REVISÃO DE CLASSIFICADORES BAYESIANOS DE

PRIMEIRA ORDEM

Kate Cerqueira Revoredo

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Gerson Zaverucha, Ph.D.

Prof. Valmir Carneiro Barbosa, Ph.D.

Prof. João Carlos Pereira da Silva, D.Sc.

Prof. Fabio Gagliardi Cozman, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

MAIO DE 2002

REVOREDO, KATE CERQUEIRA

Revisão de Classificadores Bayesianos de Primeira Ordem [Rio de Janeiro] 2002

XII, 102p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2002)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Redes Bayesianas
2. Lógica de Primeira Ordem
3. Revisão de Teoria
4. Classificadores Bayesianos

I. COPPE/UFRJ II. Título(série)

Dedicatória

A minha querida mãe, Luiza.

Agradecimentos

À minha família em especial,
- a minha querida mãe, Luiza, pelo apoio e incentivo constante em todas as etapas da minha vida e, principalmente, pelas facilidades que me proporcionou no dia a dia, duplicando o meu tempo de estudo nos últimos dois anos.
- ao meu noivo, Leandro, pela enorme paciência e compreensão nos muitos dias de ausência.
- ao meu irmão Téo, pelas sugestões técnicas na elaboração do material para apresentação da tese.

Ao meu caro orientador, Professor Doutor Gerson Zaverucha pela confiança em mim depositada, pela sua dedicação e grande disposição na transmissão de conhecimento.

Aos meus amigos pelo apoio e carinho, em especial
- a Fernanda Baião, pelo desenvolvimento em conjunto de parte da implementação, pela motivação e pelo grande exemplo.
- a Michel Carlini, por suas sugestões, fruto da sua grande capacidade de observação, e principalmente, por seu companherismo, fator importante na conclusão do trabalho.
- a Carla Delgado, pelas conversas e trocas de informações.

Aos Professores Doutores Vítor Santos Costa e João Carlos Pereira da Silva pela ajuda no esclarecimento de alguns pontos da implementação.

À Rodrigo Basílio pelo desenvolvimento em conjunto de uma das interfaces do sistema.

Ao CNPq pelo suporte financeiro.

À Deus por minha vida.

"...e porque estreita é a porta e apertado o caminho que leva a vida, e poucos há que a encontrem" (Mateus; 7,14)

Obrigado.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

REVISÃO DE CLASSIFICADORES BAYESIANOS DE PRIMEIRA ORDEM

Kate Cerqueira Revoredo

Maio/2002

Orientador: Gerson Zaverucha

Programa: Engenharia de Sistemas e Computação

Programas em Lógica Bayesianos (BLP), proposto por Kersting e De Raedt, são uma abordagem poderosa e elegante de combinar a expressividade da lógica de primeira ordem com redes Bayesianas. Eles podem representar redes Bayesianas e programas em lógica, e seu núcleo em Prolog é uma adaptação de um meta-interpretador Prolog. Foi comparado com sucesso a outras abordagens existentes na literatura, incluindo Modelos Probabilísticos Relacionais (PRM). Algoritmos para aprender as componentes qualitativa e quantitativa de PRMs e BLPs foram desenvolvidos. Nesta tese desenvolvemos um algoritmo para revisar um BLP, composto por dois procedimentos. O primeiro procedimento encontra o BLP de maior probabilidade dentre um espaço de busca composto por hipóteses modificadas em lugares que falharam na classificação. Utiliza Revisão de Teorias através do sistema FORTE, desenvolvido por Richards e Mooney, o qual induz uma revisão de programas em lógica através de exemplos , e uma adaptação do algoritmo EM, proposto por Pfeffer e Koller, para o aprendizado da componente quantitativa(CPDs) do BLP. O segundo procedimento, busca o BLP de máxima probabilidade. Propomos que para classificação, quando o BLP a ser revisto é aproximadamente correto, apenas o primeiro precise ser utilizado, e por usar técnicas de revisão de teoria e assim percorrer um espaço de hipóteses menor, pode ser uma escolha mais adequada do que os dois algoritmos de aprendizado acima mencionados.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

REVISION OF FIRST-ORDER BAYESIAN CLASSIFIERS

Kate Cerqueira Revoredo

May/2002

Advisor: Gerson Zaverucha

Program: Systems Engineering and Computer Science

Bayesian Logic Programs (BLP) by Kersting and De Raedt, is a powerful and elegant framework for combining the expressiveness of first order logic with Bayesian networks. It can represent both Bayesian networks and logic programs, and its kernel in Prolog is an adaptation of an usual Prolog meta-interpreter. It has been successfully compared to other such proposals in the literature, including Probabilistic Relational Models(PRM). Algorithms have been developed in order to learn the qualitative and quantitative components of PRMs and BLPs. In this thesis we have developed an algorithm to revise a BLP, composed of two procedures. The first procedure finds the BLP with the highest probability in a search space composed of hypotheses only modified in places that failed in classification. It uses Theory Revision, through the FORTE system, developed by Richards and Mooney, which induces a revision of Logic Programs from examples, and an adaptation of the algorithm EM proposed Koller and Pfeffer for learning the quantitative component (CPDs) of the BLP. The second procedure searches for the BLP with the highest probabilistic scoring. We argue that when the BLP is approximately correct, only the first procedure needs to be run, and because of the use of theory revision techniques, it searches a smaller hypotheses space, and so can be a more adequate choice then the others approaches aforementioned.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução

A possibilidade de representação de indivíduos , suas propriedades e as relações entre eles, fizeram da lógica de primeira ordem um sistema de representação de conhecimento muito vantajoso, mas limitado por não representar incerteza (determinístico). A Rede Bayesiana por outro lado possibilita a representação de conhecimento probabilístico, mas por ser proposicional, ou seja, baseada em atributo, não suporta a descrição do domínio em termos de regras mais gerais, o que possibilitaria a aplicação em diferentes situações. Recentemente tem sido grande o interesse em integrar estas duas abordagens, definindo uma teoria probabilística de primeira ordem, unindo o que elas têm de melhor.

Exemplo de abordagens existentes na literatura para estender redes Bayesianas para lógica de primeira ordem são: *probabilistic logic programs* (NGO, HADDAWY, 1995), *probabilistic relational models* (KOLLER, 1999) (FRIEDMAN, GETOOR, et al., 1999), *relational Bayesian nets* (JAEGER, 1997), *independent choice logic* (POOLE, 1993) (POOLE, 1998) (POOLE, 2000), *first-order Bayesian reasoning* (FABIAN, LAMBERT, 1998) e *Bayesian Logic Programs (BLP)* (KERSTING, De RAEDT, et al., 2000)(KERSTING, De RAEDT, 2000)(KERSTING, De RAEDT, 2001b). Muitas destas têcnicas utilizam a noção de Construção de Modelos de Base de Conhecimento (KBMC) (HADDAWY, 1999), onde regras de primeira ordem associadas com parâmetros probabilísticos, indicando incerteza, são usadas como base para gerar redes Bayesianas proposicionais para perguntas específicas. Em

BLP, que foi comparado a todas as abordagens mencionadas acima (KERSTING, De RAEDT, 2000), as regras de primeira ordem são representadas como um programa em lógica (como Prolog) e, como em redes Bayesianas, existe uma distribuição de probabilidades condicionais (CPD) associada a cada regra.

Algoritmos para aprender PRMs e BLPs já foram propostos na literatura (apresentar-lo-emos em mais detalhes no próximo capítulo). Em ambos, modificações em toda a estrutura são propostas, sendo a melhor escolhida através da utilização de uma função de avaliação probabilística. Uma diferença entre o algoritmo de aprendizado de um BLP e de um PRM é que o primeiro exige que todos os BLPs modificados propostos reflitam corretamente a base de dados.

Suponha que a teoria probabilística de primeira ordem fornecida seja aproximadamente correta, ou seja, apenas alguns pontos da sua estrutura estão evitando que ela reflita corretamente a base de dados. Se for possível utilizar esta base de dados para determinar quais são estes pontos, propondo então modificações apenas nestes pontos que ao final retornem uma teoria correta, seria muito mais vantajoso, ou seja mais eficiente, do que utilizar um algoritmo que proporá modificações em toda a estrutura.

Para ilustrar como a base de dados pode ser utilizada para identificar os pontos que estão evitando que uma teoria seja correta, vamos considerar que no domínio da família[1], onde a teoria seria um conjunto de cláusulas definindo relações como esposa(X,Y) ← gênero(X, feminino), casado(X,Y), a relação que define tio fosse:

tio(X,Y) ← gênero(X,masculino), tia_tio(X,Y).

tia_tio(X,Y) ← irmão_irmã(X,Z), mãe_pai(Z,Y).

irmão_irmã(X,Y) ← mãe_pai(Z,X), mãe_pai(Z,Y).

Podemos ver que no caso de alguém casado com uma pessoa que é tia, esta não será considerada como sendo um tio. Por exemplo, se quisermos saber se John é tio de Jane (tia(john,jane)), dado que sabemos que John é do gênero masculino (gênero(john,masculino)) e casado com Lisa (casado(john,lisa)), onde esta é irmã da mãe de Jane (mãe_pai(ann,kari), mãe_pai(ann,lisa), mãe_pai(kari,jane)), não conseguiríamos, pois ocorrerá uma falha no predicado irmão_irmã(john,Z). Vemos então

[1]No decorrer de toda esta tese utilizaremos o problema da família (tabela 1.1) para exemplificar as situações que estivermos considerando.

esposa(X,Y) | ← gênero(X,feminino), casado(X,Y).
marido(X,Y) ← gênero(X,masculino), casado(X,Y).
mãe(X,Y) ← gênero(X,feminino), mãe_pai(X,Y).
pai(X,Y) ← gênero(X,masculino), mãe_pai(X,Y).
filha(X,Y) ← gênero(X,feminino), mãe_pai(Y,X).
filho(X,Y) ← gênero(X,masculino), mãe_pai(Y,X).
irmã(X,Y) ← gênero(X,feminino), irmão_irmã(X,Y).
irmão(X,Y) ← gênero(X,masculino), irmão_irmã(X,Y).
tia(X,Y) ← gênero(X,feminino), tia_tio(X,Y).
tio(X,Y) ← gênero(X,masculino), tia_tio(X,Y).
tia_tio(X,Y) ← casado(X,Z), irmão_irmã(Z,W), mãe_pai(W,Y).
tia_tio(X,Y) ← irmão_irmã(X,Z), mãe_pai(Z,Y).
irmão_irmã(X,Y) ← mãe_pai(Z,X), mãe_pai(Z,Y).

Tabela 1.1: Representação do Problema da Família

que os exemplos podem ser utilizados para identificar pontos na teoria que precisam ser revistos.

Uma possível revisão para este exemplo seria a inclusão da seguinte cláusula:

$$tia\_tio(X,Y) \leftarrow casado(X,Z), irmão\_irmã(Z,W), mãe\_pai(W,Y)$$

o que permite que as duas situações em que uma pessoa pode ser definida como sendo tio de alguém estarão sendo satisfeitas.

Tendo então a diminuição do espaço de busca e conseqüentemente uma melhora em desempenho do algoritmo de busca como motivação, nós propomos um sistema de revisão de programas em lógica Bayesiano denominado por nós RBLP, que espera receber um BLP inicial e através dos exemplos descobrir pontos que falham neste BLP propondo modificações para estes e escolhendo a melhor de acordo com uma função de avaliação. O BLP retornado será correto.

O RBLP ((REVOREDO, ZAVERUCHA, 2001) e (REVOREDO, ZAVERUCHA, 2002))), é composto de dois procedimentos. O primeiro (RBLP_AP) tem por objetivo modificar minimamente um BLP fornecido para torná-lo consistente com um conjunto de exemplos de treinamento utilizando uma função de avaliação probabilística, para escolha da melhor modificação feita a este BLP. Por ter como objetivo a melhora na classificação, RBLP_AP utilizará um procedimento para identificar pontos no BLP que falharam na classificação, procurando então, dentre um conjunto de possíveis operadores de revisão, aquele que maximiza a função de avaliação, per-

mitindo que estes pontos deixem de ser falhos. RBLP_AP unifica Revisão de Teoria através do sistema FORTE (RICHARDS, MOONEY, 1995), o qual induz uma revisão de programas em lógica baseando-se nos dados, com uma adaptação do algoritmo EM (KOLLER, PFEFFER, 1997) para aprender as CPDs. Já que BLP utiliza uma representação de programas lógicos, FORTE pode ser aplicado e integrado a este. Por este motivo e pelo BLP já ter sido comparado a outras abordagens na literatura é que optamos por utiliza-lo no nosso sistema. O segundo procedimento (RBLP_MP) tem por objetivo encontrar o BLP de máxima verossimilhança.

RBLP_AP primeiro revisa os parâmetros com o objetivo de refletir melhor os dados, ou seja, de maximizar a verossimilhança, permitindo assim que exemplos que estão sendo provados com probabilidade baixa, logo estão sendo classificados incorretamente, sejam corrigidos. Caso seja verificado que a estrutura do BLP precise ser revista, esta será revista e os parâmetros retreinados. Este processo é repetido até que um treinamento adicional não forneça nenhuma melhora na classificação. BANNER (RAMACHANDRAN, MOONEY, 1998) utiliza esta abordagem para revisão de redes Bayesianas proposicionais.

A utilização do algoritmo FORTE(RICHARDS, MOONEY, 1995) como base, deu-se pelo fato de ele buscar pontos de falha na teoria e depois escolher a melhor maneira de corrigí-los. Desta forma o espaço de busca é reduzido, tornando o algoritmo menos custoso do que outros, que consideram modificações em toda a estrutura, muitas vezes tornando inviável percorrer todo o espaço de busca, por este ser muito extenso. Nesta tese mostraremos que quando o BLP é aproximadamente correto, já que estamos trabalhando com classificação, o BLP retornado pelo RBLP_AP é suficiente. Em outras situações, propomos que o RBLP_MP seja executado, com o objetivo de encontrar o BLP de máxima pontuação, de acordo com a função de avaliação que está sendo utilizada.

A presente tese é apresentada da seguinte forma: o capítulo 1, definiu a motivação e introduziu o nosso trabalho. O capítulo 2 apresenta uma breve introdução de conceitos básicos que foram importantes para a nossa pesquisa, no capítulo 3 o RBLP é definido, primeiramente uma visão geral é fornecida e depois os procedimentos para revisão dos parâmetros e da estrutura do BLP são introduzidos e

nos capítulos seguintes apresentamos conclusões e trabalhos futuros. O apêndice descreve a implementação do RBLP_AP.

Capítulo 2

Conceitos Preliminares

Nossa pesquisa baseou-se em várias idéias e técnicas. Neste capítulo apresentamos algumas desses técnicas que serão importantes para o entendimento dos capítulos seguintes. Na seção 2.1 apresentamos alguns conceitos importantes de probabilidade (MITCHELL, 1997), o leitor já familiarizado, pode encaminhar-se direto para a seção seguinte 2.2, onde é fornecida uma visão geral de redes Bayesianas (RUSSELL, NORVIG, 1995). Depois em 2.3 apresentamos programas em lógica Bayesianos (BLP) (KERSTING, De RAEDT, 2000), e por último em 2.4 refinamento de teorias (WROBEL, 1996).

2.1 Probabilidade

Um problema com a lógica de primeira ordem, e conseqüentemente com as abordagens dos agentes lógicos, é que os agentes quase sempre não têm acesso a toda a verdade sobre os assuntos que eles estão discutindo. Algumas sentenças podem ser diretamente verificadas através da percepção do agente, outras podem ser inferidas de uma percepção anterior, ou atual junto com um conhecimento sobre as propriedades do ambiente. Na maioria dos casos, entretanto, mesmo nos mais simples, existem importantes questões para as quais os agentes não encontram uma resposta categórica, logo terão de agir com um grau de incerteza. Por exemplo, suponha que um determinado agente gostaria de conduzir uma pessoa que deseja viajar até o aeroporto. Mesmo que o agente se programe para sair de casa noventa

minutos antes do avião partir e levando-se em consideração que o aeroporto localiza-se a 30km da sua casa, ele não poderá garantir que chegarão ao aeroporto a tempo, pois um dos pneus do carro pode furar ou o próprio carro pode ter qualquer tipo de pane, o trânsito pode estar extremamente lento, a gasolina acabar, enfim, vários imprevistos podem ocorrer. Logo, em um caso como este, o que o agente poderia concluir, é que ele pode chegar ao aeroporto no horário previsto com uma chance de, por exemplo, 0.9 considerando que tudo sairá como desejado.

Uma outra questão é o fato de muitas regras sobre o domínio poderem estar incompletas, ou porque existem muitas situações a serem enumeradas explicitamente ou porque algumas das condições são desconhecidas. Com isso, não pode-se ter total certeza do que é inferido a partir delas. Nesses casos o que o agente pode fornecer é um grau de crença na situação em questão. A principal ferramenta para lidar com incertezas é a teoria de probabilidade, a qual associa a cada sentença um grau de crença numérico entre 0 e 1.

Um elemento básico pertencente à linguagem que compõe a teoria de probabilidade é a variável aleatória, que representa a parte da teoria onde o valor assumido não é conhecido. No exemplo anterior, a variável aleatória estaria representando a chegada do agente ao aeroporto a tempo do vôo ou não. Cada variável aleatória tem um domínio de valores possíveis que ela pode assumir. No exemplo acima o domínio seria composto pelos valores verdadeiro, para o caso do agente chegar ao aeroporto a tempo do vôo e falso, caso contrário. As variáveis aleatórias não assumem somente valores binários, elas também podem ser discretas, onde o domínio binário é um caso em particular, assumindo valores de um domínio finito, ou podem ser contínuas onde o seu domínio é formado por números reais.

Uma noção que será útil futuramente é a de eventos atômicos. Evento atômico (interpretação) é uma completa especificação de um estado do mundo sobre o qual o agente está em dúvida, ou seja é a associação de valores para todas as variáveis aleatórias do problema que está sendo abordado. Por exemplo, se considerássemos que o nosso mundo é composto apenas pelas variáveis Chegar_ao_Aeroporto e Tráfego, então a proposição Chegar_ao_Aeroporto = verdadeiro $\wedge$ Tráfego = bom é um evento atômico. Os eventos atômicos são exclusivos, no máximo um é o realmente procu-

rado.

A probabilidade a priori de uma proposição a é o grau de crença nesta na ausência de qualquer outra informação. Por exemplo, se a probabilidade a priori do tráfego estar bom é de 0.5, escreveríamos:

P(Tráfego = bom) = 0.5

Uma distribuição de probabilidade seria a definição de probabilidades para todos os possíveis valores de uma determinada variável. Por exemplo, a distribuição de probabilidade da variável Tráfego, **P**(Tráfego), pode ser definida como:

P(Tráfego= muito_bom) = 0.2

P(Tráfego = bom) = 0.5

P(Tráfego= ruim) = 0.3

ou simplesmente **P**(Tráfego) = $< 0.2, 0.5, 0.3 >$

Como notação, consideraremos:

P(X = x) ou P(x) para denotar a probabilidade da variável X assumir o valor x e **P**(X) para denotar a distribuição de probabilidade sobre X.

Para variáveis contínuas, não é possível escrever toda a distribuição como uma tabela, porque existem infinitos valores. Defini-se então, a probabilidade de uma variável aleatória continua assumir um determinado valor x, como sendo uma função parametrizada para x. Esta função é conhecida como função de densidade de probabilidade. Ela indica a probabilidade da variável aleatória pertencer a um determinado intervalo de números reais. Como exemplo temos a função de distribuição Normal, (DEGROOT, 1987), $N(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}(\frac{(x-\mu)^2}{\sigma})}$, onde μ é a média e σ a variância.

Algumas vezes estamos interessados na probabilidade de todas as combinações de valores de um conjunto de variáveis aleatórias. Considere a expressão **P**(Tráfego,Chegar_ao_Aeroporto). Ela significa que estamos interessados em todas as combinações de possíveis valores para as variáveis Trafego e Chegar_ao_Aeroporto. Como a variável Tráfego tem um domínio composto por 3 valores e a variável Chegar_ao_Aeroporto um domínio binário, especificamos 6 valores combinados e a representação desses valores pode ser feita através de uma tabela 3 por 2. Esta distribuição é conhecida como distribuição de probabilidade conjunta das variáveis

em questão. Quando consideramos todas as variáveis do problema em questão estamos olhando para a distribuição de probabilidade conjunta total, esta especifica a probabilidade de todos os eventos atômicos.

A partir do momento em que o agente obtiver alguma evidência sobre as variáveis aleatórias, a probabilidade a priori não pode mais ser aplicada, utiliza-se então a probabilidade condicional ou posterior. A notação utilizada é P(a|b) que significa a probabilidade de a dado que tudo que sabemos é b. Por exemplo, P(Chegar_ao_Aeroporto = verdadeiro| Tráfego = muito_bom) = 0.8 indica que se o tráfego estiver muito bom e nenhuma outra informação for fornecida então a probabilidade de uma pessoa chegar ao aeroporto a tempo do vôo é de 0.8.

Probabilidades condicionais podem ser definidas em termos de probabilidades a priori:

$$P(a|b) = \frac{P(a \wedge b)}{P(b)}$$

Escrevendo de outra maneira

$$P(a \wedge b) = P(a|b)P(b)$$

que é chamada de regra do produto.

Generalizando, suponha que tenhamos as seguintes variáveis aleatórias $\{X_1, ..., X_n\}$. Pela regra do produto

$$\begin{aligned} P(x_1, ..., x_n) &= P(x_n|x_{n-1}, ..., x_1)P(x_{n-1}, ..., x_1) \\ &= P(x_n|x_{n-1}, ..., x_1)P(x_{n-1}|x_{n-2}, ..., x_1)...P(x_2|x_1)P(x_1) \\ &= \prod_{i=1}^{n} P(x_i|x_{i-1}, ..., x_1) \end{aligned} \quad (1)$$

Esta identidade é valida para qualquer conjunto de variáveis aleatórias e é chamada de regra da cadeia.

Um teorema importante na teoria de probabilidades é o teorema de Bayes definido como:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

O teorema de Bayes é útil na prática, porque existem muitos casos onde temos boas estimativas de probabilidade para os três termos da direita e precisamos computar o da esquerda.

Para o caso mais geral com várias variáveis, o teorema de Bayes pode ser escrito utilizando a notação de distribuição de probabilidades.

$$\mathbf{P}(Y|X) = \frac{\mathbf{P}(X|Y)\mathbf{P}(Y)}{\mathbf{P}(X)}$$

Os axiomas 1, 2, 3a ou 3b definidos abaixo são suficientes como fundamentos da teoria de probabilidade.

1. Para qualquer proposição $\underline{a}$, $0 \leq P(a) \leq 1$;

2. Necessariamente proposições verdadeiras têm probabilidade 1, enquanto que proposições falsas têm probabilidade 0;

3. a) $P(a \vee b) = P(a) + P(b) - P(a \wedge b)$

   b) $P(a) = \sum_{e_i \in e(a)} P(e_i)$ onde e(a) são todos os eventos atômicos onde $\underline{a}$ acontece.

A partir dos axiomas 1,2 e 3a é possível mostrar que para qualquer variável aleatória X,

$$\sum_{i=1}^{n} P(X = x_i) = 1$$

onde o domínio de X é dom(X) = $< x_1, ..., x_n >$. Ou seja, qualquer distribuição de probabilidades de uma variável (discreta) deve somar 1.

2.2 Redes Bayesianas

Uma rede Bayesiana é uma maneira compacta de representar conhecimento probabilístico.

A idéia é representar um domínio em termos de variáveis aleatórias e explicitamente modelar a interdependência das variáveis aleatórias em termos de um gráfico. Dizemos então que uma rede Bayesiana é um grafo direcionado no qual as variáveis aleatórias são os nós da rede e os arcos da rede representam dependência direta entre estas variáveis aleatórias.

Associado a cada nó está uma tabela de probabilidades condicionais, a qual fornece a probabilidade de cada valor da variável dado cada combinação possível de

valores para cada um dos seus predecessores imediatos[1] (Pais), em outras palavras é a distribuição de probabilidade condicional (CPD) de uma determinada variável aleatória com os seus pais. Podemos dizer que, a CPD define como os pais interagem entre si produzindo uma influência conjunta sobre o seu nó filho.

Uma definição importante em rede Bayesiana é a de independência condicional. Dizemos que X é condicionalmente independente de Y dado Z se a distribuição de probabilidade governando X é independente do valor de Y dado o valor de Z ou seja se

$$(\forall x_i, y_j, z_k) P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

onde x_i , y_j , z_k pertencem ao domínio das variáveis X, Y, Z respectivamente. Geralmente a fórmula acima é escrita na forma abreviada P(X|Y,Z) = P(X|Z). A definição de condicionalmente independente pode ser estendida para conjuntos de variáveis da mesma maneira.

$$P(X_1...X_l | Y_1...Y_m, Z_1...Z_n) = P(X_1...X_l | Z_1...Z_n)$$

Por construção é assumido que esta propriedade acontece em redes Bayesianas, com as suas variáveis sendo condicionalmente independentes dos seus não-descendentes na rede, dado seus predecessores imediatos. Com isso a probabilidade conjunta para qualquer associação de valores para as variáveis da rede, isto é, $P(x_1, ..., x_n)$, pode ser calculada utilizando a seguinte fórmula:

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

onde $P(x_i | Pais(X_i))$ é obtida da CPD associada a variável X_i, o que reduz a computação necessária para a definição da probabilidade conjunta das variáveis aleatórias da rede.

Comparando a equação acima com a (1), verifica-se que para toda variável X_i, na rede

$$\mathbf{P}(X_i | X_{i-1}, ...X_1) = \mathbf{P}(X_i | Pais(X_i))$$

[1]Dizemos que Y é predecessor de X se existe um caminho direto de Y até X. E dizemos que Y é predecessor imediato de X se Y for predecessor de X e não existir nenhuma outra variável entre os dois.

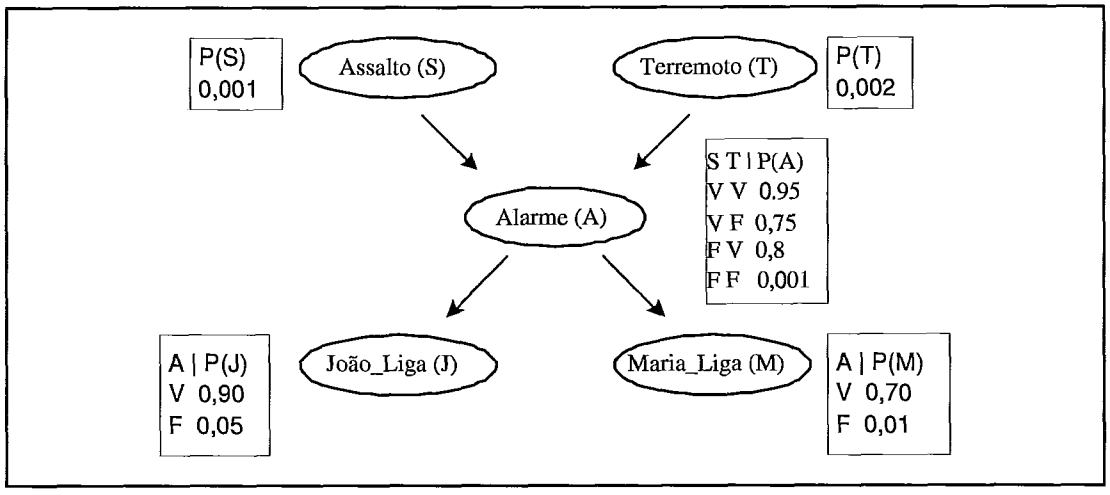

Figura 2.1: Rede Bayesiana para o sistema de alarme

onde $Pais(X_i) \subseteq \{X_{i-1}, ..., X_1\}$. Esta condição pode ser satisfeita se ordenarmos os nós de uma maneira consistente com a ordenação implícita da estrutura da rede Bayesiana.

A figura 2.1 mostra um exemplo de rede Bayesiana, (RUSSELL, NORVIG, 1995). A situação considerada é a de um sistema de alarme que pode ser ativado tanto por um assaltante como por um terremoto, este último em casos mais extremos. E dado que o alarme foi tocado espera-se que Maria e/ou João liguem para a polícia. Só que às vezes, estas duas pessoas confundem o barulho do alarme com outros como, por exemplo, o toque do telefone, com isto existe uma incerteza quanto as ligações.

A rede indica que a decisão de Maria ligar para polícia independe da de João. Já o disparo do alarme está diretamente ligada ao fato de ter ocorrido um assalto ou um terremoto.

Dadas as CPDs, a distribuição de probabilidade conjunta total da rede pode ser computada como mostrado abaixo:

$$\mathbf{P}(S, T, A, J, M) = \mathbf{P}(S)\mathbf{P}(T)\mathbf{P}(A|S, T)\mathbf{P}(J|A)\mathbf{P}(M|A)$$

Como exemplo vamos supor que gostaríamos de calcular a probabilidade do evento: o alarme tocou, mas nem um assalto nem um terremoto ocorreram, e ambos João e Maria ligaram. Considere que não existe ruído ou falha no equipamento. Utilizaremos letras minúsculas para indicar o valor que cada uma das variáveis aleatórias esta assumindo, logo:

$$P(\neg s \wedge \neg t \wedge a \wedge j \wedge m) = P(\neg s)P(\neg t)P(a|\neg s, \neg t)P(j|a)P(m|a)$$
$$= 0.999 * 0.998 * 0.001 * 0.90 * 0.70$$
$$= 0.00062$$

Relacionamentos com incerteza podem freqüentemente ser caracterizados por relações lógicas *noisy*. O exemplo padrão é a relação *noisy-or*, uma generalização do *ou* lógico. Voltando à rede Bayesiana definida na figura 2.1. Podemos dizer que o alarme pode ser disparado se e somente se ocorrer um terremoto ou um assalto. O modelo *noisy-or* permite associar uma incerteza sobre a habilidade que cada pai tem de influenciar o filho a ser verdadeiro- a relação causal entre os pais e o filho pode ser inibida. O modelo faz duas suposições. Primeiro, assume que todas as possíveis causas são listadas. Segundo, assume que a inibição de cada pai é independente da inibição de qualquer outro pai - por exemplo , qualquer que seja o motivo pelo qual um assalto não acionou o alarme independe do motivo pelo qual um terremoto também não acionou o alarme. A partir destas suposições, o alarme não disparará se e somente se todos os seus pais que são verdadeiros forem inibidos, e a probabilidade disto é o produtório das probabilidade de inibição de cada pai. Suponha as probabilidades inibidoras individuais mostradas abaixo.

$P(Alarme = F|Assalto = V, Terremoto = F) = 0.25$

$P(Alarme = F|Assalto = F, Terremoto = V) = 0.2$

A partir destas probabilidades a CPD do predicado alarme pode ser computada como mostrado na tabela 2.1.

Assalto	Terremoto	P(Alarme=V)	P(Alarme=F)
F	F	0	1
F	V	0.8	0.2
V	F	0.75	0.25
V	V	0.95	$0.2 * 0.25 = 0.05$

Tabela 2.1: CPD do predicado Alarme utilizando *noisy-or*.

Em geral, um relacionamento lógico *noisy* na qual a variável depende de k pais pode descrever sua distribuição de probabilidades condicionais utilizando O(k) parâmetros invés de $O(2^k)$. Isto torna o acesso e o aprendizado muito mais fácil.

2.2.1 Inferência

Considere a rede Bayesiana da figura 2.1. Suponha que tenha ocorrido um assalto, ou seja que o valor da variável assalto é verdadeiro, e que queremos saber a probabilidade de Maria ligar, ou seja $P(Maria\_Liga = V | Assalto = V)$. Como o valor das variáveis Terremoto e Alarme é desconhecido, inferência é necessária. A variável Maria_Liga é chamada de variável de consulta enquanto a variável Assalto é uma variável de evidência, as outras são chamadas de não-observadas (*hidden*).

Generalizando, uma rede Bayesiana pode ser utilizada para inferir o valor da variável de consulta dado valores observados para o conjunto de variáveis de evidência. Já que estamos lidando com variáveis aleatórias não seria muito correto associar à variável de consulta um determinado valor, o que realmente estamos querendo inferir é a distribuição de probabilidade para esta variável. De um modo geral, redes Bayesianas são utilizadas para inferir a distribuição de probabilidade de um conjunto de variáveis de consulta, dado um conjunto de variáveis evidências. Redes Bayesianas são flexíveis o suficiente para permitir que qualquer conjunto de nós possa servir tanto como um conjunto de variáveis de consulta como de evidência.

A estrutura da rede Bayesiana tem influência significativa na complexidade da inferência. Baseadas na sua estrutura, redes Bayesianas podem ser divididas em duas classes: *polytrees*, que têm uma estrutura simples onde cada par de variáveis está conectada por até um caminho, e redes com ciclos, que tem ciclos não-direcionados na estrutura. Como exemplo considere as figuras 2.2 e 2.3. A primeira representa um rede Bayesiana *polytree* enquanto que a segunda mostra um rede Bayesiana com ciclos (multi-conectadas).

Podemos dividir os métodos de inferência em dois grandes grupos: os métodos de inferência exata e os de inferência aproximada.

Inferência exata de probabilidades, em geral para uma rede Bayesiana arbitrária é NP-difícil (COOPER, 1990). Os de inferência aproximada, sacrificam precisão para ganhar em eficiência. Por exemplo, o método de Monte Carlo que fornece uma solução aproximada para aleatoriamente achar amostras das distribuições das variáveis não observadas (PRADHAM, DAGUM, 1996). Na teoria, até estes podem ser NP-difícil (DAGUM, LUBY, 1993). Felizmente, na prática métodos aproxima-

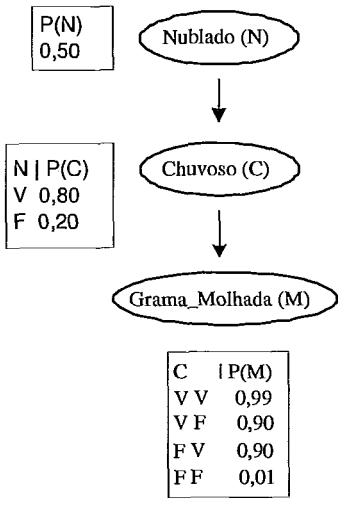

Figura 2.2: Rede *PolyTree*

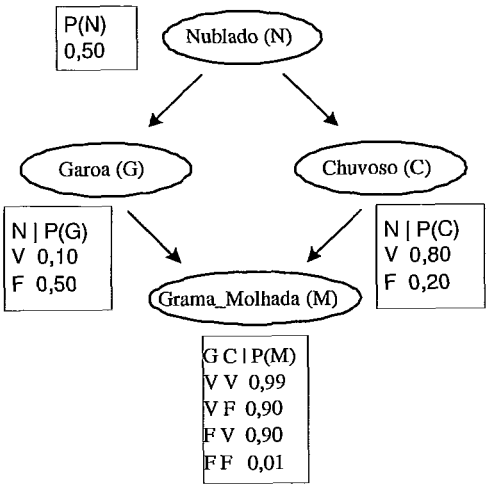

Figura 2.3: Rede com Ciclo

tivos mostram-se úteis em muitos casos.

2.2.1.1 **Inferência Exata**

Um processo simples de inferência exata é a inferência por enumeração, onde todas as variáveis do problema em questão, que não estão sendo observadas são somadas.

$$\mathbf{P}(Q|\mathbf{e}) = \frac{\mathbf{P}(Q,\mathbf{e})}{\mathbf{P}(\mathbf{e})} = \alpha\mathbf{P}(Q,\mathbf{e}) = \alpha\sum_{y}\mathbf{P}(Q,\mathbf{e},y)$$

onde Y representa as variáveis não observadas. Como P(**e**) é invariante para cada valor de Q considerado, podemos substitui-lo por uma constante, α, que chamaremos de constante de normalização, ela assegura que a soma das probabilidades para cada um dos valores de Q dará 1.

Supondo a rede Bayesiana da Figura 2.1, vamos considerar que estamos em busca da **P**(Assalto | Joao_Liga = verdadeiro , Maria_Liga = verdadeiro). Neste caso as variáveis não observadas são Terremoto e Alarme, temos então:

$$\mathbf{P}(S|j,m) = \alpha\mathbf{P}(S,j,m) = \alpha\sum_{t}\sum_{a}\mathbf{P}(S,t,a,j,m)$$

Supondo Assalto = verdadeiro a expressão fica:

$$P(s|j,m) = \alpha P(s,j,m) = \alpha\sum_{t}\sum_{a}P(s)P(t)P(a|s,t)P(j|a)P(m|a)$$

Para computar esta expressão, temos que somar quatro termos, cada um computando o produto de cinco números. No pior caso, onde temos que somar quase

todas as variáveis da rede, a complexidade do algoritmo para uma rede com n variáveis binárias é de $O(n2^n)$. Fazendo uma reorganização da expressão acima, percebendo quais das variáveis não são influenciadas pelo somatório, como mostra a fórmula a baixo, podemos reduzir a complexidade para $O(2^n)$.

$$P(s|j,m) = \alpha P(s,j,m) = \alpha P(s) \sum_t P(t) \sum_a P(a|s,t) P(j|a) P(m|a)$$

No exemplo acima o produto entre $P(j|a)$ e $P(m|a)$ é feito duas vezes, uma para t e outra para $\neg t$, o que quando trata-se de um domínio com muitas variáveis pode tornar-se computacionalmente inviável. Um outro método de inferência exata é a eliminação de variável, que se mostra mais eficiente do que o por enumeração, por exatamente evitar estas computações repetidas.

A idéia é fazer a computação apenas uma vez e guardar o resultado para usos futuros. O algoritmo analisa a equação utilizada no algoritmo de enumeração da direita para esquerda, os resultados intermediários são armazenados e os somatórios sobre variáveis são feitos somente para as partes da expressão que dependem desta variável.

Voltando ao exemplo do assalto.

$$P(s|j,m) = \alpha P(s,j,m) = \alpha P(s) \sum_t P(t) \sum_a P(a|s,t) P(j|a) P(m|a)$$

para cada parte da expressão associamos um nome, que na maioria dos casos é a inicial do nome da variável em questão, e são denominados fatores.

O Primeiro fator então, seria o fator M que estaria representando a P(m|a). Os passos então são os seguintes.

- O fator para M não necessita de um somatório sobre este, pois seu valor é fixo. Simplesmente guardamos a probabilidade dada para cada valor de $\underline{a}$ em um vetor de dois elementos, que chamaremos de $\mathbf{f}_M(A)$:

$$\mathbf{f}_M(A) = (P(m|a), P(m|\neg a))$$

- Similarmente o fator para J, $P(j|a)$, é guardado como um vetor de dois elementos, $\mathbf{f}_J(A) = (P(j|a), P(j|\neg a))$

- O fator para A é P(a|s,t), que será uma matriz 2 X 2 X 2, $\mathbf{f}_A(A, S, T)$

- Agora soma-se para A o produto desses três fatores.

$$\mathbf{f}_{AJM}(S,T) = \sum_a \mathbf{f}_A(A,S,T) * \mathbf{f}_J(A) * \mathbf{f}_M(A)$$
$$= \mathbf{f}_A(a,S,T) * \mathbf{f}_J(a) * \mathbf{f}_M(a) + \mathbf{f}_A(\neg a,S,T) * \mathbf{f}_J(\neg a) * \mathbf{f}_M(\neg a)$$

- Agora processa-se T da mesma maneira: somatório de T para o produto de $\mathbf{f}_T(T)$ e $\mathbf{f}_{AJM}(S,T)$.

$$\mathbf{f}_{TAJM}(S) = \mathbf{f}_T(t) * \mathbf{f}_{AJM}(S,t) + \mathbf{f}_T(\neg t) * \mathbf{f}_{AJM}(S,\neg t)$$

- Podemos computar o resultado simplesmente multiplicando o fator para S, isto é $\mathbf{f}_S(S) = \mathbf{P}(S)$ por $\mathbf{f}_{TAJM}(S)$.

$$\mathbf{P}(S|j,m) = \alpha \mathbf{f}_S(S) * \mathbf{f}_{TAJM}(S)$$

Note que o problema anteriormente mencionado da computação duplicada do produtório entre $P(j|a)$ e $P(m|a)$, foi resolvido, pois o fator $\mathbf{f}_{AJM}(S,T)$ armazenou este cálculo e quando deseja-se saber o valor para t e $\neg t$ apenas busca-se o valor correspondente.

O algoritmo apresentado acima é simples e eficiente para responder consultas individuais. Se quisermos computar a probabilidade posterior para todas as variáveis na rede, entretanto, pode ser menos eficiente. Por exemplo, em uma rede *polytree* seriam necessárias O(n) consultas com um custo de O(n) para cada uma, com um total de tempo $O(n^2)$.

Algoritmos de agrupamento(também conhecidos como algoritmos *join tree* (*junction tree*)(JENSEN, 1996), (COWELL, DAWID, et al., 1999)), podem reduzir o custo para O(n) em algumas situações fazendo com que sejam muitos utilizados em ferramentas comerciais de redes Bayesianas.

A idéia básica do agrupamento é agrupar nós individuais da rede para formar um agrupamento de nós de tal maneira que a rede resultante seja uma *polytree.* Por exemplo, a rede mostrada na figura 2.4 pode ser convertida em uma *polytree* através da combinação dos nós Garoa e Chuvoso em um nó agrupado chamado Garoa+Chuvoso, como mostrado na figura 2.5. Os dois nós binários são substituídos

por um mega-nó que assume os quatro possíveis valores: VV, VF, FV e FF. O mega-nó tem apenas um pai, a variável binária Nublado, então existem dois casos condicionais.

A partir do momento que a rede tem a estrutura de uma *polytree*, um algoritmo de inferência especial é aplicado. Essencialmente, o algoritmo é uma forma de propagação restrita onde as restrições garantem que os grupos vizinhos concordem na probabilidade posterior de qualquer uma das variáveis que eles tem em comum. Pode-se dizer com uma certa cautela, que este algoritmo é capaz de computar probabilidades posterior para todos os nós não evidência da rede em um tempo O(n), onde n agora é o tamanho da rede modificada. Entretanto, o problema de NP-difícil não desapareceu; se a rede precisar de tempo exponencial para a inferência com o algoritmo de eliminação de variável, então será preciso tempo exponencial para construir as CPDs da rede agrupada.

2.2.1.2 **Inferência Aproximada**

Dada a dificuldade dos métodos de inferência exata, no geral para redes com ciclos, é essencial a consideração de métodos de inferência aproximada. Definiremos nesta seção algoritmos de amostragem aleatória, também chamados de algoritmos de Monte Carlo, que fornecem respostas aproximadas e que têm sido amplamente utilizados em Ciências da Computação para estimar quantidades que são difíceis de calcular exatamente. O algoritmo *simulated annealing* é um exemplo de amostragem aleatória aplicado para problemas de otimização. Aqui estamos interessados na amostragem aplicada para a computação das probabilidades posterior. Descreveremos a seguir o algoritmo de amostragem de Cadeia de Markov Monte Carlo(MCMC).

O algoritmo MCMC, ao contrário de outros algoritmos de amostragem que geram eventos sem considerar um conhecimento prévio, gera fazendo uma mudança aleatória no evento predecessor. É, em conseqüência, proveitoso pensar na rede estando em um estado corrente particular especificando um valor para todas as variáveis. O próximo estado é gerado através de amostragem aleatória de um valor para uma das variáveis não evidências(variáveis não observadas e variáveis

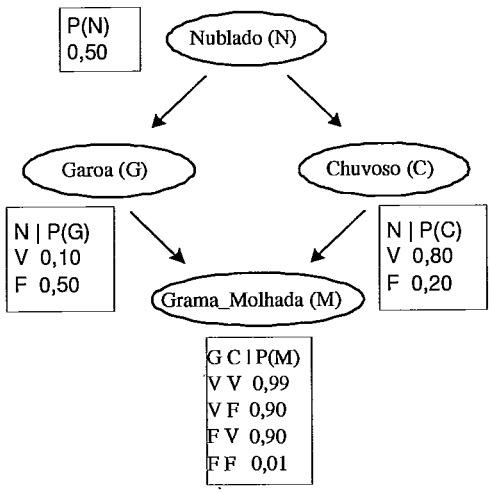

Figura 2.4: Rede com Ciclo

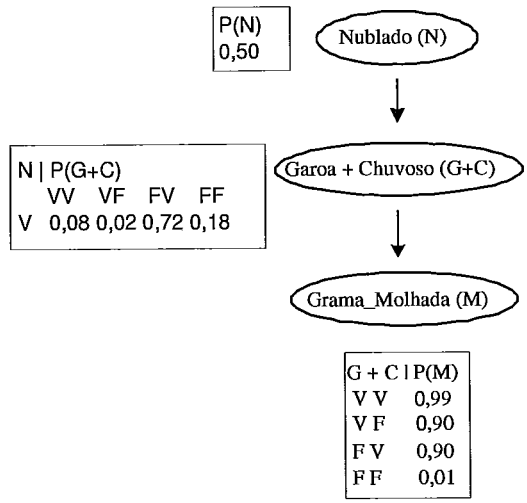

Figura 2.5: Rede *PolyTree*

de consulta) $Y_i \cup Q_i$, condicionada aos valores correntes das variáveis pertencentes a sua cobertura de Markov. A cobertura de Markov de uma determinada variável é composta pelos seus pais, filhos e os pais dos seus filhos. MCMC, então percorre aleatóriamente o domínio de possíveis estados,eventos atômicos - indo de uma variável para outra, mas mantendo as variáveis evidências fixas.

Como exemplo, considere a consulta

$$\mathbf{P}(Chuvoso|Garoa = verdadeiro, Grama\_Molhada = verdadeiro)$$

aplicado a rede da figura 2.4. As variáveis evidência Garoa e Grama_Molhada são mantidas fixas nos seus valores observados e as variáveis não observadas Nublado e Chuvoso são inicializadas aleatóriamente- digamos verdadeiro e falso respectivamente. Então, o estado inicial seria [N=verdadeiro, G=verdadeiro, C=falso, M=verdadeiro]. Agora os seguintes procedimentos são executados repetidamente:

1. Nublado é amostrado dado os valores correntes para as variáveis pertencentes a cobertura de Markov- neste caso, nós amostramos $\mathbf{P}$(Nublado | Garoa = verdadeiro, Chuvoso = verdadeiro). Suponha que isto nos forneça Nublado = falso. Então o novo estado corrente é [falso, verdadeiro, falso, verdadeiro].

2. Chuvoso é amostrado dado os valores correntes para as variáveis pertencentes a cobertura de Markov: neste caso nós amostramos $\mathbf{P}$(Chuvoso| Nublado =

falso, Garoa = verdadeiro, Grama_Molhada = verdadeiro). Suponha que encontremos Chuvoso = verdadeiro. O novo estado corrente é [falso, verdadeiro, verdadeiro, verdadeiro].

Cada estado visitado durante este processo é uma amostra que contribui para a estimação da variável Chuvoso. Se o processo obtiver 20 visitas ao estado Chuvoso = verdadeiro e 60 para o caso igual a falso, então a resposta para a consulta é $< 0.25, 0.75 >$.

Maiores detalhes sobre métodos de inferência para redes Bayesianas são fornecidos em (RUSSELL, NORVIG, 1995), (RUSSELL, NORVIG, 2000) e (JENSEN, 1996).

2.2.2 Aprendizado dos Parâmetros Probabilísticos

Quando estamos lidando com um domínio sendo representado por uma rede Bayesiana, onde esta é fixa, e o conjunto de exemplos de treinamento é completo, ou seja o valor de todas as variáveis aleatórias do domínio são especificados em todos os exemplos, nós podemos simplesmente utilizar o método da contagem para determinar a distribuição de probabilidade destas variáveis. Suponha o interesse na probabilidade P(X=x | Pa=pa). Para determinar esta probabilidade, basta contar o número de exemplos onde as variáveis X e Pa assumem os valores x e pa respectivamente e dividir pelo número de exemplos onde a variável Pa assume o valor pa.

Agora, se a base de dados não for completa, esta contagem não pode mais ser efetuada. Existem algoritmos para lidar com esta questão, como o gradiente, (MITCHELL, 1997), (BINDER, KOLLER, et al., 1997) e o *Expectation Maximization* (EM)(DEMPSTER, LAIRD, et al., 1977),(MCLACHLAN, KRISHNAN, 1997), (LAURITZEN, 1995). Nesta tese investigamos apenas o segundo e um detalhamento de como ele funciona é apresentada na próxima seção.

Expectation Maximization **(EM)**

O EM é um método proposto para o aprendizado das CPDs quando o conjunto de exemplos de treinamento é incompleto. Ele começa com CPDs iniciais e calcula a

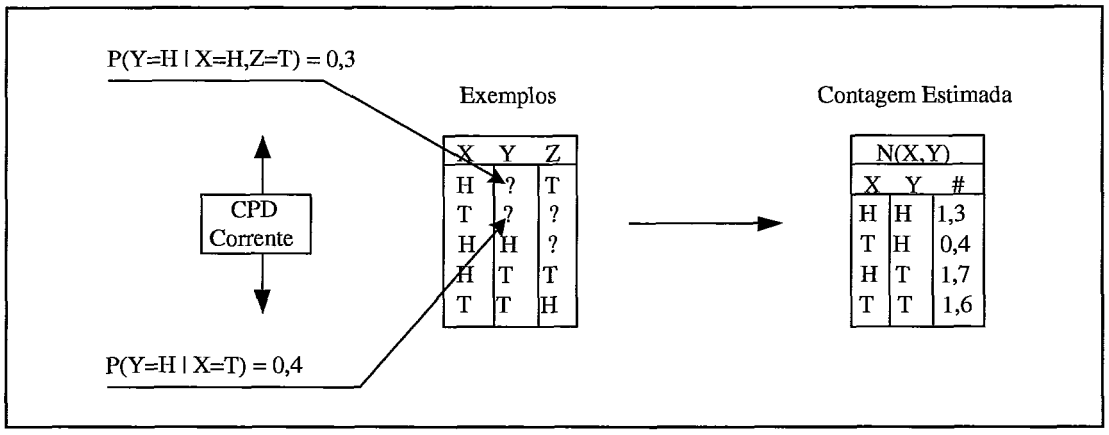

Figura 2.6: Contagem Esperada (passo-E do algoritmo EM)

verossimilhança dos exemplos utilizando estas CPDs. A cada iteração tenta encontrar novas CPDs (θ^*) que reflitam melhor o conjunto de exemplos de treinamento, ou seja tem por objetivo maximizar a verossimilhança.

$$\theta^* = \max_{\theta \in \Theta} L(\theta : \mathbf{C}) = \max_{\theta \in \Theta} \mathbf{P}(\mathbf{C}|\theta, H)$$

Intuitivamente, ele fornece uma maneira de completar a contagem usando as CPDs correntes.

O primeiro passo que é o da estimação (*expectation*) (passo-E), tem por objetivo calcular as contagens esperadas.

Suponha uma situação com três variáveis {X, Y, Z}, onde o domínio delas seja discreto, dom(X)=dom(Y)=dom(Z) = {H,T}. O primeiro passo do algoritmo EM esta representado na figura 2.6.

Como pode ser observado, quando o valor de uma variável não é determinado em um exemplo, este exemplo contribuirá para a contagem do valor esperado, com a probabilidade de que a variável em questão assuma o valor procurado, considerando as CPDs correntes. N(X=H,Y=H) = 0.3 (probabilidade inferida para Y=H, dada a CPD corrente, usando o primeiro exemplo e assumindo X e Z como evidências) + 1 (X,Y assumem o valor H no terceiro exemplo), logo N(X=H, Y=H) = 1.3, onde N é a contagem esperada.

Depois que todas as contagens esperadas relevantes para o problema em consideração tiverem sido determinadas, o passo de maximização, *maximization* (passo-M) será efetuado. Este passo vai utilizar estas contagens, para encontrar os novos

parâmetros e calcular a nova verossimilhança (L(θ | C)). Por exemplo, para encontrar a probabilidade P(Y=H | X=H), o passo-M utilizaria a seguinte equação:

$$P(Y = H|X = H) = \frac{N(X = H, Y = H)}{N(X = H)}$$

Fazendo o mesmo para cada um dos valores que Y e X assumem, a nova CDP, **P**(Y | X) seria definida.

A figura 2.7 ilustra melhor o que acabamos de discutir.

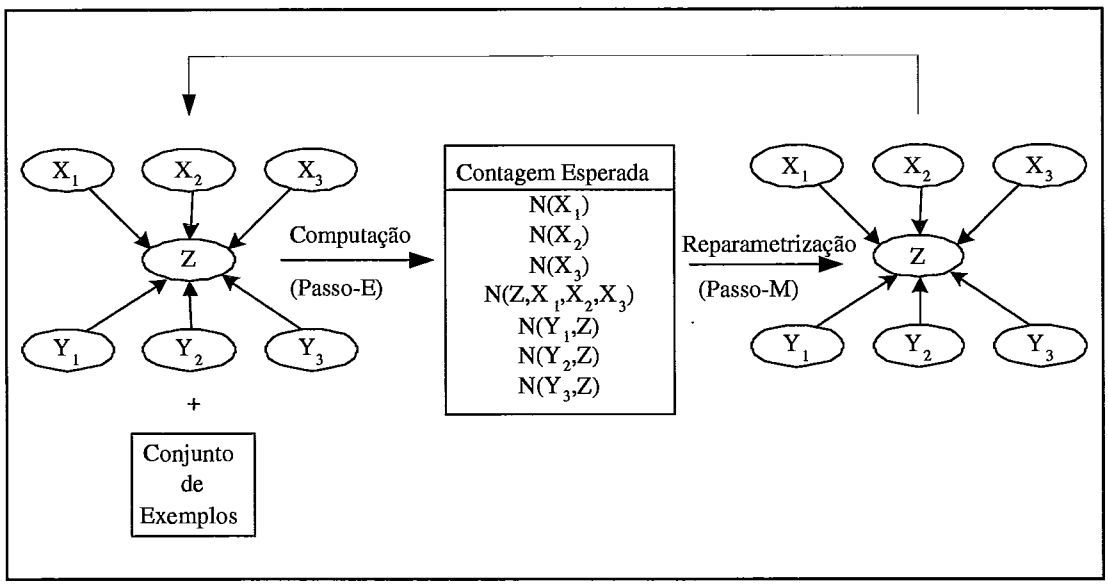

Figura 2.7: Esquema do algoritmo EM

O EM garante que a verossimilhança da iteração anterior é necessariamente menor do que a do passo corrente. Em outras palavras:

$$L(\theta_1|\mathbf{C}) \geq L(\theta_0|\mathbf{C})$$

onde θ_1 são as CPDs corrente e θ_0 as CPDs da iteração anterior.

Quando $L(\theta_1|\mathbf{C}) - L(\theta_0|\mathbf{C}) \leq \xi$, onde ξ é um erro considerado, podemos parar a computação.

2.3 BLP

Como vimos na seção anterior uma maneira de representar conhecimento é através de Redes Bayesianas. O conhecimento, então de John ser tio de Jane pode ser representado pela rede Bayesiana da figura 2.8

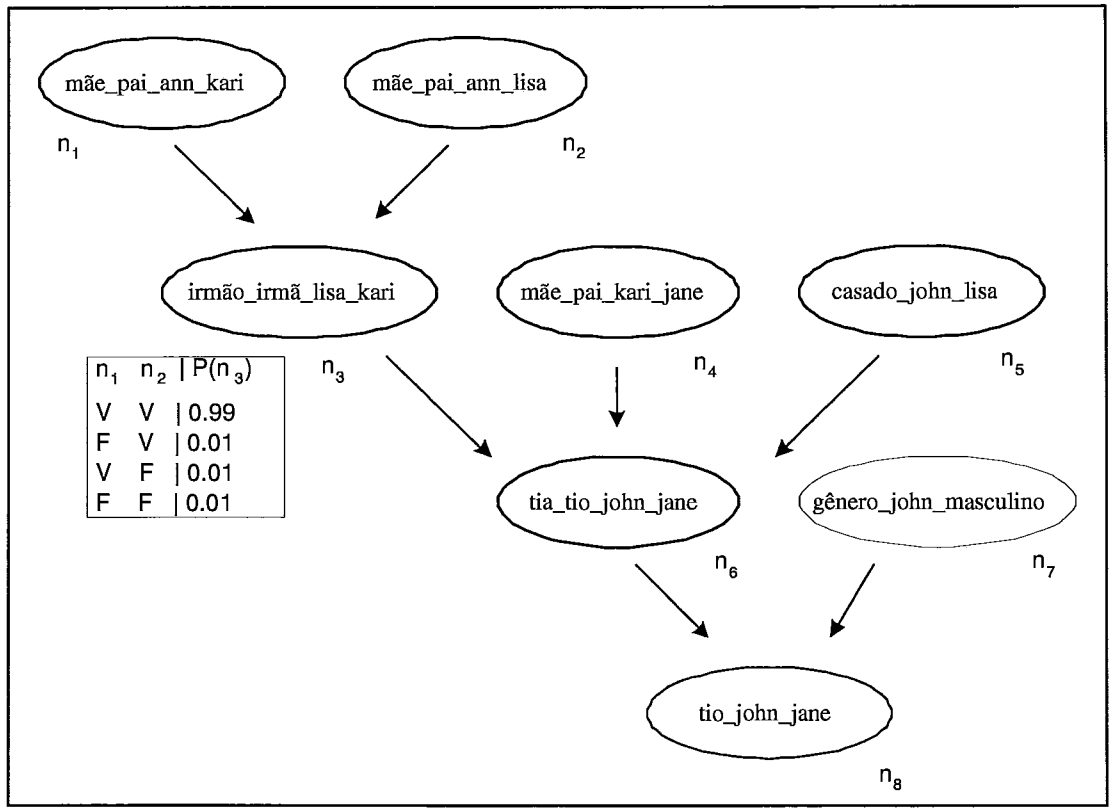

Figura 2.8: Representação do conhecimento através de Rede Bayesiana

Uma outra maneira de representar este mesmo conhecimento seria através de um conjunto de cláusulas proposicionais.

tio_john_jane ← gênero_john_masculino, tia_tio_john_jane.

tia_tio_john_jane ← casado_john_lisa, irmão_irmã_lisa_kari, mãe_pai_kari_jane.

irmão_irmã_lisa_kari ← mãe_pai_ann_kari, mãe_pai_ann_lisa.

Para capturar melhora a idéia de condicionalmente independente vamos utilizar | invés de ←, logo a teoria proposicional definida acima passa a ser representada como segue abaixo.

tio_john_jane | gênero_john_masculino, tia_tio_john_jane.

tia_tio_john_jane | casado_john_lisa, irmão_irmã_lisa_kari, mãe_pai_kari_jane.

irmão_irmã_lisa_kari | mãe_pai_ann_kari, mãe_pai_ann_lisa.

Esta teoria proposicional pode ser também representada através de uma teoria de primeira ordem básica, como é mostrado abaixo.

tio(john,jane) | gênero(john,masculino), tia_tio(john,jane).

tia_tio(john,jane) | casado(john,lisa), irmão_irmã(lisa,kari), mãe_pai(kari,jane).

irmão_irmã(lisa,kari) | mãe_pai(ann,kari), mãe_pai(ann,lisa).

Generalizando encontramos a teoria de primeira ordem definida abaixo.

tio(X,Y) | gênero(X,masculino), tia_tio(X,Y).

tia_tio(X,Y) | irmão_irmã(X,Z), mãe_pai(Z,Y).

irmão_irmã(X,Y) | mãe_pai(Z,X), mãe_pai(Z,Y).

Assim como em Redes Bayesianas cada uma das cláusulas tem uma CPD associada, definindo então uma teoria de primeira ordem Bayesiana, por exemplo a CPD da última cláusula é definida como mostra a tabela 2.2.

mãe_pai(Z,X)	mãe_pai(Z,Y)	irmão_irmã(X,Y)
V	V	0,99
F	V	0,01
V	F	0,01
F	F	0,01

Tabela 2.2: CPD da cláusula: irmão_irmã(X, Y) | mãe_pai(Z, X), mãe_pai(Z, Y) .

Note que a CPD mostrada na tabela 2.2 é a mesma definida para o nó irmão_irmã_lisa_kari da rede Bayesiana da figura 2.8.

Tendo então, a teoria de primeira ordem definida anteriormente e sabendo que John é do gênero masculino (gênero(john,masculino)) e casado com Lisa (casado(john,lisa)), onde esta é irmã da mãe de Jane (mãe_pai(ann,kari), mãe_pai(ann,lisa), mãe_pai(kari,jane)) queremos saber se John é tio de Jane (tio(john,jane)). É possível provar que John é tio de Jane, ou seja, é possível encontrar uma explicação para tio(john,jane). Uma maneira de representar esta explicação é através de uma Rede Bayesiana de primeira ordem básica, como mostra a figura 2.9.

Com a explicação para tio(john,jane) sendo representada através de uma rede Bayesiana é possível determinar com que grau de crença isto ocorre. O Procedimento de Resposta a uma Consulta do BLP, que veremos em detalhes na próxima seção, formaliza este raciocínio.

Podemos, agora formalizar o que é um programa em lógica Bayesiano (BLP) (KERSTING, De RAEDT, et al., 2000), (KERSTING, De RAEDT, 2000), (KERSTING, De RAEDT, 2001b). O BLP consiste de duas componentes: a primeira é a lógica, (LLOYD, 1989), um conjunto de cláusulas definidas Bayesianas, as quais

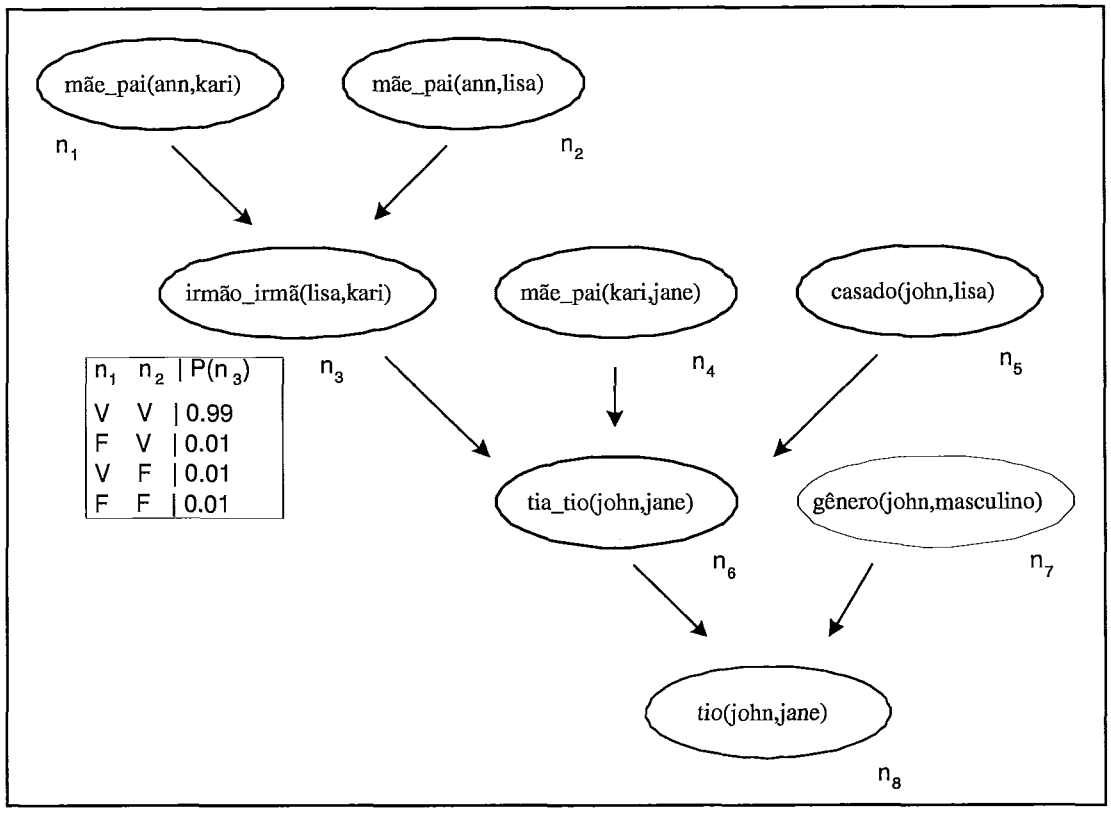

Figura 2.9: Representação através de Rede Bayesiana para a explicação para tio(john,jane)

capturam a estrutura qualitativa do domínio e baseia-se em Prolog. A segunda componente é a quantitativa, como em redes Bayesianas vai fornecer a noção de distribuição de probabilidades condicional (CPD) e regras de combinação (será definido adiante).

A teoria de primeira ordem Bayesiana que definimos acima para o problema do Tio é um programa lógico Bayesiano, que chamaremos de BLP do tio.

A cláusula irmão_irmã(X, Y) | mãe_pai(Z, X), mãe_pai(Z, Y) do BLP do Tio, é chamada de cláusula definida Bayesiana e irmão_irmã(X,Y), mãe_pai(Z,X) e mãe_pai(Z,Y) são os átomos Bayesianos. Um átomo Bayesiano é um átomo que tem um domínio associado a ele, que pode ser discreto ou contínuo. Um átomo lógico é um caso particular de átomo Bayesiano, pois seu domínio é binário.

Como vimos, podemos fazer uma determinada consulta a um BLP, encontrando uma explicação para esta consulta caso ela seja provada. Por exemplo, quando consultamos o BLP do tio para sabermos se tio(john,jane) obtemos o con-

junto de cláusulas básicas especificadas abaixo, que como vimos também podem ser representadas através de uma rede Bayesiana.

tio(john,jane) | gênero(john,masculino), tia_tio(john,jane).

tia_tio(john,jane) | casado(john,lisa), irmão_irmã(lisa,kari), mãe_pai(kari,jane).

irmão_irmã(lisa,kari) | mãe_pai(ann,kari), mãe_pai(ann,lisa).

Os átomos Bayesianos básicos, irmão_irmã(helen,kari), tia_tio(bob,jane),..., representam as variáveis aleatórias, que no exemplo considerado têm domínio binário.

Generalizando, uma cláusula Bayesiana c é uma expressão da forma:

$$A|A_1, ..., A_n$$

Onde $n \geq 0, A, A_1, ..., A_n$ são átomos Bayesianos e todas as variáveis são implicitamente universalmente quantificados, como no Prolog. Nós definimos cabeça(c) := A e corpo(c) := $\{A_1, ..., A_n\}$. Dizemos que a variável da cabeça está sendo diretamente influenciada pelas variáveis do corpo. As diferenças entre uma cláusula Bayesiana e uma lógica são: (1) os átomos $p(t_1, ..., t_n)$ que surgirem são Bayesianos, o que significa que eles têm um domínio dom(p) associado a eles que pode ser discreto ou contínuo e (2) utiliza-se "|"invés de "←", puramente uma convenção sintática.

Suponha que o BLP do tio definido acima, também tivesse a seguinte cláusula definida Bayesiana.

$tia\_tio(X, Y)|irmão\_irmã(X, Z), mãe\_pai(Z, Y)$

Logo quando tentássemos provar tio(bob,jane), para o conjunto de evidências especificado acima obteríamos também a seguinte cláusula básica:

$tia\_tio(bob, jane)|irmão\_irmã(helen, kari), mãe\_pai(kari, jane)$

Teríamos então, duas cláusulas básicas, cada uma especificando uma distribuição de probabilidade

P(tia_tio(bob,jane) | casado(bob,helen), irmão_irmã(helen,kari),
mãe_pai(kari,jane))

P(tia_tio(bob,jane) | irmão_irmã(helen,kari), mãe_pai(kari,jane))

respectivamente, mas o que precisa ser especificado é a distribuição de probabilidade combinada. A solução padrão para obter esta distribuição é a utilização das regras de combinação, *combining rules*. (KERSTING, De RAEDT, 2001b) utilizou a idéia de

(NGO, HADDAWY, 1997) e por motivos computacionais considerou que o conjunto de antecedentes era finito.

Na teoria, uma regra de combinação é qualquer algoritmo que mapeia um conjunto finito de CPDs $\{\mathbf{P}(A|A_{i_1},...,A_{i_n i})|1 \leq i \leq m, n_i \geq 0\}$, em uma única CPD chamada CPD combinada, $\mathbf{P}(A|B_1,...,B_n)$ com $\{B_1,...,B_n\} \subseteq \bigcup\{A_{i_1},...,A_{i_n i}\}$.

Como exemplo, a regra de combinação *max* é definida como:

$$\mathbf{P}(A|\cup_{i=1}^{n} A_{i1},...,A_{in_i}) = \max_{i=1}^{n}\{\mathbf{P}(A|A_{i1},...,A_{in_i})\}$$

Outra regra de combinação, para predicados Bayesianos que tenham domínio binário, é *noisy-or*. Como vimos na seção anterior, as regras *noisy-or* generalizam o *ou* lógico com algumas considerações. Formalizando, tendo as distribuições de probabilidade $\{P(A|A_i)|1 \leq i \leq m\}$ com variáveis aleatórias binárias $A, A_1, ..., A_m$ é mapeado em $P(A|A_1,...,A_m)$ com
$P(A = falso|A_1 = a_1,...,A_m = a_m) = \prod_{i=1}^{m} r_i$, onde

$$r_i = \begin{cases} P(A = falso|A_i = a_i), a_i = verdadeiro \\ 1, a_i = falso \end{cases}$$

$P(A = verdadeiro|A_1 = a_1,...,A_m = a_m) = 1 - P(A = falso|A_1 = a_1,...,A_m = a_m)$

BLP restringe as cláusulas a serem *range-restricted*, ou seja todas as variáveis que ocorrem na cabeça também ocorrem no corpo.

Vamos considerar que HB(B) é a Base de Herbrand (LLOYD, 1989),(CASANOVA, GIORNO, et al., 1987) do BLP B.

Considerando a consulta P(tio(bob,jane) | casado(bob,helen)=V, gênero(bob,masculino)=V, mãe_pai(ann,kari)=V, mãe_pai(ann,helen) =V, mãe_pai(kari,jane)=V) feita ao BLP do Tio, em termos lógicos podemos dizer que os átomos

{irmão_irmã(helen,kari), mãe_pai(ann,kari), mãe_pai(ann,helen), tia_tio(bob,jane), casado(bob,helen), irmão_irmã(helen,kari), mãe_pai(kari,jane), tio(bob,jane), tia_tio(bob,jane), gênero(bob,masculino) }
formam o modelo mínimo de Herbrand (LH(B)), ou seja, são todos os átomos básicos que foram derivados a partir do BLP em questão.

Como vimos anteriormente, considerando a relação de influência entre as variáveis, podemos construir um grafo de dependência que acrescido das distribuições de probabilidade condicionais combinadas pode ser considerado como uma rede Bayesiana.

Para que um programa lógico Bayesiano represente uma rede Bayesiana é preciso garantir que o grafo de dependências seja acíclico.

Definição 2.3.1 *Seja B um BLP. Se*

1. *LH(B)* $\neq \{\}$
2. *o grafo de dependência é acíclico, e*
3. *cada variável aleatória em LH(B) é somente influenciada por um conjunto finito de variáveis aleatórias*

então B é chamado de bem definido.

Utilizando-se, então, um BLP bem definido, pode-se dizer intuitivamente que cada programa lógico Bayesiano especifica uma rede Bayesiana.

2.3.1 Procedimento de Resposta a uma Consulta

Esta seção mostrará como responder a uma consulta probabilística a partir de um BLP, e em conseqüência como construir a rede Bayesiana correspondente a esta consulta.

Definição 2.3.2 *Uma consulta a um programa em lógica Bayesiano B é uma expressão da forma* $?\text{-}Q_1, ..., Q_n | E_1 = e_1, ..., E_m = e_m$ *com* $n > 0$, $m \geq 0$. *Pergunta-se pela distribuição de probabilidade condicional* $\mathbf{P}(Q_1, ..., Q_n | E_1 = e_1, ..., E_m = e_m)$ *das variáveis de consulta* $Q_1, ..., Q_n$, *onde* $\{Q_1, ..., Q_n, E_1 = e_1, ..., E_m = e_m\} \subset HB(B)$. *Uma consulta com* $m = 0$ *é chamada de sem evidências.*

Dizemos que uma resposta é definida se e somente se $\{Q_1, ..., Q_n, E_1 = e_1, ..., E_m = e_m\} \subset LH(B)$, onde $LH(B)$ é o modelo mínimo de Herbrand.

Definição 2.3.3 *Seja B um BLP, N sua rede Bayesiana e* $X_1, ..., X_m, m > 0$, *nós de N. A rede suporte* $N(X_1, ..., X_m)$ *é um grafo formado pela união de todas as redes suportes simples* $N(X_i)$.

Para construir uma consulta foi feita uma adaptação nos dois passos estratégicos da abordagem do KBMC(*Knowledge-based model construction*)(BREESE, GOLDMAN, et al., 1997) (HADDAWY, 1999): primeiro constrói-se uma rede Bayesiana N e, depois aplica-se um algoritmo de inferência em N de maneira a responder a pergunta.

Para encontrar a rede suporte (KERSTING, De RAEDT, 2001b) adaptou um algoritmo apresentado em (RUSSELL, NORVIG, 1995).

Entrada: B, um BLP bem definido; Q, uma consulta.
Saída: N, a rede suporte de ?-Q.

N ← Rede_Bayesiana_Vazia;
T ← Arvore_SLD_Vazia;
Arvore_SLD(B,Q,T);
Computa_Rede_Suporte(T,N);
N ← Poda(N);

Figura 2.10: Algoritmo para encontrar a rede suporte de um BLP

No algoritmo apresentado na figura 2.10 primeiro inicializa-se a rede suporte N e a árvore SLD T com vazio, para então poder computar as duas utilizando os procedimentos Arvore-SLD e Computa_Rede_Suporte, que serão apresentados a seguir.

O procedimento Arvore-SLD vai primeiro verificar se algum fato Bayesiano unifica com a consulta Q. Em caso positivo, são inseridos na árvore SLD T os arcos correspondentes. Depois busca todas as cláusulas Bayesianas cuja a cabeça unifica com a consulta Q e tenta provar o corpo dessas cláusulas. Árvore-SLD processa o corpo da cláusula selecionada átomo por átomo, construindo toda a árvore SLD T.

Computa_Rede_Suporte inspeciona todos os caminhos de sucesso P em T e constrói a rede suporte correspondente N. Funciona inserindo para uma cláusula

básica $C(e)\theta$ (adquirida das informações associadas a um arco e em P) todos os nós correspondentes e arcos na rede suporte N. Depois de construir estas versões de rede suporte N sem combinação, aplica-se as regras de combinação correspondentes a cada nó. Isto pode ocasionar a exclusão de alguns arcos.

Uma implementação desses procedimentos foi apresentada em (KERSTING, De RAEDT, 2001b), e pode ser encontrada no apêndice B. É construído sobre um meta interpretador Prolog. Existem vários tipos de meta interpretador que utilizam árvores SLD, (STERLING, SHAPIRO, 1986), (BRATKO, 1986). Uma adaptação simples da busca em profundidade foi utilizada.

Resumindo o que o procedimento de resposta a uma consulta faz é exatamente determinar a rede suporte correspondente a uma determinada consulta. Os nós dessa rede serão as variáveis aleatórias (átomos básicos) relevantes para a consulta em questão. Por exemplo, suponha que no BLP especificado acima,tivéssemos ainda a cláusula esposa(X,Y) | gênero(X,feminino), casada(X,Y). Esta cláusula não é utilizada na prova da consulta,**P**(tio(bob,jane) | casado(bob,helen)=V, gênero(bob,masculino)=V, mãe_pai(ann,kari)=V, mãe_pai(ann,helen) =V, mãe_pai(kari,jane)=V), logo, ela não fará parte da rede suporte construída. A partir dos princípios do KBMC, cada consulta feita ao BLP especifica uma rede Bayesiana proposicional que pode ser consultada utilizando ferramentas usuais de inferência de redes Bayesianas.

2.3.2 Mapeamento de um PRM em um BLP

Modelos Probabilísticos Relacionais (PRM) é uma linguagem para descrever modelos probabilísticos baseados na lógica de primeira ordem. Permitem que o domínio seja representado em termos de objetos, suas propriedades e relações entre eles. Estes modelos representam incerteza sobre propriedades de um objeto, representando sua dependência probabilística em outra propriedade deste objeto ou em propriedades de objetos relacionados. Podem até representar incerteza sobre a própria estrutura relacionada. Existem atributos que são fixos, ou seja, seus valores são conhecidos. Pode-se dizer então que PRMs estendem redes Bayesianas com o conceito de indivíduos, suas propriedades e relações entre eles. De uma maneira,

eles são para Redes Bayesianas o que a lógica de primeira ordem é para a lógica proposicional.

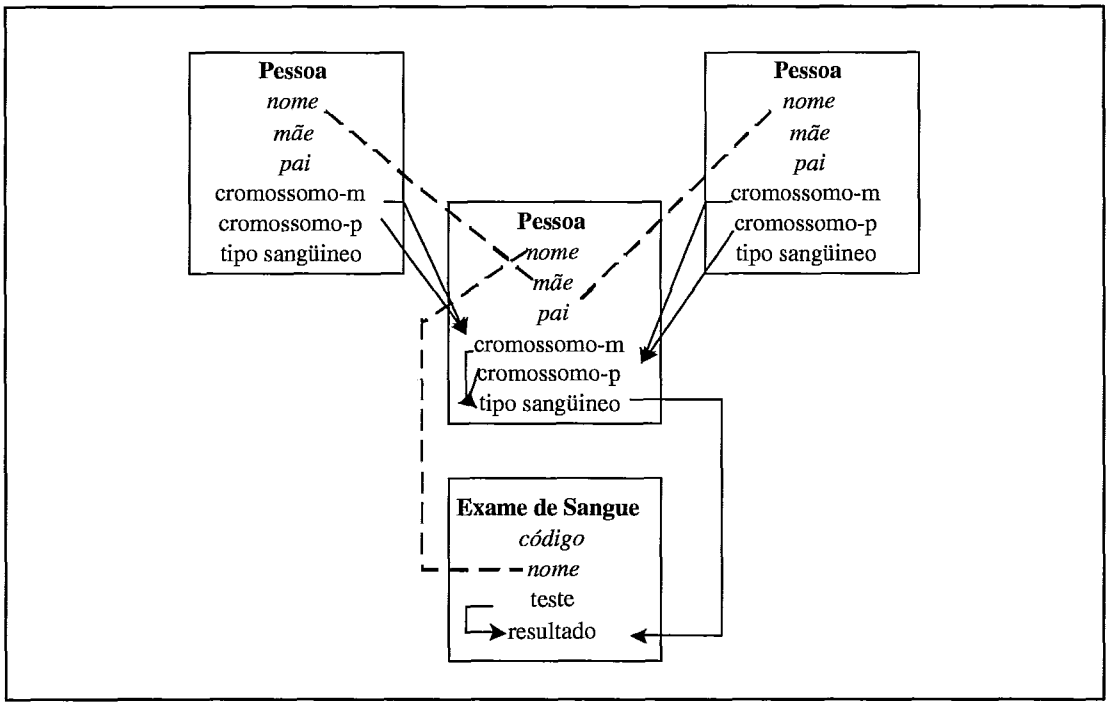

Figura 2.11: Estrutura do PRM para um domínio simples de genética. Linha pontilhada indica relação entre objetos, enquanto que linhas cheias indicam dependência probabilística. Atributos fixos estão em itálico, enquanto que atributos probabilísticos em fonte regular (FRIEDMAN, GETOOR, et al., 1999).

O exemplo na figura 2.11 representa um modelo simples de genética para a herança de um único gene que determina o tipo sanguíneo de uma pessoa. Cada pessoa tem duas cópias do cromossomo contendo este gen, um herdado de sua mãe e outro herdado de seu pai. Existe também um teste que tenta reconhecer o tipo sanguíneo da pessoa. O esquema contém duas classes: *Pessoa* e *Exame_de_Sangue* e três relações: *Pai*, *Mãe* e *Teste_de*. São três os objetos da classe *Pessoa*. Os atributos de *Pessoa* são *nome*, $cromossomo - p$(cromossomo herdado do pai), $cromossomo - m$(cromossomo herdado da mãe). O atributo *nome* da classe *Pessoa* é representado da seguinte maneira: *Pessoa.nome*.

Generalizando, o vocabulário de um modelo relacional consiste de um conjunto de classes $X_1, ...X_n$ e um conjunto de relações $R_1, ...R_m$. Cada classe é associada a um conjunto de atributos $\Lambda(X_i)$. Cada atributo $A_j \in \Lambda(X_i)$ assume algum valor do domínio finito $V(A_j)$. Usa-se $X.A$ para denotar o atributo de um objeto na classe

X. As classes e relações definem o esquema do modelo.

Se $R(X_1, ...X_m)$ é qualquer relação, pode-se projetar R nos seus argumentos i-ésimo e j-ésimo para obter uma relação binária $\rho(X_i, X_j)$, que podemos ver como uma abertura (*slot*) de X_i. Para qualquer objeto x da classe X_i, pode-se denotar $x.\rho$ como representando todos os elementos y em X_j tal que $\rho(x, y)$ é satisfeito. Estas aberturas (*slot*) podem ser concatenadas formando uma cadeia de aberturas (*chain slot*) $\tau = \rho_1, ..., \rho_m$ definida como composição de relações binárias.

Uma instância I de um esquema é uma interpretação para o modelo relacional. Um modelo probabilístico relacional define uma distribuição de probabilidade sobre um conjunto de instâncias de um esquema. Considera-se que o conjunto de objetos e de relações entre eles são fixos, logo o PRM define somente uma distribuição de probabilidades sobre os atributos dos objetos do modelo.

O PRM especifica a distribuição de probabilidades usando o mesmo princípio de redes Bayesianas; assume que cada variável aleatórias, neste caso os atributos $x.a$ de um objeto x, é diretamente influenciada por um conjunto de outros atributos.

Formalmente o PRM consiste de duas componentes: uma estrutura qualitativa de dependência, S, e os parâmetros associados a esta, θ_S. A estrutura de dependência é definida através da associação a cada $X.A$ de um conjunto de pais, $Pa(X.A)$. Existem dois tipos de pai. O atributo $X.A$ pode depender de um outro atributo probabilístico B de X, ou seja para qualquer objeto x, $x.a$ dependerá probabilisticamente de $x.b$. Ou o atributo $X.A$ pode depender de um atributo de um objeto relacionado a X, $X.\tau.B$, onde τ é uma abertura encadeada (*slot chain*), ou seja, exceto em casos onde a abertura encadeada é garantidamente um valor único, é necessário a especificação da dependência probabilística de $x.a$ para todo o conjunto de valores $y.b : y \in x.\tau$. Para isso PRM utiliza a noção de agregação da teoria de banco de dados.

Em (KERSTING, De RAEDT, 2000) BLP é relacionado a outras abordagens que também estendem redes Bayesianas à lógica de primeira ordem, entre elas o PRM.

Para representar um PRM através de um BLP considera-se que cada atributo $X.A$ é um predicado Bayesiano, $a(X)$ e cada relação r de aridade n, é um predicado

de aridade n, r. O atributo $X.A$ com um pai $X.B$ será equivalente a cláusula $a(X)|b(X)$, e caso $X.A$ dependa de outro atributo em uma classe relacionada tem-se $a(X)|b(Y)$, $\tau(X,Y)$.

Voltando ao exemplo da determinação do tipo sanguíneo, tem-se o seguinte conjunto de cláusulas para representa-lo no BLP.

cromossomo-m(X) | mãe(X,Y), cromossomo-p(Y), cromossomo-m(Y).

cromossomo-p(X) | pai(X,Y), cromossomo-p(Y), cromossomo-m(Y).

tipo-sanguíneo(X) | cromossomo-m(X), cromossomo-p(X).

resultado(X) | teste-de(X,Y), teste(X), tipo-sanguíneo(Y).

Enquanto PRM utiliza a noção de agregação para trabalhar com múltiplas instanciações, BLP usa as regras de combinação.

2.3.3 Aprendizado da estrutura de um PRM e de um BLP

Em (FRIEDMAN, GETOOR, et al., 1999), um algoritmo de aprendizado de PRMs que automaticamente constrói um modelo probabilístico relacional é proposto. Este algoritmo gera possíveis estruturas, que sejam acíclicas, e com uma heurística, função de avaliação, e um algoritmo de busca, vai verificar a melhor de acordo com o conjunto de exemplos de treinamento.

O algoritmo de busca é de *hill-climbing*. A estrutura corrente é mantida e iterativamente tenta-se melhorá-la. A cada iteração, considera-se um conjunto de simples transformações locais, avaliando cada uma e escolhendo a de maior valor(aquela que maximiza a função de avaliação). Como a função de avaliação é a probabilidade posterior da estrutura, esta pode ser decomposta em componentes locais, logo se o algoritmo de busca considera uma modificação em um determinado atributo $X.A$, somente a componente da função de avaliação relacionada a este atributo precisará ser avaliada, diminuindo assim a computação.

As modificações locais são modificações no conjunto de pais de um determinado atributo. Como são muitos os possíveis conjuntos de pais e como mesmo localmente a computação da função de avaliação é custosa, a cada fase do algoritmo utiliza-se um conjunto de pais potenciais, onde as modificações locais serão feitas utilizando apenas este conjunto. Conhecendo então, os possíveis pais, a computação mais custosa,

a probabilidade conjunta e a agregação, podem ser pré-computadas. Essas considerações permitem que o passo de busca seja feito com muita eficiência. Para a escolha dos pais potenciais (FRIEDMAN, GETOOR, et al., 1999) baseou-se em (FRIEDMAN, NACHMAN, et al., 1999), o qual examina uma abordagem semelhante para o contexto de redes Bayesianas. A idéia para determinação dos pais potenciais é uma abordagem iterativa que começa com uma estrutura e seleciona $Pot_k(X.A)$ (pais potenciais para o atributo X.A na fase k) baseado nesta estrutura. O procedimento de busca é então executado e uma nova estrutura escolhida. Escolhe-se então um novo conjunto de pais potenciais baseado nesta nova estrutura e repete-se o procedimento parando quando nenhuma melhora for mais observada. Inicializa-se $Pot_1(X.A)$ com o conjunto de atributos de X. Nas fases seguintes $Pot_{k+1}(X.A)$ consistirá de todos os $Pot_k(X.A)$, assim como todos os atributos que estão relacionados com X via uma cadeia de aberturas de tamanho menor que k. Como o conjunto de pais é expandido a cada fase, este pode tornar-se muito grande, logo utiliza-se um algoritmo de refinamento que somente adiciona um pai ao conjunto $Pot_{k+1}(X.A)$ se este realmente for acrescentar algo de acordo com $Pot_k(X.A)$. Para mais detalhes (FRIEDMAN, GETOOR, et al., 1999).

Para aprender a estrutura de um BLP, (KERSTING, De RAEDT, 2001c) fornece uma abordagem mais lógica, onde o espaço de hipóteses é composto por BLPs válidos, ou seja que provam todos os exemplos no conjunto de treinamento. Os exemplos contem duas partes a primeira é a lógica, onde cada um dos exemplos é um modelo mínimo de Herbrand para um BLP desconhecido e que será utilizado para verificar se o BLP corrente é valido. A segunda parte é a probabilística que é uma associação parcial de valores para as variáveis aleatórias do exemplo, definindo quais são as variáveis evidência desse exemplo.

O algoritmo de aprendizado de um BLP utiliza a forma de aprendizado através de interpretação (*learning from interpretation*) de ILP (De RAEDT, 1997) para aprender a estrutura lógica de um BLP, que deve ser satisfeito por todos os exemplos no conjunto de treinamento.

O algoritmo (veja a figura 2.12) prossegue propondo modificações no BLP inicial, modificações estas executadas por operadores de refinamento. Estes operadores

```
Seja H' uma hipótese inicial válida;
S(H') := Função_Avaliação(H');
repetir
   H := H';
   S(H) := S(H');
   para cada H"∈ ρ_g(H') ∪ ρ_s(H') faça
      se H"é (logicamente) válida sobre C então
         se a rede Bayesiana induzida por H"for acíclica então
            se Função_Avaliação(H")> S(H') então
               H' := H";
               S(H') := Função_Avaliação(H");
    fim
até S(H')≤ S(H);
retornar H';
```

Figura 2.12: Algoritmo de aprendizado da estrutura de um BLP, proposto em (KERSTING, De RAEDT, 2001c)

podem adicionar (ρ_s) um antecedente ao corpo de uma cláusula ou excluir (ρ_g) do corpo. Cada BLP proposto (hipótese) tem que ser consistente com o conjunto de treinamento (**C**) e acíclico. A melhor estrutura é escolhida usando uma função de avaliação probabilística. A função de avaliação escolhida foi a verossimilhança.

2.4 Refinamento de Teoria

A aquisição de conhecimento é uma tarefa difícil, demorada e com chances de erro. O processo de automaticamente melhorar uma base de conhecimento existente utilizando métodos de aprendizado é executada pelos sistemas de refinamento de teoria (WROBEL, 1996).

Experiências com sistemas especialistas e sistemas baseados em regras, rapidamente mostraram-se não muito eficientes. O fato do conhecimento ser fornecido por especialistas torna a tarefa demorada e conseqüentemente custosa. Fora isto, muitas das informações poderiam estar incompletas ou até mesmo incorretas. Uma maneira encontrada para resolver este problema foi o aprendizado indutivo(MITCHELL, 1997), onde a idéia era minimizar o conhecimento fornecido por um especialista fazendo com que o sistema extraísse o conhecimento através de exemplos. Tomando

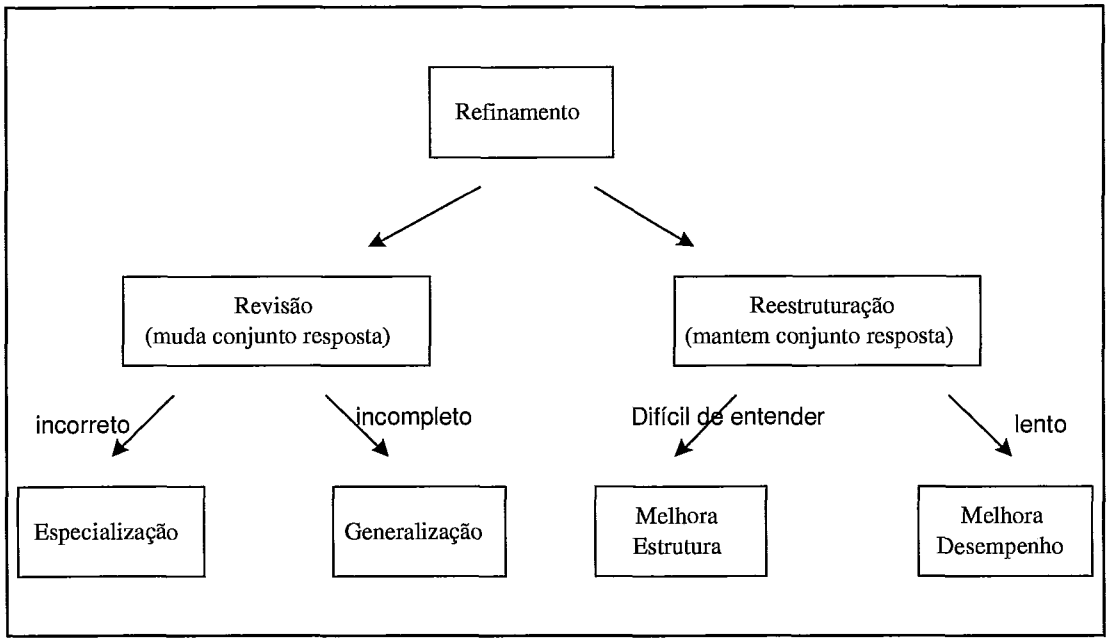

Figura 2.13: Esquema de Refinamento de Teoria definido em (WROBEL, 1996)

classificação como exemplo, o sistema receberia instâncias de classes e aprenderia a generalização destas, utilizando-se dos exemplos. Com a generalização seria possível resolver problemas futuros de classificação. Apesar de muitas técnicas desenvolvidas para este fim terem sido bem sucedidas elas requeriam uma base de exemplos de treinamento muito grande.

Uma outra abordagem seria o refinamento de teoria.

O processo de refinamento de uma teoria pode ser dividido em dois tipos: revisão e reestruturação de uma teoria. Ambos têm por objetivo aumentar a qualidade da teoria.

A tarefa de revisar significa mudar o conjunto de respostas de uma dada teoria, isto é melhorar sua capacidade de inferência proporcionando assim o adicionamento de respostas que haviam sido perdidas ou a remoção de respostas incorretas. No primeiro caso falamos em generalização da teoria e no segundo de especialização. Já a tarefa de reestruturação não muda o conjunto de resposta da teoria fornecida, o objetivo é a melhora de como as respostas são encontradas . Pode-se distinguir dois grupos de reestruturação, um com objetivo de melhorar o desempenho e o outro em facilitar o entendimento do usuário. O gráfico na figura 2.13 ilustra a idéia acima.

Como nesta tese o que utilizamos é revisão de teoria, o enfoque será nesta

questão, para mais detalhes (WROBEL, 1996).

Ao contrário do que ocorre no aprendizado a partir de exemplos em ILP (*ILP prediction learning problem*), onde são fornecidos exemplos e um conhecimento preliminar assumidamente correto (não serão modificados), no problema de revisão de teoria começa-se com uma teoria inicial a qual desejamos minimamente modificar, através de algum critério de minimalidade, para que represente melhor o conjunto de exemplos . Assume-se então que o conhecimento preliminar também esta incluído na teoria e que também pode ser modificado [2].

O problema de revisão de uma teoria pode ser definido da seguinte maneira, (WROBEL, 1996):

Definição 2.4.1 *Dados:*

- *uma teoria inicial* T
- *um conjunto de exemplos positivos e negativos* C^+ *e* C^-.

Achar:

- *uma teoria revista* T'
- *que derive todos os exemplos positivos (completo),* $T' \models C^+$
- *e nenhum dos exemplos negativos (consistência),* $\forall c^- \in C^- : T' \not\models c^-$
- *e satisfaz um critério de minimalidade.*

No nosso caso a teoria será composta de cláusulas definidas de primeira ordem e os exemplos serão átomos básicos.

Uma importante diferença entre sistemas de aprendizado a partir de exemplos e sistemas de revisão de teoria é a condição adicional que é pedida pela Teoria de Revisão, condição esta que garante que a teoria será revista o mínimo necessário. Em sistemas de revisão de teoria que não utilizam tais critérios adicionais, o resultado da revisão é avaliado somente em termos do desempenho, isto é, se a teoria tornou-se

[2]Em muitos sistemas, revisões em certas partes da teoria podem ser evitadas pelo usuário, desta maneira, caso deseje-se, pode-se deixar o conhecimento preliminar invariante.

completa e consistente, logo tornou-se correta. Tais sistemas usam a teoria inicial somente como um conhecimento preliminar para ajudá-los a alcançar o desempenho desejado melhor e mais rápido. Geralmente todos os exemplos são fornecidos de uma só vez (Batch em vez de incremental). Outros sistemas de revisão de teoria, entretanto, incorporam o aspecto de manutenção e requerem que a teoria revista seja encontrada a partir de uma revisão mínima da teoria inicial.

Como nosso sistema está baseado no sistema FORTE, definido na próxima seção, utilizaremos o mesmo critério de minimalidade que ele utiliza, que é o critério sintático, que parte do princípio que a revisão minimal é aquela que minimiza as aplicações dos operadores de revisão.

Resumindo, dado uma teoria inicial, esta podendo estar incompleta ou até mesmo incorreta, sistemas de revisão de teoria modificam minimamente esta teoria inicial tendo como objetivo a melhora na representação da base de dados. Técnicas (simbólicas) têm sido desenvolvidas para revisar teorias proposicionais e de primeira ordem(OURSTON, MOONEY, 1994) (COHEN, 1992) (WOGULIS, PAZZANI, 1993), (WOGULIS, 1994) (BAFFES, MOONEY, 1993) (BAFFES, 1994) (RICHARDS, MOONEY, 1995). Existem também as técnicas conexionista e as probabilísticas: (TOWELL, SHAVLIK, 1994), (GARCEZ, ZAVERUCHA, 1999), (FRIEDMAN, GETOOR, et al., 1999), (KERSTING, De RAEDT, 2001c),(BUNTINE, 1991). Nesses trabalhos, resultados experimentais mostram que sistemas de revisão de teorias podem aprender teorias mais precisas e com menos dados do que sistemas puramente indutivos.

Aplicaremos Revisão de Teoria para aprender a parte qualitativa de um programa em lógica Bayesiano (BLP) e conseqüentemente uma rede Bayesiana de primeira ordem é o objetivo desta tese.

Um exemplo de sistema de refinamento de teoria é o FORTE (RICHARDS, MOONEY, 1995), que veremos a seguir.

2.4.1 FORTE

First-Order Revision of Theories from Examples (FORTE) é um sistema para revisão de teorias, compostas por cláusulas definidas de primeira ordem, através de

um conjunto de exemplos de treinamento utilizando diferentes técnicas de revisão.

FORTE é um algoritmo iterativo de *hill-climbing* onde em cada iteração tenta melhorar o desempenho através da identificação de pontos responsáveis pela classificação incorreta de exemplos. Estes pontos precisam ser revistos e são chamados de pontos de revisão.

Vamos considerar o exemplo de uma teoria proposicional mostrado na figura 2.14 para entendermos mais facilmente o raciocínio de identificação das causas de uma classificação incorreta. Considere V como verdadeiro e F como falso.

Teoria :
H $\leftarrow$ X, Z.
H $\leftarrow$ Y, G.
X $\leftarrow$ A, B.
Y $\leftarrow$ C, D.
Exemplos:
1.G = V, A = F, B = F, D = F, Z = F, H = T
2.G = F, A = V, B = V, D = V, Z = V, H = F

Figura 2.14: Exemplo proposicional de identificação das causas de uma classificação incorreta

A teoria classifica o primeiro exemplo, dito um exemplo positivo de H, como um exemplo negativo, indicando que a teoria é muito específica. Isto poderia estar ocorrendo porque

1. Deveriam existir mais regras para a prova de H, ou
2. uma ou mais regras que provam H tem mais antecedentes do que o necessário, ou
3. as regras para X ou Y são muito específicas,isto é, tem mais antecedentes do que o necessário, ou
4. deveriam existir mais regras para a conclusão de X ou Y.

O segundo exemplo, dito um exemplo negativo de H, é classificado como um exemplo positivo pela teoria. Isto significa que a teoria é muito geral, o que pode estar ocorrendo porque:

1. Existem muitas maneiras de concluir H, isto é, uma das regras que esta provando H deveria ser excluída, ou

2. as regras para concluir H são muito gerais e precisam de antecedentes adicionais, ou

3. existem mais regras do que o necessário concluindo X ou Y, ou

4. as regras para concluírem X ou Y são muito gerais e precisam de antecedentes adicionais.

Estas observações não somente diagnosticam os erros na teoria, como também sugerem maneiras pelas quais a teoria poderia ser revista para corrigir as falhas.

Vamos exemplificar agora para o caso de primeira ordem. Considere a teoria de primeira ordem e o conjunto de exemplos de treinamento descritos na figura 2.15.

Teoria:
1. avo(A,B)| pai(A,C),mãe_pai(C,B).
2. avo(A,B)| pai(A,B).
3. mãe_pai(A,B) | pai(A,B).
4. mãe_pai(A,B) | mãe(A,B), gênero(B,feminino).

Exemplos:
Positivo: avo(walter,jim)| pai(walter,carol),mae(carol,jim).
Negativo: avo(lee,jim) | pai(lee,jim).

Figura 2.15: Exemplo de primeira ordem de identificação das causas de classificação incorreta

Como se pode ver o exemplo positivo não está sendo provado, pois falha no predicado gênero(carol,feminino). Uma alteração então, na teoria poderia ser a retirada do antecedente gênero(B,feminino) da cláusula 4. Já o exemplo negativo é provado, o que não deveria acontecer, e podemos verifica que a responsável é a cláusula 2, pois ela diz que basta A ser pai de B para que A também seja avó de B o que sabemos não ser verdade, logo uma revisão seria a exclusão desta cláusula da teoria. Teríamos então como teoria resultante a mostrada na figura 2.16.

Em cada iteração, para cada exemplo, FORTE anota os pontos de revisão:

- Generalização - o literal em uma cláusula responsável pela falha na prova de uma instância (exemplo) positiva (ponto de falha) e outros antecedentes

Teoria Revista:

1. avo(A,B) | pai(A,C) , mãe_pai(C,B).
3. mãe_pai(A,B) | pai(A,B).
4. mãe_pai(A,B) | mae(A,B).

Figura 2.16: Exemplo de primeira ordem, teoria resultante

(pontos de contribuição) que podem ter contribuído para esta falha atribuindo valores incorretos para as variáveis; Como exemplo, considere a relação abaixo também do problema da família

irmã(X,Y) | filha(X,Z), mãe_pai(Z,Z).

Quando tentarmos executar o predicado irmã, o antecedente mãe_pai será guardado como um ponto falho, pois não tem como alguém ser pai dele mesmo, e o antecedente filha como um ponto de contribuição, pois foi dele a escolha da variável Z responsável pela falha no predicado mãe_pai.

Qualquer predicado destes literais anotados também é considerado um ponto de revisão (baseado em predicado). Estes pontos de revisão são utilizados pelo operador de identificação (a ser definido), o qual procura generalizar a definição desses predicados.

- Especialização - cláusulas usadas em provas de instâncias negativas.

Cada ponto de revisão tem um potencial, que é o é o máximo crescimento em desempenho que pode ser proporcionado à teoria a partir de modificações neste ponto. Ou seja, é o número de exemplos que verificaram a necessidade de revisão neste ponto, e que podem passar a ser classificados corretamente após uma revisão ser feita.

Para efetuar a revisão em cada ponto de revisão são utilizados operadores.

Os operadores para especialização são: ·

- Exclusão de regra - existem duas restrições para este operador. Ele não pode excluir uma cláusula que seja ou a cláusula base de um predicado recursivo ou a única cláusula que define um determinado predicado. Neste último substitui-se a cláusula a ser excluída pela regra *conceito :- fail.*

- Adição de antecedente - adiciona-se antecedentes a uma cláusula na tentativa de fazer com que não sejam provadas todas as instâncias negativas. Se o adicionamento destes antecedentes fizer com que instâncias positivas deixem de ser provadas, adiciona-se esta cláusula especializada à teoria e recomeça a especialização com a cláusula original, procurando especializações alternativas que retenham a prova das outras instâncias positiva enquanto eliminando as negativas. Existem dois algoritmos para adicionar antecedentes à uma cláusula: o primeiro é o adição de antecedentes *hill-climbing* - este adiciona um antecedente de cada vez procurando pelo que proporciona o melhor desempenho da teoria. Algumas vezes nenhum dos antecedentes diminui o desempenho, mas também não aumenta. Para estes casos é preciso que vários antecedentes sejam adicionados de uma só vez, utiliza-se então o segundo algoritmo, *relational pathfinding*, o qual tenta encontrar os antecedentes que tem relação entre si, para mais informações (RICHARDS, MOONEY, 1995).

Os operadores para generalização são:

- Exclusão de antecedente - temos dois métodos: a)o primeiro é a exclusão de antecedentes *hill-climbing*. Este método vai excluir um antecedente de cada vez na cláusula, tornando-a mais geral. O antecedente a ser escolhido é aquele que aumenta o desempenho ao máximo, enquanto não prova nenhuma instância negativa. Este processo é repetido até que o desempenho da teoria não possa ser melhorado. b) exclusão de múltiplos antecedentes- algumas vezes a prova de uma instância apenas é afetada com a exclusão de mais de um antecedente de uma só vez. Primeiro todos os antecedentes cuja exclusão não permite que instâncias negativas sejam provadas, são reunidos; então combinações destes antecedentes são feitas na procura daquela que permite o maior número de instâncias positivas provadas sem provar negativas. O algoritmo não pára quando todas as instâncias positivas foram provadas, continua excluindo tantos antecedentes quanto puder. Este algoritmo é computacionalmente caro; entretanto só é chamado quando o *hill-climbing* não proporciona nenhuma revisão.

- Adiciona regra - é uma revisão baseada em cláusula. Deixa a cláusula original na teoria e gera novas baseadas nesta. O processo é feito em duas etapas. Primeiro copia a cláusula original e usando o algoritmo de exclusão de antecedente *hill-climbing*, exclui antecedentes sem permitir que nenhuma instância negativa seja provada e permitindo também que algumas positivas, que antes não eram provadas passem a ser (mesmo que isto permita a prova de negativas). Então cria uma ou mais especializações desta regra, utilizando o operador de adicionamento de antecedentes, para permitir a prova das instâncias positivas desejadas enquanto elimina as negativas.

- Identificação - constrói uma cláusula nova para generalizar a definição de um antecedente que falhou na prova de uma instância positiva. Melhor que desenvolver a cláusula do nada, executa um passo de resolução inversa (MUGGLETON, 1997) usando duas regras existentes no domínio da teoria. Por exemplo considere o conjunto de cláusulas tiradas do problema da família.

  tio(X,Y) | gênero(X,masculino),tia_tio(X,Y).

  tio(X,Y) | gênero(X,masculino), irmão_irmã(X,Z), mãe_pai(Z,Y).

  tia_tio(X,Y) | casado(X,Z), irmão_irmã(Z,W), mãe_pai(W,Y).

  tia(X,Y) | gênero(X,feminino), tia_tio(X,Y).

  Quando uma instância de tia é apresentada onde esta é tia de sangue, esta instância não será provada. Um dos pontos de falha é a chamada para tia_tio. O operador de Identificação procurará por maneiras de fornecer uma outra cláusula para este predicado, e encontra uma das duas cláusulas para tio. A revisão proposta substituirá a segunda cláusula para tio por:

  tia_tio(X,Y) | irmão_irmã(X,Z), mãe_pai(Z,Y).

- Absorção - Vai substituir uma cláusula já existente por uma nova a partir da substituição dos seus antecedentes falhos pela cabeça de uma cláusula, cujo o conjunto dos seus antecedentes contém os antecedentes definidos como sendo falhos. Suponha como exemplo o conjunto de cláusulas abaixo.

  tio(X,Y) | gênero(X,masculino), irmão_irmã(X,Z), mãe_pai(Z,Y).

repetir
Gera pontos de revisão
Ordena pontos de revisão por potencial(maior para menor)
Para cada ponto de revisão
Gera possíveis revisões
Atualiza melhor revisão encontrada
Até que o potencial do próximo ponto de revisão seja menor que a pontuação da melhor revisão atualizada
Se a melhor revisão melhora a teoria
Implementa melhor revisão
Até que nenhuma revisão melhore a teoria

Figura 2.17: Algoritmo FORTE, proposto em (RICHARDS, MOONEY, 1995)

tia_tio(X,Y) | irmão_irmã(X,Z), mãe_pai(Z,Y).

tia_tio(X,Y) | casado(X,Z), irmão_irmã(Z,W), mãe_pai(W,Y).

Quando uma instância para tio, onde este não é um tio de sangue, é apresentada, esta não será provada. O ponto de falha será em irmão_irmã ou mãe_pai. O operador para absorção achará um antecedente similar na segunda cláusula para tia_tio, então substituirá a cláusula para tio por

tio(X,Y) | gênero(X,masculino), tia_tio(X,Y).

Quando cada um dos operadores é aplicado, a sua pontuação é guardada. Pontuação de um operador é o crescimento real proporcionado ao desempenho da teoria após a revisão no ponto de revisão em que foi aplicado. Ou seja, é a diferença entre instâncias que passaram a ser classificadas corretamente e instâncias que deixaram de ser classificadas corretamente.

Por este motivo, potencial e pontuação de um operador, teriam o mesmo valor apenas quando a aplicação deste, não permitir que instâncias que anteriormente eram classificadas de modo correto deixem de ser.

Para cada ponto de revisão escolhe-se a melhor revisão proposta, e esta será a proporcionada pelo operador de maior pontuação.

A revisão que realmente será efetuada é aquela definida como sendo a melhor revisão proposta. No caso de um empate, utiliza-se o critério de minimalidade. O critério de minimalidade do FORTE é o sintático, minimiza o número de operações que um operador de revisão efetua.

A revisão escolhida, para ser implementada, tem que proporcionar melhora em desempenho para a teoria.

Já que a cada iteração do algoritmo é exigido um aumento em desempenho onde este é limitado a cem porcento, o algoritmo garantidamente terminará.

O algoritmo pode ser visto na figura 2.17.

Caso todas as modificações propostas para a teoria, incluindo a escolhida como sendo a melhor, não proporcionarem melhora alguma a esta, a pontuação final é a pontuação inicializada antes dos operadores serem aplicados. Caso isto aconteça, não existe revisão a ser implementada, a teoria final é a teoria inicial. O último teste do algoritmo FORTE, falha quando a situação acima mencionada acontece.

Quando aplicando um operador de revisão não é preciso utilizar todo o conjunto de exemplos de treinamento, somente um subconjunto consistindo daquelas instâncias onde as provas dependem das cláusulas que estão sendo revistas.

Capítulo 3

Revisão de Programas em Lógica Bayesianos (RBLP)

Neste capítulo introduzimos RBLP (*Revision of Bayesian Logic Programs*).

Como vimos na seção 2.3.3, algoritmos de aprendizado de um PRM e de um BLP foram propostos, (FRIEDMAN, GETOOR, et al., 1999) e (KERSTING, De RAEDT, 2001c) respectivamente. Esses dois algoritmos, têm por objetivo encontrar a melhor estrutura dentre um conjunto de possíveis (espaço de hipóteses), utilizando uma função de avaliação probabilística para a escolha da melhor. Uma diferença importante entre estas duas abordagens é que enquanto o algoritmo de aprendizado de um PRM, utiliza um espaço de hipóteses onde as hipóteses são estruturas encontradas através de modificações locais em toda a rede (mudanças no conjunto de pais de um determinado atributo), o BLP tem uma visão mais lógica. Ele determina que o espaço de hipótese deve ser formado por estruturas consistentes com a base de dados, ou seja, as hipóteses devem ser BLPs válidos (em termos lógicos provam todos os exemplos contidos no conjunto de exemplos de treinamento).

O sistema por nós proposto (RBLP), é um algoritmo de dois procedimentos que assim como o algoritmo de aprendizado de um BLP, vai utilizar uma abordagem lógica. O primeiro procedimento é o de aproximação (RBLP_AP), este unifica Revisão de Teoria, baseada no FORTE para procurar por uma revisão mínima de um BLP fornecido, consistente com o conjunto de treinamento disponível, com uma adaptação do algoritmo EM (KOLLER, PFEFFER, 1997) para aprender as CPDs.

O BLP encontrado tem a melhor pontuação probabilística para o espaço de busca composto por hipóteses modificadas em lugares que falharam em classificação. O segundo procedimento então, procurará pelo BLP com máxima pontuação probabilística em todo espaço de busca (restrito aos BLPs consistentes com o conjunto de treinamento), maximizando então o BLP resultante do RBLP_AP.

O RBLP_AP espera receber uma estrutura inicial, verificando se existem pontos nesta, onde uma modificação precisa ser feita. Pontos da estrutura que precisam ser revisto, são pontos que falharam na classificação de algum exemplo de treinamento. Utilizando operadores para revisão, modificações são propostas e as estruturas resultantes formam o espaço de busca. Com uma função de avaliação probabilística, a melhor estrutura, é então, escolhida e implementada.

Por determinar os pontos onde a estrutura deve ser revista e assim limitar o espaço de busca, o RBLP_AP torna-se menos custoso do que os dois algoritmos acima mencionados.

O RBLP_AP modifica a estrutura do BLP apenas nos pontos onde exemplos indicam falhas, por tratar-se de um sistema de revisão. Ao contrário dos sistemas de aprendizado que partem do zero com o objetivo de encontrar a estrutura que melhor reflete o conjunto de treinamento, o RBLP já tem uma estrutura representando o domínio em questão e o objetivo é melhorá-la através de modificações onde ela falha em classificação. O BLP encontrado é um BLP que em muitos casos é suficiente, mas em outras situações, ou quando tiver sido verificado que o RBLP_AP não proporcionou a melhora esperada em classificação, ou quando existir tempo para uma resposta mais exata, o RBLP_MP, que procura pelo BLP de maior pontuação probabilística em todo o espaço de busca (restrito aos BLPs consistentes com a base de dados), pode ser executado.

Nas seções seguintes detalharemos estes dois procedimentos.

3.1 Procedimento RBLP_AP

Definimos o RBLP_AP como sendo um sistema de revisão de programas em lógica Bayesianos, que utiliza um conjunto de exemplos de treinamento para encontrar pontos de falha na estrutura e através de operadores de revisão gerar um

conjunto de possíveis revisões (hipótese), de onde será escolhida a melhor utilizando uma função de avaliação probabilística. Esta estrutura resultante será uma modificação minima do BLP fornecido, e será consistente com o conjunto de exemplos de treinamento.

O sistema RBLP_AP, então, espera receber um programa em lógica Bayesiano inicial e um conjunto de exemplos e após o processamento, ele retorna um BLP revisto que estará correto e será o de máxima probabilidade de acordo com o espaço de busca composto por BLPs modificados em pontos de falha em classificação. O diagrama da figura 3.1 mostra melhor esta idéia.

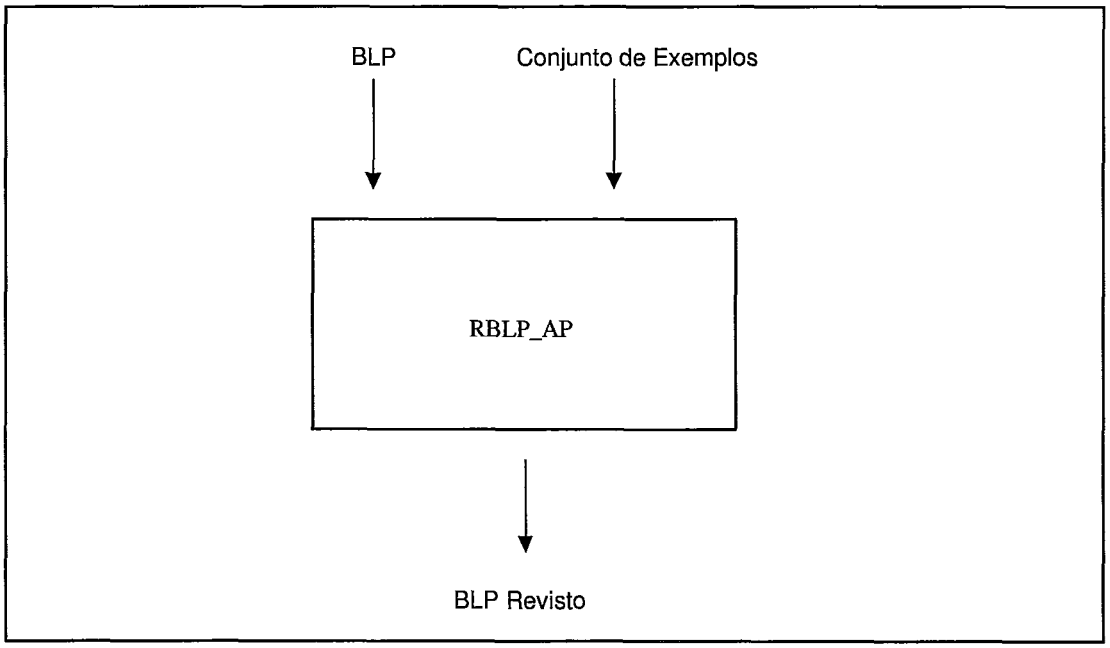

Figura 3.1: Diagrama de representação do procedimento RBLP_AP

Tendo recebido um BLP inicial, o primeiro passo do RBLP_AP é fazer o refinamento dos parâmetros probabilísticos (CPDs). Para isto é preciso primeiro construir a rede Bayesiana que cada um dos exemplos de treinamento define, utilizando o procedimento de resposta a uma consulta do BLP, onde este, para cada exemplo de treinamento tentará construir um rede Bayesiana proposicional. Dado este conjunto de redes Bayesianas os parâmetros podem agora ser aprendidos.

Na tentativa de gerar as estruturas de rede para cada um dos exemplos, já foi possível verificar quais os exemplos não foram classificados corretamente. Para os casos dos exemplos que falharam em classificação, por não serem provados, isto pôde

ser verificado, quando o algoritmo falhou na construção das suas redes Bayesianas, sendo estes então, guardados como exemplos falhos e da mesma maneira que o FORTE define um ponto de revisão, nós o faremos.

Depois de retreinados os parâmetros e se tiver sido verificado falha em classificação em algum exemplo, a componente qualitativa do BLP é revista. O esquema dos processos executados pelo RBLP_AP podem ser vistos na figura 3.2.

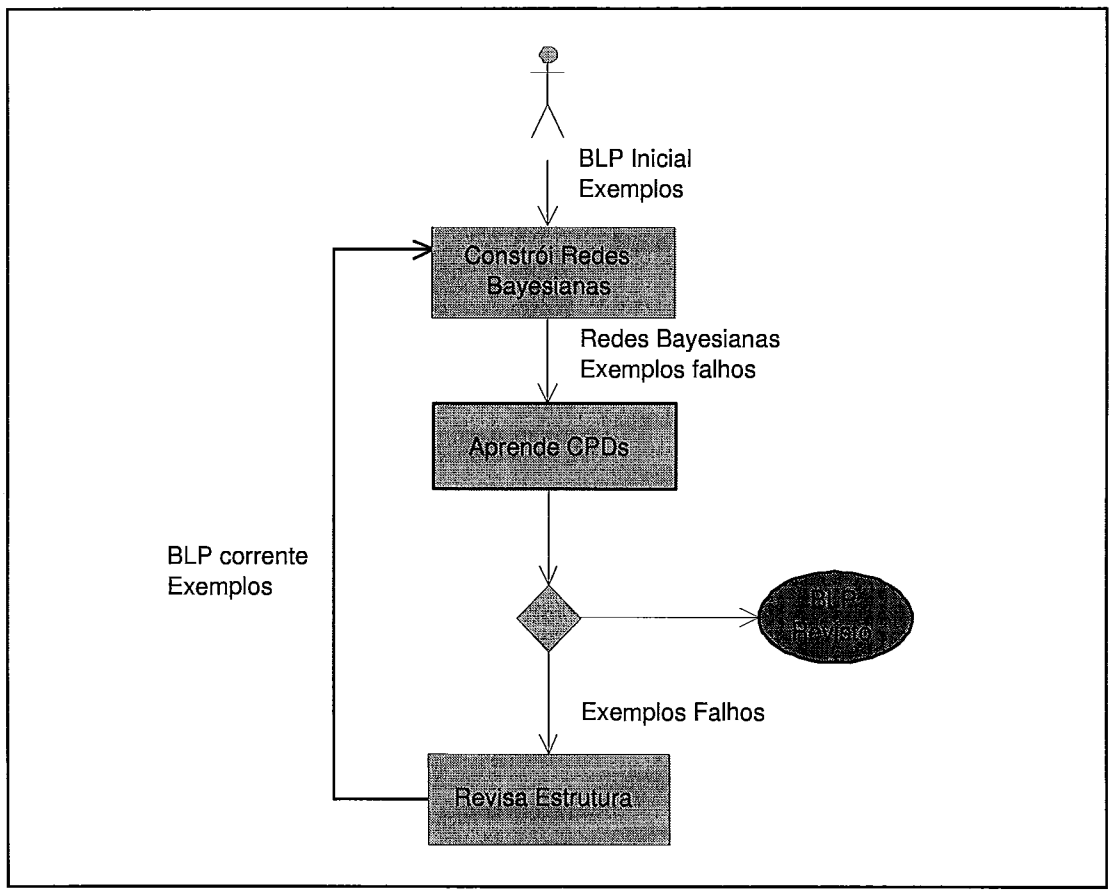

Figura 3.2: Diagrama de representação dos processos do procedimento RBLP_AP, onde os retângulos representam processos, o losango uma situação condicional e os textos as entradas dos processos.

A revisão é feita, utilizando os operadores de revisão definidos para o FORTE. Os operadores geram o espaço de hipótese para cada um dos pontos de revisão e com a função de avaliação probabilística, a melhor hipótese é escolhida. Os operadores que são avaliados pela função probabilística são aqueles que proporcionam alguma melhora lógica para o BLP. Apenas uma revisão é implementada a cada iteração do algoritmo e é aquela que maximiza a função de avaliação. O algoritmo prossegue

até que não seja mais possível melhorar logicamente o BLP corrente.

O RBLP_AP utiliza todos os pontos de revisão para escolher a melhor revisão, ao contrário do FORTE, que pára quando o potencial do próximo ponto a ser analisado é menor que a pontuação do ponto corrente. Isto ocorre porque como estamos lidando com uma função de avaliação probabilística, não há garantias de que a revisão com maior pontuação lógica também seria a de maior pontuação probabilística.

O Processo de revisão de um BLP pode ser visto em detalhes no diagrama da figura 3.3.

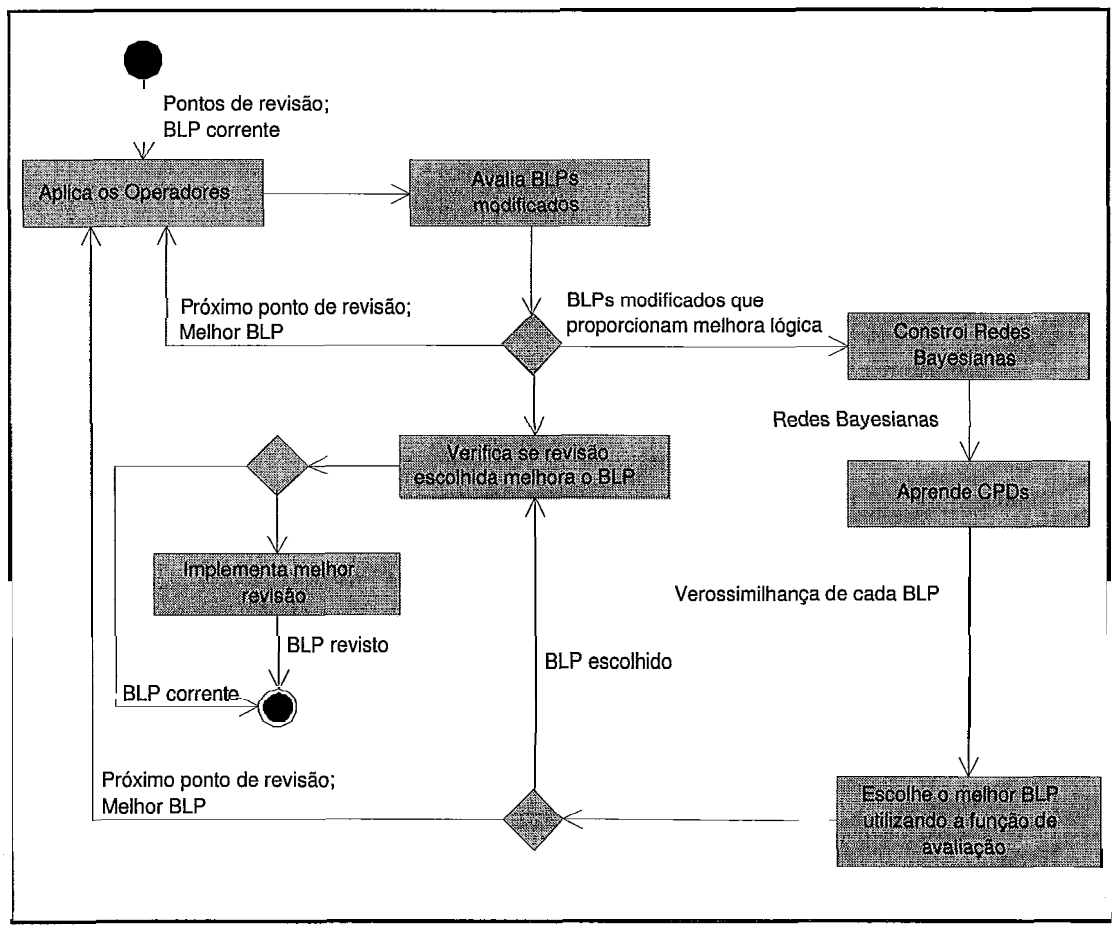

Figura 3.3: Diagrama de representação do processo de Revisão da Estrutura, onde os retângulos representam processos, os losangos uma situação condicional e os textos as entradas dos processos.

A seguinte notação será utilizada no decorrer da tese.

C : conjunto de exemplos de treinamento.

D^C : conjunto de evidências para o exemplo C.

N^C : Rede Bayesiana construída a partir do exemplo C usando o procedimento de consulta do BLP.

Θ : CPDs

Θ_i : CPDs da iteração i.

$\Theta_{i,j}$: CPDs da cláusula j para a iteração i.

$\theta_{i.j,x,pa}$: probabilidade da cláusula j para a iteração i quando X=x and Pa(X)=pa.

O conjunto de exemplos de treinamento será composto por modelos mínimos de Herbrand, representando a parte lógica do exemplo, com associação de valores para alguns dos seus termos, definindo as variáveis de evidência, para representar a parte probabilística do exemplo.

A probabilidade de um exemplo é definida como a probabilidade conjunta das variáveis evidência assumirem o valor definido para cada uma delas. Como a consulta tem um valor definido, ela também é considerada como sendo uma variável evidência. Mais precisamente, $P(C_i) = P(D^{C_i}), \forall i \in [1, m]$, onde m é número de exemplos de treinamento.

Consideramos que a função de avaliação probabilística é a verossimilhança ($L(H : \mathbf{C})$), onde H é a hipótese corrente, ou seja, é a componente qualitativa do BLP corrente e $\mathbf{C}$ o conjunto de exemplos de treinamento.

$$L(H : \mathbf{C}) = \mathbf{P}(\mathbf{C}|H, \Theta)$$

Nós consideramos que os exemplos são independentes, logo

$$P(\mathbf{C}|\Theta, H) = \prod_{i=1}^{m} P(C_i|\Theta, H)$$

onde m é o número de exemplos no conjunto de treinamento.

A melhor revisão (H*), então, é aquela que maximiza a função de avaliação probabilística.

$$H^* = \max_{H \in \mathbf{H}} L(H : \mathbf{C}) = \max_{H \in \mathbf{H}} \mathbf{P}(\mathbf{C}|H, \Theta)$$

onde $\mathbf{H}$ é o conjunto de hipóteses. Se a melhor revisão proporcionar melhora para o BLP, ou seja, se a verossimilhança do BLP modificado for maior que a verossimilhança do BLP inicial, então esta revisão será implementada.

O algoritmo pode ser visto na figura 3.4.

```
Seja H' o BLP inicial;
S_MR := score_C(H');
Melhor_revisão := H';
se tiverem sido encontrados exemplos falhos então
   faça
     faça para cada ponto de revisão
        gera possíveis revisões (H);
        para cada H"∈ H faça
           se H"melhorar logicamente o BLP então
              S_H" := score_C(H");
              se S_H" > S_MR então
                 Melhor_revisão := H";
                 S_MR := S_H";
              fim se
           fim se
        fim
     até terminar o conjunto de pontos de revisão
     se Melhor_revisão melhorar o BLP
        H' := Melhor_revisão
     fim se
   até nenhuma revisão melhorar logicamente o BLP
fim se
retorne H';
```

Figura 3.4: Algoritmo RBLP_AP

Nossa abordagem não somente tenta encontrar um BLP consistente com o conjunto de exemplos de treinamento, mas também tenta achar uma teoria que melhor reflita os dados.

O melhor operador de revisão será implementado se ele realmente proporcionar alguma melhora no desempenho da teoria, ou seja, se a verossimilhança dele for maior que a verossimilhança do BLP corrente.

O algoritmo termina quando nenhuma outra modificação proporcionar uma melhora lógica para o BLP corrente. Mantendo este critério garantimos que o BLP final será consistente com a base de dados.

Uma outra abordagem para reparar uma teoria, empregada por muitas técnicas existentes para aprendizado de redes Bayesianas (FRIEDMAN, 1997) (SINGH, 1997) (KWOH, GILLIES, 1996) (COOPER, HERSKOVITS, 1992) (RAMONI, SEBAS-

TIANI, 1997) é considerar todas as possíveis revisões e escolher a que proporcionar o melhor desempenho de acordo com uma função de avaliação. Já que geralmente existem muitas revisões possíveis para uma estrutura de rede, e somente uma pequena parcela dessas são necessárias para enquadrar os exemplos de treinamento, tal abordagem, gere e depois teste (MITCHELL, 1997) é computacionalmente cara.

A função $score_{\mathbf{C}}$(H') é responsável pela construção das redes Bayesianas, aprendizado das CPDs e determinação da verossimilhança. Discutir-la-emos em mais detalhes na próxima seção.

3.1.1 Revisão dos Parâmetros: Revisando as CPDs

Como discutido anteriormente o objetivo desta pesquisa é desenvolver um algoritmo que revise um programa em lógica Bayesiano. Para isto uma componente importante precisa ser revista, a componente quantitativa, ou seja, as distribuições de probabilidades condicionais (CPDs). É esta parte do algoritmo que iremos verificar nesta seção.

Como em nossa abordagem temos um programa em lógica Bayesiano e para cada exemplo do conjunto de treinamento uma rede Bayesiana é construída utilizando o procedimento de resposta a uma consulta, estas redes Bayesianas são diferentes uma das outras. As variáveis aleatórias dessas redes são variáveis relevantes, logo as variáveis que não aparecem não podem ser consideradas não-observadas e por isto não devem interferir na contagem estimada. O algoritmo EM, visto na seção 2.2.2, então não pode ser utilizado, senão as contagens esperadas seriam calculadas erradamente.

Propostas já foram feitas para o aprendizado das CPDs nestas situações, como em (KOLLER, PFEFFER, 1997), onde uma adaptação do algoritmo EM (DEMPSTER, LAIRD, et al., 1977) é utilizada ou em (KERSTING, De RAEDT, 2001a), que utiliza um procedimento baseado no gradiente (BINDER, KOLLER, et al., 1997). Nós utilizamos a primeira abordagem.

EM adaptado

No EM, que vimos na seção anterior, depois das contagens esperadas serem determinadas (passo-E), no passo-M, as probabilidades são calculadas utilizando estas contagens, por exemplo

$$P(X = x|Pa(X) = pa) = \frac{N(X = x, Pa(X) = pa)}{N(Pa(X) = pa)}$$

Para a revisão dos parâmetros de um BLP, não podemos estimar as contagens corretamente, pois cada exemplo constrói uma rede Bayesiana diferente, como foi explicado anteriormente. Por este motivo, uma modificação na maneira de estimar as contagens é necessária, e foi proposto em (KOLLER, PFEFFER, 1997). O que esta abordagem determina é que devemos olhar para as estruturas de rede construídas para cada exemplo, pois dessa maneira sabemos quais as variáveis relevantes para cada contagem, assim, exemplos que não as têm não influenciarão na contagem. Formalizando,

$$N(X, Pa(X)) = \sum_{\mathbf{C}} \sum_{X \in X_r^C} P(X = x, Pa(X) = pa|D^C)$$

onde X_r^C é o conjunto de variáveis aleatórias na cláusula r para o exemplo C e por isso relevantes para a contagem.

A fórmula proposta em (KOLLER, PFEFFER, 1997) para a determinação das distribuições de probabilidades das cláusulas do BLP é:

$$\theta_{r,x,pa} \leftarrow \frac{N(X, Pa(X))}{N(Pa(X))} = \frac{\sum_{C} \sum_{X \in X_r^c} P(X = x, Pa(X) = pa|D^C)}{\sum_{C} \sum_{X \in X_r^c} P(Pa(X) = pa|D^C)}$$

onde $\theta_{r,x,pa}$ significa a probabilidade da cláusula r, quando a variável X (cabeça da cláusula) assume valor x e seus pais (corpo da cláusula) (Pa(X)) assumem valor pa. Logo aplicando a fórmula para todos os possíveis valores de X e Pa(X) temos a distribuição de probabilidades da cláusula r.

Como dissemos anteriormente o conjunto de exemplos de treinamento é composto por átomos básicos. Cada exemplo em (**C**) é composto por um conjunto de evidências (D^c) e uma consulta. Dado um exemplo particular C, N^c é a rede de

conhecimento construída a partir deste utilizando o procedimento de resposta a uma consulta.

A fórmula representa o passo-M do algoritmo EM adaptado. Ela tem por objetivo calcular as distribuições de probabilidade para cada uma das cláusulas.

O critério de parada do EM adaptado é o mesmo do EM original, considera-se um erro e para-se quando

$$L(\theta_{i+1}|\mathbf{C}) - L(\theta_i|\mathbf{C}) \leq \xi$$

Vale lembrar que os exemplos que não geraram uma rede Bayesiana quando o procedimento de resposta a uma consulta foi aplicado não influenciarão na estimativa dos parâmetros.

3.1.2 Exemplo do procedimento RBLP_AP

Para ilustrar o funcionamento do RBLP_AP vamos considerar o seguinte BLP.

1. esposa(X,Y) | gênero(X,feminino),casado(X,Y).
2. irmão_irmã(X,Y) | mãe_pai(Z,X),mãe_pai(Z,Y).
3. tia_tio(X,Y) | casado(X,Z), irmão_irmã(Z,W), mãe_pai(W,Y).
4. tio(X,Y) | gênero(X,masculino), tia_tio(X,Y).

O domínio de todas as variáveis aleatórias do problema é binário

CPDs Iniciais (θ_0)

$\mathbf{P}$(casado(X,Y)) = $< 0.90, 0.10 >$

$\mathbf{P}$(gênero(X,Y)) = $< 0.95, 0.05 >$

$\mathbf{P}$(mãe_pai(X,Y)) = $< 0.85, 0.15 >$

casado(X,Y)	gênero(X,feminino)	**P**(esposa(X,Y))
V	V	0.35
F	V	0.01
V	F	0.01
F	F	0.01

Tabela 3.1: CPD da cláusula esposa(X,Y) | gênero(X,feminino), casado(X,Y)

mãe_pai(Z,X)	mãe_pai(Z,Y)	**P**(irmão_irmã(X,Y))
V	V	0.97
F	V	0.01
V	F	0.01
F	F	0.01

Tabela 3.2: CPD da cláusula irmão_irmã(X, Y) | mãe_pai(Z, X), mãe_pai(Z, Y)

casado(X,Z)	irmão_irmã(Z,W)	mãe_pai(W,Y)	**P**(tia_tio(X,Y))
V	V	V	0.95
F	V	V	0.01
V	F	V	0.01
V	V	F	0.01
F	F	V	0.01
V	F	F	0.01
F	V	F	0.01
F	F	F	0.01

Tabela 3.3: CPD da cláusula tia_tio(X, Y) | casado(X,Z), irmão_irmã(Z, W), mãe_pai(W,Y)

tia_tio(X,Y)	gênero(X,masculino)	**P**(tio(X,Y))
V	V	0.90
F	V	0.01
V	F	0.01
F	F	0.01

Tabela 3.4: CPD da cláusula tio(X, Y) | tia_tio(X, Y), gênero(X, masculino)

Vamos considerar que o conjunto de exemplos de treinamento é formado pelos quatro exemplos a seguir:

1. esposa(alice,art)=V| gênero(alice,feminino)=V, casado(alice,art)=V

2. tio(bob,jane)=V| gênero(bob,masculino)=V, casado(bob,helen)=V, mãe_pai(ann,helen)=V, mãe_pai(ann,kari)=V, mãe_pai(kari,jane)=V, irmão_irmã(helen,kari), tia_tio(bob,jane)

3. tio(john,jane)=V| gênero(john,masculino)=V, casado(john,lisa)=V, mãe_pai(ann,kari)=V, mãe_pai(ann,lisa)=V, mãe_pai(kari,jane)=V, irmão_irmã(lisa,kari), tia_tio(john,jane)

4. tio(eric,jane)=V| gênero(eric,masculino)=V, mãe_pai(ann,kari)=V, mãe_pai(ann,eric)=V, mãe_pai(kari,jane)=V, irmão_irmã(eric,kari), tia_tio(eric,jane)

Cada um dos termos dos exemplos, cujo predicado não é cabeça de uma cláusula do BLP, é adicionada ao mesmo como cabeça de uma cláusula com antecedente *true.*

Aplicando o algoritmo da figura 3.4, vamos computar a função $score_{\mathbf{C}}(BLP)$.

Utilizando o procedimento de resposta à uma consulta obtemos as 3 redes Bayesianas mostradas nas figuras 3.5,3.6 e 3.7, onde os nós cinza representam as evidências.

Durante o processo de construção das redes Bayesianas, como já havíamos comentado, o sistema identifica quais exemplos foram classificados incorretamente, logo note que os exemplos de treinamento 1 e 4 ambos não foram classificados corretamente. O primeiro foi classificado como falso, enquanto que o quarto não pôde nem ser provado. Devido ao exemplo 4, o sistema após aprender os parâmetros terá que revisar a estrutura do BLP. A única forma do exemplo 1 ser corrigido será através do aprendizado dos parâmetros, pois isto não causa revisão da estrutura no RBLP_AP

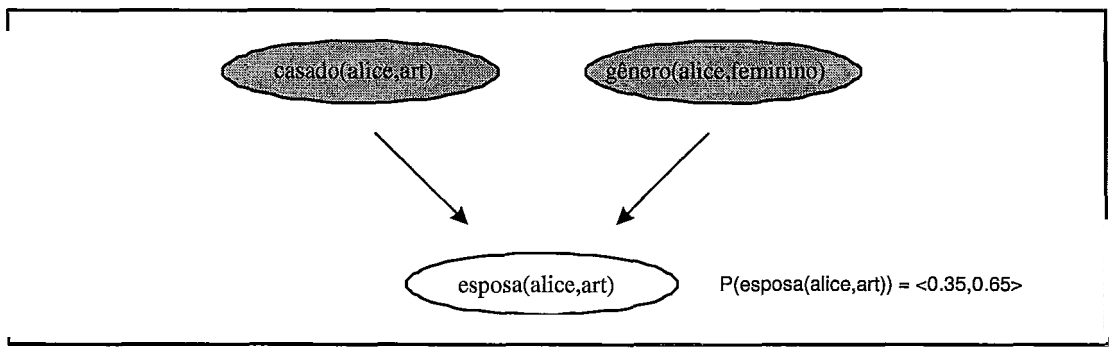

Figura 3.5: Rede Bayesiana construída para o Exemplo 1, durante o aprendizado das CPDs

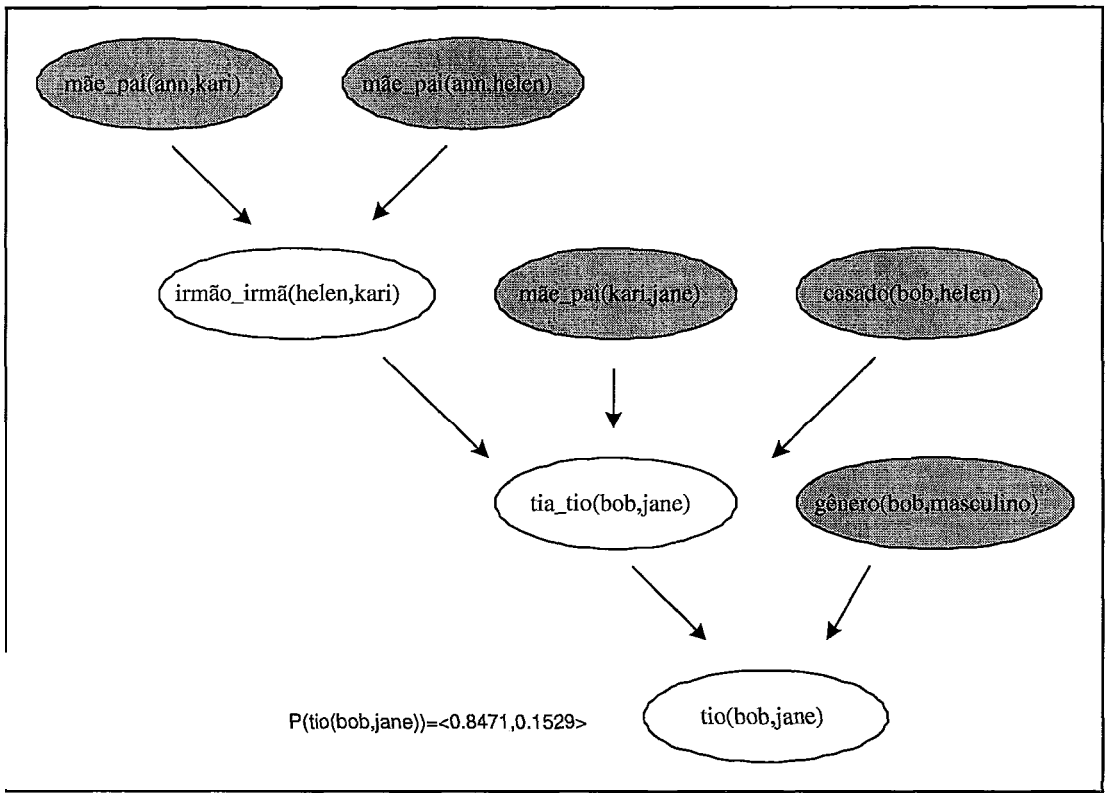

Figura 3.6: Rede Bayesiana construída para o Exemplo 2, durante o aprendizado das CPDs

Com as CPDs iniciais θ_0 podemos calcular a verossimilhança inicial.

$$L(\theta_0 : \mathbf{C}) = \mathbf{P}(\mathbf{C}|\theta_0, H) = 0.8347$$

onde H é a componente qualitativa do BLP

Ainda computando a função $score_{\mathbf{C}}(BLP)$, vamos aprender a componente quantitativa do BLP (CPDs), para tal utilizaremos o EM adaptado descrito na seção 3.1.1. A seguir mostramos os passos do processo de aprendizado de parâmetros.

Aprendizado da CPD para a Cláusula 1

Como a cláusula esposa(X,Y)| gênero(X,feminino), casado(X,Y) apenas é utilizada na construção da rede Bayesiana para o exemplo 1 , apenas este gerará um conjunto de variáveis aleatórias e conseqüentemente influenciará na contagem.

$$X_r^{C_1} = \{esposa(alice, art), gênero(alice, feminino), casado(alice, art)\}$$

$$N(esposa(X,Y), Pa(esposa(X,Y))) = \sum_{X \in X_1^{C_1}} P(X, Pa(X)|D^{C_1})$$

$$= (\mathbf{P}(esposa(alice, art), gênero(alice, feminino), casado(alice, art)|D^{C_1}) +$$

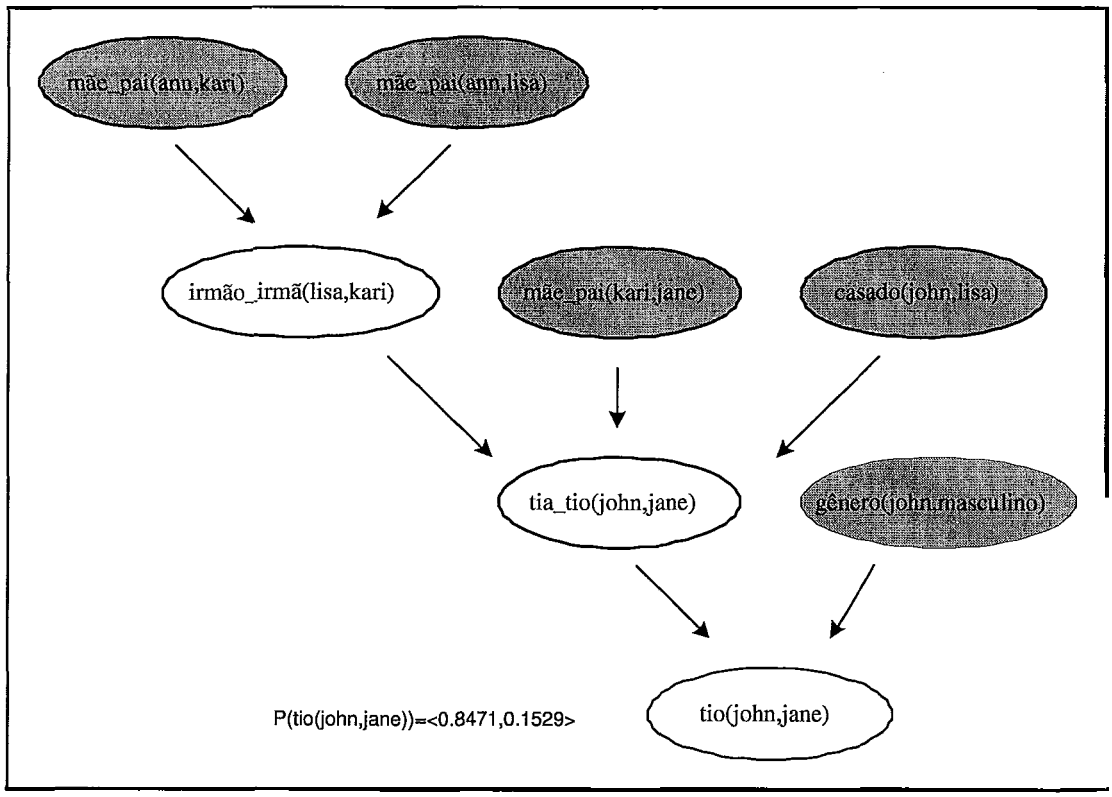

Figura 3.7: Rede Bayesiana construída para o Exemplo 3, durante o aprendizado das CPDs

$$\mathbf{P}(gênero(alice, feminino)|D^{C_1})+$$
$$\mathbf{P}(casado(alice, art)|D^{C_1}))$$
$$N(Pa(esposa(X,Y))) = \sum_{X \in X_r^{C_1}} P(Pa(X)|D^{C_1})$$
$$= \mathbf{P}(gênero(alice, feminino), casado(alice, art)|D^{C_1})$$

onde $D^{C_1} = \{gênero(alice, feminino), casado(alice, art)\}$

Podemos agora, calcular a CPD para a cláusula 1

$$\theta_{1.1} \leftarrow \frac{N(esposa(X,Y), Pa(esposa(X,Y)))}{N(Pa(esposa(X,Y)))}$$

onde $\theta_{1.1}$ representa a distribuição de probabilidades da cláusula 1 utilizando θ_0.

Aplicando algum método de inferência para redes Bayesianas as probabilidades das parcelas dos somatórios podem ser computadas, e em seguida a CPD para a cláusula em questão é calculada. Para este exemplo, utilizamos o método de inferência por enumeração. Este procedimento foi implementado em Matlab utilizando o Bayes Net Toolbox (MURPHY), (MURPHY, 2001).

A tabela 3.5 mostra a CPD retreinada para a cláusula 1.

$\theta_{1.1,V,(V,V)}$=P(esposa(X,Y) | gênero(X,feminino),casado(X,Y))	0.7833
$\theta_{1.1,V,(V,F)}$=P(esposa(X,Y) | gênero(X,feminino),¬casado(X,Y))	0.50
$\theta_{1.1,V,(F,F)}$=P(esposa(X,Y) |¬ gênero(X,feminino),¬ casado(X,Y))	0.50
$\theta_{1.1,V,(F,V)}$=P(esposa(X,Y) |¬ gênero(X,feminino),casado(X,Y))	0.50
$\theta_{1.1,F,(V,V)}$=P(¬ esposa(X,Y) | gênero(X,feminino),casado(X,Y))	0.2167
$\theta_{1.1,F,(V,F)}$=P(¬ esposa(X,Y) | gênero(X,feminino),¬casado(X,Y))	0.50
$\theta_{1.1,F,(F,F)}$=P(¬ esposa(X,Y) |¬ gênero(X,feminino),¬ casado(X,Y))	0.50
$\theta_{1.1,F,(F,V)}$=P(¬ esposa(X,Y) |¬ gênero(X,feminino), casado(X,Y))	0.50

Tabela 3.5: CPD retreinada para a cláusula esposa(X,Y) | gênero(X,feminino), casado(X,Y).

$\theta_{1.1,V,(V,V)}$ indica a probabilidade da cláusula 1, quando esposa(X,Y) = V e (gênero(X,feminino),casado(X,Y)) = (V,V), utilizando θ_0.

Vale lembrar que todas as cláusulas básicas têm a mesma distribuição de probabilidade da cláusula de onde foram originadas.

O mesmo procedimento é feito para as outras cláusulas do BLP. Como pode ser observado, ambas as redes Bayesianas construídas para os exemplos 2 e 3 utilizam as cláusulas 2,3 e 4.

Aprendizado da CPD para a Cláusula 2

$$X_r^{C_2} = \{irmão\_irmã(helen, kari), mãe\_pai(ann, helen), mãe\_pai(ann, kari)\}$$

$$X_r^{C_3} = \{irmão\_irmã(lisa, kari), mãe\_pai(ann, kari), mãe\_pai(ann, lisa)\}$$

$$N(irmão\_irmã(X,Y), Pa(irmão\_irmã(X,Y))) = \sum_{i\in\{2,3\}} \sum_{X\in X_r^{C_i}} P(X, Pa(X)|D^{C_i})$$

$$\begin{aligned}
&= (\mathbf{P}(irmão_irmã(helen, kari), mãe_pai(ann, helen), mãe_pai(ann, kari)|D^{C_2})+ \\
&\quad \mathbf{P}(mãe_pai(ann, kari)|D^{C_2})+ \\
&\quad \mathbf{P}(mãe_pai(ann, helen)|D^{C_2}))+ \\
&\quad (\mathbf{P}(irmão_irmã(lisa, kari), mãe_pai(ann, kari), mãe_pai(ann, lisa)|D^{C_3})+ \\
&\quad \mathbf{P}(mãe_pai(ann, kari)|D^{C_3})+ \\
&\quad \mathbf{P}(mãe_pai(ann, lisa)|D^{C_3}))
\end{aligned}$$

$$\begin{aligned}
N(Pa(irmão_irmã(X,Y))) &= \sum_{i\in\{2,3\}} \sum_{X\in X_r^{C_i}} P(Pa(X)|D^{C_i}) \\
&= \mathbf{P}(mãe_pai(ann, kari), mãe_pai(ann, helen)|D^{C_2})+ \\
&\quad \mathbf{P}(mãe_pai(ann, kari), mãe_pai(ann, lisa)|D^{C_3})
\end{aligned}$$

onde $D^{C_2} = \{mãe\_pai(ann, kari), mãe\_pai(ann, helen), gênero(bob, masculino),$
$casado(bob, helen)\}e$

D^{C_3} = $\{mãe\_pai(ann, kari), mãe\_pai(ann, lisa), gênero(john, masculino)\}$
$casado(john, lisa)$

Podemos agora, calcular a CPD para a cláusula 2.

$$\theta_{1.2} \leftarrow \frac{N(irmão\_irmã(X,Y), Pa(irmão\_irmã(X,Y)))}{N(Pa(irmão\_irmã(X,Y)))}$$

onde $\theta_{1.2}$ representa a distribuição de probabilidades da cláusula 2, utilizando θ_0.

A tabela 3.6 mostra a CPD retreinada para a cláusula 2.

$\theta_{1.2,V,(V,V)}$=P(irmão_irmã(X,Y) | mãe_pai(Z,X),mãe_pai(Z,Y))	0.99
$\theta_{1.2,V,(V,F)}$=P(irmão_irmã(X,Y) | mãe_pai(Z,X),¬mãe_pai(Z,Y))	0.50
$\theta_{1.2,V,(F,F)}$=P(irmão_irmã(X,Y) |¬ mãe_pai(Z,X),¬ mãe_pai(Z,Y))	0.50
$\theta_{1.2,V,(F,V)}$=P(irmão_irmã(X,Y) |¬ mãe_pai(Z,X),mãe_pai(Z,Y))	0.50
$\theta_{1.2,F,(V,V)}$=P(¬ irmão_irmã(X,Y) | mãe_pai(Z,X),mãe_pai(Z,Y))	0.01
$\theta_{1.2,F,(V,F)}$=P(¬ irmão_irmã(X,Y) | mãe_pai(Z,X),¬mãe_pai(Z,Y))	0.50
$\theta_{1.2,F,(F,F)}$=P(¬ irmão_irmã(X,Y) |¬ mãe_pai(Z,X),¬ mãe_pai(Z,Y))	0.50
$\theta_{1.2,F,(F,V)}$=P(¬ irmão_irmã|¬ mãe_pai(Z,X), mãe_pai(Z,Y))	0.50

Tabela 3.6: CPD retreinada para a cláusula irmão_irmã(X,Y) | mãe_pai(Z,X), mãe_pai(Z,Y).

$\theta_{1.2,V,(V,V)}$ indica a probabilidade da cláusula 2, quando irmão_irmã(X,Y) = V e (mãe_pai(Z,X),mãe_pai(Z,Y)) = (V,V), utilizando θ_0.

Aprendizado da CPD para a Cláusula 3

$X_r^{C_2} = \{irmão\_irmã(helen, kari), mãe\_pai(kari, jane), casado(bob, helen),$
$tia\_tio(bob, jane)\}$

$X_r^{C_3} = \{irmão\_irmã(lisa, kari), mãe\_pai(kari, jane), casado(john, helen),$
$tia\_tio(john, jane)\}$

$$N(tia\_tio(X,Y), Pa(tia\_tio(X,Y))) = \sum_{i\in\{2,3\}} \sum_{X\in X_r^{C_i}} P(X, Pa(X)|D^{C_i})$$

$= (\mathbf{P}(tia\_tio(bob, jane), irmão\_irmã(helen, kari), mãe\_pai(kari, jane),$
$casado(bob, helen)|D^{C_2})+$
$\mathbf{P}(irmão\_irmã(helen, kari), mãe\_pai(ann, kari), mãe\_pai(ann, helen)|D^{C_2})+$
$\mathbf{P}(mãe\_pai(kari, jane))|D^{C_2}))+$
$\mathbf{P}(casado(bob, helen))|D^{C_2}))+$
$(\mathbf{P}(tia\_tio(john, jane), irmão\_irmã(lisa, kari), mãe\_pai(kari, jane),$
$casado(john, helen)|D^{C_3})+$
$\mathbf{P}(irmão\_irmã(lisa, kari), mãe\_pai(ann, kari), mãe\_pai(ann, lisa)|D^{C_3})+$

$$\mathbf{P}(mãe\_pai(kari, jane))|D^{C_3}))+$$
$$\mathbf{P}(casado(john, helen))|D^{C_3}))$$
$$N(Pa(tia\_tio(X,Y))) = \sum_{i\in\{2,3\}} \sum_{X\in X_r^{C_i}} P(Pa(X)|D^{C_i})$$
$$= \mathbf{P}(irmão\_irmã(lisa, kari), mãe\_pai(kari, jane),$$
$$casado(john, lisa)|D^{C_3})+$$
$$\mathbf{P}(irmão\_irmã(ann, kari), mãe\_pai(ann, lisa)|D^{C_3})$$

onde $D^{C_2} = \{mãe\_pai(ann, kari), mãe\_pai(ann, helen), mãe\_pai(kari, jane),$
$casado(bob, helen), gênero(bob, masculino)\}$ e

$D^{C_3} = \{mãe\_pai(ann, kari), mãe\_pai(ann, lisa), mãe\_pai(kari, jane),$
$casado(john, helen), gênero(john, masculino)\}$

Podemos agora, calcular a CPD para a cláusula 3.

$$\theta_{1.3} \leftarrow \frac{N(tia\_tio(X,Y), Pa(tia\_tio(X,Y)))}{N(Pa(tia\_tio(X,Y)))}$$

onde $\theta_{1.3}$ representa a distribuição de probabilidades da terceira cláusula utilizando θ_0.

A tabela 3.7 mostra a CPD retreinada para a cláusula 3.

$\theta_{1.3,V,(V,V,V)}$=tia_tio(X,Y) | casado(X,Z), irmão_irmã(Z,W), mãe_pai(W,Y)	0.9802
$\theta_{1.3,V,(V,V,F)}$=tia_tio(X,Y) | casado(X,Z), irmão_irmã(Z,W), ¬ mãe_pai(W,Y)	0.50
$\theta_{1.3,V,(V,F,V)}$=tia_tio(X,Y) | casado(X,Z), ¬ irmão_irmã(Z,W), mãe_pai(W,Y)	0.0032
$\theta_{1.3,V,(F,V,V)}$=tia_tio(X,Y) | ¬ casado(X,Z), irmão_irmã(Z,W), mãe_pai(W,Y)	0.50
$\theta_{1.3,V,(F,F,V)}$=tia_tio(X,Y) | ¬ casado(X,Z), ¬irmão_irmã(Z,W), mãe_pai(W,Y)	0.50
$\theta_{1.3,V,(V,F,F)}$=tia_tio(X,Y) | casado(X,Z), ¬irmão_irmã(Z,W), ¬ mãe_pai(W,Y)	0.50
$\theta_{1.3,V,(F,V,F)}$=tia_tio(X,Y) | ¬ casado(X,Z), irmão_irmã(Z,W), ¬ mãe_pai(W,Y)	0.50
$\theta_{1.3,V,(F,F,F)}$=tia_tio(X,Y) | ¬ casado(X,Z), ¬ irmão_irmã(Z,W), ¬ mãe_pai(W,Y)	0.50
$\theta_{1.3,F,(V,V,V)}$= ¬ tia_tio(X,Y) | casado(X,Z), irmão_irmã(Z,W), mãe_pai(W,Y)	0.0198
$\theta_{1.3,F,(V,V,F)}$= ¬ tia_tio(X,Y) | casado(X,Z), irmão_irmã(Z,W), ¬ mãe_pai(W,Y)	0.50
$\theta_{1.3,F,(V,F,V)}$= ¬ tia_tio(X,Y) | casado(X,Z), ¬ irmão_irmã(Z,W), mãe_pai(W,Y)	0.9968
$\theta_{1.3,F,(F,V,V)}$= ¬ tia_tio(X,Y) | ¬ casado(X,Z), irmão_irmã(Z,W), mãe_pai(W,Y)	0.55
$\theta_{1.3,F,(F,F,V)}$= ¬ tia_tio(X,Y) | ¬ casado(X,Z), ¬irmão_irmã(Z,W), mãe_pai(W,Y)	0.50
$\theta_{1.3,F,(V,F,F)}$= ¬ tia_tio(X,Y) | casado(X,Z), ¬irmão_irmã(Z,W), ¬ mãe_pai(W,Y)	0.50
$\theta_{1.3,F,(F,V,F)}$= ¬ tia_tio(X,Y) | ¬ casado(X,Z), irmão_irmã(Z,W), ¬ mãe_pai(W,Y)	0.50
$\theta_{1.3,F,(F,F,F)}$= ¬ tia_tio(X,Y) | ¬ casado(X,Z), ¬ irmão_irmã(Z,W), ¬ mãe_pai(W,Y)	0.50

Tabela 3.7: CPD retreinada para a cláusula irmão(X,Y) | irmão_irmã(X,Y), gênero(X,_).

$\theta_{1.3,V,(V,V,V)}$ indica a probabilidade da cláusula 3, quando tia_tio(X,Y) = V e (casado(X,Z),irmão_irmã(Z,W), mãe_pai(W,Y)) = (V,V,V), utilizando θ_0 .

Aprendendo a CPD para a cláusula 4

$$X_r^{C_2} = \{tio(bob, jane), tia\_tio(bob, jane), gênero(bob, masculino)\}$$

$$X_r^{C_3} = \{tio(john, jane), tia\_tio(john, jane), gênero(john, masculino)\}$$

$$N(tio(X,Y), Pa(tio(X,Y))) = \sum_{i\in\{2,3\}} \sum_{X\in X_r^{C_i}} P(X, Pa(X)|D^{C_i})$$

$$= (\mathbf{P}(tio(bob, jane), tia\_tio(bob, jane), gênero(bob, masculino)|D^{C_2})+$$

$$\mathbf{P}(tia\_tio(bob, jane), casado(bob, helen), irmão\_irmã(helen, kari),$$

$$mãe\_pai(kari, jane)|D^{C_2})+$$

$$\mathbf{P}(gênero(bob, masculino)|D^{C_2}))+$$

$$(\mathbf{P}(tio(john, jane), tia\_tio(john, jane), gênero(john, masculino)|D^{C_3})+$$

$$\mathbf{P}(tia\_tio(john, jane), casado(john, lisa), irmão\_irmã(lisa, kari),$$

$$mãe\_pai(kari, jane)|D^{C_3})+$$

$$\mathbf{P}(gênero(john, masculino)|D^{C_3}))+$$

$$N(Pa(tio(X,Y))) = \sum_{i\in\{2,3\}} \sum_{X\in X_r^{C_i}} P(Pa(X)|D^{C_i})$$

$$= \mathbf{P}(tia\_tio(bob, jane), gênero(bob, masculino)|D^{C_2})+$$

$$\mathbf{P}(casado(bob, helen), irmão\_irmã(helen, kari),$$

$$mãe_pai(kari, jane)|D^{C_2})+$$

$$\mathbf{P}(tia\_tio(john, jane), gênero(john, masculino)|D^{C_3})+$$

$$\mathbf{P}(casado(john, lisa), irmão\_irmã(lisa, kari),$$

$$mãe\_pai(kari, jane)|D^{C_3})+$$

onde $D^{C_2} = \{mãe\_pai(ann, kari), mãe\_pai(ann, helen), gênero(bob, masculino),$

$casado(bob, helen)\}e$

$D^{C_3} = \{mãe\_pai(ann, kari), mãe\_pai(ann, lisa), gênero(john, masculino)\}$

$casado(john, lisa)\}$

Podemos agora, calcular a CPD para a cláusula 4.

$$\theta_{1.4} \leftarrow \frac{N(tio(X,Y), Pa(tio(X,Y)))}{N(Pa(tio(X,Y)))}$$

onde $\theta_{1.4}$ representa a distribuição de probabilidades da cláusula 4. utilizando θ_0.

A tabela 3.8 mostra a CPD retreinada para a cláusula 4.

$\theta_{1.4,V,(V,V)}$ indica a probabilidade da cláusula 4, quando tio(X,Y) = V e (tia_tio(X,Y), gênero(X,_)) = (V,V), utilizando θ_0.

$\theta_{1.4,V,(V,V)}$=P(tio(X,Y) | tia_tio(X,Y),gênero(X,_))	0.9514
$\theta_{1.4,V,(F,V)}$=P(tio(X,Y) | tia_tio(X,Y),gênero(X,_))	0.006
$\theta_{1.4,V,(V,F)}$=P(tio(X,Y) | tia_tio(X,Y),gênero(X,_))	0.50
$\theta_{1.4,V,(F,F)}$=P(tio(X,Y) | tia_tio(X,Y),gênero(X,_))	0.50
$\theta_{1.4,F,(V,V)}$= ¬ P(tio(X,Y) | tia_tio(X,Y),gênero(X,_))	0.0486
$\theta_{1.4,F,(F,V)}$= ¬ P(tio(X,Y) | ¬ tia_tio(X,Y),gênero(X,_))	0.9940
$\theta_{1.4,F,(V,F)}$= ¬ P(tio(X,Y) | tia_tio(X,Y), ¬ gênero(X,_))	0.50
$\theta_{1.4,F,(F,F)}$= ¬ P(tio(X,Y) | ¬ tia_tio(X,Y),¬ gênero(X,_))	0.50

Tabela 3.8: CPD retreinada para a cláusula tio(X,Y) | tia_tio(X,Y), gênero(X,_).

Com as CPDs encontradas para cada cláusula do problema, podemos calcular a verossimilhança corrente.

$$L(\theta_1 : \mathbf{C}) = P(\mathbf{C}|\theta_1, \mathbf{H}) = 0.9020$$

Considerando um erro de 0.10 ($\xi = 0.10$)[1], temos que:

$$L(\theta_1 : \mathbf{C}) - L(\theta_0 : \mathbf{C}) = 0.0673 < \xi$$

logo podemos considerar as CPDs encontradas na primeira iteração como sendo as novas CPDs do problema.

Note que a nova probabilidade para a consulta P(esposa(alice,art)), após o aprendizado dos parâmetros é

$$P(esposa(alice, art)) =< 0.7833, 0.2167 >$$

fazendo com que o exemplo 1 passe a ser classificado corretamente, mostrando que o aprendizado de parâmetros pode ajustar as probabilidades e fazer com que exemplos que são classificados incorretamente passem a ser classificados corretamente.

Ao término da função $score_C(BLP)$ da figura 3.4, como o exemplo 4 foi falho, teremos a revisão de estrutura.

Durante o aprendizado dos parâmetros, RBLP_AP anotou o exemplo 4 como sendo um exemplo falho. Possíveis modificações serão propostas para o BLP corrente, através de operadores de revisão (gera **H** na figura 3.4), na tentativa de encontrar o melhor BLP que classifique corretamente os exemplos de treinamento.

[1]Como o exemplo que estamos utilizando é muito simples o erro considerado foi suficiente para obtermos os resultados esperados, mas em situações mais complexas um erro de valor mais baixo ($\xi = 0.01$) seria mais apropriado.

Para revisar este BLP operadores de generalização serão aplicados. Estes podem ser: (a) o operador Exclui Antecedentes para excluir o antecedente $casado(X, Z)$ que foi o responsável pela falha em classificação, gerando a cláusula

$$5.tia\_tio(X, Y)|irmão\_irmã(X, Z), mãe\_pai(Z, Y).$$

ou (b) o operador para Adicionar uma Nova Cláusula ao BLP, que neste caso também seria a cláusula 5.

1°Revisão (Operador de Exclusão de Antecedente)

Com a aplicação do operador de revisão a cláusula 5 é gerada e a CPD para ela é inicializada randômicamente (tabela 3.9).

irmão_irmã(Z,W)	mãe_pai(W,Y)	**P**(tia_tio(X,Y))
V	V	0.85
F	V	0.01
V	F	0.01
F	F	0.01

Tabela 3.9: CPD da cláusula tia_tio(X, Y) | irmão_irmã(X, Z), mãe_pai(Z,Y)

Após a aplicação do operador de revisão, a rede Bayesiana para o exemplo 4 é construída utilizando o procedimento de resposta à uma consulta (figura 3.8).

As redes Bayesianas para os outros exemplos são as mesmas construídas na seção anterior. Com as redes Bayesianas construídas, os parâmetros probabilísticos são novamente aprendidos. As novas CPDs encontradas são mostradas nas tabelas 3.10, 3.11 e 3.12.

mãe_pai(Z,X)	mãe_pai(Z,Y)	**P**(irmão_irmã(X,Y))
V	V	0.9967
F	V	0.50
V	F	0.50
F	F	0.50

Tabela 3.10: CPD da cláusula irmão_irmã(X, Y) | mãe_pai(Z, X), mãe_pai(Z, Y)

Aplicando a função de avaliação ao BLP corrente, encontramos a verossimilhança abaixo.

$$S_{H_1} = \mathbf{P}(\mathbf{C}|H_1, \theta) = 0.9838$$

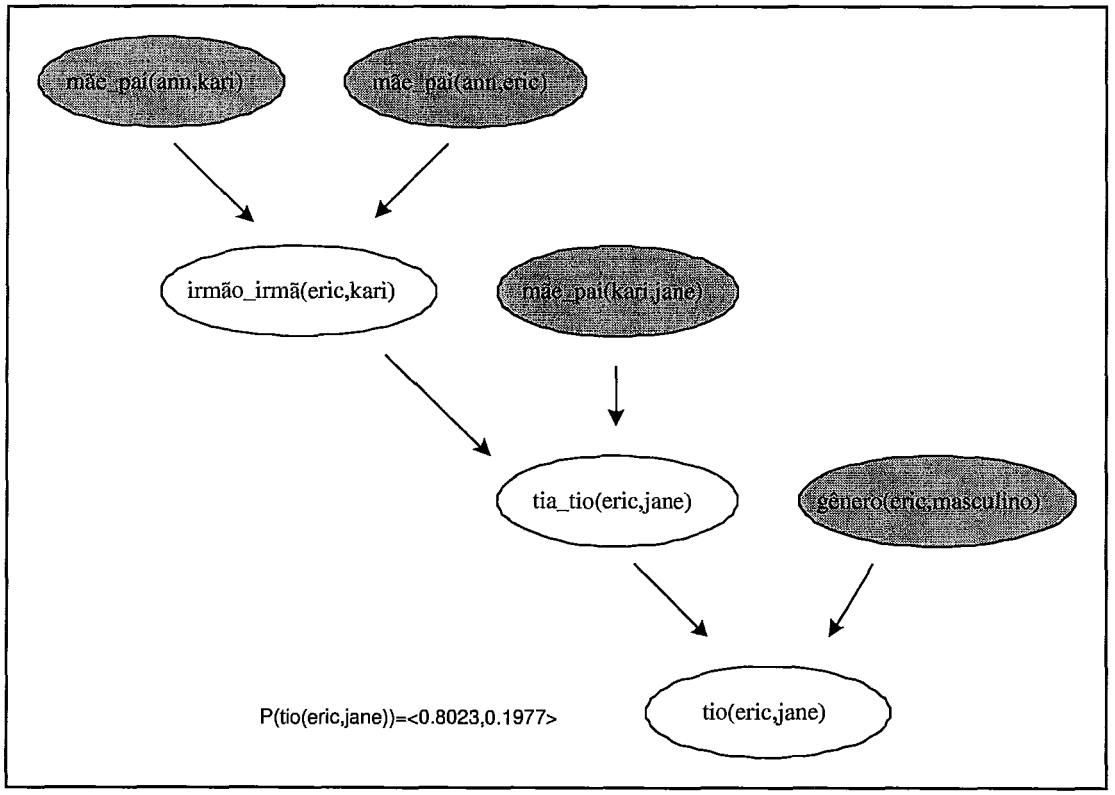

Figura 3.8: Rede Bayesiana construída a partir do exemplo 4

tia_tio(X,Y)	gênero(X,masculino)	**P**(tio(X,Y))
V	V	0.9669
F	V	0.0066
V	F	0.50
F	F	0.50

Tabela 3.11: CPD da cláusula tio(X, Y) | tia_tio(X, Y), gênero(X, masculino)

irmão_irmã(X,Z)	mãe_pai(Z,Y)	**P**(tia_tio(X,Y))
V	V	0.9470
F	V	0.0006
V	F	0.50
F	F	0.50

Tabela 3.12: CPD da cláusula tia_tio(X, Y) | irmão_irmã(Z, W), mãe_pai(W,Y)

onde H_1 é a estrutura do BLP para a proposta de revisão encontrada após a aplicação do primeiro operador (cláusulas 1,2,4 e 5).

Já que $S_{H_1} > S_{H_0} \therefore Melhor\_Revisão := S_{H_1}$

2°Revisão (Operador para Acrescentar uma Nova Cláusula)

Com a aplicação do segundo operador de revisão a cláusula $tia\_tio(X,Y)|irmão\_irmã(X,Z), mãe\_pai(Z,Y)$ é acrescida ao BLP e vamos considerar que a sua CPD inicial seja a mesma assumida durante a aplicação do 1° operador de revisão.

Da mesma maneira que foi feito para o 1° operador, a rede Bayesiana para o exemplo 4 é construída (figura 3.8) e em seguida as CPDs são aprendidas. As mesmas CPDs são encontradas com exceção da CPD para a cláusula 3 (tabela 3.13).

casado(X,Z)	irmão_irmã(Z,W)	mãe_pai(W,Y)	**P**(tia_tio(X,Y))
V	V	V	0.9968
F	V	V	0.50
V	F	V	0.50
V	V	F	0.0018
F	F	V	0.50
V	F	F	0.50
F	V	F	0.50
F	F	F	0.50

Tabela 3.13: CPD da cláusula tia_tio(X, Y) | casado(X,Z), irmão_irmã(Z, W), mãe_pai(W,Y)

Aplicando a função de avaliação.

$$S_{H_2} = \mathbf{P}(\mathbf{C}|H_2, \theta) = 0.9150$$

onde H_2 é a estrutura do BLP para a proposta de revisão do segundo operador (cláusulas 1,2,3,4 e 5).

Já que $S_{H_2} < S_{H_1}$ a melhor revisão continua sendo H_1 e esta será a hipótese escolhida. Como H_1 melhora o desempenho do BLP corrente, então ela será implementada.

Já que nenhuma outra melhora lógica pode ser proporcionada ao BLP corrente, o BLP final é:

1. esposa(X,Y) | gênero(X,feminino),casado(X,Y)

2. irmão_irmã(X,Y) | mãe_pai(Z,X), mãe_pai(Z,Y).

4. tio(X,Y) | gênero(X, masculino), tia_tio(X,Y).

5. tia_tio(X,Y)| irmão_irmã(X,Z), mãe_pai(Z,Y).

Pela figura 3.8 pode-se perceber que o exemplo 4 passou a ser classificado corretamente.

Para evitar *overfitting* utilizamos o método de validação cruzada (*cross-validation*) (MITCHELL, 1997).

O BLP revisto é consistente com a base de dados e é o de máxima probabilidade, considerando o espaço de busca dos pontos de revisão.

3.2 Procedimento RBLP_MP

O procedimento de aproximação discutido na seção anterior, encontra um BLP correto e que maximiza a função de avaliação probabilística considerando que o espaço de busca é composto por estruturas modificadas somente nos pontos de revisão.

Para muitas situações, e especialmente para classificação, onde o BLP inicial é aproximadamente correto, o BLP encontrado por RBLP_AP deve ser suficiente, como ocorreu para o exemplo visto na seção anterior. Se este não for o caso, ou foi verificado que o RBLP_AP não proporcionou a melhora em classificação esperada, e existir tempo para uma resposta mais exata, propomos a aplicação do procedimento de maximização após ter executado o de aproximação. O procedimento RBLP_MP é baseada no algoritmo de aprendizado da estrutura de um BLP visto em 2.3.3 (figura 2.12).

RBLP_MP tenta encontrar a estrutura com a maior pontuação probabilística em todo o espaço de busca, restrito a BLPs consistentes e evitando os pontos já revistos por RBLP_AP.

Se fosse do interesse, poderíamos aplicar o RBLP_MP no BLP retornado pelo exemplo da seção anterior. Este proporia modificações em toda a estrutura do BLP retornado por RBLP_AP, com o objetivo de encontrar um BLP de verossimilhança maior do que o já encontrado. Como o BLP inicial estava aproximadamente correto, e foi verificado que os pontos que falhavam em classificação foram corrigidos por RBLP_AP, aplicar RBLP_MP seria desnecessário.

Capítulo 4

Conclusão

Teoria de revisão é uma área de pesquisa que está baseada na idéia de que, quando somente um conjunto limitado de dados está disponível, pode-se melhorar o aprendizado focalizando o espaço de busca das hipóteses possíveis. Sistemas de revisão de teoria assumem que o sistema de aprendizado tem uma base de conhecimento inicial incompleta e/ou incorreta (geralmente obtida a partir de um especialista), a qual é indutivamente revista para refletir os dados. Experimentos com bases de dados que refletem problemas reais tem demonstrado que revisão de uma teoria aproximadamente correta, produz melhores resultados do que apenas aprendendo a partir de exemplos de treinamento.

Recentemente, novas linguagens para representação, que integram Lógica de primeira ordem com redes Bayesianas, têm sido desenvolvidas. Programas em Lógica Bayesiano(BLP) (KERSTING, De RAEDT, 2000), e Modelos Probabilísticos Relacionais (PRM)(KOLLER, 1999) são exemplos. Aprendizado tanto da parte quantitativa como da parte qualitativa destes modelos já foram propostos (KERSTING, De RAEDT, 2001c), (FRIEDMAN, GETOOR, et al., 1999) e se relacionam com a proposta desta tese.

Nas duas abordagens acima mencionadas, o objetivo é o aprendizado das estruturas. Tem-se então, um conjunto de possíveis estruturas (espaço de hipóteses), e com uma função de avaliação tenta-se encontrar a melhor. Podemos indicar como uma diferença importante entre os algoritmos de aprendizado de um PRM e de

um BLP, a abordagem totalmente probabilística do primeiro contra a abordagem lógica-probabilística do segundo. O algoritmo para aprender um PRM não garante que a estrutura encontrada seja consistente com o conjunto de treinamento; está preocupado em encontrar a estrutura de maior pontuação probabilística. Por outro lado, o algoritmo para aprender a estrutura de um BLP encontra necessariamente um BLP consistente com os exemplos de treinamento e que também tenha a maior pontuação probabilística.

Nosso objetivo nesta tese foi o de revisar um classificador de primeira-ordem Bayesiano, para isto, apresentamos uma integração de técnicas para aprender as CPDs e para modificar a estrutura de uma rede Bayesiana de primeira-ordem. O algoritmo resultante, Revisão de Programas Lógicos Bayesiano (RBLP) une redes Bayesianas com programação em lógica indutiva (Revisão de Teoria). O uso do BLP para representar redes Bayesianas de primeira-ordem permitiu que esta integração fosse mais fácil.

RBLP é um algoritmo de dois procedimentos. O procedimento de aproximação (RBLP_AP) encontra um BLP consistente e que tenha maximizado a função de avaliação probabilística, considerando que o espaço de busca é composto pelas estruturas modificadas somente nos pontos responsáveis pela falha em classificação. O procedimento de maximização (RBLP_MP), que é basicamente um algoritmo de aprendizado de BLP (KERSTING, De RAEDT, 2001c), procura pelo BLP de maior pontuação probabilística em todo o espaço de busca (restrito aos BLPs consistentes com os exemplos de treinamento).

Uma diferença importante entre RBLP_AP e o algoritmo de aprendizado de um PRM é a mesma mencionada acima entre os algoritmos de aprendizado de um BLP e de um PRM, já que ele também garante que o BLP final é consistente com o conjunto de treinamento (RBLP_AP baseia-se em FORTE e este garante a consistência da teoria resultante).

Para classificação, se a teoria inicial estiver aproximadamente correta, é mais eficiente propor modificações somente em lugares responsáveis por falha em classificação, analisando estas possíveis revisões e escolhendo a de maior pontuação probabilística. Conseqüentemente, a Revisão de Teoria é uma boa escolha , já que

identifica os lugares a serem reparados usando os exemplos de treinamento, e modifica somente estes. A teoria encontrada tem a maior pontuação probabilística para o espaço de busca acima mencionado. RBLP_AP utiliza esta idéia definindo, então uma outra diferença importante entre ele e as duas abordagens mencionadas acima. Sob a consideração de uma teoria aproximadamente correta, o BLP encontrado pelo RBLP_AP e o algoritmo de aprendizado de um BLP não diferem muito, tornando o RBLP_AP uma escolha mais adequada, já que o número de operações a executar é menor, mas se houver interesse ou se tiver sido verificado que o o procedimento RBLP_AP não proporcionou a melhora em classificação esperada e existir tempo para uma resposta mais exata, RBLP_MP pode ser usado com o objetivo de procurar o BLP com maior pontuação probabilística em todo o espaço de busca (restrito às teorias consistentes com os exemplos de treinamento). Se o BLP inicial não estiver aproximadamente correto, pode ser mais eficiente executar diretamente o RBLP_MP. Os resultados serão similares aos dados pelo algoritmo de aprendizado de um BLP, mas sem nenhuma melhora em eficiência.

Uma contribuição importante desta tese foi mostrar que o RBLP_AP pode ser mais eficiente que os dois algoritmos acima mencionados, quando utilizando um classificador Bayesiano de primeira-ordem aproximadamente correto.

A idéia de um procedimento de aproximação pode ser similarmente aplicado a um PRM, assim como a idéia de um procedimento de maximização pode ser aplicado ao BANNER para o caso proposicional. No futuro executaremos experimentos mais extensivos incluindo o uso da função de avaliação *Minimum Description Length* (MDL) e a utilização de outros métodos de inferência como o método de Monte Carlo.

Referências Bibliográficas

BAFFES, P., 1994, *Automatic Student Modeling and Bug Library Construction using Theory Refinement*, Ph.D. thesis, University of Texas, Austin, TX.

BAFFES, P., MOONEY, R., 1993, "Symbolic revision of theories with M-of-N rules", In:*Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1135–1140, Chambery, France.

BINDER, J., KOLLER, D., RUSSELL, S., et al., 1997, "Adaptive probabilistic networks with hidden variables", *Machine Learning*, v. 29, n. 2-3, pp. 213–244.

BOUTILIER, C., FRIEDMAN, N., GOLDSZMIDT, M., et al., 1996, "Context-specific independence in Bayesian networks", In:*Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 115–123.

BRATKO, I., 1986, *PROLOG: Programming for Artificial Intelligence*, Addison-Wesley.

BREESE, J., GOLDMAN, R., WELLMAN, M., 1997, "Introduction to the special section on knowledge-based construction of probabilistic and decision models", *Cybernetics*, v. 24, n. 11, pp. 1577–1579.

BUNTINE, W., 1991, "Theory Refinement on Bayesian Networks", In:*Proceedings Seventeenth Conference Uncertainty in Artificial Intelligence*, pp. 52–60, San Mateo, CA.

CASANOVA, M., GIORNO, F. A. C., FURTADO, A. L., 1987, *Programação em Lógica e a Linguagem Prolog*, EDGARD BLUCHER.

COHEN, W., 1992, "Compiling prior knowledge into an explicit bias", In:*Proceedings of the Nineth International Conference on Machine Learing*, pp. 102–110, Aberdeen, Scotland.

COOPER, G., 1990, "Computational complexity of probabilistic inference using Bayesian belief networks (research note)", *Artificial Intelligence*, v. 42, pp. 393–405.

COOPER, G., HERSKOVITS, E., 1992, "A Bayesian method for the induction of probabilistic networks from data", *Machine Learining*, v. 9, pp. 309–147.

COWELL, R. G., DAWID, P. P., LAURITZEN, S. L., et al., 1999, *Probabilistic Networks and Expert Systems*, Springer Verlag, New York.

DAGUM, P., LUBY, M., 1993, "Approximating probabilistic reasoning in Bayesian belief networks is NP-hard", *Artificial Intelligence*, v. 60, n. 1, pp. 141–153.

De RAEDT, L., 1997, "Logical Settings for Concept-Learning", *Artificial Intelligence*, v. 95, n. 1, pp. 187–201.

DEGROOT, M. H., 1987, *Probability and Statistics*, Addison-Wesley.

DEMPSTER, A., LAIRD, N., RUBIN, D., 1977, "Maximum likelihood from incomplete data", v. 39, n. Series B, pp. 1–38.

FABIAN, I., LAMBERT, D. A., 1998, "First-Order Bayesian Reasoning", In:*Proceedings Eleventh Australian Joint Conference on Artificial Intelligence*, number 1502, pp. 131–142.

FLACH, P., 1994, *Simply logical: intelligent reasoning by example.*, John Wisley and Sons Ltd.

FRIEDMAN, N., 1997, "Learning belief networks in the presence of missing values and hidden variables", In:*Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 125–133, Nashville, Tennessee.

FRIEDMAN, N., GEIGER, D., GOLDSZMIDT, M., 1997, "Bayesian Network Classifiers", *Machine Learning*, v. 29, pp. 131–163.

FRIEDMAN, N., GETOOR, L., KOLLER, D., et al., 1999, "Learning Probabilistic Relational Models", In:*Proceedings of the Sixteenth International Join Conference on Artificial Inteligence*, pp. 1300–1309, Stockholm, Sweeden.

FRIEDMAN, N., NACHMAN, I., PEéR, D., 1999, "Learning Bayesian Network Structure from Massive Datasets: The Sparse Candidate Algorithm", In:*Proceedings of the Fifteenth Conference on Uncertainty in Articial Intelligence*, pp. 206–215.

GARCEZ, A., ZAVERUCHA, G., 1999, "The Connectionist Inductive Learning and Logic Programming System", *Applied Intelligence*, v. 11, pp. 59–77.

GETOOR, L., 2001, "Multi-Relational Data Mining using Probabilistic Models Research Summary", In: *Workshop Multi-Relational Data Mining-MRDM 2001*, Freiburg, Germany.

GETOOR, L., FRIEDMAN, N., KOLLER, D., et al., 2001, "Learning Probabilistic Models of Relational Structure", In:*Proceedings of the International Conference on Machine Learning*, Williamstown, MA Morgan Kaufman, , June.

HADDAWY, P., 1999, "An overview of some recent developments on Bayesian problem solving techniques", v. 20, n. 2, pp. 11–29.

HALPERN, J. Y., 1989, "An analysis of first-order logics of probability", *Artificial Intelligence*, v. 46, pp. 311–350.

HECKERMAN, D., 1996, "A tutorial on learning with Bayesian networks", Technical Report 6, Microsoft Research.

HUGIN EXPERT (A), *Hugin Expert A/S. Hugin Help. The Net Language*, `http://www.hugin.dk/hugintro/language-pane.html`.

HUGIN EXPERT (B), *Hugin Expert A/S. Hugin API reference Manual, 2001*, `http://www.hugin.com/documentation.html`.

HUGIN EXPERT (C), *Hugin Expert A/S.*, `http://www.hugin.com`.

JAEGER, M., 1997, "Relational Bayesian Networks", In:*Proceedings of the Thirteenth Conference on Uncertainty in AI*, pp. 266–273.

JENSEN, F. V., 1996, *An introduction to Bayesian networks*, Springer Verlag, New York.

KERSTING, K., De RAEDT, L., 2000, "Bayesian Logic Programs", In:*Proceedings. of the Work-in-Progress Track at the Tenth International Conference on Inductive Logic Programming.*

KERSTING, K., De RAEDT, L., 2001a, "Adaptive Bayesian Logic Programs", pp. 104–117, Strasbourg, France.

KERSTING, K., De RAEDT, L., 2001b, "Bayesian Logic Programs", Technical Report 151, University of Freiburg, Institute for Computer Science, Freiburg, German, , April.

KERSTING, K., De RAEDT, L., 2001c, "Towards Combining Inductive Logic Programming with Bayesian Networks", pp. 118–131, Strasbourg, France.

KERSTING, K., De RAEDT, L., KRAMER, S., 2000, "Interpreting Bayesian Logic Programs", *In Working Notes of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data (SRL).*

KOLLER, D., 1999, "Probabilistic Relational Models", In:*Proceedings of the Nineth Int. Workshop on ILP*, number 1634, pp. 3–13.

KOLLER, D., PFEFFER, A., 1997, "Learning probabilities for noisy first-order rules", In:*Proceedings of the Fifteenth International Joint Conference on Artficial Intelligence (IJCAI-97)*, pp. 1316–1323.

KWOH, C. K., GILLIES, D., 1996, "Using hidden nodes in Bayesian networks", *Artificial Intelligence*, v. 88, n. 1-2, pp. 1–38.

LANGLEY, P., SIMON, H. A., 1995, "Applications of machine learning and rule induction", *Communications of the Association for Computing Machinery*, v. 38, n. 11, pp. 55–64.

LAURITZEN, S. L., 1995, "The EM algorithm for graphical association models with missing data", *Computational Statistics and Data Analysis*, v. 19, pp. 191–201.

LLOYD, J., 1989, *Foundations of Logic Programming*, 2 ed., Springer Verlag.

MCLACHLAN, G. J., KRISHNAN, T., 1997, *The EM algorithm and Extensions*, 1 ed., Wiley Interscience, New York.

MITCHELL, T., 1997, *Machine Learning*, McGraw-Hill, New York.

MUGGLETON, S., 1997, *Inductive logic programming*, McGraw-Hill, New York.

MURPHY, *Bayes Net Toolbox for Matlab. U.C. Berkeley.*, `http://www.cs.berkley.edu/murphyk/Bayes/bnt.html`.

MURPHY, K., 2001, "The Bayes Net Toolbox for Matlab", v. 33.

NGO, L., HADDAWY, P., 1995, "Probabilistic logic programming and Bayesian networks", In:*In Algorithms, Concurrency and Knowledge: Proceedings of the Asian Computing Science Conference*, pp. 286–300, Pathumthai, Thailand, December.

NGO, L., HADDAWY, P., 1997, "Answering queries from context-sensitive probabilistic knowledge bases", *Theoretical Computer Science*, v. 171, pp. 147–177.

OPITZ, D. W., SHAVLIK, J. W., 1993, "Heuristically expanding knowledge-based neural networks", In:*Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 512–517, Chambery, France.

OURSTON, D., MOONEY, R. J., 1994, "Theory refinement combining analytical and empirical methods", *Artificial Intelligence*, v. 66, pp. 311–344.

PEARL, J., 1991, *Reasoning in Intelligent Systems: Networks of Plausible Inference*, 2 ed., Morgan Kaufman.

POOLE, D., 1993, "Probabilistic Horn abduction and Bayesian networks", *Artificial Intelligence*, v. 64, n. 1, pp. 81–129.

POOLE, D., 1998, "Learning, Bayesian Probability, Graphical Models, and Abduction", *In: Abduction and Induction: essays on their relation and integration.*

POOLE, D., 2000, "Abducing Through Negation as Failure: Stable models within the independent choice logic", *Journal of Logic Programming*, v. 44, pp. 5–35.

PRADHAM, M., DAGUM, P., 1996, "Optimal Monte Carlo estimation of belief network inference", In:*Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 446–453.

QUINLAN, J. R., 1993, *C4.5:Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CP.

RAMACHANDRAN, S., MOONEY, R., 1998, "Theory Refinement of Bayesian Networds with Hidden Variables", In:*Proceedings of the fifteenth International Conference on Machine Learning (ICML)*, pp. 454–462.

RAMONI, M., SEBASTIANI, P., 1997, "Learning Bayesian networks from incomplete databases", In:*Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence* Morgan Kaufmann.

REVOREDO, K., ZAVERUCHA, G., 2001, "Theory Refinement of Bayesian Logic Programs", In:*Eigth International Conference on Neural Information Processing (ICONIP)*, pp. 1088–1092, Shanghai, China.

REVOREDO, K., ZAVERUCHA, G., 2002, "Revision of First-Order Bayesian Classifiers", In:*to appear in the Twelfth International Conference on Inductive Logic Programming.*

RICHARDS, B. L., MOONEY, R. J., 1995, "Automated Refinement of First-Order Horn-Clause Domain Theories", *Machine Learning*, v. 19, pp. 95–131.

RUSSELL, S., NORVIG, P., 1995, *Artificial intelligence: A modern approach*, Englewook Cliffs, NJ: Prentice-Hall.

RUSSELL, S., NORVIG, P., 2000, *Artificial intelligence: A modern approach*, chapter 15,16. Draft version, 2 ed.

SINGH, M., 1997, "Learning Bayesian networks from incomplete data", In:*Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pp. 534–539, Providence, Rhode Island.

STERLING, L., SHAPIRO, E., 1986, *The Art of Prolog: Advanced programming Techniques*, The MIT Press.

TOWELL, G., SHAVLIK, J., 1994, "Knowledge-Based Artificial Neural Networks", *Artificial Intelligence*, v. 70, n. 1–2, pp. 119–165.

WOGULIS, J., 1994, *An Approach to Repairing and Evaluationg First-Order Theories Containing Multiple Concepts and Negation*, Ph.D. thesis, University of California, Irvine, CA.

WOGULIS, J., PAZZANI, M., 1993, "A methodology for evaluationg theory revision systems: Results with Audrey II", In:*Proceedings of the Thirteenth International Join Conference on Artificial Intelligence*, pp. 1128–1134, Chambery, France.

WROBEL, S., 1996, "First-order Theory Refinement", *Advances in Inductive Logic Programming*, pp. 14–33.

```
call_query(Q, Theory, Arq) :-
    erase_module(pdom),
    compile(Arq), assert(current_threory(Theory)), query([Q],[]),
    retract(current_threory(Theory)).
```

O arquivo "pdom"contém a descrição do domínio na mesma formatação especificada no apêndice B. Como o BLP inicial foi recebido e compilado, não podemos compilar um arquivo com uma nova [2] descrição para a componente qualitativa. Por este motivo sua componente qualitativa não é definida neste arquivo e sim enviada como parâmetro (Theory) para o predicado *call_query*. Arq contém o nome do arquivo "pdom"que será recompilado.

O predicado prove/3 também precisou ser modificado, já que o BLP corrente agora é recebido como parâmetro.

```
prove(true,P,P):- !.
prove([],P,P) :- !.
prove([GA|GB],OP,NP) :- !,
    prove(GA,OP,PA), prove(GB,PA,NP).
prove(G,OP,[G-B|P]) :-!,
    current_theory(Theory), v_resolve(Theory,G, B),
    prove(B,OP,P).


v_resolve(_,Goal, true) :-
    built_in(Goal), !.
v_resolve(_, Goal,true) :-
    current_predicate(_, fdt:Goal), !,
    call(fdt:Goal).
v_resolve([predicate(Template, Clauses1, Clauses2)|_], Head, Body) :-
    no_bindings((Template = Head)), !,
    select_clause(Clauses1, Clauses2, Popsicle),
    thaw(Popsicle, Clause),
```

[2] A componente qualitativa será a mesma na primeira vez que executarmos este procedimento, mas quando uma modificação para esta for proposta ela já passará a ser outra.

```
    unify(Clause, [Head|Body]).
v_resolve([_|Preds], Head, Body) :-
    v_resolve(Preds, Head, Body).

built_in(true). built_in(fail). built_in(_ is _). built_in(_ = _). built_in(_ \ = _).
built_in(_ == _). built_in(_ \ == _). built_in(_ ; _). built_in(_ > _). built_in(_ =< _).
built_in(_ >= _).
```

Se a rede Bayesiana para a consulta em questão for construída, então, uma ordem de numeração para os nós desta rede, *toposort/3*, e a identificação dos nós pais para cada um dos nós, *dependency_list/2*, precisa feita para que o arquivo com a estrutura da rede Bayesiana, *create_network_matlab/2*, e as CPDs, *create_cpd_matlab/2*, possa ser criado. Com isso o predicado *solution_graph/2* também teve que ser alterado. O arquivo gerado utiliza a formatação do Bayes Net Toolbox (MURPHY).

```
solution_graph([],_).
solution_graph([G|Gs],S) :-
    prove(G,[],P), assertz(proof(P)), topsort(P,[],OrdP), dependency_list(P,LP),
    open('Exemplo.m',append,File),
    create_network_matlab(LP, OrdP,File), create_cpd_matlab(LP, OrdP,File),
solution_graph(Gs,S),
    assertz(proof(end)), collect(US), sort(US,S), !.

topsort([],Ord,OrdI):-
    invert_order(Ord,[],OrdI),!.
topsort(L0,Ord0,Ord) :-
    pick_sort(L0,Ln,Ord0,Ord1),
    topsort(Ln,Ord1,Ord).

pick_sort([],[],Ord,Ord):-!.
pick_sort([(A-true)|T],Nt,Ord0,Ordf) :-
    pick_sort(T,Nt,[A|Ord0],Ordf).
```

```
pick_sort([(A-Parents)|T],Nt,Ord0, OrdF) :-
    my_member(Ord0, Parents), !,
    pick_sort(T,Nt,[A|Ord0],OrdF).
pick_sort([(A-Parents)|T],[A-Parents|Nt],Ord0,Ordf) :-
    pick_sort(T,Nt,Ord0,Ordf).

my_member([],_) :- !, fail.
my_member(_,[]) :- !.
my_member(L0, [H|T]) :-
    member(H, L0),!,
    my_member(L0,T).

invert_order([],L,L) :- !.
invert_order([H|T],Lp,Lf) :-
    inverte_ordem(T,[H|Lp],Lf).

dependency_list([],[]):-!.
dependency_list([(A-true)|T],[[A,[]]|DL]) :- !,
    dependency_list(T,DL).
dependency_list([(A-Parents)|T],[[A,Parents]|DL]) :-
    dependency_list(T,DL).

create_network_matlab(L, OrdL, Filename) :-
    network_matlab(L, OrdL,Filename,1).
create_cpd_matlab(L, OrdL, Filename) :-
    network_matlab(L, OrdL,Filename,2).
network_matlab([],_,_,_):-!.
network_matlab([H|T],OrdL, File, Type) :-
    network_matlab2(H,OrdL, File, Type),
    network_matlab(T,OrdL, File, Type).
network_matlab2([H|[T]],OrdL, File,1) :-
```

nth1(Pos,OrdL,H),

write_network_matlab(T,Pos,OrdL, File).

network_matlab2([H|[T]],OrdL, File, 2) :-

cpd1(H,T,Cpd),

nth1(Pos,OrdL,H),

format(File, 'bnet.CPD$\tilde{p}$ = tabular_CPD(bnet, $\tilde{p}$, $\tilde{p}$); n',[Pos,Pos, Cpd]).

write_network_matlab([],_,_,_) :-!.

write_network_matlab([Parent|Parents],PosH,OrdL, File) :-

nth1(Pos,OrdL,Parent),

format(File, 'dag($\tilde{p}$,$\tilde{p}$)=1; n',[Pos,PosH]),

write_network_matlab(Parents,PosH,OrdL,File).

Considerando a consulta **P**(esposa(alice,art) | gênero(alice,feminino)=V, casado(alice,art)=V), o predicado *topsort/3* retorna a seguinte lista:

$$L = [genero(alice, feminino), casado(alice, art), esposa(alice, art)]$$

A dependência entre os nós da rede é definida da seguinte maneira:

$[[esposa(alice, art), [genero(alice, feminino), casado(alice, art)]],$

$[genero(alice, feminino), []], [casado(alice, art), []]]$

Vamos considerar o mapeamento:

genero(alice,femminino) = 1;

casado(alice,art)=2;

esposa(alice,art) = 3;

A rede Bayesiana construída terá a seguinte formatação:

N = 3;

dag = zeros(N,N);

dag(1,3) =1; dag(2,3) = 1;

*ns = 2*ones(1,N);*

bnet = mk_bnet(dag, ns);

bnet.CPD1 = tabular_CPD(bnet, 1, [0.05, 0.95]);

bnet.CPD2 = tabular_CPD(bnet, 2, [0.01, 0.90]);

bnet.CPD3 = tabular_CPD(bnet, 3, [0.99,0.99,0.99,0.65,0.01,0.01,0.01,0.35]);

```
engine = enumerative_inf_engine(bnet);
evidence = cell(1,N);
evidence1 = 2;
evidence2 = 2;
[engine, ll] = enter_evidence(engine, evidence);
[m,ll] = marginal_nodes(engine,[1 2]);
```

São definidos n arquivos como este, onde n é o número de redes Bayesianas construídas.

A implementação deste processo foi feita em Quintus Prolog Release 3.4.

A.2 Processo - Aprende CPDs

O processo Aprende as CPDs recebe, então, as Redes Bayesianas construídas no processo anterior e os exemplos que foram definidos como sendo falhos.

Para aprender as CPDs, o EM adaptado, está sendo implementado em MatLab, com o intuito de utilizar as ferramentas de inferência do Bayes Net Toolbox (MURPHY), (MURPHY, 2001). A função mostrada abaixo ainda precisa ser parametrizada.

```
[Num_network,Num_regras] = retorna_parametros;
Prob_true = 0; Prob_false = 0; Prob_Pais_true = 0; Prob_Pais_false = 0;
for k=1:Num_network
   Example_name = strcat('exemploteseEM',num2str(k));
   [m,regra,dag,num_nos_rede, evidence,bnet, cabeca_regra] =
                              feval(Example_name,1,1,Num_regras);
   no_cabecaAux = cabeca_regra(1);
   no_cabeca = no_cabecaAux{1};
   nos = regra{1};
   if isempty(nos)==0
      qtd_nos = length(nos);
      for j=1:qtd_nos
         num_nos = quantos_nos(dag,num_nos_rede,nos(j));
         vetor_consulta_pais = cria_vetor(num_nos,dag,num_nos_rede, nos(j),2);
```

```
        if isempty(vetor_consulta_pais) == 0
            eh_evidencia = verifica_evidencia(evidence,vetor_consulta_pais);
            if eh_evidencia ==2
                Prob_Pais_atual_true = 1;
                Prob_Pais_atual_false = 0;
            else
                marginal = feval(Example_name,3,vetor_consulta_pais,Num_regras);
                Prob_Pais_atual_true = marginal.T(2);
                Prob_Pais_atual_false = marginal.T(1);
            end
            Prob_Pais_true = Prob_Pais_true + Prob_Pais_atual_true;
            Prob_Pais_false = Prob_Pais_false + Prob_Pais_atual_false;
        end
        num_nos = num_nos + 1;
        vetor_consulta = cria_vetor(num_nos,dag,num_nos_rede, nos(j),1);
        num_nos_regra = length(vetor_consulta);
        if evidence{nos(j)} == 2
            Prob_true = Prob_true + 1;
            Prob_false = Prob_false + 0;
        else
            marginal = feval(Example_name,3,vetor_consulta,Num_regras);
            Prob_true = Prob_true + marginal.T(2);
            Prob_false = Prob_false + marginal.T(1);
        end
    end
  end
end
Prob_regra_true = Prob_true/Prob_Pais_true;
Prob_regra_false = Prob_false/Prob_Pais_true;
[Prob_regra_normalizada_true,Prob_regra_normalizada_false]
                        = normaliza(Prob_regra_true,Prob_regra_false);
```

As funções auxiliares utilizadas pela função principal estão descritas abaixo.

```
function num_nos = quantos_nos(dag, num_nos_rede, no_cabeca)
num_nos = 0;
for i=1:num_nos_rede
   if dag(i,no_cabeca) == 1
      num_nos = num_nos + 1;
   end
end

function vetor = cria_vetor(tamanho, dag, num_nos_rede, no_cabeca, Tipo)
vetor = 1:tamanho;
k = 1;
if tamanho ¿ 1
   for i=1:num_nos_rede
      if dag(i,no_cabeca) == 1
         vetor(k) = i;
         k = k + 1;
      end
   end
end
if Tipo == 1
   vetor(k) = no_cabeca;
end

function  eh_evidencia = verifica_evidencia(evidence,consulta)
qtd = length(consulta);
eh_evidencia = 2;
for j=1:qtd
   if isempty(evidenceconsulta(j))
      eh_evidencia = 1;
      return;
   end
```

end

function *[Prob_true,Prob_false] = normaliza(Prob_inicial_true,Prob_inicial_false)*

soma = Prob_inicial_true + Prob_inicial_false;

alpha = 1/soma;

*Prob_true = alpha*Prob_inicial_true;*

*Prob_false = alpha*Prob_inicial_false;*

Após o processamento do aprendizado das CPDs, se o pontos que falharam em classificação passarem a ser classificados corretamente, o BLP com as CPDs modificadas é retornado para o usuário e o RBLP_AP termina, caso contrário o processo Revisa Estrutura é chamado, onde este recebe os exemplos que foram classificados como falhos no processo Constrói rede Bayesianas.

A.3 Processo - Revisa Estrutura

Para revisar a componente qualitativa do BLP modificamos alguns predicados importantes do sistema FORTE e que são parte fundamental no processo de revisão.

O principal predicado do algoritmo FORTE é *forte/2*, que espera receber uma teoria inicial e tem por objetivo retornar uma teoria revista. Este predicado chama *forte1/3*, que é um predicado recursivo, parando quando nenhuma outra modificação proposta na teoria, proporcionar melhora para esta.

Para o RBLP_AP dois predicados chamados de dentro do *forte1/3* precisam ser adaptados, *revisions/4* é um deles. Ele espera receber o conjunto de pontos de revisão, ordenados do com maior potencial para o de menor potencial, a maior pontuação encontrada até o momento e a teoria inicial, retornando a melhor revisão para cada um dos pontos de revisão. Este predicado é recursivo, ele analisa cada um dos pontos até que uma modificação proposta em um deles proporcione uma pontuação maior do que o potencial do próximo a ser analisado. Este é um dos motivos pelo qual este predicado precisa ser modificado, pois no RBLP_AP todos os pontos de revisão são analisados, já que probabilisticamente não é possível garantir que um ponto que tenha um potencial menor do que a pontuação lógica do maior ponto analisado até o momento, não vá permitir que o BLP tenha um desempenho

probabilístico melhor.

Por este motivo o teste

$$Potential >= Best\_so\_far,$$

foi retirado do predicado, pois quando esta consideração não era satisfeita, os próximos pontos de revisão não eram analisados.

O predicado *revisions/4* chama todos os operadores para cada ponto de revisão. Com a aplicação dos operadores a especificação das CPDs de cada uma das cláusulas deve ser altera. Por exemplo, se o operador Adiciona uma Nova Cláusula for aplicado, a CPD desta deve ser inicializada randômicamente, para que o procedimento de aprendizado das CPDs seja executado. Em vista disso, alteramos o predicado /it revise_theory/4, para *revise_probabilistic_theory/4*

```
revise_probabilistic_theory(Olds, News, Old_thy, New_thy) :-
   read_cpd(Old_Prob_Thy),
   revise_probabilistic_theory1(Olds, News, Old_thy, New_thy, Old_Prob_Thy,
      New_Prob_Thy),
   write_cpd(New_Prob_Thy).

revise_probabilistic_theory1([], [], Thy, Thy, Prob_Thy, Prob_Thy).
revise_probabilistic_theory1([Old|Olds], [New|News], Old_thy, New_thy,
      Old_Prob_Thy, New_Prob_Thy) :-
   revise_probabilistic_theory2(Old, New, Old_thy, Temp_thy,
      Old_Prob_Thy, Temp_Prob_Thy),
   revise_probabilistic_theory1(Olds, News, Temp_thy, New_thy, Temp_Prob_Thy,
      New_Prob_Thy).

revise_probabilistic_theory2(Old_pred, New_pred, Old_theory, New_theory,
      Old_prob_theory, New_prob_theory) :-
   pe_theory([New_pred], Temp),
   de_theory(Temp, New),
   ( New == []
      -> Add_pred = none
```

```
    ; New = [Add_pred]
    ),
  once(revise_probabilistic_theory3(Old_pred, Add_pred, Old_theory, New_theory,
    Old_prob_theory, New_prob_theory)).

revise_probabilistic_theory3(none, New_pred, Old_theory, [New_pred|Old_theory],
    Old_prob_theory, New_prob_theory) :- !,
  add_prob_theory(New_pred,Old_prob_theory,New_prob_theory).
revise_probabilistic_theory3(Old_pred, none, Old_theory, New_theory,
    Old_prob_theory, New_prob_theory) :- !,
  delete( Old_theory, Old_pred, New_theory),
  find_probabilistic_predicate(Old_pred, Old_prob_theory, New_prob_theory).
revise_probabilistic_theory3(Old_pred, New_pred, Old_theory, New_theory,
    Old_prob_theory, New_prob_theory) :-
  select(Old_pred, Old_theory, New_pred, New_theory),
  find_probabilistic_predicate(Old_pred, Old_prob_theory, Aux_prob_theory),
  add_prob_theory(New_pred,Aux_prob_theory,New_prob_theory).

read_cpd(CPDs) :-
  concat_atom(['Family', '.pcpd'], Cpd_filename),
  tt:read_cpd_file(Cpd_filename, CPDs).

write_cpd(Prob_theory) :-
  concat_atom(['Family', '.pcpd'], Cpd_filename),
  open(Cpd_filename, write, File),
  tt:write_prob_theory(File,Prob_theory),
  close(File).

find_probabilistic_predicate(Old_pred, Old_prob_theory, New_prob_theory) :-
  concat_atom(['Family', '.pcpd'], Cpd_filename),
  tt:delete_old_predicate(Cpd_filename, Old_pred, Old_prob_theory, New_prob_theory).

add_pred_theory([],_, _).
```

```
add_prob_theory(predicate(Template, Clauses, _),Aux_prob_theory,
    New_prob_theory) :-

add_prob_theory1(Template, Clauses, Aux_prob_theory, New_prob_theory).
add_prob_theory1(_,[],New_prob_theory, New_prob_theory).
add_prob_theory1(Template, [clause(Clause)|Clauses], Aux_prob_theory,
    New_prob_theory) :-
  length(Clause, Length_clause),
  ( Length_clause == 1
    -> compose_list_prob(0.99,[0.01],Cpds)
    ; calculate_num_exp(2, Length_clause, 1, Num_prob),
    Actual_num_prob is Num_prob - 1,
    search_cpd(Actual_num_prob, [0.99], Cpds)
    ),
  New_pred = predicate(Template,clause(Clause),Cpds),
  add_prob_theory1(Template,Clauses, [New_pred—Aux_prob_theory],
    New_prob_theory).

search_cpd(1,Aux_cpds,[0.99|Aux_cpds]).
search_cpd(Num_prob, Aux_cpds, Cpds) :-
  Actual_num_prob is Num_prob -1,
  search_cpd(Actual_num_prob, [0.01|Aux_cpds], Cpds).
  compose_list_prob(Prob,Current_list, [Prob|Current_list]).

calculate_num_exp(Base, 1, Parcial_num_prob, Total_num_prob) :-
  Total_num_prob is Parcial_num_prob * Base.
calculate_num_exp(Base, Length_clause, Parcial_num_prob, Total_num_prob) :-
  Current_length_clause is Length_clause - 1,
  Parcial_num_prob1 is Parcial_num_prob * Base,
  calculate_num_exp(Base, Current_length_clause, Parcial_num_prob1,
    Total_num_prob).
```

Os operadores precisam ser avaliados. Como vimos, nos alteramos a função de

avaliação do FORTE para ser uma função de avaliação probabilística, logo o predicado *calc_score/4* responsável por buscar a pontuação do operador sendo considerado precisava ser alterado *calc_score_probabilistic/4*.

```
calc_score_probabilistic(Revision, Theory, Correct, Incorrect):-
    Revision = revision(Score, New_preds, Old_preds, _),
    revise_probabilistic_theory(Old_preds, New_preds, Theory, New_theory),
    prove_instances(Correct, New_theory, _, Wrong),
    prove_instances(Incorrect, New_theory, Right, _),
    Improvement is Right - Wrong,
    calc_size(New_preds, New_size),
    calc_size(Old_preds, Old_size),
    Compaction is Old_size - New_size,
    prove_instances_probabilistic(Incorrect, New_theory, Right, _, Likelihood),
    /* Aqui chama o aprendizado dos parâmetros e que também retornará a likelihood */
    Score = score(Improvement, Compaction, Likelihood).

theory_accuracy_probabilistic([], _, _, [], [], Correct, Correct, Incorrect,
                              Incorrect, _, Likelihood, Likelihood).
theory_accuracy_probabilistic([Instance|Instances], Theory, Depth_limit,
        [Instance|Provable], Unprovable,Cum_correct, Num_correct, Cum_incorrect,
        Num_incorrect, NumExample, Cum_Likelihood, Likelihood) :-
    assert_instance(Instance),
    Instance =.. [Type, Assertion, _],
    call_query(Assertion, Theory , 'dominio.pdom'),
    New_correct is Cum_correct + 1,
    New_incorrect = Cum_incorrect
    theory_accuracy_probabilistic(Instances, Theory, Depth_limit, Provable,
        Unprovable, New_correct, Num_correct, New_incorrect,
        Num_incorrect, New_NumExample, New_Likelihood, Likelihood).
theory_accuracy_probabilistic([Instance|Instances], Theory, Depth_limit,
```

[Instance|Provable], Unprovable, Cum_correct, Num_correct,Cum_incorrect,

Num_incorrect, NumExample, Cum_Likelihood, Likelihood) :-

Instance =.. [Type|_],

New_incorrect is Cum_incorrect + 1, New_correct = Cum_correct

theory_accuracy_probabilistic(Instances, Theory, Depth_limit, Provable,

Unprovable, New_correct, Num_correct, New_incorrect,

Num_incorrect, New_NumExample, Cum_Likelihood, Likelihood).

Um outro predicado chamado por *revisions/4* é o *best_new_revision/2*, utilizado para determinar a melhor revisão para o ponto de revisão sendo considerado. Ele mantém a revisão de melhor pontuação e compara-a com a próxima revisão proposta. *greater_score/2* é utilizado então para fazer esta comparação. Como nós propomos uma nova função de avaliação para determinar o melhor operador, criamos um novo predicado,*greater_probabilistic_score/2*, para substituir o *greater_score/2*

greater_probabilistic_score(score(Accuracy1, _,Likelihood1), score(_, _,Likelihood2)) :-

Accuracy1 > 0,

Likelihood1 > Likelihood2, !.

greater_probabilistic_score(score(Accuracy1, Compaction1,Likelihood1),

score(_, _,Likelihood2)):-

Accuracy1 =:= 0,

Compaction1 > 0,

Likelihood1 > Likelihood2, !.

greater_probabilistic_score(score(Accuracy1, _,Likelihood1),

score(Accuracy2, _,Likelihood2)):-

Likelihood1 = Likelihood2,

Accuracy1 > Accuracy2,!.

greater_probabilistic_ score(score(Accuracy1, Compaction1, Likelihood1),

score(Accuracy2, Compaction2, Likelihood2)):-

Likelihood1 = Likelihood2,

Accuracy1 = Accuracy2,

Compaction1 > Compaction2.

A *Accuracy* considerada é a melhora lógica, logo apenas avaliaremos os operadores que estiverem proporcionando alguma melhora lógica, dessa maneira podemos garantir, assim como o FORTE, que o BLP final é consistente com conjunto de exemplos de treinamento.

Tendo a melhor revisão para cada um dos pontos de revisão, estas revisões serão ordenadas decrescentemente e a primeira, conseqüentemente a melhor, será escolhida e se realmente proporcionar melhora para a teoria será implementada.

Se não for mais possível proporcionar melhora lógica ao BLP corrente, a revisão termina e o BLP revisto é retornado para o Usuário.

Outros predicados que são auxiliares para o desenvolvimento da revisão também foram implementados.

Apêndice B

Implementação do Procedimento de Resposta a uma Consulta

Em (KERSTING, De RAEDT, 2001b) uma implementação feita em Prolog da primeira fase (computar a rede Bayesiana para uma determinada consulta) do procedimento de resposta a uma consulta de um BLP é apresentado. Nós a reproduziremos aqui. O interpretador tem por objetivo construir a rede suporte de acordo com um BLP e uma consulta fornecidos. Esta rede suporte será escrita na linguagem do software HUGIN ((HUGIN EXPERT (A)), (HUGIN EXPERT (B))), onde uma versão de demonstração pode ser obtida em (HUGIN EXPERT (C)). Esta rede será guardada no arquivo 'out.net', que depois de executado no HUGIN permitirá a execução do segundo passo do procedimento de resposta a uma consulta, que é o passo de inferência e determinação da distribuição de probabilidade para a consulta feita. O coração do interpretador (prove/3) é uma adaptação de um meta interpretador Prolog que pode ser encontrado em livros introdutórios de Prolog como em (STERLING, SHAPIRO, 1986), (BRATKO, 1986), e (FLACH, 1994). O código foi desenvolvido em SICStus 3.8.1.

O arquivo com a definição do BLP deve ter a seguinte formatação:

Um predicado *domain* que define os predicados que fazem parte do BLP que estiver sendo considerado. Nós utilizaremos o problema da família para exemplificar.

domain(gênero/2,discrete,[true,false]).

domain(casado/2,discrete,[true,false]).
domain(mãe_pai/2,discrete,[true,false]).
domain(esposa/2,discrete,[true,false]).
domain(tio/2,discrete,[true,false]).
domain(irmão_irmã/2,discrete,[true,false]).
domain(tia_tio/2,discrete,[true,false]).

O primeiro parâmetro do predicado *domain* indica quais os predicados Bayesianos que fazem parte do BLP da família, o segundo parâmetro indica qual o tipo de domínio desses predicados, que no nosso exemplo são todos discretos, e por ultimo o domínio destes predicados. p/N indica que o predicado p tem aridade N.

O predicado *combining_rule* determina qual a regra de combinação será utilizada para cada um dos predicados Bayesianos do BLP considerado.

combining_rule(gênero/2,id).
combining_rule(casado/2,id).
combining_rule(mãe_pai/2,id).
combining_rule(esposa/2,id).
combining_rule(tio/2,id).
combining_rule(irmão_irmã/2,id).
combining_rule(tia_tio/2,id).

Para o BLP da família as regras de combinação utilizadas serão a identidade.

O aspecto qualitativo do BLP da família pode ser definido como segue

gênero(bob, masculino). *gênero(alice,feminino).* *gênero(john,masculino).*
casado(alice,art). *mãe_pai(ann,kari).* *mãe_pai(ann,helen)*
mãe_pai(kari,jane). *mãe_pai(ann,lisa).* *casado(john,lisa)*
casado(bob,helen).

esposa(A, B) | gênero(A, feminino), casado(A, B).

irmão_irmã(A, B) | mãe_pai(C, A), mãe_pai(C, B).
tia_tio(X,Y) | casado(X,Z), irmão_irmã(Z,W), mãe_pai(W,Y).
tio(X,Y) | gênero(X,masculino), tia_tio(X,Y).

As distribuições de probabilidade condicionais são especificadas no predicado cpd/2.

cpd(gênero(_,_),[0.95,0.05]).
cpd(casado(_,_),[0.90,0.10]).
cpd(mãe_pai(_,_),[0.85,0.15]).

cpd((esposa(A, B) | (gênero(A, feminino), casado(A, B))),
[0.99,0.99,0.99,0.65,0.01,0.01,0.01,0.35]).
cpd((tia_tio(X,Y) | casado(X,Z), irmão_irmã(Z,W), mãe_pai(W,Y))),
[0.99,0.99,0.99,0.99,0.99,0.99,0.99,0.03,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.95]).
cpd((irmão_irmã(A, B) |(mãe_pai(C, A), mãe_pai(C, B))),
[0.99,0.99,0.99,0.03,0.01,0.01,0.01,0.97]).
cpd(tio(X,Y) | gênero(X,masculino), tia_tio(X,Y))),
[0.99,0.99,0.99,0.03,0.01,0.01,0.01,0.90]).

O primeiro argumento corresponde a cláusula Bayesiana e o segundo a distribuição de probabilidade desta.

Agora reproduziremos o interpretador BLoP, responsável pela construção da rede Bayesiana para uma determinada consulta. O interpretador pode ser inicializado digitando *blop_shell* no promt do seu Prolog. Um prompt interno será criado. Primeiramente o arquivo com a definição do BLP deve ser fornecido para que este possa ser consultado, logo para o nosso exemplo forneceremos o arquivo, ['family.blp']. Depois de compilado com sucesso podemos fazer a consulta, por exemplo *tio(bob,jane).*

O esqueleto do algoritmo apresentado em 2.10 é representado pelo predicado *query/2.*

```
query(Q,E) :-
        retractall(proof(_)), retractall(network(_)),
        assert(proof([])),
        evidence_variables(E,EVars), append(Q,EVars,Vars),
        solution_graph(Vars,S),
        support_network(S,SNC),
        apply_comb(SNC,SN), asserta(network(SN)),
        write_support_network(SN).

evidence_variables([],[]).
evidence_variables([V=_|Vs],[V|Vars]) :-
        evidence_variables(Vs,Vars).
```

Depois de extrair as variáveis randômicas que ocorrem nas evidências, *evidence_variable(E,EVars)*, *query/2* computa a grafo solução S para as variáveis Vars chamando o predicado *solution_graph(Vars,S)*.

```
prove(true,P,P):- !.
prove((GA,GB),OP,NP) :- !,
    prove(GA,OP,PA), prove(GB,PA,NP).
prove(G,OP,[G-B|P]) :-
    clause(G,B), prove(B,OP,P).

solution_graph([G|Gs],S) :-
    prove(G,[],P), assertz(proof(P)),fail;
    solution_graph(Gs,S);
    assertz(proof(end)), collect(US), sort(US,S).

collect(L) :-
    retract(proof(X)),!,
    (X==end,!,L=[];
    collect(R), append(X,R,L)).
```

O procedimento *solution_graph/2* é parecido com o *findall/3* definido em

(BRATKO, 1986). O resultado S é uma lista Prolog ordenada de arcos A-B de um nó ou A para um nó e B no grafo solução de Vars. Para computar os caminhos bem sucedidos utiliza-se o predicado *prove/3* descrito em (STERLING, SHAPIRO, 1986). Este retorna a lista de caminhos bem sucedidos no ultimo parâmetro. Tendo o grafo solução S, o predicado *support_network/2* procura as CPDs associadas a cada cláusula utilizada para construir S:

```
support_network([],[]).
support_network([N-Pa|Es],[(N,T,D,[Parents|Pas],[CPD|CPDs])|SN]) :-
        functor(N,P,NA), domain(P/NA,T,D),
        cpd(N,Pa,Parents,CPD),
        support_network(N,Es,Pas,CPDs,Rest),
        support_network(Rest,SN).
support_network(_,[],[],[],[]).
support_network(N,[N-Pa|R],[Parents|Pas],[CPD|CPDs],Rest) :-
        cpd(N,Pa,Parents,CPD),
        support_netwok(N,R,Pas,CPDs,Rest).
support_network(_,R,[],[],R).
```

O ultimo passo é computar as distribuições de probabilidade condicionais combinadas. Isto é feito utilizando o predicado *apply_comb/2*:

```
apply_comb([],[]).
apply_comb([F|Fs],[CF|CFs]) :-
        combine(F,CF),
        apply_comb(Fs,CFs).

combine((N,T,D,LPa,LCPDs), (N,T,D,Pa,CPD)) :-
        functor(N,P,NA),
        clause(combining_rule(P/NA,CR),_),
        C=..[CR,LPa,LCPDs,Pa,CPD],
        call(C).
```

A regra de combinação correspondente ao predicado P é chamada por *call(C)*. Uma regra de combinação cr é vista como um predicado prolog *cr(PaL,CPDL,*

Pa, CPD). O argumento PaL é a lista do conjunto de pais. O argumento CPDL especifica a CPD correspondente. Então, o i-ézimo elemento de CPDL são as CPDs associadas ao i-ezimo conjunto de pais PaL. O conjunto de pais resultantes e as CPDs combinadas são retornadas em Pa e CPD.

Depois de terminado todos os passos, a rede suporte é fornecida na variável SN. É escrita para um arquivo chamado 'out.net'. O resto do código implementa um prompt onde espera-se receber o arquivo com a teoria, que será utilizada para encontrar a rede Bayesiana, e a consulta.

```
:-use_module(library(listing)).
:-use_module(library(charsio)).
:-op(500,xfy,'|').
:-dynamic domain/3, combining_rule/2,network/1.

term_expansion((Head|Body1,Body2), (Head:-Body1,Body2)).
term_expansion((Head|Body),(Head:-Body)).
term_expansion(domain(A/N,T,D),(:-dynamic A/N)) :-
        retractall(domain(A/N,_,_)),assert(domain(A/N,T,D)).
term_expansion(combining_rule(A/N,B),(:-dynamic A/N)) :-
        retractall(combining_rule(A/N,_)),assert(combining_rule(A/N,B)).
term_expansion(cpd((H|B),CPD),cpd(H,B,BL,CPD)) :-
        conj_2_list(B,BL).
term_expansion(cpd(H,CPD),cpd(H,true,[],CPD)) :- !.

conj_2_list((A,B),L ) :-
        conj_2_list(A,LA),
        conj_2_list(B,LB),
        append(LA,LB,L).
conj_2_list((A),[A]).  Codificação do promt
blop_shell :-
        assert(network([])),blop_shell_help,blop_shell(next).
blop_shell(next) :- !,
```

```
    blop_shell_promt, char_conversion('|',';'),
    read(Goal),char_conversion('|','|'),blop_shell(Goal).
blop_shell(exit).

blop_shell_help :- !,nl,
write(' ************************************************************ '),nl,
write(' *          Blop - Interpretador de programa em lógica Bayesiano          *'),nl,
write(' ************************************************************ '),nl,
write('*ajuda.                          |+ esta mensagem                  *'),nl,
write('*Q1,...,Qn.                      |+rede suporte para computar      *'),nl,
write('*                                | p(Q1,...,Qn)                    *'),nl,
write('*Q1,...,Qn|E1 = e1,...,EM = eM.  |+ rede suporte para computar *'),nl,
write('*                                | p(Q1,...,Qn|E1 = e1,...,EM = eM) *'),nl,
write('*["familia.blp"].                |+ consulta arquivo BLP"familia.blp"*'),nl,
write('sair.                            |+ sai do promot                  *'),nl,
write(' ************************************************************ '),nl,

blop_shell(help) :-
    blop_shell_help, blop_shell(next).
blop_shell([X]) :-
    consult(X), !, blop_shell(next).
blop_shell((Q;E)) :- !,
    query([Q],[E]),blop_shell(next).
blop_shell(Q) :- !,
    query([Q],[]),blop_shell(next).
blop_shell(_) :- !,
    format('~n Comando desconhecido !~n~n',[]),blop_shell(next).

blop_shell_promt :-
    write('<BLop>?- '),flush_output(user_output).
```

Regra de combinação identidade

```
id([Pa],[CPD],Pa,CPD).
```

Escrevendo a rede suporte na linguagem HUGIN.

```
write_support_network(SN) :-
    flush_output(user_output), tell('out.net'),
    format('net~n {~~n node_size = (100 40);~n }~n~n',[]),
    write_nodes(SN), write_potentials(SN), told.

write_nodes([]).
write_nodes([(N,T,D,_,_)|R]) :- write_node(N,T,D),write_nodes(R).

write_node(N,continuous,_) :- format('nó continuo~ @~ n {~ n label="~ q";~ n }~
n~n ',[write_no_brackets(N),N].
write_node(N,discrete,D):- format('nó discreto~ @~ n {~ n states=(~ @);~ n label="~
q";~ n }~n~n ',[write_no_brackets(N),write_ddomain(D),N].

write_ddomain([]).
write_ddomain([D|Ds]) :- format('"~ q" ',[D]), write_ddomain(Ds).

write_potentials([]).
write_potentials([(N,continuous,_,Pa,CPD)|R]) :-
    format('potenciais (~ @~ @ )~n {~n data=(~ @);~n {~n~n ',
    [write_no_brackets(N), write_cond(Pa), write_continuous_cpd(CPD)]),
    write_potentials(R).
write_potentials([(N,discrete,_,Pa,CPD)|R]) :-
    format('potenciais (~ @~ @ )~n {~n data=(~ @);~n {~n~n ',
    [write_no_brackets(N), write_cond(Pa), write_list(CPD)]),
    write_potentials(R).

write_cond([]).
write_cond(P) :- format('|~ @',[write_list(P)]).
```

```
write_continuous_cpd([]).
write_continuous_cpd([normal(M,V)|Ps]) :-
        format('normal(~ @,~ @)', [ write_no_brackets(M),write_no_brackets(V)]),
        write_continuous_cpd(Ps).
write_list([]).
write_list([P|Ps]) :-
        format('~ @',[write_no_brackets(P)]), write_list(Ps).

write_no_brackets(T) :-
        write_to_chars(T,C), delete(C,40,C1), delete(C1,41,C2),
        substitute(44,C2,95,NT), name(N,NT), write(N).
```

Apêndice A

Implementação do RBLP_AP

Aqui apresentamos uma idéia geral da implementação [1] do RBLP_AP. A figura 3.2 representa os processos de revisão de um programa em lógica Bayesiano. As seções seguintes detalham a implementação de cada um desses processos.

A.1 Processo - Constrói Rede Bayesiana

A revisão começa com o usuário fornecendo um conjunto de exemplos e o BLP inicial. As redes Bayesianas para cada exemplo de treinamento são, então, construídas. Para isto utilizamos o interpretador descrito no apêndice B com algumas modificações. Como o interpretador é executado como um programa independente e para isto espera receber um arquivo descrevendo o BLP ao qual será feita a consulta, foram necessárias modificações para incorpora-lo ao RBLP_AP. Um novo predicado, *call_query/3*, foi criado para permitir que este interpretador modificado, que chamamos de *probabilisticprover.pl*, pudesse ser chamado pelos outros programas Prolog do RBLP_AP. Para cada exemplo de treinamento, então, o predicado *call_query/3* é executado na tentativa de construir a rede Bayesiana correspondente.

```
call_query((Q|E), Theory, Arq) :-
    erase_module(pdom),
    compile(Arq), assert(current_threory(Theory)), query([Q],[E]),
    retract(current_threory(Theory)).
```

[1] A implementação ainda não foi concluída, e por este motivo alguns procedimentos, que serão descrito nesta seção, ainda não foram suficientemente testados.