Algoritmo de Eliminação de Variáveis em Primeira Ordem Aplicado a uma Lógica de Descrição Probabilística

Felipe I. Takiyama[1], **Fabio G. Cozman**[1]

[1] Laboratório de Tomada de Decisão
Escola Politécnica da Universidade de São Paulo (Poli-USP)
Av. Prof. Mello de Morais, 2231 – 05508-900 – São Paulo – SP – Brazil

{felipe.takiyama,fgcozman}@usp.br

Abstract. *There is an ongoing effort to embed probabilities to description logics in the last decade. The combination of graphical probabilistic models (such as Bayesian Networks) with description logics significantly increases the expressivity and flexibility of the latter, but it also introduces a number of complexities when making inferences. In this paper we study the application of a first-order variable elimination algorithm – the* AC-FOVE *to a specific probabilistic description logic – the* CR$\mathcal{ALC}$*. We present the basic theory and demonstrate the computation of inferences through examples.*

Resumo. *A incorporação de probabilidades a lógicas de descrição tem atraído um grande esforço na última década. A combinação de modelos probilísticos gráficos (tais como Redes Bayesianas) e lógicas de descrição permite um aumento significativo na expressividade e flexibilidade desta última, porém adiciona um fator de complexidade na realização de inferências. Neste artigo deseja-se verificar a viabilidade da aplicação de um algoritmo de eliminação de variáveis em primeira ordem – o* AC-FOVE *– à ontologias especificadas em uma lógica de descrição probabilística – o* CR$\mathcal{ALC}$*. Apresenta-se a teoria básica e demonstra-se o cálculo de inferências através de exemplos.*

1. Introdução

Lógicas de descrição são utilizadas para apoiar as tecnologias inseridas no contexto da Web Semântica por oferecer boa flexibilidade e por permitir meios tratáveis para se realizar raciocínio lógico. A ideia básica da lógica de descrição consiste na definição de conceitos, que podem ser usados por classificadores na realização de inferências. Entretanto, a caracterização de conceitos nem sempre pode ser feita de forma categórica, pois alguns termos (principalmente aqueles que envolvem conceitos do mundo real) podem possuir propriedades "opcionais" ou que assumem um valor probabilístico [Heinsohn 1994]. Nesse sentido, existe um esforço crescente para estender lógicas de descrição de maneira que estas possam representar e manipular incertezas [Polastro and Cozman 2008a].

Na teoria de probabilidades, redes Bayesianas são amplamente utilizadas para representar dependências entre variáveis aleatórias. Entretanto, estas redes são representações proposicionais, isto é, a representação de informações acerca de vários indivíduos é proporcional ao tamanho da população. Por outro lado, lógicas de descrição tradicionais podem manipular relações e quantificações de variáveis lógicas, mas não o podem fazer diante de incerteza.

Algumas representações que unem os lógicas de descrição com modelos probabilísticos já foram propostas ([Getoor and Taskar 2007, Raedt et al. 2008]). O problema deste modelos, no entanto, é a dificuldade para se realizar inferência. Alguns algoritmos de inferência exata, como a eliminação de variáveis em primeira ordem [Poole 2003] e inferência "lifted" com fórmulas de contagem [Milch et al. 2008] utilizam artifícios para melhorar o desempenho da inferência, mas exibem problemas de escalabilidade. Existem na literatura propostas de algoritmos de inferência aproximada, que podem utilizar técnicas de amostragem [Gilks and Spiegelhalter 1996] e propagação de crença, tais como o Loopy Belief Propagation [Pearl 1988] e o L2U [Ide and Cozman 2008]. O problema dos algoritmos aproximados é que não existe garantia de convergência, embora resultados empíricos se mostrem bastante razoáveis [Polastro 2012].

Um dos problemas que surgem em inferência "lifted" é a existência de parâmetros extras em um nó pai: durante a proposicionalização, o nó filho terá um número de pais que dependerá do tamanho da população. [Kisynski and Poole 2009] propuseram a criação de uma nova estrutura de dados, denominada fator paramétrico de agregação, que permite esta representação de maneira independente do tamanho da população. Esta estrutura de dados é introduzida no algoritmo C-FOVE [Milch et al. 2008] para realizar inferências, dando origem ao algoritmo AC-FOVE.

Este trabalho propõe aplicar o algoritmo AC-FOVE [Kisynski and Poole 2009] a ontologias especificadas na linguagem de descrição proposta por [Polastro and Cozman 2008b], o Credal $\mathcal{ALC}$ (CR$\mathcal{ALC}$). Um algoritmo de inferência aproximada já existe para o CR$\mathcal{ALC}$ [Polastro and Cozman 2008b], entretanto não existe nenhum algoritmo de inferência exata que já tenha sido aplicado à lógica. Inicia-se introduzindo a teoria básica concernente ao algoritmo e à lógica, e em seguida exibem-se exemplos de inferência para demonstrar a viabilidade desta aplicação.

O restante do trabalho é organizado como se segue. Na Seção 2, introduz-se os conceitos associados à lógica de descrição probabilística CR$\mathcal{ALC}$. A Seção 3 fornece conceitos e definições usados no algoritmo AC-FOVE e descreve o seu funcionamento. Na Seção 4, ilustram-se com exemplos a computação de inferências sobre a lógica CR$\mathcal{ALC}$ usando o AC-FOVE. As conclusões e sugestões para trabalhos futuros são apresentadas na Seção 5.

2. A Lógica de Descrição Probabilística CR$\mathcal{ALC}$

CR$\mathcal{ALC}$ é uma lógica de descrição probabilística proposta por [Polastro and Cozman 2008b] que combina estruturas da lógica $\mathcal{ALC}$ com asserções probabilísticas e uso limitado de nominais. Inicialmente, introduz-se alguns conceitos relacionados à lógica de descrição $\mathcal{ALC}$ e, em seguida, as suposições assumidas pela lógica CR$\mathcal{ALC}$.

Um vocabulário é uma estrutura contendo *indivíduos, conceitos e papéis* [Baader et al. 2003]. *Construtores* são usados para combinar conceitos e papéis. Dados dois conceitos C e D e um papel r, os seguintes construtores estão disponíveis: *conjunção* ($C \sqcup D$), *disjunção* ($C \sqcap C$), *negação* ($\neg C$), *restrição existencial* ($\exists r.C$) e *restrição de valor* ($\forall r.C$). Denota-se uma inclusão de conceito por $C \sqsubseteq D$ e uma definição de conceito por $C \equiv D$. Denomina-se *terminologia* um conjunto de de inclusões e definições. Representa-se o conceito $C \sqcup \neg C$ por $\top$ e o conceito $C \sqcap \neg C$ por $\bot$. Se uma inclusão

ou definição contém um conceito C no lado direito e o conceito D no lado direito, diz-se que C *utiliza diretamente* D. Indica-se o fecho transitivo "utiliza diretamente" por *utiliza*. Uma terminologia é *acíclica* se nenhum conceito utiliza a si mesmo.

A semântica do $\mathcal{ALC}$ é construída a partir de um domínio $\mathcal{D}$ (conjunto não vazio) e uma interpretação $\mathcal{I}$, que mapeia indivíduos para elementos do domínio, nomes de conceito para subconjuntos do domínio e nomes de papéis para relações binárias no conjunto $\mathcal{D} \times \mathcal{D}$. A interpretação $\mathcal{I}$ extende-se para outros conceitos da seguinte maneira: $\mathcal{I}(C_1 \sqcup C_2) = \mathcal{I}(C_1) \cup \mathcal{I}(C_2)$, $\mathcal{I}(C_1 \sqcap C_2) = \mathcal{I}(C_1) \cap \mathcal{I}(C_2)$, $\mathcal{I}(\neg C_1) = \mathcal{D} \backslash \mathcal{I}(C_1)$, $\mathcal{I}(\exists r.C_1) = \{x \in \mathcal{D} \mid \exists y : (x, y) \in \mathcal{I}(r) \ \wedge \ y \in \mathcal{I}(C_1)\}$, $\mathcal{I}(\forall r.C_1) = \{x \in \mathcal{D} \mid \forall y : (x, y) \in \mathcal{I}(r) \rightarrow y \in \mathcal{I}(C_1)\}$. Diz-se que $C_1 \sqsubseteq C_2$ se, e somente se, $\mathcal{I}(C) \subseteq \mathcal{I}(C_2)$ e que $C_1 \equiv C_2$ se, e somente se, $\mathcal{I}(C_1) = \mathcal{I}(C_2)$.

Existem dois tipos de semântica que podem ser adotados por uma lógica de descrição probabilística: na *semântica baseada em domínio*, as probabilidades são designadas para subconjuntos do domínio $\mathcal{D}$; já na *semântica baseada em interpretação*, as probabilidades são especificadas sobre as próprias interpretações. Por exemplo, numa semântica baseada em domínio, a sentença $P_{\mathcal{D}}(Animal) = \alpha$ indica que, ao selecionar aleatoriamente um indivíduo do domínio, a probabilidade de que ele seja um $Animal$ é α; numa semântica baseada em interpretação, a sentença $P(Animal(Formiga)) = \beta$ atribui o valor β para todas as interpretações onde $Formiga$ é um $Animal$.

A partir dos conceitos supracitados, constrói-se a lógica CR$\mathcal{ALC}$ da seguinte maneira:

Inclusões probabilísticas e inferências. A base da lógica CR$\mathcal{ALC}$ é um fragmento da lógica $\mathcal{ALC}$ excluindo-se os papéis temporariamente. Assim, se C e D são conceitos, $\neg C$, $C \sqcap D$ e $C \sqcup D$ também serão conceitos. A inclusão e a definição de conceitos são expressas por $C \sqsubseteq D$ e $C \equiv D$, respectivamente. São permitidas também inclusões probabilísticas $P(C|D) = \alpha$, em que D é um conceito e C é um nome de conceito. Se D for $\top$, escreve-se simplesmente $P(C) = \alpha$.

Aciclicidade. Toda terminologia $\mathcal{T}$ na lógica CR$\mathcal{ALC}$ é acíclica, o que torna possível a sua representação usando um grafo acíclico direto $\mathcal{G}(\mathcal{T})$. Cada nó corresponde a um conceito, e se um conceito C utiliza diretamente o conceito D, então D é *pai* de C em $\mathcal{G}(\mathcal{T})$.

Semântica. Na lógica de descrição CR$\mathcal{ALC}$, utiliza-se semântica baseada em interpretação. A semântica para $P(C|D) = \alpha$ é $\forall x \in \mathcal{D} : P(C(x)|D(x)) = \alpha$, que permite realizar inferências em $P(A(a)|B(b))$ para os conceitos A e B e os indivíduos a e b. Note que não há contradição entre $\forall x : P(C(x)) = \alpha$ e uma observação $C(a) = verdadeiro$ já que $P(C(a)|C(a)) = verdadeiro$ enquanto ainda se tem $P(C(a)) = \alpha$.

Papéis e redes Bayesianas relacionais. Acrescentam-se agora as restrições $\exists r.C$ e $\forall r.C$ à lógica. Suponha que C é um nome de conceito (que pode ter uma definição arbitrariamente complexa). Asserções do tipo $P(\exists r.C|D) = \alpha$ não são permitidas, pois o condicionante deve ser um nome de conceito. A representação gráfica destas restrições é feita criando-se um nó no grafo e conectando-o aos nós que representam o papel r e o conceito C. Como estas restrições usam diretamente r e C, a seta deve partir de r e C apontar para a restrição, conforme mostrado na Figura 1.

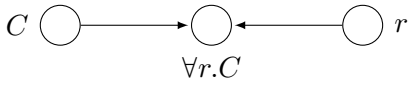

Figura 1. Representação da restrição $\forall r.C$

Independência e condição de Markov. Dada uma terminologia, para todo conceito C e para cada elemento $x \in \mathcal{D}$, $C(x)$ é independente de todas as sentenças que não utilizam $C(x)$, dadas as sentenças que utilizam diretamente C. De maneira análoga, para todo papel r e para cada par $(x, y) \in \mathcal{D} \times \mathcal{D}$, $r(x, y)$ é independente de todas as sentenças que não utilizam $r(x, y)$, dadas as sentenças que utilizam diretamente r.

Homogeneidade. Suponha que um conceito C tenha m pais, $P_1, \ldots, P_m$. Para cada conjunção de conceitos $\pm P_m$, em que o sinal $\pm$ indica se P_m está negado ou não, tem-se $P(C|\pm P_1 \sqcap \pm P_2 \sqcap \cdots \sqcap \pm P_m) = \gamma$ para γ arbitrário. Esta condição permite a representação da terminologia por um grafo acíclico direcionado em que cada nó está associado a uma relação. Caso cada probabilidade tenha um valor preciso, a terminologia pode ser representada por uma rede Bayesiana relacional. Se nem todas as probabilidades forem especificadas precisamente, a proposicionalização gera uma *rede credal* [Cozman 2000]. Assume-se neste texto que todas as probabilidades tenham valores precisos.

Unicidade. Cada indivíduo possui um nome único no domínio. Assume-se também que um conceito C ou é especificado por uma definição ou possui um pai P para o qual são feitas inclusões probabilísticas com respeito a P e $\neg P$. Impõe-se também que para cada papel r existe uma atribuição de probabilidade $P(r) = \alpha$, cuja semântica é dada por $\forall x, y : P(r(x, y)) = \alpha$.

3. O Algoritmo AC-FOVE

O AC-FOVE é um algoritmo proposto por [Kisynski and Poole 2009] para realizar inferências sobre modelos probabilísticos em primeira ordem. Na Seção 3.1 definem-se os conceitos relacionados a modelos probabilísticos de primeira ordem, e na Seção 3.2 explica-se o funcionamento do algoritmo.

3.1. Conceitos e definições

Uma *população* é um conjunto de *indivíduos*, e um *parâmetro* é uma variável lógica atrelada a uma população. Dado um parâmetro A, representa-se a sua população por $\mathcal{D}(A)$. Um *termo* é uma variável lógica ou constante denotando um indivíduo da população. Uma *variável aleatória parametrizada* possui a forma $f(t_1, \ldots, t_k)$, em que f é um functor (isto é, um símbolo de função ou de predicado) e t_i são termos. Cada functor possui um conjunto de valores denominado *coleção* do functor. A coleção do functor f é denotada por $range(f)$. O conjunto de parâmetros de uma variável aleatória parametrizada $f(t_1, \ldots, t_k)$ é denotada por $param(f(t_1, \ldots, t_k))$. Uma *substituição* θ em um conjunto de variáveis lógicas distintas $\{X_1, \ldots, X_k\}$ é representada na forma $\{X_1/t_1, \ldots, X_k/t_k\}$ em que cada X_i é um parâmetro e cada t_i é um parâmetro atrelado a uma população ou a uma constante. Uma *substituição básica* é uma substituição em que cada t_i é uma constante. Uma variável aleatória parametrizada $f(t_1, \ldots, t_k)$ representa um conjunto de variáveis aleatórias, uma para cada substituição básica possível para cada um dos seus

parâmetros. Denota-se este conjunto como $ground(f(t_1, \ldots, t_k))$. Uma *atribuição de valores* a variáveis aleatórias é expressa pela função $\mathbf{v}$, que recebe uma variável aleatória e retorna seu valor. Pode-se estender este conceito para variáveis aleatórias parametrizadas.

Para exemplificar estes conceitos, considere que $Pessoa$ seja uma variável lógica atrelada à população de todas as pessoas em uma universidade $\{p_1, \ldots, p_n\}$. Seja $estudante(Pessoa)$ uma variável aleatória parametrizada, em que $estudante$ é um functor com coleção $\{verdadeiro, falso\}$. Tem-se $range(estudante) = \{verdadeiro, falso\}$ e $param(estudente(Pessoa)) = \{Person\}$. A variável aleatória parametrizada $estudante(Pessoa)$ representa um conjunto de n variáveis aleatórias, uma para cada substituição $\{Pessoa/p_1, \ldots, Pessoa/p_n\}$. Se $\mathbf{v}$ é uma atribuição de valores tal que $\mathbf{v}(estudante(Pessoa)) = falso$ então cada uma das variáveis aleatórias no conjunto $ground(estudante(Pessoa))$ recebe o valor *falso* pelo operador $\mathbf{v}$.

Uma *fórmula de contagem* [Milch et al. 2008] é representada por $\#_{A:\mathcal{C}}[f(\ldots, A, \ldots)]$ em que A é um parâmetro *vinculado* pelo símbolo $\#$, $\mathcal{C}$ é um conjunto de restrições de inegualdade envolvendo A e $f(\ldots, A, \ldots)$ é uma variável aleatória parametrizada. Uma fórmula de contagem é uma variável aleatória parametrizada que representa o histograma dos possíveis valores de uma variável aleatória parametrizada $f(\ldots, a, \ldots)$ satisfazendo $a \in \mathcal{D}(A) : C$. Dada uma atribuição de valores $\mathbf{v}$, o valor de $\#_{A:\mathcal{C}}[f(\ldots, A, \ldots)]$ é o histograma h definido por $h(x) = |\{a \in (\mathcal{D}(A) : \mathcal{C}) : \mathbf{v}(f(\ldots, a, \ldots)) = x\}|$.

Um *fator* representa uma função de tuplas de variáveis aleatórias a números reais. Ele é usado para armazenar a tabela de probabilidades condicionais inicial de cada variável aleatória e os cálculos intermediários durante a inferência. Um *fator paramétrico* ou *parfactor* é uma tripla $\langle \mathcal{C}, \mathcal{V}, \mathcal{F} \rangle$, em que $\mathcal{C}$ é um conjunto de restrições de inegualdade, $\mathcal{V}$ é um conjunto de variáveis aleatórias parametrizadas e $\mathcal{F} : \times_{i \in \mathcal{V}} range(i) \rightarrow \mathbb{R}$ é um fator.

Em modelos direcionados de primeira ordem, se um nó filho possui um nó pai com parâmetros extras, na proposicionalização o nó filho terá um número de pais que depende do tamanho da população dos parâmetros extras. Esta "agregação" de maneira independente do tamanho da população não pode ser feita com parfactors e fórmulas de contagem, o que torna o algoritmo C-FOVE ineficiente na presença desta estrutura. Surge assim a necessidade de criar um operador que descreva a dependência entre variável pai e variável filho.

Neste texto, os modelos probabilísticos de primeira ordem devem satisfazer as seguintes condições:

1. Cada variável aleatória parametrizada possui um pai com no máximo um parâmetro extra;
2. Se um variável aleatória parametrizada c tem um pai p com um parâmetro extra A então:
   (a) p é o único pai de c;
   (b) $range(p) \subseteq range(c)$;
   (c) $c = \bigotimes_{a \in \mathcal{D}(A) p(\ldots, a, \ldots)}$.

Embora as condições pareçam muito restritivas, a expressividade dos modelos não é comprometida. Distribuições mais complicadas podem ser obtidas combinando-se múltiplas agregações e introduzindo-se variáveis auxiliares.

Um *parfactor de agregação* é uma hextupla $\langle \mathcal{C}, p, c, \mathcal{F}_p, \otimes, \mathcal{C}_A \rangle$, em que p e c são variáveis aleatórias parametrizadas (p é pai de c); $range(p) \subseteq range(c)$; A é o único parâmetro em p que não está em c; $\mathcal{C}$ é um conjunto de restrições de inegualdade que não envolvem A; $\mathcal{F}_p$ é um fator em $range(p) \rightarrow \mathbb{R}$; $\otimes$ é um operador binário comutativo e associativo sobre a coleção de c e $\mathcal{C}_A$ é um conjunto de restrições de inegualdade envolvendo A.

Parfactors de agregação representam um conjunto de fatores, um para cada substituição básica em todas as variáveis lógicas de $param(c)$ que satisfazem as restrições em $\mathcal{C}$. Parfactors de agregação podem ser convertidos para parfactors simples usando fórmulas de contagem, o que torna possível sua incorporação ao algoritmo C-FOVE. Entretanto, em algumas situações específicas é possível realizar operações diretamente sobre parfactors de agregação, o que aumenta a eficiência do cálculo durante a inferência.

3.2. Funcionamento do algoritmo

O AC-FOVE utiliza o algoritmo C-FOVE [Milch et al. 2008] como base, incorporando o conceito de parfactors de agregação quando necessário e incluindo o tratamento destes nas operações básicas.

O C-FOVE calcula a distribuição marginal $\mathcal{J}_{\mathbf{Q}}(\Phi) = \sum_{\mathbf{U} \setminus \mathbf{Q}} \mathcal{J}(\Phi)$, em que Φ é um conjunto de parfactors (que podem ser simples ou de agregação), $\mathbf{U}$ e $\mathbf{Q}$ são conjuntos de variáveis aleatórias parametrizadas ($\mathbf{Q} \subseteq \mathbf{U}$) e $\mathcal{J}(\Phi)$ denota o produto de todos os fatores representados pelos elementos de Φ. O cálculo é feito eliminando as variáveis de $\mathbf{U} \setminus \mathbf{Q}$ de maneira não instanciada quando possível.

O algoritmo C-FOVE pode ser descrito a partir de cinco macro-operações básicas:

Particionamento Esta operação manipula um conjunto de parfactors de maneira a garantir que, para quaisquer variáveis aleatórias parametrizadas presentes neles, os conjuntos de variáveis aleatórias representadas por elas sejam idênticos ou disjuntos.

Eliminação Global Dado um produto de parfactors, esta operação elimina uma variável aleatória parametrizada deste produto, isto é, faz a eliminação de cada uma das variáveis aleatórias que ela representa.

Contagem Dado um parfactor e uma variável lógica livre, esta operação elimina esta variável do parfactor. A variável lógica deve aparecer em apenas uma variável aleatória parametrizada do parfactor.

Proposicionalização Esta operação proposicionaliza uma variável lógica livre de um parfactor.

Expansão Total Expande uma fórmula de contagem em toda constante que pertença ao domínio da variável contada e satisfaça as restrições.

Dado um conjunto de parfactors Φ e um conjunto de variáveis aleatórias $\mathbf{Q}$, o algoritmo AC-FOVE calcula o valor da distribuição marginal $\mathcal{J}_{\mathbf{Q}}(\Phi)$. Inicialmente, realiza-se uma macro-operação de Particionamento. O Particionamento também é realizado com respeito às variáveis aleatórias em $\mathbf{Q}$. Em seguida, o algoritmo elimina as variáveis não consultadas de Φ realizando uma das outras quatro macro-operações. As operações são selecionadas segundo uma política de busca gulosa, em que o custo de cada operação é definido como o tamanho total dos parfactors que ela cria. As operações de Eliminação Global e Contagem nem sempre podem ser realizadas devido às suas pré-condições; em

contrapartida as operações de Proposicionalização e Expansão Total sempre podem ser executadas. Em casos extremos, todo o conjunto de parfactors pode ser proposicionalizado, o que implica na completeza do algoritmo.

4. Análise

Nesta seção, serão apresentados três exemplos de cálculo de inferência sobre ontologias simples em CR$\mathcal{ALC}$. No primeiro exemplo explora-se o uso do algoritmo AC-FOVE sem agregações. No segundo exemplo utiliza-se uma restrição na consulta para demonstrar sua aplicabilidade na lógica CR$\mathcal{ALC}$. Nestes exemplos serão utilizados trechos da extensa ontologia Wine, extraídos de um arquivo OWL do repositório de ontologias do *Temporal Knowledge Base Group* da Universidade Jaume I[1]. No terceiro exemplo, utiliza-se uma versão adaptada do arquivo de exemplo do sistema *CEL* (um classificador para a lógica de descrição $\mathcal{EL}+$[2]), em que se faz uso de agregações para realizar inferências.

4.1. Exemplo 1 - Expressividade da Lógica

Considere a o trecho da ontologia Wine que define o conceito "Vinho do Porto", já adaptado para a lógica CR$\mathcal{ALC}$:

$$\begin{aligned}
\mathsf{Port(x)} \equiv & \mathsf{RedWine(x)} \sqcap \mathsf{HasWineBodyFull(x)} \\
& \sqcap \mathsf{HasWineFlavourStrong(x)} \sqcap \mathsf{HasWineSugarSweet(x)} \\
& \sqcap \mathsf{LocatedInPortugalRegion(x)} \\
\mathsf{RedWine(x)} \equiv & \mathsf{Wine(x)} \sqcap \mathsf{HasWineColorRed(x)} \\
P(\mathsf{Wine(x)}) = & 0.9 \\
P(\mathsf{HasWineColorRed(x)}) = & 0.33 \\
P(\mathsf{HasWineBodyFull(x)}) = & 0.33 \\
P(\mathsf{HasWineFlavourStrong(x)}) = & 0.33 \\
P(\mathsf{HasWineSugarSweet(x)}) = & 0.33 \\
P(\mathsf{LocatedInPortugalRegion(x)}) = & 0.15
\end{aligned}$$

Suponha que $\mathcal{D}(\mathsf{x}) = \{x_1, \ldots, x_n\}$. O grafo da ontologia é ilustrado na Figura 2.

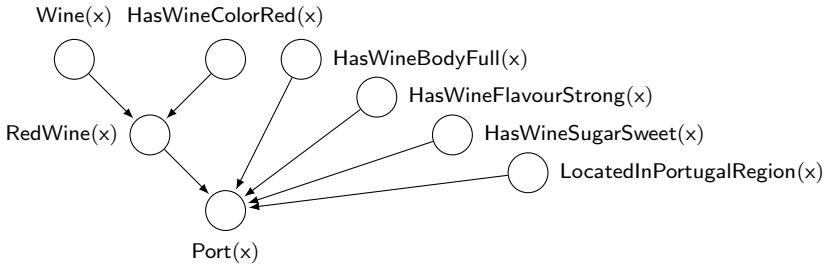

Figura 2. Trecho da ontologia Wine que define o conceito "Vinho do Porto"

[1]Disponível em `http://krono.act.uji.es/Links/ontologies/wine.owl/view`
[2]Disponível em `http://lat.inf.tu-dresden.de/~meng/ontologies/kangaroo.cl`

Suponha que se deseja calcular $P(\mathsf{Port(x)}) = \mathcal{J}_{\mathsf{Port(x)}}(\Phi_0)$, em que Φ_0 é o conjunto de parfactors inicial:

$$\begin{aligned}
\Phi_0 = \{&\langle \varnothing, \{\mathsf{Wine(x)}\}, \mathcal{F}_1\rangle, && [1]\\
&\langle \varnothing, \{\mathsf{HasWineColorRed(x)}\}, \mathcal{F}_2\rangle, && [2]\\
&\langle \varnothing, \{\mathsf{RedWine(x)}, \mathsf{Wine(x)}, \mathsf{HasWineColorRed(x)}\}, \mathcal{F}_3\rangle, && [3]\\
&\langle \varnothing, \{\mathsf{HasWineBodyFull(x)}\}, \mathcal{F}_4\rangle, && [4]\\
&\langle \varnothing, \{\mathsf{HasWineFlavourStrong(x)}\}, \mathcal{F}_5\rangle, && [5]\\
&\langle \varnothing, \{\mathsf{HasWineSugarSweet(x)}\}, \mathcal{F}_6\rangle, && [6]\\
&\langle \varnothing, \{\mathsf{LocatedInPortugalRegion(x)}\}, \mathcal{F}_7\rangle, && [7]\\
&\langle \varnothing, \{\mathsf{RedWine(x)}, \mathsf{HasWineBodyFull(x)}, \mathsf{HasWineFlavourStrong(x)},\\
&\qquad \mathsf{HasWineSugarSweet(x)}, \mathsf{LocatedInPortugalRegion(x)}, \mathsf{Port(x)}\}, \mathcal{F}_8\rangle\} && [8]
\end{aligned}$$

O algoritmo calculará primeiro o produto dos parfactors [1], [2] e [3], e em seguida eliminará as variáveis $\mathsf{Wine(x)}$ e $\mathsf{HasWineColorRed(x)}$ resultando no parfactor

$$\langle \varnothing, \{\mathsf{RedWine(x)}\}, \mathcal{F}_9\rangle \qquad [9]$$

em que $\mathcal{F}_9 = \sum_{\substack{\mathsf{Wine(x)}\\ \mathsf{HasWineColorRed(x)}}} \mathcal{F}_1 \odot \mathcal{F}_2 \odot \mathcal{F}_3$. Após esta operação, o algoritmo calcula o produto dos parfactors [4], [5], [6], [7], [8] e [9] e elimina as variáveis $\mathsf{RedWine(x)}$, $\mathsf{HasWineBodyFull(x)}$, $\mathsf{HasWineFlavourStrong(x)}$, $\mathsf{HasWineSugarSweet(x)}$ e $\mathsf{LocatedInPortugalRegion(x)}$ resultando no parfactor

$$\Phi_1 = \{\langle \varnothing, \{\mathsf{Port(x)}\}, \mathcal{F}_{10}\rangle\} \qquad [10]$$

em que $\mathcal{F}_{10} = \sum_{\substack{\mathsf{RedWine(x)},\\ \mathsf{HasWineBodyFull(x)},\\ \mathsf{HasWineFlavourStrong(x)},\\ \mathsf{HasWineSugarSweet(x)},\\ \mathsf{LocatedInPortugalRegion(x)}}} \mathcal{F}_4 \odot \mathcal{F}_5 \odot \mathcal{F}_6 \odot \mathcal{F}_7 \odot \mathcal{F}_8 \odot \mathcal{F}_9$. Tem-se $\mathcal{J}_{\mathsf{Port(x)}}(\Phi_0) = \mathcal{J}(\Phi_1)$. Numericamente, tem-se $P(\mathsf{Port(x)}) = 0,0016$.

4.2. Exemplo 2 - Utilização de Restrição

Considere a o trecho da ontologia Wine que define o conceito "Vinho Americano", já adaptado para a lógica CR$\mathcal{ALC}$: $\mathsf{AmericanWine(x)} \equiv \mathsf{Wine(x)} \sqcap \mathsf{LocatedInUSRegion(x)}$, $P(\mathsf{Wine(x)}) = 0.9$, $P(\mathsf{LocatedInUSRegion(x)}) = 0.15$. O grafo da ontologia é ilustrado na Figura 3.

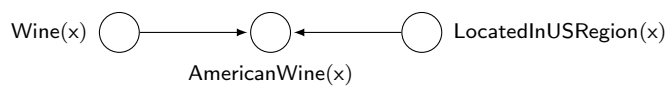

Figura 3. Trecho da ontologia Wine que define o conceito "Vinho Americano"

Suponha que $\mathcal{D}(x) = \{x_1, \ldots, x_n\}$ e que foi observado que x_1 é um vinho americano (representado pelo parfactor $\langle \varnothing, \{\mathsf{AmericanWine(x_1)}\}, \mathcal{F}_4\rangle$). Deseja-se calcular

$\mathcal{J}_{\text{ground(AmericanWine(x)):}\{x \neq x_1\}}(\Phi_0)$, em que Φ_0 é o conjunto de parfactors inicial:

$$\Phi_0 = \{\langle \varnothing, \{\mathsf{Wine(x)}\}, \mathcal{F}_1\rangle, \quad [1]$$
$$\langle \varnothing, \{\mathsf{LocatedInUSRegion(x)}\}, \mathcal{F}_2\rangle, \quad [2]$$
$$\langle \varnothing, \{\mathsf{AmericanWine(x)}, \mathsf{Wine(x)}, \mathsf{LocatedInUSRegion(x)}\}, \mathcal{F}_3\rangle, \quad [3]$$
$$\langle \varnothing, \{\mathsf{AmericanWine}(\mathsf{x}_1)\}, \mathcal{F}_4\rangle\} \quad [4]$$

O algoritmo inicia invocando a operação de Particionamento sobre os parfactors [1], [2] e [3] na substituição $\{x/x_1\}$, resultando no conjunto de parfactors Φ_1:

$$\Phi_0 = \{\langle \{x \neq x_1\}, \{\mathsf{Wine(x)}\}, \mathcal{F}_1\rangle, \quad [5]$$
$$\langle \{x \neq x_1\}, \{\mathsf{LocatedInUSRegion(x)}\}, \mathcal{F}_2\rangle, \quad [6]$$
$$\langle \{x \neq x_1\}, \{\mathsf{AmericanWine(x)}, \mathsf{Wine(x)}, \mathsf{LocatedInUSRegion(x)}\}, \mathcal{F}_3\rangle, \quad [7]$$
$$\langle \varnothing, \{\mathsf{AmericanWine}(\mathsf{x}_1)\}, \mathcal{F}_4\rangle, \quad [8]$$
$$\langle \varnothing, \{\mathsf{Wine}(\mathsf{x}_1)\}, \mathcal{F}_1\rangle, \quad [9]$$
$$\langle \varnothing, \{\mathsf{LocatedInUSRegion}(\mathsf{x}_1)\}, \mathcal{F}_2\rangle, \quad [10]$$
$$\langle \varnothing, \{\mathsf{AmericanWine}(\mathsf{x}_1), \mathsf{Wine}(\mathsf{x}_1), \mathsf{LocatedInUSRegion}(\mathsf{x}_1)\}, \mathcal{F}_3\rangle, \} \quad [11]$$

Em seguida, o algoritmo multiplica os parfactors [5], [6] e [7] e elimina as variáveis Wine(x) e LocatedInUSRegion(x). De maneira análoga, os parfactors [8], [9], [10] e [11] são multiplicados e as variáveis $\mathsf{Wine}(\mathsf{x}_1)$ e $\mathsf{LocatedInUSRegion}(\mathsf{x}_1)$ são eliminadas. O conjunto de parfactors se torna então

$$\Phi_1 = \{\langle \{x \neq x_1\}, \{\mathsf{AmericanWine(x)}\}, \mathcal{F}_4\rangle, \quad [12]$$
$$\langle \varnothing, \{\mathsf{AmericanWine}(\mathsf{x}_1)\}, \mathcal{F}_5\rangle\} \quad [13]$$

em que $\mathcal{F}_4 = \sum_{\mathsf{Wine(x)},\mathsf{LocatedInUSRegion(x)}} \mathcal{F}_1 \odot \mathcal{F}_2 \odot \mathcal{F}_3$ e $\mathcal{F}_5 = \sum_{\mathsf{Wine}(\mathsf{x}_1),\mathsf{LocatedInUSRegion}(\mathsf{x}_1)} \mathcal{F}_1 \odot \mathcal{F}_2 \odot \mathcal{F}_3$. Não é possível multiplicar os fatores [12] e [13], de maneira que é necessário eliminar a variável x de [12] realizando uma contagem nesta variável [Milch et al. 2008]. Ao realizar esta operação, obtém-se um novo parfactor $\langle \varnothing, \{\#_{x:\{x \neq x_1\}}[\mathsf{AmericanWine(x)}]\}, \mathcal{F}_6\rangle$, que por sua vez pode ser multiplicado por [13]; do resultado se elimina a variável $\mathsf{AmericanWine}(\mathsf{x}_1)$ para obter

$$\Phi_2 = \{\langle \varnothing, \{\#_{x:\{x \neq x_1\}}[\mathsf{AmericanWine(x)}]\}, \mathcal{F}_7\rangle\} \quad [14]$$

em que $\mathcal{F}_7 = \sum_{\mathsf{AmericanWine}(\mathsf{x}_1)} \mathcal{F}_5 \odot \mathcal{F}_6$. Tem-se $\mathcal{J}_{\text{ground(AmericanWine(x)):}\{x \neq x_1\}}(\Phi_0) = \mathcal{J}(\Phi_2)$.

4.3. Exemplo 3 - Utilização de Agregação

Considere a versão adaptada da ontologia Kangaroo, definida da seguinte maneira: $P(\mathsf{Animal(x)}) = 0.9$, $P(\mathsf{Rational(x)}) = 0.6$, $P(\mathsf{hasChild(x,y)}) = 0.3$, $\mathsf{Human(x)} \equiv \mathsf{Animal(x)} \sqcap \mathsf{Rational(x)}$, $\mathsf{Beast(x)} \equiv \mathsf{Animal(x)} \sqcap \neg\mathsf{Rational(x)}$, $\mathsf{Kangaroo(x)} \sqsubseteq \mathsf{Beast(x)}$, $P(\mathsf{Kangaroo(x)}|\mathsf{Beast(x)}) = 0.4$, $\mathsf{Parent(x)} \equiv \exists\mathsf{hasChild(x,y)}.\mathsf{Human(y)}$, $\mathsf{MaternityKangaroo(x)} \equiv \mathsf{Kangaroo(x)} \sqcap \exists\mathsf{hasChild(x,y)}.\mathsf{Kangaroo(y)}$. Usando a teoria apresentada na Seção 2, pode-se construir o grafo da ontologia, exibido na Figura 4.

Suponha que se deseja calcular $P(\mathsf{Parent(x)})$ e que todos os conceitos e relações tenham domínio booleano. Considere que x tenha uma população finita $\mathcal{D}$. No resto

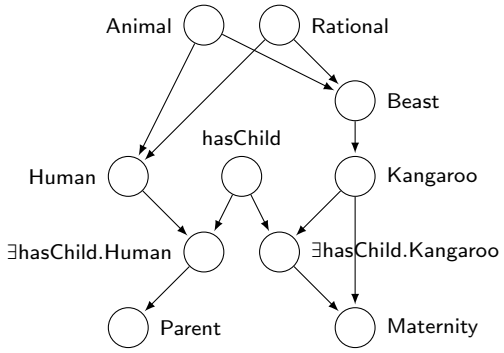

Figura 4. Ontologia Kangaroo representada como uma rede Bayesiana

deste artigo, far-se-á referência aos nós por suas iniciais, e os nós ∃hasChild.Human e ∃hasChild.Kangaroo serão referidos por ∃h.H e ∃h.K, respectivamente. Como não existe nenhuma evidência, tem-se $\mathsf{H}, \mathsf{hH}, \mathsf{P} \perp\!\!\!\perp \mathsf{B}, \mathsf{K}, \mathsf{hK}, \mathsf{M} \mid \mathsf{A}, \mathsf{R}, \mathsf{h}$, ou seja, os nós $\mathsf{B}, \mathsf{K}, \mathsf{hK}, \mathsf{M}$ não influenciam no resultado de $P(\mathsf{Parent(x)})$ e portanto podem ser eliminados. Os nós Animal, Rational e Human podem ser agregados em um único nó calculando-se a distribuição de probabilidade conjunta deles e eliminando-se as variáveis Animal e Rational.

Para utilizar o AC-FOVE, o nó agregado deve ser o único pai de seu nó filho. Assim, introduz-se um nó auxiliar $C(x, y) = h(x, y) \sqcap H(y)$. A rede resultante é exibida na Figura 5.

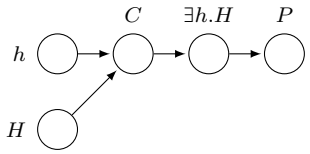

Figura 5. Representação reduzida do grafo da ontologia Kangaroo

Calcular $P(\mathsf{P(x)})$ é o mesmo que computar $\mathcal{J}(\Phi_0)$, em que Φ_0 é o conjunto de parfactors inicial:

$$\begin{aligned}
\Phi_0 = \{&\langle \varnothing, \{\mathsf{H(y)}\}, \mathcal{F}_H \rangle, && [1]\\
&\langle \varnothing, \{\mathsf{h(x,y)}\}, \mathcal{F}_h \rangle, && [2]\\
&\langle \varnothing, \{\mathsf{H(y)}, \mathsf{h(x,y)}, \mathsf{C(x,y)}\}, \mathcal{F}_C \rangle, && [3]\\
&\langle \varnothing, \mathsf{C(x,y)}, \exists\mathsf{h.H(x)}, \mathcal{F}, \text{OR}, \varnothing \rangle, && [4]\\
&\langle \varnothing, \{\exists\mathsf{h.H(x)}, \mathsf{P(x)}\}, \mathcal{F}_P \rangle\} && [5]
\end{aligned}$$

O algoritmo inicia multiplicando os parfactors [1], [2] e [3], e em seguida eliminando as variáveis $\mathsf{H(y)}$ e $\mathsf{h(x,y)}$ resultando em

$$g_1 = \langle \varnothing, \{\mathsf{C(x,y)}\}, \mathcal{F}_1 \rangle \quad [6]$$

em que $\mathcal{F}_1 = \sum_{\mathsf{H(y)},\mathsf{h(x,y)}} \mathcal{F}_H \odot \mathcal{F}_h \odot \mathcal{F}_C$. Multiplica-se então o parfactor [6] com o parfactor de agregação [4], elimina-se a variável $\mathsf{C(x,y)}$, multiplica-se o resultado pelo parfactor [5] e finalmente elimina-se a variável $\exists\mathsf{h.H(x)}$ para se obter

$$\Phi_1 = \{\langle \varnothing, \{\mathsf{P(x,y)}\}, \mathcal{F}_2 \rangle\} \quad [7]$$

em que $\mathcal{F}_2 = \sum_{\exists \mathsf{h}.\mathsf{H}(\mathsf{x})} \left(\mathcal{F}_P \odot \left(\sum_{\mathsf{C}(\mathsf{x})} \mathcal{F}_1 \odot \mathcal{F} \right) \right)$. Tem-se $\mathcal{J}(\Phi_0) = \mathcal{J}(\Phi_1)$.

5. Conclusões e Trabalhos Futuros

Neste artigo, mostrou-se que é viável aplicar o algoritmo de inferência exata em primeira ordem AC-FOVE para realizar inferências em ontologias especificadas usando a lógica de descrição probabilística CR$\mathcal{ALC}$. O algoritmo AC-FOVE utiliza estruturas de dados específicas para reduzir a necessidade de proposicionalização da rede e, consequentemente, aumentar a eficiência do cálculo. Em particular, os parfactors de agregação permitem expressar relações mais complexas independentemente do tamanho da população.

Nos exemplos fornecidos, objetivou-se explorar os diversos conceitos utilizados tanto pelo algoritmo como pela lógica. No primeiro exemplo, ilustrou-se a expressividade da lógica ao se mostrar o potencial de descrição de uma ontologia extensa e complexa. No segundo exemplo, demonstra-se a aplicabilidade de restrições na realização de inferências. O uso de restrições permite realizar consultas mais flexíveis (por exemplo, permitindo a exclusão de um indivíduo da população) ao custo de maior complexidade computacional. Finalmente, no terceiro exemplo, utilizou-se parfactors de agregação para reduzir o tempo de computação da inferência. Caso não fossem utilizados parfactors de agregação, isto é, fosse utilizado apenas o algoritmo C-FOVE, haveria a necessidade de proposicionalizar o modelo, o que tornaria a inferência inviável para populações maiores.

Para trabalhos futuros, sugere-se investigar a possibilidade de ampliar a expressividade dos parfactors, de maneira que se possa realizar consultas mais flexíveis, como por exemplo permitir o uso de outros tipos de restrições além da inegualdade. Este aumento de expressividade aumenta também o tempo para o processamento das restrições, e analisar o ganho obtido (se ele existir) é um desafio interessante. Outra possibilidade de extensão seria investigar a possibilidade de se realizar inferências com domínios infinitos. O próximo passo deste trabalho é desenvolver um software que utilize o algoritmo AC-FOVE para realizar inferência sobre a lógica CR$\mathcal{ALC}$ e posteriormente utilizá-lo em problemas do mundo real que envolvam ontologias mais complexas e domínios maiores, de maneira a testar a escalabilidade do algoritmo.

6. Agradecimentos

O segundo autor é parcialmente apoiado pelo CNPq. O trabalho aqui apresentado recebeu financiamento da FAPESP, projeto 2008/03995-5.

Referências

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Cozman, F. G. (2000). Credal networks. *Artificial Intelligence*, 120(2):199–233.

Getoor, L. and Taskar, B. (2007). *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Gilks, W. and Spiegelhalter, D. (1996). *Markov chain Monte Carlo in practice*. Chapman & Hall CRC.

Heinsohn, J. (1994). Probabilistic description logics. In *UAI*, pages 311–318.

Ide, J. S. and Cozman, F. G. (2008). Approximate algorithms for credal networks with binary variables. *Int. J. Approx. Reasoning*, 48(1):275–296.

Kisynski, J. and Poole, D. (2009). Lifted aggregation in directed first-order probabilistic models. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 1922–1929, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Milch, B., Zettlemoyer, L. S., Kersting, K., Haimes, M., and Kaelbling, L. P. (2008). Lifted probabilistic inference with counting formulas. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, AAAI'08, pages 1062–1068. AAAI Press.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Polastro, R. B. (2012). *Lógica Probabilística Baseada em Redes Bayesianas Relacionais com Inferência em Primeira Ordem*. PhD thesis, Escola Politécnica da Universidade de São Paulo.

Polastro, R. B. and Cozman, F. G. (2008a). Inference in probabilistic ontologies with attributive concept descriptions and nominals. In *URSW*, volume 423 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Polastro, R. B. and Cozman, F. G. (2008b). Loopy propagation in a probabilistic description logic. In *Proceedings of the 2nd international conference on Scalable Uncertainty Management*, SUM '08, pages 120–133, Berlin, Heidelberg. Springer-Verlag.

Poole, D. (2003). First-order probabilistic inference. In *Proceedings of the 18th international joint conference on Artificial intelligence*, IJCAI'03, pages 985–991, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Raedt, L. D., Frasconi, P., Kersting, K., and Muggleton, S., editors (2008). *Probabilistic Inductive Logic Programming - Theory and Applications*, volume 4911 of *Lecture Notes in Computer Science*. Springer.