feita uma segunda ainda na área de robótica móvel, agora não apenas para extrair o contexto do robô mas sim utilizar essa informação para melhorar uma tarefa de mapeamento. Aplicamos aqui uma ontologia semelhante à utilizada na Seção 7.1 com o intuito de melhorar o desempenho de técnicas de mapeamento e facilitar a criação de mapas anotados ou ainda mapas semânticos, inspirado no trabalho de (HERTZBERG; SAFFIOTTI, 2008; ANGUELOV et al., 2005; GALINDO et al., 2008).

O desafio aqui é fazer o mapeamento tridimensional de uma área, separando dados de diferentes ambientes, de forma a criar um mapa com as anotações de rótulo de cada ambiente e também informações topológicas do mesmo. Por exemplo, quais ambientes estão ligados permitindo acesso direto de um ao outro. Além desse ganho semântico, o fato de poder dividir o mapa em pequenas partes deve ainda diminuir o custo computacional do mapeamento uma vez que pode-se computar cada parte separadamente, ao invés de tratar todos os dados de uma vez só.

Para tanto, percorre-se a área a ser mapeada com um robô que vai coletando imagens do local, dados de odometria e de um sensor laser. Tradicionalmente, utilizam-se todos os dados da odometria e laser em um algoritmo de SLAM. Ao invés disso, utilizou-se aqui uma ontologia probabilística para identificar os diferentes ambientes percorridos, de forma a realizar o SLAM em cada ambiente separadamente. A ontologia foi construída com base na apresentada na Seção 7.1, porém modificada para melhor atender o novo objetivo. Da mesma forma como foi feito anteriormente, as inferências são realizadas em redes instanciadas de acordo com objetos identificados por um algoritmo de SIFT nas imagens captadas pelo robô.

A ontologia é definida como segue.

Dois conceitos primitivos Object e $\mathsf{Environment}$ identificam respectivamente os objetos encontrados pelo robô e os ambientes a serem mapeados. CR$\mathcal{ALC}$ requer que sejam especificadas probabilidades a priori para conceitos primitivos, mas uma vez que esses conceitos sempre aparecerão como nós observados no cálculo da inferência, o valor da probabilidade não interfere em nada. Foi utilizado

$$P(\mathsf{Object}) \in [0.2, 0.8]\,, \qquad P(\mathsf{Environment}) \in [0.2, 0.8]\,.$$

Dois papéis $\mathsf{contains}(x, y)$ e $\mathsf{near}(x, y)$ representam que dois indivíduos do domínio são relacionados: respectivamente, que um ambiente contêm um objeto e que um objeto está perto de outro objeto. Como não há informação sobre a disposição dos objetos a serem encontrados, assume-se os mesmos intervalos

$$P(\mathsf{contains}) \in [0.2, 0.8]\,, \qquad P(\mathsf{near}) \in [0.2, 0.8]\,.$$

Um conceito $\mathsf{Connector}$ para representar conexão de dois corredores e objetos como portas; e um papel $\mathsf{accesses}(x, y)$ que diz quais ambientes podem ser acessados de quais conectores. Novamente adota-se os mesmos intervalos:

$$P(\mathsf{Connector}) \in [0.2, 0.8]\,, \qquad P(\mathsf{accesses}) \in [0.2, 0.8]\,.$$

É estabelecida uma hierarquia entre os objetos a serem identificados, através de construtores como:

$$
\begin{aligned}
\mathsf{InteriorObject} &\sqsubseteq \mathsf{Object}, \\
\mathsf{ExteriorObject} &\sqsubseteq \mathsf{Object}, \\
\mathsf{OfficeObject} &\sqsubseteq \mathsf{Object}, \\
\mathsf{Table} &\sqsubseteq \mathsf{InteriorObject} \\
\mathsf{Printer} &\sqsubseteq \mathsf{InteriorObject} \sqcap \mathsf{OfficeObject}, \\
\mathsf{Door} &\sqsubseteq \mathsf{Object} \sqcap \mathsf{Connector}.
\end{aligned}
$$

São definidos objetos compostos de forma a facilitar a descrição dos ambientes:

$$
\begin{aligned}
\mathsf{Desk} &\equiv \mathsf{Table} \sqcap \exists \mathsf{near}.\mathsf{Chair}, \\
\mathsf{Entrance} &\equiv \mathsf{Door} \sqcap \exists \mathsf{near}.\mathsf{Sign}.
\end{aligned}
$$

E por ultimo, temos a descrição dos ambientes propriamente ditos:

$$
\begin{aligned}
\mathsf{Room} &\equiv \mathsf{Environment} \\
&\quad \sqcap \exists \mathsf{contains}.\mathsf{Door} \\
&\quad \sqcap \exists \mathsf{contains}.\mathsf{Table} \\
&\quad \sqcap \exists \mathsf{contains}.\mathsf{Chair} \\
&\quad \sqcap \neg\exists \mathsf{contains}.\mathsf{ExteriorObjects}, \\
\mathsf{Office} &\equiv \mathsf{Room} \\
&\quad \sqcap \exists \mathsf{contains}.\mathsf{Desk} \\
&\quad \sqcap \exists \mathsf{contains}.\mathsf{Cabinet} \\
&\quad \sqcap \exists \mathsf{contains}.\mathsf{Monitor}, \\
&\vdots \\
\mathsf{Hallway} &\equiv \mathsf{Environment} \\
&\quad \sqcap \exists \mathsf{contains}.\mathsf{Entrance} \\
&\quad \sqcap \exists \mathsf{contains}.\mathsf{Extinguisher} \\
&\quad \sqcap \exists \mathsf{contains}.\mathsf{Switchbox} \\
&\quad \sqcap \neg\exists \mathsf{contains}.\mathsf{InteriorObjects}.
\end{aligned}
$$

Tabela 7.1: Ambientes identificados para 6 pontos de coleta de dados.

Pontos de Coleta						
Local	1	5	7	10	13	17
Observações						
Objetos	3 cadeiras mesa monitor armário porta	2 cadeiras mesa 2 monitores armário porta	3 portas 2 placas extintor caixa de força	2 portas 2 placas extintor caixa de força quadro	2 cadeiras 2 mesas monitor armário porta	2 cadeiras mesa monitor 2 armários porta
Resultado da inferência						
P(Escritório)	[**0.000**, **0.502**]	[**0.000**, **0.602**]	[0.000, 0.001]	[0.000, 0.001]	[**0.000**, **0.554**]	[**0.000**, **0.554**]
P(Sala de aula)	[0.000, 0.450]	[0.000, 0.532]	[0.000, 0.060]	[0.000, 0.060]	[0.000, 0.385]	[0.000, 0.385]
P(Corredor)	[0.000, 0.020]	[0.000, 0.020]	[**0.000**, **0.767**]	[**0.000**, **0.767**]	[0.000, 0.020]	[0.000, 0.020]
Área	1		2		3	

A ontologia inteira é muito extensa e não será apresentada aqui. Para uma versão completa da mesma, consulte o Apêndice C que apresenta o conteúdo do arquivo KRSS com a ontologia.

Apresentamos aqui os resultados de um experimento onde um robô percorrendo três diferentes ambientes de uma área interna de um edifício, sendo eles um laboratório, a sala de um professor e o corredor que dá acesso aos dois. Foram adquiridos dados em 18 posições do percurso, sendo que em cada posição foram coletadas imagens e pontos 3D de um sensor laser, onde as coordenadas das posições vieram da odometria do robô e um giroscópio. O robô utilizado foi um Pioneer 3-AT equipado com um laser SICK e uma câmera de vídeo.

Para cada um dos pontos de coleta de dados do percurso, foi utilizado um algoritmo SIFT para identificar objetos do ambiente. A ontologia foi instanciada com os objetos encontrados, e cada ponto foi rotulado como "escritório" ou "corredor", de acordo com o resultado de inferência mais significativo. Sempre que dois pontos consecutivos do trajeto receberam rótulos diferentes, os dados coletados foram segmentados em um novo subconjunto para a área do novo ambiente. Nesse experimento foram encontradas 3 áreas distintas. A Tabela 7.1 mostra os valores da inferência para 2 pontos de cada área. Note que na ontologia os ambientes possíveis não foram marcado como mutualmente exclusivos, portanto a soma de todos eles não precisa chegar a 100%.

Com os dados dos sensores divididos nas 3 áreas identificadas, foram produzidos ma-

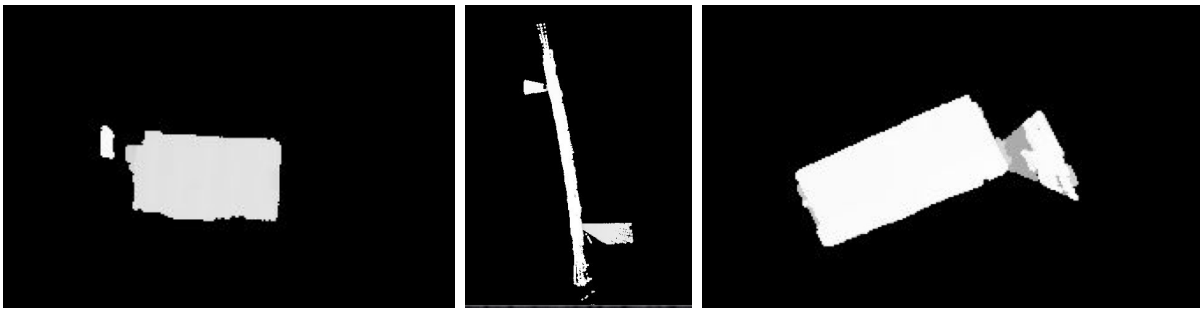
pas individualmente pra cada uma delas, com o software *6D SLAM*[2]. Utilizando as informações topológicas da rede, e o posicionamento dos conectores (portas), os mapas foram montados de forma a produzir um mapa geral com as três áreas. Comparativamente, foi feito um mapa geral único usando todos os dados no SLAM. Na Tabela 7.2 encontra-se os tempos despendidos para gerar o mapa por partes e o tempo para gerar o mapa único. Nas Figura 7.1 pode-se ver o mapa das três áreas mapeadas. Para efeitos ilustrativos, na Figura 7.2 temos uma vista tridimensional produzida do interior de um dos ambientes.

Na Tabela 7.2 pode-se ver que o mapeamento das áreas separadas é efetivamente mais rápido do que o processamento de todos os dados de uma só vez. A diferença observada não é muito grande, mas deve-se lembrar que foram utilizados apenas 18 pontos de coletas de dados percorrendo 3 ambientes. Ao se considerar um mapa maior e mais complexo, a diferença será mais significativa uma vez que o tempo despendido para o mapeamento cresce consideravelmente para grandes quantidades de dados, pois mais comparações precisam ser feitas. Além do que a separação em áreas independentes nos permite o processamento das mesmas em paralelo, diminuindo o tempo de processamento para o tempo da área mais demorada. Além do ganho computacional, há ainda o valor agregado aos mapas pela rotulação das áreas e o estabelecimento de sua topologia.

Tabela 7.2: Comparação de tempos (em milissegundos) para criação dos mapas.

Por partes		Mapa único
Sala A	52186	
Corredor	31670	114703
Laboratório	7267	
Total	91123	114703

[2]Disponível em `http://openslam.org` (NCHTER, 2009).



(a) Laboratório (b) Corredor (c) Sala

Figura 7.1: Mapas criados para as três áreas identificadas.

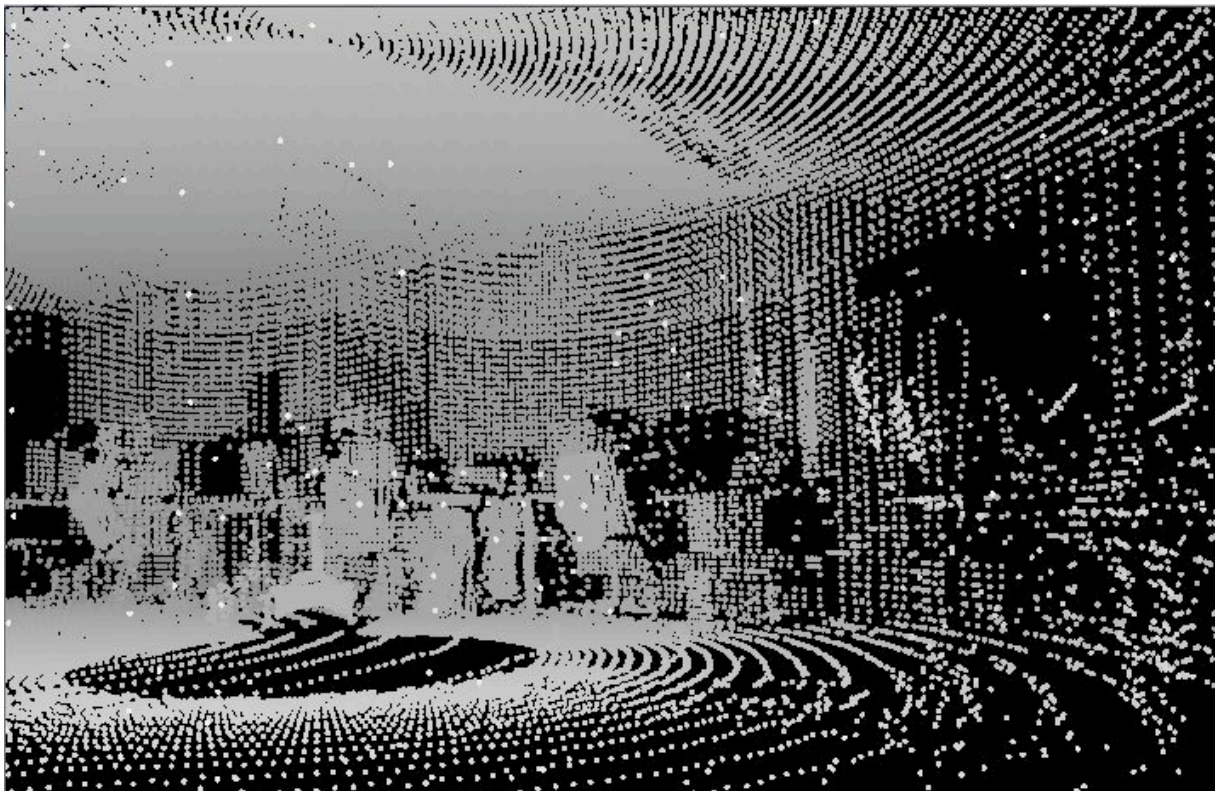


Figura 7.2: Vista tridimensional da área 1.

8 Publicações

Durante o desenvolvimento de todo o trabalho, os resultados mais significativos foram publicados em artigos e congressos da área de inteligência artificial.

Logo no início do trabalho foi feito um estudo sobre modelos lógico-probabilisticos. Durante esse estudo foi produzido e publicado um artigo intitulado *Learning Classifiers With Markov Logic Networks* (SILVA; COZMAN; POLASTRO, 2008), o qual foi apresentado no *XVII Congresso Brasileiro de Automática – CBA2008*. Nesse artigo foi desenvolvida uma metodologia para se aprender classificadores utilizando-se do modelo de *Markov Logic Networks*, um dos modelos mais promissores na época.

Posteriormente, com o desenvolvimento da CR$\mathcal{ALC}$ mais um artigo foi produzido. Intitulado *Loopy Propagation in a Probabilistic Description Logic* (COZMAN; POLASTRO, 2008) e apresentado no *2nd International Conference on Scalable Uncertainty Management – SUM2008*. O artigo continha a nova lógica de descrição probabilística chamada e o algoritmo de Loopy Propagation em primeira ordem.

Logo na sequência, mais um artigo foi publicado. *Inference in Probabilistic Ontologies with Attributive Concept Descriptions and Nominals* (POLASTRO; COZMAN, 2008) foi apresentado no *4th International Workshop on Uncertainty Reasoning for the Semantic Web – URSW2008*, na *7th International Semantic Web Conference – ISWC08*. Nesse artigo, foi proposta a extensão da CR$\mathcal{ALC}$ na qual é possível tratar *nominais*.

No ano seguinte, foi publicado um artigo contendo análises de complexidade e a proposta de método variacional para realização de inferência na CR$\mathcal{ALC}$ (e possivelmente em outras lógicas de descrição probabilísticas). Intitulado *Complexity Analysis and Variational Inference for Interpretation-based Probabilistic Description Logics* (COZMAN; POLASTRO, 2009), foi apresentado na *25th Conference on Uncertainty in Artificial Intelligence – UAI2009*.

Nesse mesmo período iniciava-se a abordagem de problemas reais da área de inteligência artificial. Um *extended abstract* intitulado *Probabilistic Description Logics for*

Context Awareness in Mobile Robotics: A Case Study in Imprecisely Specified Probabilities foi aceito no *6th International Symposium on Imprecise Probability: Theories and Applications – ISIPTA09*, na qual foi exposto um pôster com o material relevante. O trabalho aborda o problema de *extração de contexto* em robótica móvel.

As idéias contidas no extended abstract foram posteriormente desenvolvidas, levando aos resultados apresentados na Seção 7.1 deste trabalho, os quais foram publicados no artigo intitulado *Semantic Mapping with a Probabilistic Description Logic* (POLASTRO et al., 2010) no *20th Brazilian Symposium on Artificial Intelligence – SBIA 2010*.

Ainda na vertente de extração de contexto, foi feita uma segunda aplicação em problemas reais, aproveitando os rótulos extraídos para o auxilio na criação de mapas semânticos e mapas anotados. Os resultados obtidos foram apresentados na Seção 7.2 deste trabalho e foram publicados no artigo intitulado *Dealing with Semantic Knowledge in Robotics: Spatial Reasoning using a Probabilistic Description Logic* (CORRÊA et al., 2010) apresentado no *XI Argentine Symposium on Artificial Intelligence – ASAI 2010*.

9 Considerações Finais

Este trabalho apresentou três contribuições para o campo de inteligência artificial, sendo i. uma lógica de descrição probabilística, ii. um algoritmo de inferência em primeira ordem e iii. aplicações praticas em problemas reais. A primeira contribuição tem como intuito facilitar a representação do conhecimento de agentes ou robôs, enquanto que a segunda nos dá uma ferramenta para melhor raciocinar sobre o conhecimento representado.

A lógica proposta é uma versão probabilística da lógica de descrição $\mathcal{ALC}$ com semântica baseada em interpretações, permitindo inclusões e consultas sobre a probabilidade de asserções. A lógica apresenta um bom equilíbrio entre expressividade e complexidade, mas pode facilmente ser estendida, por exemplo, para permitir hierarquia de papéis, restrições de cardinalidade e relações de transitividade. Uma possível extensão foi abordada nos experimentos realizados, a respeito da representação de nominais na lógica CR$\mathcal{ALC}$. Tal expansão se mostrou viável, embora valha ressaltar que o uso de tipagem de variáveis pode levar a uma solução ainda mais elegante.

O algoritmo de inferência proposto é um algoritmo de loopy propagation em primeira-ordem para ser usado em representações shattered de terminologias com domínio finito. É um algoritmo desenvolvido para o uso em terminologias representadas na lógica CR$\mathcal{ALC}$ mas que pode ser utilizado por outros modelos probabilísticos de primeira ordem. Os testes realizados mostram que o algoritmo escapa dos problemas de escalabilidade dos algoritmos tradicionais, podendo ser utilizado em ontologias realistas na literatura. O algoritmo foi aplicado com sucesso em dois problemas reais na área de robótica móvel, e embora as aplicações apresentadas possam parecer simples, elas são base de muitas outras tarefas que agora poderão ser abordados sem restrições quanto à escalabilidade.

Assim, pode-se afirmar que os objetivos propostos na Seção 1.1 foram atingidos, pois eram justamente: o desenvolvimento de uma linguagem formal para a especificação de modelos lógico-probabilísticos; o desenvolvimento de um algoritmo de inferência proba-

bilística; o teste do algoritmo e a aplicação em problemas reais. Os resultados obtidos foram bastante satisfatórios, rendendo publicações conforme descrito no Capítulo 8.

Quanto a trabalhos futuros, alguns tópicos são deixados em aberto para mais discussão, como a inferência em domínios infinitos e a extensão da lógica até um modelo probabilístico completo para a linguagem OWL. Algumas das hipóteses adotadas podem ser futuramente removidas, como a questão de aciclicidade ou a de homogeneidade. O uso de modelos probabilísticos de grafos não direcionados (RICHARDSON; DOMINGOS, 2006; TASKAR; ABBEEL; KOLLER, 2002) pode facilitar a questão sobre a hipótese de aciclicidade, embora esses modelos falhem na implicação da fatorização de probabilidades na presença de restrições lógicas (MOUSSOURIS, 1974) restando ainda muita discussão para esse caso. Também pode ser desejável a remoção da hipótese de domínio fechado, já que muitas lógicas de descrição tradicionais não assumem hipóteses sobre a cardinalidade do domínio.

Referente ao desempenho, pode-se ver que o algoritmo aqui proposto tem um custo computacional inferior em comparação com os algoritmos tradicionais. E ao mesmo tempo manteve a precisão no cálculo da inferência, sendo que na maioria dos casos foi ainda mais preciso que outros algoritmos aproximados.

Referências Bibliográficas

ANGUELOV, D. et al. Discriminative learning of Markov random fields for segmentation of 3d scan data. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2005. p. 169–176.

BAADER, F. et al. *Description Logic Handbook*. Cambridge University Press, 2002.

BACCHUS, F. *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach*. Cambridge: MIT Press, 1990.

BALDI, P.; BRUNAK, S. *Bioinformatics: The Machine Learning Approach (2nd ed.)*. MIT Press, 2001.

BOOLE, G. *The Laws of Thought*. Dover edition, 1958.

BRAZ, R. de S.; AMIR, E.; ROTH, D. Lifted first-order probabilistic inference. In: *International Joint Conference in Artificial Intelligence (IJCAI)*. 2006.

BRAZ, R. de S.; AMIR, E.; ROTH, D. MPE and partial inversion in lifted probabilistic variable elimination. In: *AAAI*. 2006.

CHARNIAK, E. Bayesian networks without tears. *AI Magazine*, p. 50–63, Fall 1991.

CHAVIRA, M.; DARWICHE, A.; JAEGER, M. Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning*, v. 42, p. 4–20, 2006.

CORRÊA, F. et al. Dealing with semantic knowledge in robotics with a probabilistic description logic. In: *Simpósio Argentino de Inteligência Artificial*. Buenos Aires: , 2010.

COSTA, P. C. G.; LASKEY, K. B. PR-OWL: A framework for probabilistic ontologies. In: *Conference on Formal Ontology in Information Systems*. 2006.

COSTA, P. C. G. da; LASKEY, K. B. Of Klingons and starships: Bayesian logic for the 23rd century. In: *Conference on Uncertainty in Artificial Intelligence*. 2005.

COZMAN, F. G. Credal networks. *Artificial Intelligence*, v. 120, p. 199–233, 2000.

COZMAN, F. G.; POLASTRO, R. B. Loopy propagation in a probabilistic description logic. In: GRECO, S.; LUKASIEWICZ, T. (Ed.). *Second International Conference on Scalable Uncertainty Management*. Springer, 2008. (Lecture Notes in Artificial Intelligence (LNAI 5291)), p. 120–133.

COZMAN, F. G.; POLASTRO, R. B. Complexity analysis and variational inference for interpretation-based probabilistic description logics. In: *Uncertainty in Artificial Intelligence*. 2009.

D'AMATO, C.; FANIZZI, N.; LUKASIEWICZ, T. Tractable reasoning with Bayesian description logics. In: *Scalable Uncertainty Management*. Springer Berlin / Heidelberg, 2008, (Lecture Notes in Computer Science, v. 5291). p. 146–159.

DEGROOT, M. H. *Probability and Statistics*. Addison-Wesley, 1986.

DING, Z.; PENG, Y.; PAN, R. BayesOWL: Uncertainty modeling in semantic web ontologies. In: *Soft Computing in Ontologies and Semantic Web*. Berlin/Heidelberg: Springer, 2006, (Studies in Fuzziness and Soft Computing, v. 204). p. 3–29. ISBN 978-3-540-33472-9.

DÜRIG, M.; STUDER, T. Probabilistic ABox reasoning: preliminary results. In: *Description Logics*. 2005. p. 104–111.

FINETTI, B. de. *Theory of Probability, vol. 1-2*. New York: Wiley, 1974.

GALINDO, C. et al. Robot task planning using semantic maps. *Robotics and Autonomous Systems*, v. 11, p. 955–966, 2008.

GEORGAKOPOULOS, G.; KAVVADIAS, D.; PAPADIMITRIOU, C. H. Probabilistic satisfiability. *Journal of Complexity*, v. 4, p. 1–11, 1988.

GETOOR, L. et al. Learning probabilistic models of relational structure. In: *International Conference on Machine Learning*. 2001. p. 170–177.

GETOOR, L.; TASKAR, B. *Introduction to Statistical Relational Learning*. MIT Press, 2007. ISBN 0262072882.

GILKS, W. R.; RICHARDSON, S.; SPIEGELHALTER, D. J. *Markov Chain Monte Carlo in Practice*. London, England: Chapman and Hall, 1996.

GIUGNO, R.; LUKASIEWICZ, T. P-SHOQ(D): A probabilistic extension of SHOQ(D) for probabilistic ontologies in the semantic web. In: FLESCA, S. et al. (Ed.). *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA)*. Cosenza, Italy: Lecture Notes in Artificial Intelligence, Springer, 2002. v. 2424, p. 86–97.

HAILPERIN, T. Best possible inequalities for the probability of a logical function of events. *American Mathematical Monthly*, v. 72, p. 343–359, 1965.

HALPERN, J. Y. An analysis of first-order logics of probability. *Artificial Intelligence*, v. 46, p. 311–350, 1990.

HALPERN, J. Y. *Reasoning about Uncertainty*. Cambridge, Massachusetts: MIT Press, 2003.

HEINSOHN, J. Probabilistic description logics. In: *Conference on Uncertainty in Artificial Intelligence*. 1994. p. 311–318.

HERTZBERG, J.; SAFFIOTTI, A. Using semantic knowledge in robotics. *Robotics and Autonomous Systems*, v. 56, p. 875–877, 2008.

HORROCKS, I.; PATEL-SCHNEIDER, P. F.; HARMELEN, F. van. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, v. 1, n. 1, p. 7–26, 2003.

HOWSON, C.; URBACH, P. *Scientific Reasoning: the Bayesian Approach*. Chicago, Illinois: Open Court Publishing Company, 1993.

HUNG, E.; GETOOR, L.; SUBRAHMANIAN, V. S. Probabilistic interval XML. *ACM Transactions on Computational Logic*, v. 8, n. 4, 2007.

IDE, J. S.; COZMAN, F. G. Approximate algorithms for credal networks with binary variables. *International Journal of Approximate Reasoning*, v. 48, n. 1, p. 275–296, 2008.

JAEGER, M. Probabilistic reasoning in terminological logics. In: *Principles of Knowledge Representation (KR)*. 1994. p. 461–472.

JAEGER, M. Relational Bayesian networks. In: GEIGER, D.; SHENOY, P. P. (Ed.). *Conference on Uncertainty in Artificial Intelligence*. San Francisco, California: Morgan Kaufmann, 1997. p. 266–273.

JAEGER, M. Convergence results for relational Bayesian networks. In: *LICS*. 1998.

JAEGER, M. Reasoning about infinite random structures with relational Bayesian networks. In: *Knowledge Representation (KR)*. San Francisco, California: Morgan Kaufman, 1998.

JAEGER, M. Complex probabilistic modeling with recursive relational Bayesian networks. *Annals of Mathematics and Artificial Intelligence*, v. 32, p. 179–220, 2001.

JAUMARD, B.; HANSEN, P.; AO, M. P. de A. Column generation methods for probabilistic logic. *ORSA Journal on Computing*, v. 3, n. 2, p. 135–148, Spring 1991.

JENSEN, F. V. *An Introduction to Bayesian Networks*. New York: Springer Verlag, 1996.

KERSTING, K.; DICK, U. Balios - the engine for Bayesian logic programs. In: *European Conference on Principles and Practice of Knowledege Discovery in Databases*. 2004. p. 549–551.

KOLLER, D.; PFEFFER, A. Object-oriented Bayesian networks. In: *Conference on Uncertainty in Artificial Intelligence*. 1997. p. 302–313.

Kyburg Jr., H. E.; TENG, C. M. *Uncertain Inference*. Cambridge University Press, 2001.

LIAO, L.; FOX, D.; KAUTZ, H. Location-based activity recognition using relational Markov networks. In: *International Joint Conference on Artifical Intelligence*. 2005.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, v. 60, n. 2, p. 91, nov. 2004.

LUKASIEWICZ, T. Probabilistic logic programming with conditional constraints. *ACM Transactions on Computational Logic*, v. 2, n. 3, p. 289–339, 2001.

LUKASIEWICZ, T. Expressive probabilistic description logics. *Artif. Intell.*, Elsevier Science Publishers Ltd., Essex, UK, v. 172, n. 6-7, p. 852–883, abr. 2008. ISSN 0004-3702.

LUKASIEWICZ, T.; STRACCIA, U. Managing uncertainty and vagueness in description logics for the semantic web. *Web Semant.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, v. 6, n. 4, p. 291–308, nov. 2008. ISSN 1570-8268.

MILCH, B. et al. BLOG: Probabilistic models with unknown objects. In: *IJCAI*. 2005.

MILCH, B.; RUSSELL, S. First-order probabilistic languages: into the unknown. In: *Int. Conference on Inductive Logic Programming*. 2007.

MOUSSOURIS, J. Gibbs and Markov random systems with constraints. *Journal of Statistical Physics*, v. 10, n. 1, p. 11–33, January 1974. ISSN 0022-4715 (Print) 1572-9613 (Online).

MUGGLETON, S.; RAEDT, L. D. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, v. 20, p. 629–679, 1994.

NCHTER, A. *3D Robotic Mapping*. Springer Tracts in Advanced Robotics, 2009.

NEAPOLITAN, R. E. *Learning Bayesian Networks*. Prentice Hall, 2003.

NGO, L.; HADDAWY, P. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, v. 171, n. 1–2, p. 147–177, 1997.

NILSSON, N. J. Probabilistic logic. *Artificial Intelligence*, v. 28, p. 71–87, 1986.

NOTTELMANN, H.; FUHR, N. Adding probabilities and rules to OWL lite subsets based on probabilistic datalog. *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems*, v. 14, n. 1, p. 17–42, 2006.

PEARL, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, California: Morgan Kaufmann, 1988.

POLASTRO, R. B. et al. Semantic mapping with a probabilistic description logic. In: *20th Brazilian Symposium on Artificial Intelligence (SBIA)*. Sao Bernardo do Campo, Brazil: Springer, 2010. (Lecture Notes in Computer Science).

POLASTRO, R. B.; COZMAN, F. G. Inference in probabilistic ontologies with attributive concept descriptions and nominals. In: *4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW) at the 7th International Semantic Web Conference (ISWC)*. Karlsruhe, Germany: Springer, 2008.

POOLE, D. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, v. 64, p. 81–129, 1993.

POOLE, D. First-order probabilistic inference. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2003. p. 985–991.

RAEDT, L. D.; KERSTING, K. Probabilistic inductive logic programming. In: RAEDT, L. D. et al. (Ed.). *Probabilistic Inductive Logic Programming - Theory and Applications*. Springer, 2008, (Lecture Notes in Computer Science). p. 1–27.

RICHARDSON, M.; DOMINGOS, P. Markov logic networks. *Machine Learning*, v. 62, n. 1-2, p. 107–136, 2006.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach (2nd ed.)*. New Jersey: Upper Saddle River, Prentice Hall, 1995.

SATO, T.; KAMEYA, Y. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, v. 15, p. 391–454, 2001.

SCHMIDT-SCHAUSS, M.; SMOLKA, G. Attributive concept descriptions with complements. *Artificial Intelligence*, v. 48, p. 1–26, 1991.

SEBASTIANI, F. A probabilistic terminological logic for modelling information retrieval. In: CROFT, W.; RIJSBERGEN, C. v. (Ed.). *17th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)*. Dublin, Ireland: Springer-Verlag, 1994. p. 122–130.

SHACHTER, R. D. Evaluating influence diagrams. *Operations Research*, v. 34, n. 6, p. 873–882, November/December 1986.

SILVA, V. A.; COZMAN, F. G.; POLASTRO, R. B. Learning classifiers with Markov logic networks. In: *Congresso Brasileiro de Automática (CBA)*. Brasil: Sociedade Brasileira de Automática, 2008.

STAKER, R. Reasoning in expressive description logics using belief networks. In: *International Conference on Information and Knowledge Engineering*. Las Vegas, USA: , 2002. p. 489–495.

TASKAR, B.; ABBEEL, P.; KOLLER, D. Discriminative probabilistic models for relational data. In: *Conference on Uncertainty in Artificial Intelligence*. Edmonton, Canada: Morgan Kaufmann, 2002. p. 485–492.

THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic Robotics*. MIT Press, 2005.

YEDIDIA, J. S.; FREEMAN, W. T.; WEISS, Y. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, v. 51, p. 2282–2312, 2005.

YELLAND, P. M. *Market analysis using a combination of Bayesian networks and description logics*. 1999.

Apêndice A – Ontologia Wine.

Conteúdo do arquivo wine.krss usado para o experimento apresentado na Seção 6.3

```
(probability wine(x) 0.9)

;;madeFromGrape
(probability madeFromCabernetFrancGrape(x) 0.5)
(probability madeFromCabernetSauvignonGrape(x) 0.5)
(probability madeFromChardonnayGrape(x) 0.5)
(probability madeFromCheninBlancGrape(x) 0.5)
(probability madeFromGamayGrape(x) 0.5)
(probability madeFromMalbecGrape(x) 0.5)
(probability madeFromMerlotGrape(x) 0.5)
(probability madeFromPetiteSyrahGrape(x) 0.5)
(probability madeFromPetiteVerdotGrape(x) 0.5)
(probability madeFromPinotBlancGrape(x) 0.5)
(probability madeFromPinotNoirGrape(x) 0.5)
(probability madeFromRieslingGrape(x) 0.5)
(probability madeFromSangioveseGrape(x) 0.5)
(probability madeFromSauvignonBlancGrape(x) 0.5)
(probability madeFromSemillonGrape(x) 0.5)
(probability madeFromZinfandelGrape(x) 0.5)

;;Regions
(probability locatedInAustralianRegion(x) 0.15)
(probability locatedInFrenchRegion(x) 0.15)
(probability locatedInGermanyRegion(x) 0.15)
(probability locatedInItalianRegion(x) 0.15)
(probability locatedInNewZealandRegion(x) 0.15)
(probability locatedInPortugalRegion(x) 0.15)
(probability locatedInUSRegion(x) 0.15)

;;regiao contida em AustralianRegion
(conditional-probability locatedInSouthAustraliaRegion(x) locatedInAustralianRegion(x) 0.5)
(define-primitive-concept locatedInSouthAustraliaRegion(x) locatedInAustralianRegion(x) )

;;regiao contida em FrenchRegion
(conditional-probability locatedInAlsaceRegion(x) locatedInFrenchRegion(x) 0.2)
(define-primitive-concept locatedInAlsaceRegion(x) locatedInFrenchRegion(x) )
(conditional-probability locatedInBeaujolaisRegion(x) locatedInFrenchRegion(x) 0.2)
(define-primitive-concept locatedInBeaujolaisRegion(x) locatedInFrenchRegion(x) )
(conditional-probability locatedInBordeauxRegion(x) locatedInFrenchRegion(x) 0.2)
(define-primitive-concept locatedInBordeauxRegion(x) locatedInFrenchRegion(x) )
(conditional-probability locatedInBourgogneRegion(x) locatedInFrenchRegion(x) 0.2)
(define-primitive-concept locatedInBourgogneRegion(x) locatedInFrenchRegion(x) )
(conditional-probability locatedInLoireRegion(x) locatedInFrenchRegion(x) 0.2)
(define-primitive-concept locatedInLoireRegion(x) locatedInFrenchRegion(x) )

;;regiao contida em ItalianRegion
(conditional-probability locatedInChiantiRegion(x) locatedInItalianRegion(x) 0.5)
(define-primitive-concept locatedInChiantiRegion(x) locatedInItalianRegion(x) )

;;regiao contida em USRegion
(conditional-probability locatedInCaliforniaRegion(x) locatedInUSRegion(x) 0.5)
(define-primitive-concept locatedInCaliforniaRegion(x) locatedInUSRegion(x) )
(conditional-probability locatedInTexasRegion(x) locatedInUSRegion(x) 0.5)
(define-primitive-concept locatedInTexasRegion(x) locatedInUSRegion(x) )
```

```
;;regiao contida em BordeauxRegion
(conditional-probability locatedInMedocRegion(x) locatedInBordeauxRegion(x) 0.333)
(define-primitive-concept locatedInMedocRegion(x) locatedInBordeauxRegion(x) )
(conditional-probability locatedInSauterneRegion(x) locatedInBordeauxRegion(x) 0.333)
(define-primitive-concept locatedInSauterneRegion(x) locatedInBordeauxRegion(x) )
(conditional-probability locatedInStEmilionRegion(x) locatedInBordeauxRegion(x) 0.333)
(define-primitive-concept locatedInStEmilionRegion(x) locatedInBordeauxRegion(x) )

;;regiao contida em BourgogneRegion
(conditional-probability locatedInCotesDOrRegion(x) locatedInBourgogneRegion(x) 0.5)
(define-primitive-concept locatedInCotesDOrRegion(x) locatedInBourgogneRegion(x) )
(conditional-probability locatedInMeursaultRegion(x) locatedInBourgogneRegion(x)0.5)
(define-primitive-concept locatedInMeursaultRegion(x) locatedInBourgogneRegion(x) )
;;adicionar Zenode

;;regiao contida em CaliforniaRegion
(conditional-probability locatedInArroyoGrandeRegion(x) locatedInCaliforniaRegion(x) 0.125)
(define-primitive-concept locatedInArroyoGrandeRegion(x) locatedInCaliforniaRegion(x) )
(conditional-probability locatedInCentralCoastRegion(x) locatedInCaliforniaRegion(x) 0.125)
(define-primitive-concept locatedInCentralCoastRegion(x) locatedInCaliforniaRegion(x) )
(conditional-probability locatedInEdnaValleyRegion(x) locatedInCaliforniaRegion(x) 0.125)
(define-primitive-concept locatedInEdnaValleyRegion(x) locatedInCaliforniaRegion(x) )
(conditional-probability locatedInMendocinoRegion(x) locatedInCaliforniaRegion(x) 0.125)
(define-primitive-concept locatedInMendocinoRegion(x) locatedInCaliforniaRegion(x) )
(conditional-probability locatedInNapaRegion(x) locatedInCaliforniaRegion(x) 0.125)
(define-primitive-concept locatedInNapaRegion(x) locatedInCaliforniaRegion(x) )
(conditional-probability locatedInSantaBarbaraRegion(x) locatedInCaliforniaRegion(x) 0.125)
(define-primitive-concept locatedInSantaBarbaraRegion(x) locatedInCaliforniaRegion(x) )
(conditional-probability locatedInSantaCruzMountainsRegion(x) locatedInCaliforniaRegion(x) 0.125)
(define-primitive-concept locatedInSantaCruzMountainsRegion(x) locatedInCaliforniaRegion(x) )
(conditional-probability locatedInSonomaRegion(x) locatedInCaliforniaRegion(x) 0.125)
(define-primitive-concept locatedInSonomaRegion(x) locatedInCaliforniaRegion(x) )

;;regiao contida em LoireRegion
(conditional-probability locatedInAnjouRegion(x) locatedInLoireRegion(x) 0.25)
(define-primitive-concept locatedInAnjouRegion(x) locatedInLoireRegion(x) )
(conditional-probability locatedInMuscadetRegion(x) locatedInLoireRegion(x) 0.25)
(define-primitive-concept locatedInMuscadetRegion(x) locatedInLoireRegion(x) )
(conditional-probability locatedInSancerreRegion(x) locatedInLoireRegion(x) 0.25)
(define-primitive-concept locatedInSancerreRegion(x) locatedInLoireRegion(x) )
(conditional-probability locatedInToursRegion(x) locatedInLoireRegion(x)0.25)
(define-primitive-concept locatedInToursRegion(x) locatedInLoireRegion(x) )

;;regiao contida em TexasRegion
(conditional-probability locatedInCentralTexasRegion(x) locatedInTexasRegion(x) 0.5)
(define-primitive-concept locatedInCentralTexasRegion(x) locatedInTexasRegion(x) )

;;regiao contida em MedocRegion
(conditional-probability locatedInMargauxRegion(x) locatedInMedocRegion(x) 0.5)
(define-primitive-concept locatedInMargauxRegion(x) locatedInMedocRegion(x) )
(conditional-probability locatedInPauillacRegion(x) locatedInMedocRegion(x) 0.5)
(define-primitive-concept locatedInPauillacRegion(x) locatedInMedocRegion(x) )

;;Descriptor
```

```
;;WineColor
(probability hasWineColorRed(x) 0.333)
(probability hasWineColorRose(x) 0.333)
(probability hasWineColorWhite(x) 0.333)
;;WineBody
(probability hasWineBodyLight(x) 0.333)
(probability hasWineBodyMedium(x) 0.333)
(probability hasWineBodyFull(x) 0.333)
;;WineFlavour
(probability hasWineFlavourDelicate(x) 0.333)
(probability hasWineFlavourModerate(x) 0.333)
(probability hasWineFlavourStrong(x) 0.333)
;;WineSugar
(probability hasWineSugarDry(x) 0.333)
(probability hasWineSugarOffDry(x) 0.333)
(probability hasWineSugarSweet(x) 0.333)

;;wines
(define-concept EarlyHarvest(x)
    (and
        wine(x)
        (or hasWineSugarDry(x) hasWineSugarOffDry(x))
    )
)

(define-concept LateHarvest(x)
    (and
        wine(x)
        (or hasWineFlavourModerate(x) hasWineFlavourStrong(x))
        hasWineSugarSweet(x)
    )
)

(define-concept AlsatianWine(x)
    (and
        wine(x)
        locatedInAlsaceRegion(x)
    )
)

(define-concept AmericanWine(x)
    (and
        wine(x)
        locatedInUSRegion(x)
    )
)

(define-concept Beaujolais(x)
    (and
        wine(x)
        locatedInBeaujolaisRegion(x)
    )
)

(define-concept Bordeaux(x)
```

```
    (and
        wine(x)
        locatedInBordeauxRegion(x)
    )
)

(define-concept Sauternes(x)
    (and
        Bordeaux(x)
        LateHarvest(x)
        hasWineColorWhite(x)
        hasWineBodyMedium(x)
        locatedInSauterneRegion(x)
    )
)

(define-concept Burgundy(x)
    (and
        wine(x)
        locatedInBourgogneRegion(x)
    )
)

(define-concept CabernetFrank(x)
    (and
        wine(x)
        madeFromCabernetFrancGrape(x)
        hasWineBodyMedium(x)
        hasWineColorRed(x)
        hasWineFlavourModerate(x)
        hasWineSugarDry(x)
    )
)

(define-concept CabernetSauvignon(x)
    (and
        wine(x)
        madeFromCabernetSauvignonGrape(x)
        (or hasWineBodyMedium(x) hasWineBodyFull(x))
        hasWineColorRed(x)
        (or hasWineFlavourModerate(x) hasWineFlavourStrong(x))
        hasWineSugarDry(x)
    )
)

(define-concept CaliforniaWine(x)
    (and
        wine(x)
        locatedInCaliforniaRegion(x)
    )
)

(define-concept Chardonnay(x)
    (and
        wine(x)
```

```
        madeFromChardonnayGrape(x)
        (or hasWineBodyFull(x) hasWineBodyMedium(x))
        hasWineColorWhite(x)
        (or hasWineFlavourStrong(x) hasWineFlavourModerate(x))
    )
)

(define-concept CheninBlank(x)
    (and
        wine(x)
        madeFromCheninBlancGrape(x)
        (or hasWineBodyFull(x) hasWineBodyMedium(x))
        hasWineColorWhite(x)
        hasWineFlavourModerate(x)
        (or hasWineSugarDry(x) hasWineSugarOffDry(x))
    )
)

(define-concept DessertWine(x)
    (and
        wine(x)
        (or hasWineSugarOffDry(x)  hasWineSugarSweet(x))
    )
)

(define-concept IceWine(x)
    (and
        DessertWine(x)
        LateHarvest(x)
        hasWineColorWhite(x)
        (or hasWineBodyMedium(x) hasWineBodyFull(x))
        (or hasWineFlavourModerate(x) hasWineFlavourStrong(x))
    )
)

(define-concept Riesling(x)
    (and
        wine(x)
        madeFromRieslingGrape(x)
        hasWineColorWhite(x)
    )
)

(define-concept SweetRiesling(x)
    (and
        Riesling(x)
        hasWineSugarSweet(x)
        DessertWine(x)
        hasWineBodyFull(x)
        (or hasWineFlavourModerate(x) hasWineFlavourStrong(x))
    )
)

(define-concept DryRiesling(x)
    (and
```

```
        Riesling(x)
        hasWineSugarDry(x)
        (or hasWineBodyLight(x) hasWineBodyMedium(x))
        hasWineColorWhite(x)
        hasWineFlavourDelicate(x)
    )
)

(define-concept DryWine(x)
    (and
        wine(x)
        hasWineSugarDry(x)
    )
)

(define-concept RedWine(x)
    (and
        wine(x)
        hasWineColorRed(x)
    )
)

(define-concept DryRedWine(x)
    (and
        DryWine(x)
        RedWine(x)
    )
)

(define-concept WhiteWine(x)
    (and
        wine(x)
        hasWineColorWhite(x)
    )
)

(define-concept DryWhiteWine(x)
    (and
        DryWine(x)
        WhiteWine(x)
    )
)

(define-concept FrenchWine(x)
    (and
        wine(x)
        locatedInFrenchRegion(x)
    )
)

(define-concept FullBodyWine(x)
    (and
        wine(x)
        hasWineBodyFull(x)
    )
```

```
)

(define-concept Gamay(x)
    (and
        wine(x)
        madeFromGamayGrape(x)
    )
)

(define-concept GermanWine(x)
    (and
        wine(x)
        locatedInGermanyRegion(x)
    )
)

(define-concept ItalianWine(x)
    (and
        wine(x)
        locatedInItalianRegion(x)
    )
)

(define-concept Chianti(x)
    (and
        ItalianWine(x)
        (or hasWineBodyLight(x) hasWineBodyMedium(x))
        hasWineColorRed(x)
        hasWineFlavourModerate(x) hasWineSugarDry(x) locatedInChiantiRegion(x)
madeFromSangioveseGrape(x) )
)

(define-concept Loire(x)
    (and
        wine(x)
        locatedInLoireRegion(x)
    )
)

(define-concept Anjou(x)
    (and
        Loire(x)
        locatedInAnjouRegion(x)
        hasWineBodyLight(x)
        hasWineColorRose(x)
        hasWineFlavourDelicate(x)
        hasWineSugarOffDry(x)
    )
)

(define-concept Meritage(x)
    (and
        wine(x)
        (or
            (and madeFromCabernetSauvignonGrape(x) madeFromCabernetFrancGrape(x))
```

```
            (and madeFromCabernetSauvignonGrape(x) madeFromMalbecGrape(x))
            (and madeFromCabernetSauvignonGrape(x) madeFromPetiteVerdotGrape(x))
            (and madeFromCabernetSauvignonGrape(x) madeFromMerlotGrape(x))
            (and madeFromCabernetFrancGrape(x) madeFromMalbecGrape(x))
            (and madeFromCabernetFrancGrape(x) madeFromPetiteVerdotGrape(x))
            (and madeFromCabernetFrancGrape(x) madeFromMerlotGrape(x))
            (and madeFromMalbecGrape(x) madeFromPetiteVerdotGrape(x))
            (and madeFromMalbecGrape(x) madeFromMerlotGrape(x))
            (and madeFromPetiteVerdotGrape(x) madeFromMerlotGrape(x))
        )
        hasWineColorRed(x)
    )
)

(define-concept Merlot(x)
    (and
        wine(x)
        madeFromMerlotGrape(x)
        (or hasWineBodyLight(x) hasWineBodyMedium(x))
        hasWineColorRed(x)
        (or hasWineFlavourModerate(x) hasWineFlavourDelicate(x))
        hasWineSugarDry(x)
    )
)

(define-concept PetiteSyrah(x)
    (and
        wine(x)
        madeFromPetiteSyrahGrape(x)
        (or hasWineBodyMedium(x) hasWineBodyFull(x))
        hasWineColorRed(x)
        (or hasWineFlavourModerate(x) hasWineFlavourStrong(x))
        hasWineSugarDry(x)
    )
)

(define-concept PinotBlanc(x)
    (and
        wine(x)
        madeFromPinotBlancGrape(x)
        hasWineColorWhite(x)
    )
)

(define-concept PinotNoir(x)
    (and
        wine(x)
        madeFromPinotNoirGrape(x)
        hasWineColorRed(x)
    )
)

(define-concept Port(x)
    (and
        RedWine(x)
```

```
        hasWineBodyFull(x)
        hasWineFlavourStrong(x)
        hasWineSugarSweet(x)
        locatedInPortugalRegion(x)
    )
)

(define-concept RedBordeaux(x)
    (and
        Bordeaux(x)
        RedWine(x)
        (or madeFromCabernetSauvignonGrape(x) madeFromMerlotGrape(x))
    )
)

(define-concept RedBurgundy(x)
    (and
        Burgundy(x)
        RedWine(x)
        madeFromPinotNoirGrape(x)
    )
)

(define-concept CotesDOr(x)
    (and
        RedBurgundy(x)
        locatedInCotesDOrRegion(x)
        hasWineFlavourModerate(x)
    )
)

(define-concept RoseWine(x)
    (and
        wine(x)
        hasWineColorRose(x)
    )
)

(define-concept SemillonOrSauvignonBlanc(x)
    (and
        wine(x)
        (or madeFromSemillonGrape(x) madeFromSauvignonBlancGrape(x))
        (or hasWineBodyMedium(x) hasWineBodyFull(x))
        hasWineColorWhite(x)
    )
)

(define-concept SauvignonBlanc(x)
    (and
        SemillonOrSauvignonBlanc(x)
        madeFromSauvignonBlancGrape(x)
    )
)

(define-concept Semillon(x)
```

```
    (and
        SemillonOrSauvignonBlanc(x)
        madeFromSemillonGrape(x)
    )
)

(define-concept SweetWine(x)
    (and
        wine(x)
        hasWineSugarSweet(x)
    )
)

(define-concept TableWine(x)
    (and
        wine(x)
        hasWineSugarDry(x)
    )
)

(define-concept RedTableWine(x)
    (and
        TableWine(x)
        hasWineColorRed(x)
    )
)

(define-concept TexasWine(x)
    (and
        wine(x)
        locatedInTexasRegion(x)
    )
)

(define-concept WhiteBordeaux(x)
    (and
        Bordeaux(x)
        WhiteWine(x)
        (or madeFromSemillonGrape(x) madeFromSauvignonBlancGrape(x))
    )
)

(define-concept WhiteBurgundy(x)
    (and
        Burgundy(x)
        WhiteWine(x)
        madeFromChardonnayGrape(x)
    )
)

(define-concept Meursault(x)
    (and
        WhiteBurgundy(x)
        locatedInMeursaultRegion(x)
        hasWineBodyFull(x)
```

```
    )
)

(define-concept WhiteLoire(x)
    (and
        Loire(x)
        WhiteWine(x)
        (or madeFromCheninBlancGrape(x) madeFromPinotBlancGrape(x) madeFromSauvignonBlancGrape(x))
    )
)

(define-concept WhiteNonSweetWine(x)
    (and
        WhiteWine(x)
        (or hasWineSugarDry(x) hasWineSugarOffDry(x))
    )
)

(define-concept Zinfandel(x)
    (and
        wine(x)
        madeFromZinfandelGrape(x)
        (or hasWineBodyFull(x) hasWineBodyMedium(x))
        hasWineColorRed(x)
        (or hasWineFlavourModerate(x) hasWineFlavourStrong(x))
        hasWineSugarDry(x)
    )
)
```

Apêndice B – Ontologia utilizada para extração de contexto em robótica móvel.

Conteúdo do arquivo context.krss usado para a aplicação apresentada na Seção 7.1

```
;;Relations
(probability contains(x,y) [0.25, 0.75])

;;Indoors
(probability IndoorObjects(x) [0.25, 0.75])

(conditional-probability Computer(x) IndoorObjects(x) [0.25, 0.75])
(define-primitive-concept Computer(x) IndoorObjects(x) )
(conditional-probability Tv(x) IndoorObjects(x) [0.25, 0.75])
(define-primitive-concept Tv(x) IndoorObjects(x) )
(conditional-probability RestingObjects(x) IndoorObjects(x) [0.25, 0.75])
(define-primitive-concept RestingObjects(x) IndoorObjects(x) )
(conditional-probability KitchenObjects(x) IndoorObjects(x) [0.25, 0.75])
(define-primitive-concept KitchenObjects(x) IndoorObjects(x) )

(conditional-probability Couch(x) RestingObjects(x) [0.25, 0.75])
(define-primitive-concept Couch(x) RestingObjects(x) )
(conditional-probability Bed(x) RestingObjects(x) [0.25, 0.75])
(define-primitive-concept Bed(x) RestingObjects(x) )

(conditional-probability Refrigerator(x) KitchenObjects(x) [0.25, 0.75])
(define-primitive-concept Refrigerator(x) KitchenObjects(x) )
(conditional-probability Oven(x) KitchenObjects(x) [0.25, 0.75])
(define-primitive-concept Oven(x) KitchenObjects(x) )

(probability Chair(x) [0.25, 0.75])
(probability Table(x) [0.25, 0.75])

(define-concept EContainsIndoorObjects(x)
    (some contains IndoorObjects )
)
(define-concept EContainsRestingObjects(x)
    (some contains RestingObjects )
)
(define-concept EContainsKitchenObjects(x)
    (some contains KitchenObjects )
)

(define-concept EContainsChair(x)
    (some contains Chair )
)
(define-concept EContainsTable(x)
    (some contains Table )
)
(define-concept EContainsComputer(x)
    (some contains Computer )
)
(define-concept EContainsCouch(x)
    (some contains Couch )
)
(define-concept EContainsTv(x)
    (some contains Tv )
)
(define-concept EContainsRefrigerator(x)
    (some contains Refrigerator )
```

```
)
(define-concept EContainsOven(x)
    (some contains Oven )
)
(define-concept EContainsBed(x)
    (some contains Bed )
)

;;Combinations
(define-concept Desk(x)
    (and
        Table(x)
        EContainsChair(x)
    )
)
(define-concept EContainsDesk(x)
    (some contains Desk )
)

;;Outdoors
(probability OutdoorObjects(x) [0.25, 0.75])

(conditional-probability Vegetation(x) OutdoorObjects(x) [0.25, 0.75])
(define-primitive-concept Vegetation(x) OutdoorObjects(x) )
(conditional-probability Water(x) OutdoorObjects(x) [0.25, 0.75])
(define-primitive-concept Water(x) OutdoorObjects(x) )
(conditional-probability Building(x) OutdoorObjects(x) [0.25, 0.75])
(define-primitive-concept Building(x) OutdoorObjects(x) )
(conditional-probability Sand(x) OutdoorObjects(x) [0.25, 0.75])
(define-primitive-concept Sand(x) OutdoorObjects(x) )
(conditional-probability Road(x) OutdoorObjects(x) [0.25, 0.75])
(define-primitive-concept Road(x) OutdoorObjects(x) )
(conditional-probability Car(x) OutdoorObjects(x) [0.25, 0.75])
(define-primitive-concept Car(x) OutdoorObjects(x) )

(conditional-probability Lawn(x) Vegetation(x) [0.25, 0.75])
(define-primitive-concept Lawn(x) Vegetation(x) )
(conditional-probability Bush(x) Vegetation(x) [0.25, 0.75])
(define-primitive-concept Bush(x) Vegetation(x) )
(conditional-probability Tree(x) Vegetation(x) [0.25, 0.75])
(define-primitive-concept Tree(x) Vegetation(x) )

(probability Cement(x) [0.25, 0.75])

(define-concept EContainsOutdoorObjects(x)
    (some contains OutdoorObjects )
)
(define-concept EContainsVegetation(x)
    (some contains Vegetation )
)
(define-concept EContainsWater(x)
    (some contains Water )
)
(define-concept EContainsBuilding(x)
    (some contains Building )
```

```
)
(define-concept EContainsSand(x)
    (some contains Sand )
)
(define-concept EContainsRoad(x)
    (some contains Road )
)
(define-concept EContainsCar(x)
    (some contains Car )
)
(define-concept EContainsLawn(x)
    (some contains Lawn )
)
(define-concept EContainsBush(x)
    (some contains Bush )
)
(define-concept EContainsTree(x)
    (some contains Tree )
)
(define-concept EContainsCement(x)
    (some contains Cement )
)


;;Context / Labels

(define-concept Office(x)
    (and
        EContainsDesk(x)
        (and
            EContainsComputer(x)
            (and
                (not EContainsBed(x) )
                (not EContainsOutdoorObjects(x) )
            )
        )
    )
)

(define-concept LivingRoom(x)
    (and
        EContainsCouch(x)
        (and
            (not EContainsKitchenObjects(x) )
            (and
                (not EContainsBed(x))
                (not EContainsOutdoorObjects(x))
            )
        )
    )
)

(define-concept Dormitory(x)
    (and
        EContainsBed(x)
```

```
        (not EContainsOutdoorObjects(x))
    )
)

(define-concept Kitchen(x)
    (and
        EContainsOven(x)
        (and
            EContainsRefrigerator(x)
            (and
                (not EContainsRestingObjects(x))
                (not EContainsOutdoorObjects(x))
            )
        )
    )
)

(define-concept ParkingLot(x)
    (and
        EContainsCar(x)
        (and
            EContainsCement(x)
            (and
                (not EContainsIndoorObjects(x) )
                (and
                    (not EContainsBuilding(x) )
                    (and
                        (not EContainsSand(x) )
                        (and
                            (not EContainsLawn(x) )
                            (not EContainsWater(x) )
                        )
                    )
                )
            )
        )
    )
)

(define-concept UrbanEnv(x)
    (and
        EContainsRoad(x)
        (and
            EContainsBuilding(x)
            (not EContainsIndoorObjects(x) )
        )
    )
)

(define-concept Park(x)
    (and
        EContainsLawn(x)
        (and
            (not EContainsCar(x) )
            (and
```

```
                (not EContainsIndoorObjects(x))
                (and
                    (not EContainsSand(x) )
                    (not EContainsCement(x) )
                )
            )
        )
    )
)

(define-concept Beach(x)
    (and
        EContainsSand(x)
        (and
            EContainsWater(x)
            (and
                (not EContainsLawn(x))
                (and
                    (not EContainsIndoorObjects(x))
                    (and
                        (not EContainsCar(x) )
                        (not EContainsCement(x) )
                    )
                )
            )
        )
    )
)

(define-concept Forest(x)
    (and
        EContainsTree(x)
        (and
            (not EContainsSand(x))
            (and
                (not EContainsCement(x))
                (and
                    (not EContainsIndoorObjects(x))
                    (and
                        (not EContainsTable(x))
                        (not EContainsChair(x))
                    )
                )
            )
        )
    )
)

(define-concept Desert(x)
    (and
        EContainsSand(x)
        (and
            (not EContainsWater(x) )
            (and
                (not EContainsLawn(x) )
```

```
                (and
                    (not EContainsCement(x))
                    (and
                        (not EContainsIndoorObjects(x))
                        (and
                            (not EContainsTable(x))
                            (not EContainsChair(x))
                        )
                    )
                )
            )
        )
    )
)
```

Apêndice C – Ontologia utilizada para criação de mapas semânticos.

Conteúdo do arquivo map.krss usado para a aplicação apresentada na Seção 7.2

```
(probability Ambient(x) [0.2, 0.8] )
(probability Object(x) [0.2, 0.8] )

;; Relations:

(probability contains(x,y) [0.2, 0.8] )
(probability near(x,y) [0.2, 0.8] )

;; Object Hierarchy:

(conditional-probability InsObject(x) Object(x) [0.2, 0.8] )
(define-primitive-concept InsObject(x) Object(x) )

(conditional-probability OutObject(x) Object(x) [0.2, 0.8] )
(define-primitive-concept OutObject(x) Object(x) )

(conditional-probability OfcObject(x) Object(x) [0.2, 0.8] )
(define-primitive-concept OfcObject(x) Object(x) )

;; Placing each object in its specific group:

(conditional-probability Table(x) InsObject(x) [0.2, 0.8] )
(define-primitive-concept Table(x) InsObject(x) )

(conditional-probability Chair(x) InsObject(x) [0.2, 0.8] )
(define-primitive-concept Chair(x) InsObject(x) )

(conditional-probability Cabinet(x) (and InsObject(x) OfcObject(x)) [0.2, 0.8] )
(define-primitive-concept Cabinet(x) (and InsObject(x) OfcObject(x)) )

(conditional-probability Monitor(x) (and InsObject(x) OfcObject(x)) [0.2, 0.8] )
(define-primitive-concept Monitor(x) (and InsObject(x) OfcObject(x)) )

(conditional-probability Printer(x) (and InsObject(x) OfcObject(x)) [0.2, 0.8] )
(define-primitive-concept Printer(x) (and InsObject(x) OfcObject(x)) )

(conditional-probability Sign(x) OutObject(x) [0.2, 0.8] )
(define-primitive-concept Sign(x) OutObject(x) )

(conditional-probability Extinguisher(x) OutObject(x) [0.2, 0.8] )
(define-primitive-concept Extinguisher(x) OutObject(x) )

(conditional-probability Switchboard(x) OutObject(x) [0.2, 0.8] )
(define-primitive-concept Switchboard(x) OutObject(x) )

(conditional-probability Door(x) Object(x) [0.2, 0.8] )
(define-primitive-concept Door(x) Object(x) )

(conditional-probability Board(x) Object(x) [0.2, 0.8] )
(define-primitive-concept Board(x) Object(x) )

;; Existential restrictions that will be needed in composite object descriptors

(define-concept ChairNear(x)
    (some near(x,y) Chair(y) )
```

```
)

(define-concept SignNear(x)
    (some near(x,y) Sign(y) )
)

;; Existential restrictions that will be needed in ambient descriptors

(define-concept ContainsDoor(x)
    (some contains(x,y) Door(y) )
)

(define-concept ContainsTable(x)
    (some contains(x,y) Table(y) )
)

(define-concept ContainsChair(x)
    (some contains(x,y) Chair(y) )
)

(define-concept ContainsCabinet(x)
    (some contains(x,y) Cabinet(y) )
)

(define-concept ContainsMonitor(x)
    (some contains(x,y) Monitor(y) )
)

(define-concept ContainsBoard(x)
    (some contains(x,y) Board(y) )
)

(define-concept ContainsExtinguisher(x)
    (some contains(x,y) Extinguisher(y) )
)

(define-concept ContainsSwitchboard(x)
    (some contains(x,y) Switchboard(y) )
)

(define-concept ContainsInsObject(x)
    (some contains(x,y) InsObject(y) )
)

(define-concept ContainsOutObject(x)
    (some contains(x,y) OutObject(y) )
)

(define-concept ContainsOfcObject(x)
    (some contains(x,y) OfcObject(y) )
)

;; Composite objects

(define-concept Desk(x)
```

```
    (and
        Table(x)
        ChairNear(x)
    )
)
(define-concept ContainsDesk(x)
    (some contains(x,y) Desk(y) )
)

(define-concept Entrance(x)
    (and
        Door(x)
        SignNear(x)
    )
)
(define-concept ContainsEntrance(x)
    (some contains(x,y) Entrance(y) )
)

;; Ambient Descriptors

(define-concept Room(x)
    (and
        (and
            (and
                (and
                    Ambient(x)
                    ContainsDoor(x)
                )
                ContainsTable(x)
            )
            ContainsChair(x)
        )
        (not ContainsOutObject(x))
    )
)

(define-concept Office(x)
    (and
        (and
            (and
                Room(x)
                ContainsDesk(x)
            )
            ContainsCabinet(x)
        )
        ContainsMonitor(x)
    )
)

(define-concept Classroom(x)
    (and
        (and
            Room(x)
            ContainsBoard(x)
```

```
        )
        (not ContainsOfcObject(x))
    )
)

(define-concept Hallway(x)
    (and
        (and
            (and
                (and
                    Ambient(x)
                    ContainsEntrance(x)
                )
                ContainsExtinguisher(x)
            )
            ContainsSwitchboard(x)
        )
        (not ContainsInsObject(x))
    )
)

(define-concept Unknown(x)
    (and
        (and
            Ambient(x)
            (not Room(x))
        )
        (not Hallway(x))
    )
)
```