

COMPUTAÇÃO DE RESPOSTAS EM SISTEMAS BASEADOS EM ELIMINAÇÃO DE MODELOS E DEFAULTS

Marco A. Casanova¹, Ramiro Guerreiro¹, e Andrea Silva^{1,2}

¹Centro Científico Rio - IBM Brasil
Caixa Postal 4624

22.071, Rio de Janeiro, RJ

²Departamento de Informática - PUC/RJ

R. Marquês de S. Vicente, 209

22.453, Rio de Janeiro, RJ

SUMÁRIO

Este trabalho aborda os fundamentos de sistemas para Programação em Lógica baseados no método de eliminação de modelos, acrescido de defaults. Inicialmente exibe-se uma adaptação do método que é correta e completa com respeito à computação de respostas quando os programas são conjuntos arbitrários de cláusulas. Em seguida, rediscute-se o problema de computar respostas quando os programas incluem também um conjunto finito de defaults.

1. INTRODUÇÃO

Eliminação de modelos (Loveland [1968, 1969, 1978]) é uma alternativa bastante interessante para a construção de sistemas para Programação em Lógica. Em particular, os sistemas Prolog baseiam-se em um método de eliminação de modelos, chamado Resolução-SLD, amplamente estudado, inclusive no que concerne a computação de respostas (ver, por exemplo, Lloyd [1984]). Porém, Resolução-SLD trabalha apenas com cláusulas definidas, forçando de certa forma a adoção da negação por falha finita (Clark [1978]) para ampliar o poder de expressão dos programas e consultas.

Este trabalho aborda os fundamentos de sistemas para Programação em Lógica baseados em um outro método, chamado *eliminação de modelos fraca (EMF)* (Loveland [1969]), acrescido ainda de defaults (Reiter [1980]).

O método *EMF* possui várias características atraentes. Ele trabalha com cláusulas genéricas, ou seja, com cláusulas com um número arbitrário de literais positivos ou negativos, é linear de entrada, não utiliza fatoração e, apesar destas características, é refutacionalmente completo. A primeira contribuição deste trabalho consiste em definir uma adaptação de *EMF* que é correta e completa com respeito à computação de respostas quando os programas e as consultas são expressos por conjuntos de cláusulas genéricas. A demonstração destes dois resultados é consideravelmente mais complexa do que a de resultados semelhantes para Resolução-SLD.

A negação dentro do contexto do método *EMF* possui o sentido clássico. A segunda contribuição deste trabalho refere-se então à introdução de defaults ao

método *EMF* para capturar raciocínio não monotônico, incluindo como caso particular a negação por falha. Neste contexto mais amplo, a própria noção de resposta de uma consulta a um programa em lógica apresenta nuances interessantes, que são brevemente discutidas.

Este trabalho está organizado da seguinte forma. A seção 2 recorda o método de eliminação de modelos fraca. A seção 3 estende o método para computação de respostas e enuncia os resultados principais. Finalmente, a seção 4 discute o uso de defaults e as suas implicações para a noção de resposta.

2. ELIMINAÇÃO DE MODELOS FRACA

O método de eliminação de modelos fraca (*EMF*) admite apenas refutações lineares de entrada, não utiliza fatoração e, apesar destas características, é refutacionalmente completo. Para tal, *EMF* mantém os literais resolvidos dentro das cláusulas derivadas, ordena os literais (resolvidos ou não) dentro das cláusulas e contém regras de inferência que utilizam tal informação.

Para definir de forma precisa o método *EMF*, considere inicialmente que os alfabetos de primeira ordem incluem também os colchetes esquerdo e direito, “[” e “]”. Todas as definições que se seguem referem-se a um alfabeto de primeira ordem A fixado a priori.

Um *literal* é qualquer fórmula atômica, negada ou não, um *literal resolvido* (ou um *R-literal*) é uma expressão da forma $[L]$, onde L é um literal, e um *elemento* é um literal resolvido ou não. Uma *cadeia elementar* é uma seqüência de literais e uma *cadeia* é uma seqüência de elementos. Denotaremos uma cadeia justapondo os elementos que a compõem e usaremos “ \square ” para representar a cadeia vazia (que por definição é elementar). Cada cadeia C representa, por convenção, o fecho universal da disjunção dos seus literais, no sentido de que uma estrutura para o alfabeto A satisfaz C se e somente se satisfizer a fórmula que C representa. Logo, os R-literais de uma cadeia não influenciam o seu significado.

Definiremos agora as regras de inferência de *EMF*. Usaremos $B'B''$ para denotar a concatenação de duas cadeias B' e B'' e $|L|$ para indicar a fórmula atômica P , se L for o literal P ou o literal $\neg P$. Dois literais L' e L'' são *canceláveis por uma substituição* θ se e somente se possuem sinais opostos e θ é um unificador mais geral (u.m.g.) de $\{L', L''\}$.

Sejam A' e A'' cadeias e β uma renomeação das variáveis de A'' tal que $A''\beta$ tenha variáveis distintas das de A' . Seja L' o elemento mais à esquerda de A' e suponha que L' seja um literal. Uma cadeia A é uma *extensão* de A' por A'' se e somente se existe um literal L'' de A'' e uma substituição θ tais que L' e $L''\beta$ são canceláveis por θ e $A = B'B'$, onde B'' é a cadeia $A''\beta\theta$ com o literal $L''\beta\theta$ removido e B' é a cadeia $A'\theta$ com o literal $L'\theta$ substituído por $[L'\theta]$.

Seja A' uma cadeia. Seja L' o elemento mais à esquerda de A' e suponha que L' seja um literal. Uma cadeia A é uma *redução* de A' se e somente se existe um R-literal M' de A' e uma substituição θ tais que L' e M' são canceláveis por θ e A é $A'\theta$ com o literal $L'\theta$ removido.

Uma cadeia A é a *contração* de uma cadeia A' se e somente se A é obtida removendo-se repetidamente o elemento mais à esquerda de A' até que este seja um literal ou que A' se transforme na cadeia vazia.

Uma cadeia A é uma *extensão plena* de A' por A'' se somente se for a contração da extensão de A' por A'' . Uma cadeia A é uma *redução plena* de uma cadeia A' se somente se for a contração da redução de A' .

O sistema formal de eliminação de modelos fraca, EM , admite como classe de linguagens os conjuntos das cadeias sobre alfabetos de primeira ordem, não possui axiomas e trabalha com duas regras de inferência, *extensão plena* e *contração plena*, definidas da seguinte forma:

Extensão Plena:

se A' e A'' são cadeias e
 A é uma extensão plena de A' por A''
então derive A de A' e A'' .

Redução Plena:

se A' é uma cadeia e
 A é uma redução plena de A'
então derive A de A' .

Uma *EMF-dedução* de uma cadeia C a partir de um conjunto de cadeias elementares S é uma seqüência de cadeias E tal que E possui um *prefixo* E_1, \dots, E_i composto apenas de cadeias em S , C é a última cadeia de E e toda cadeia em E a partir da *cadeia inicial* E_i é derivada da imediatamente anterior, a *cadeia-pai*, por redução ou extensão plenas, e de uma cadeia do prefixo de E , a *cadeia auxiliar*, no caso de extensão plena. É possível restringir ainda mais as deduções admissíveis através de filtros que eliminam cadeias derivadas (veja Loveland [1978]). Uma *EMF-refutação* a partir de um conjunto de cadeias elementares S é uma EMF-dedução da cadeia vazia a partir de S .

Finalmente, o *método da eliminação de modelos fraca* é o par $EMF = (EM, D)$, onde D é o conjunto das deduções lineares de entrada em EM . Conforme mencionado anteriormente, este método é refutacionalmente correto e completo (para uma demonstração, veja Loveland [1969]).

3. COMPUTAÇÃO DE RESPOSTAS ATRAVÉS DO MÉTODO EMF

Esta seção introduz inicialmente os conceitos de programa em cláusulas genéricas, consulta e resposta. Em seguida, apresenta uma adaptação do método *EMF* para computação de respostas, terminando com o enunciado de dois resultados básicos.

3.1 Programas, Consultas e Respostas

Um *programa* P é um conjunto finito de cadeias elementares. Da mesma forma, uma *consulta* A é um conjunto finito de cadeias elementares.

Um programa consiste então de uma série de cadeias formalizando um determinado universo de discurso. Por exemplo, o conjunto de cadeias abaixo forma um programa, que chamaremos **DIC**:

1. programa(a,fortran)
2. programa(b,pascal)
3. programa(c,fortran) programa(c,pascal)
4. chama(a,b)
5. chama(b,c)
6. \neg chama(x,y) depende(x,y)
7. \neg chama(x,z) \neg depende(z,y) depende(x,y)

Por exemplo, a cadeia (3) indica que "c" é um programa escrito em "fortran" ou "pascal" e as cadeias (6) e (7) indicam que x depende de y se x chama y direta ou transitivamente.

Já a noção de consulta deve ser interpretada de acordo com o seguinte exemplo. Suponha que se queira determinar de que programas escritos em "fortran" ou "pascal" o programa "a" depende. O processo de formalização desta pergunta consiste de dois passos. Primeiro, reescreve-se a pergunta como uma fórmula de primeira ordem:

$$Q. \text{ depende}(a,x) \wedge (\text{programa}(x,\text{fortran}) \vee \text{programa}(x,\text{pascal}))$$

Em seguida, obtém-se uma representação clausal da negação de Q:

8. \neg depende(a,x) \neg programa(x,fortran)
9. \neg depende(a,x) \neg programa(x,pascal)

Estas duas cadeias formam então a consulta correspondente à pergunta original. Denotaremos esta consulta por DEP[a].

Trataremos agora da noção de resposta. Se C é uma cadeia elementar, denote por \bar{C} a conjunção dos literais complementares aos literais de C e, se $R = \{R_1, \dots, R_n\}$ é um conjunto de cadeias elementares, denote por \bar{R} o fecho universal da fórmula $(\bar{R}_1 \vee \dots \vee \bar{R}_n)$.

Seja P um programa e A uma consulta. Uma *resposta* de A a P é um conjunto finito R de instâncias de A sobre o alfabeto de P U A, ou seja, um conjunto de cadeias obtidas das cadeias em A substituindo-se variáveis por termos do alfabeto em questão. Uma resposta R está *correta* se e somente se P implica logicamente \bar{R} . Finalmente, uma resposta R de A a P é *mais geral* do que uma resposta S de A a P se e somente se \bar{R} implica logicamente \bar{S} .

Esta noção de resposta deve ser interpretada como sugere a seguinte continuação do exemplo anterior. Recorde que DIC e DEP[a] denotavam o programa e a consulta, respectivamente. Então, uma resposta correta de DEP[a] a DIC seria:

$$10. \neg \text{depende}(a,b) \neg \text{programa}(b,\text{pascal})$$

De fato, esta cadeia é uma instância da cadeia (9) e, portanto, qualifica-se como uma resposta de DEP[a] a DIC. Além disso, esta resposta é correta pois DIC implica logicamente a fórmula "depende(a,b) \wedge programa(b,pascal)".

Uma segunda resposta correta de DEP[a] a DIC seria:

11. $\neg \text{depende}(a,c) \neg \text{programa}(c,\text{fortran})$
12. $\neg \text{depende}(a,c) \neg \text{programa}(c,\text{pascal})$

pois DIC implica logicamente a fórmula:

$$(\text{depende}(a,c) \wedge \text{programa}(c,\text{fortran})) \vee (\text{depende}(a,c) \wedge \text{programa}(c,\text{pascal}))$$

3.2 Computação de Respostas

Dada uma EMF-refutação E a partir de um programa P e de uma consulta A a P , é possível mostrar que as substituições efetuadas nas variáveis livres de cadeias em A durante a construção de E induzem uma resposta correta de A a P . Porém, recuperar tais substituições não é exatamente simples pois A possivelmente contém várias cadeias, que podem ser inclusive reusadas em E . Esta seção revê então a noção de cadeia e redefine as regras de inferência do método *EMF* de tal forma a registrar tais substituições.

Dado um conjunto de cadeias S e uma substituição θ , denotaremos por $S\theta$ o conjunto de cadeias resultante da aplicação de θ a cada cadeia em S .

Uma *cadeia ativada* é um par da forma (A,L) , onde A é uma cadeia e L é um conjunto de literais. A *ativação* de um programa P é o conjunto de todos os pares da forma (A,\emptyset) , onde $A \in P$. A *ativação* de uma consulta $A = \{A_1, \dots, A_n\}$ é o conjunto de todos os pares da forma $(A_i, \{R_i(\bar{x}_i)\})$, $i = 1, \dots, n$, onde \bar{x}_i é uma lista das variáveis de A_i e R_i é um novo símbolo predicativo de aridade igual ao comprimento de \bar{x}_i . O literal $R_i(\bar{x}_i)$ é o *literal de resposta* para A_i .

Uma cadeia ativada (A,L) é uma *redução plena ativada* de uma cadeia ativada (A',L') se e somente se A é uma redução plena de A' com u.m.g. θ e $L = L'\theta$. Uma cadeia ativada (A,L) é uma *extensão plena ativada* de (A',L') por (A'',L'') se e somente se A é uma extensão plena de A' por A'' , com u.m.g. θ e renomeação β de A'' , e $L = L'\theta \cup L''\beta\theta$. Os conceitos de *EMF-dedução ativada* e de *EMF-refutação ativada* seguem diretamente dos de *EMF-dedução* e *EMF-refutação* apenas substituindo-se 'cadeia' por 'cadeia ativada', 'redução plena' por 'redução plena ativada' e 'extensão plena' por 'extensão plena ativada'.

Seja P um programa e $A = \{A_1, \dots, A_n\}$ uma consulta a P . Suponha que as ativações de P e A sejam P' e A' , respectivamente. Uma resposta R de A a P é *EMF-computada* se e somente se existe uma EMF-refutação ativada E a partir de $P' \cup A'$ tal que E termina em (\square, S) e $B \in R$ se e somente se existe $(A_i, \{R_i(\bar{x}_i)\}) \in A'$ e existe $R_i(\bar{t}_i) \in S$ tais que $B = A_i\{\bar{x}_i/\bar{t}_i\}$.

Para exemplificar o conceito de refutação ativada, considere novamente o programa DIC e a consulta DEP[a] introduzidos na seção 3.1. Uma EMF-refutação ativada a partir da ativação das cadeias em DIC e em DEP[a] seria:

1. (programa(a,fortran), \emptyset)
2. (programa(b,pascal), \emptyset)
3. (programa(c,fortran) programa(c,pascal), \emptyset)
4. (chama(a,b), \emptyset)
5. (chama(b,c), \emptyset)
6. (\neg chama(x,y) depende(x,y), \emptyset)
7. (\neg chama(x,z) \neg depende(z,y) depende(x,y), \emptyset)
8. (\neg depende(a,u) \neg programa(u,fortran), $\{R_1(u)\}$)
9. (\neg depende(a,v) \neg programa(v,pascal), $\{R_2(v)\}$)
10. (\neg chama(a,y) [\neg depende(a,y)] \neg programa(y,pascal), $\{R_2(y)\}$)
 . extensão plena de 9 por 6
11. (\neg programa(b,pascal), $\{R_2(b)\}$) . extensão plena de 10 por 4
12. (\square , $\{R_2(b)\}$) . extensão plena de 11 por 2

Logo, $R = \{\neg \text{depende}(a,b) \neg \text{programa}(b,\text{pascal})\}$ é uma resposta EMF-computada de DEP[a] a DIC pois $R_2(b)$, o literal resposta em (12), indica que a variável v da cadeia em (9) foi substituída por b.

O método *EMF*, modificado como descrito acima, é correto e completo para computação de respostas no seguinte sentido:

Teorema 1: (Correção e Computação de Respostas por *EMF*)

Para todo programa **P**, para toda consulta **A**,

- (a) toda resposta EMF-computada de **A** a **P** é correta;
- (b) para toda a resposta correta de **A** a **P**, existe outra mais geral que é EMF-computada.

A demonstração deste resultado pode ser encontrada em Silva-[1988].

Estes resultado generaliza então resultados semelhantes para computação de respostas por Resolução-SLD (ver, por exemplo, Lloyd [1984]). Note, porém, que a noção de SLD-refutação é bastante mais simples do que a de EMF-refutação pois Resolução-SLD trabalha apenas com programas restritos a conjuntos finitos de cláusulas definidas e com consultas consistindo apenas de uma cláusula contendo somente literais negativos. Como consequência, torna-se muito mais simples identificar a seqüência de substituições efetuadas nas variáveis da cláusula da consulta e, logo, a demonstração da correção e completude da computação de respostas por Resolução-SLD.

Um resultado semelhante para resolução pode ser encontrado em Luckham [1971] e em Casanova [1987]. Estas referências também utilizam a técnica de literal de resposta, introduzida em Green [1969], para computar respostas. Porém, no caso de *EMF*, esta técnica teve que ser adaptada adotando-se pares da forma (C,B) essencialmente porque as regras de inferências de *EMF* trabalham com seqüências ordenadas de literais.

4. EXTENSÃO DO MÉTODO PARA INCORPORAR DEFAULTS

Esta seção resume brevemente os conceitos necessários sobre defaults para em seguida discutir o problema de computar respostas quando os programas incluem também um conjunto finito de defaults. Assumiremos uma certa familiaridade com Reiter [1980].

4.1 Defaults Clausais

Um *default clausal* é uma expressão da forma "A:C", onde A, o *antecedente* do default, é uma conjunção de literais, e C, o *conseqüente* do default, é uma cadeia elementar. Uma *teoria com defaults clausais* é um par $\Delta = (P, D)$, onde P é um conjunto finito de cadeias e D é um conjunto finito de defaults clausais.

O default clausal "A:C" deve ser visto como uma forma mais conveniente para expressar o default normal aberto $\frac{A:MF}{F}$, na notação usual, onde F é a disjunção dos literais em C. Assim, ao nos restringirmos a defaults clausais, estaremos considerando um caso particular de defaults normais abertos.

A semântica para teorias com defaults clausais segue do conceito de extensão para teorias com defaults abertos. Assim, dada uma teoria $\Delta = (P, D)$, um default clausal "A:C" em D deve ser interpretado como gerando o conjunto de defaults "A θ :C θ ", para toda substituição θ das variáveis de A e C por termos do universo de Herbrand sobre o alfabeto em questão. Intuitivamente, "A θ :C θ " indica que a afirmação C θ poderá ser assumida "por default" sempre que o antecedente A θ puder ser provado a partir de Δ e for consistente assumir C θ .

O resto desta seção aborda o conceito de refutação, seguindo Reiter [1980]. No que se segue, seja $\Delta = (P, D)$ uma teoria com defaults clausais e A um conjunto qualquer de cadeias elementares.

Intuitivamente, uma refutação a partir de Δ e A é uma seqüência de EMF-refutações tal que: (1) a primeira é uma EMF-refutação a partir das cadeias em $P \cup A$ e dos conseqüentes dos defaults em D; (2) após a primeira, cada EMF-refutação tem como objetivo refutar a negação dos antecedentes dos defaults utilizados na EMF-refutação anterior, com as substituições apropriadas, a partir das cadeias em P (mas não em A) e dos conseqüentes dos defaults em D. A seqüência deve satisfazer também um teste de consistência global, sancionando o disparo dos defaults.

Para monitorar as substituições afetando cada utilização de cada default em uma refutação, utilizaremos uma técnica idêntica àquela adotada na seção 3 para computar respostas (porém utilizando aqui o adjetivo 'indexada').

Assim, a *indexação* de Δ é o conjunto de pares da forma (C_i, \emptyset) , onde $C_i \in P$, ou da forma $(C_i, \{\delta_i(\bar{x}_i)\})$, para cada default "A_i:C_i" em D, onde \bar{x}_i é uma lista das variáveis ocorrendo em A_i e C_i e δ_i é um novo símbolo predicativo com aridade igual ao comprimento de \bar{x}_i . A *indexação* de A é o conjunto de pares da forma (C_i, \emptyset) , onde $C_i \in A$. Semelhantemente aos literais resposta utilizados na seção 3, $\delta_i(\bar{x}_i)$ registrará as substituições aplicadas nas variáveis ocorrendo no default "A_i:C_i". Para tal, as regras de inferência de EMF, bem como as noções de dedução e refutação, devem ser modificadas exatamente como descrito na seção 3. Em particular, quando uma cadeia indexada (B, D) for estendida por outra

$(C_i, \{\delta_i(\bar{x}_i)\})$, oriunda de um default, todas as variáveis de $\delta_i(\bar{x}_i)$ devem ser renomeadas para evitar conflitos com as de B.

Seja E uma EMF-refutação indexada a partir de cadeias na indexação de Δ e \mathbf{A} . Suponha que E termine em (\square, \mathbf{S}) . Um default ψ é *retornado* por E se e somente se existir um par $(C_i, \{\delta_i(\bar{x}_i)\})$ na indexação de Δ , correspondente a um default " $A_i:C_i$ " em \mathbf{D} , e existir um literal da forma $\delta_i(\bar{t}_i)$ em \mathbf{S} tal que ψ pode ser escrito como " $A_i\theta:C_i\theta$ ", onde $\theta = \{\bar{x}_i/\bar{t}_i\}$.

Seja " $A_i:C_i$ " um default em \mathbf{D} e $(C_i, \{\delta_i(\bar{x}_i)\})$ o par correspondente na indexação de Δ . Este default é *disparado* em E se e somente se existir uma cadeia indexada em E obtida por extensão plena indexada tendo como cadeia auxiliar o par $(C_i, \{\delta_i(\bar{x}_i)\})$. Assim, cada disparo do default " $A_i:C_i$ " em E gera um default *descendente* dentre os retornados por E .

Uma *EMF-refutação com defaults admissível* a partir de $\Delta = (\mathbf{P}, \mathbf{D})$ e \mathbf{A} é uma sequência finita E_0, \dots, E_k de EMF-refutações indexadas tal que:

1. E_0 é uma EMF-refutação indexada a partir das cadeias na indexação de Δ e \mathbf{A} ;
2. para $0 \leq i \leq k$, seja \mathbf{D}_i o conjunto dos defaults retornados por E_i e, para $1 \leq i \leq k$, seja $\mathbf{D}^{(i-1)}$ o conjunto dos defaults em \mathbf{D}_{i-1} que são descendentes dos defaults disparados em E_{i-1} . Então
 - a. Seja B a cadeia representando a negação da conjunção de todos os antecedentes dos defaults em $\mathbf{D}^{(i-1)}$. Então, E_i é uma EMF-refutação indexada a partir da indexação de Δ , acrescida do par (B, \mathbf{D}_{i-1}) ;
 - b. $\mathbf{D}^{(k)} = \emptyset$;
 - c. (*Teste de Consistência*). Seja \mathbf{C} o conjunto de todos os consequentes dos defaults ocorrendo em \mathbf{D}_k . Então, existe uma substituição θ das variáveis ocorrendo em \mathbf{C} por termos do universo de Herbrand sobre o alfabeto em questão tal que $\mathbf{P} \cup \mathbf{C}\theta$ é satisfável.

Finalmente, uma EMF-refutação com defaults admissível E_0, \dots, E_k a partir de $\Delta = (\mathbf{P}, \mathbf{D})$ e \mathbf{A} é uma *EMF-refutação com defaults* se e somente se, para todo $i \in [0, k]$, para toda cadeia-pai indexada (C_j, \mathbf{D}_j) de E_i que também ocorre como cadeia auxiliar indexada em E_i , os defaults descendentes de \mathbf{D}_j na cadeia-pai e \mathbf{D}_j na cadeia auxiliar são idênticos.

Segue dos resultados em Reiter [1980] que o método *EMF*, adaptado para trabalhar com defaults clausais como descrito acima, é refutacionalmente correto e completo.

4.2. Computação de Respostas com Defaults Clausais

Esta seção discute muito brevemente como estender as noções de programa, consulta e resposta para incorporar defaults. Um *programa com defaults* é um par $\Delta = (\mathbf{P}, \mathbf{D})$, onde \mathbf{P} é um conjunto finito de cadeias e \mathbf{D} é um conjunto finito de defaults clausais. As noções de consulta e resposta permanecem inalteradas e a noção de resposta correta passa a se basear na noção de extensão de teorias com defaults.

A computação de respostas pelo método *EMF* com defaults combina as duas noções de ativação e indexação, descritas nas seções 3 e 4. Assim, as regras de inferência passam a trabalhar com triplas da forma $(C, \mathbf{L}, \mathbf{M})$, onde C é uma cadeia, \mathbf{L} é um conjunto de literais para computação de respostas a uma

consulta, como descrito na seção 3, e M é um conjunto de literais para monitorar os defaults utilizados em uma refutação, como descrito na seção 4.

A definição de EMF-refutação com defaults induz imediatamente a seguinte noção de resposta computada. Seja $\Delta = (P, D)$ um programa com defaults e A uma consulta a Δ . Uma resposta R de A a Δ é *EMF-computada por defaults* se e somente se existe uma EMF-refutação com defaults E a partir de Δ e A tal que a última refutação em E termina em (\square, S, E) , o teste de consistência de E utiliza a substituição θ e $B \in R$ se e somente se existe $(A_i, \{R_i(\bar{x}_i)\}, \emptyset)$ na ativação de A e existe $R_i(\bar{t}_i) \in S$ tais que $B = A_i \gamma \theta$ onde $\gamma = \{\bar{x}_i/\bar{t}_i\}$.

Esta definição não é razoável, porém, pois admite respostas com uma componente completamente arbitrária, advinda da substituição θ gerada para o teste de consistência. Por outro lado, pela própria semântica dos defaults abertos, não é possível abandonar tal substituição sob pena de invalidar completamente a correção das EMF-refutações com defaults.

Por exemplo, seja $\Delta = (P, D)$ um programa, onde $P = \{\text{pássaro}(z), \neg \text{voa}(\text{pinguim}), \neg \text{voa}(\text{ema}), \text{amarelo}(\text{sabiá})\}$ e $D = \{\text{pássaro}(y) : \text{voa}(y)\}$. Seja a consulta $A = \{\neg \text{voa}(x)\}$. Considere a EMF-refutação por defaults $E = (E_0, E_1)$, a partir de Δ e A , construída da seguinte forma. Primeiro, E_0 é uma EMF-refutação indexada e ativada a partir da indexação e ativação de Δ e de A :

- 0.1 $(\text{voa}(y), \emptyset, \delta(y))$ (oriunda de Δ)
- 0.2 $(\neg \text{voa}(x), R(x), \emptyset)$ (oriunda de A)
- 0.3 $(\square, R(x), \delta(x))$ (extensão de 0.2 por 0.1)

Note que E_0 retorna o default "pássaro(x) : voa(x)". Logo, E_1 deve ser uma EMF-refutação indexada e ativada a partir das cadeias na indexação e ativação de Δ e da cadeia representando a negação do pre-requisito de "pássaro(x) : voa(x)" (a cadeia (1.2) abaixo):

- 1.1 $(\text{pássaro}(z), \emptyset, \emptyset)$ (oriunda de Δ)
- 1.2 $(\neg \text{pássaro}(x), R(x), \delta(x))$ (oriunda da negação do pre-requisito)
- 1.3 $(\square, R(x), \delta(x))$ (extensão de 1.2 por 1.1)

Pela definição de EMF-refutação com defaults, é necessário testar ainda a consistência do conjunto $E = \{\text{pássaro}(z), \neg \text{voa}(\text{pinguim}), \neg \text{voa}(\text{ema}), \text{amarelo}(\text{sabiá})\} \cup \{\text{voa}(x)\theta\}$, para alguma substituição θ de x por um termo do universo de Herbrand do alfabeto em questão. De fato, tomando-se $\theta = \{x/\text{sabiá}\}$ o conjunto anterior E torna-se consistente. Logo, $\{\neg \text{voa}(\text{sabiá})\}$ é a resposta EMF-computada pela refutação E , para esta escolha de θ .

Note que a escolha de θ é inteiramente arbitrária. Por outro lado, não é possível ignorar θ , pois o conjunto $\{\text{pássaro}(z), \neg \text{voa}(\text{pinguim}), \neg \text{voa}(\text{ema}), \text{amarelo}(\text{sabiá})\} \cup \{\text{voa}(x)\}$, não é satisfatível. Intuitivamente, o default em D não pode ser disparado para uma substituição de x por, por exemplo, "pinguim" ou "ema".

Para solucionar o impasse acima, propomos então basear o teste de consistência sempre em uma substituição fixada a priori, que troca cada variável por uma nova constante cuja semântica seria "o indivíduo típico tal que ...".

No exemplo corrente, introduza uma nova constante, p_0 , interpretada como "o pássaro típico". Considere novamente a EMF-refutação com defaults E , exceto que a substituição do teste de consistência passa a ser, por definição, $\{x/p_0\}$. Como, para esta escolha de θ , o conjunto $\{\text{pássaro}(z), \neg \text{voa}(\text{pinguim}), \neg \text{voa}(\text{ema}), \text{amarelo}(\text{sabiá})\} \cup \{\text{voa}(x)\theta\}$, é consistente, temos que $\{\neg \text{voa}(p_0)\}$ é a nova resposta computada pela EMF-refutação E . Tal resposta intuitivamente indica que "o pássaro típico" voa. Note que a introdução de p_0 é inteiramente semelhante à Skolemização de $\exists x(\text{voa}(x))$, exceto no que concerne à interpretação intuitiva.

O uso de "indivíduos típicos" pode ser ampliado considerando-se respostas corretas que:

1. envolvam apenas os "indivíduos típicos";
2. envolvam os "indivíduos típicos" e apontem algum "indivíduo atípico";
3. envolvam os "indivíduos típicos" e apontem todos os "indivíduos atípicos".

O esboço de um mecanismo de teste aproveitando estas sugestões seria o seguinte.

Seja $E = (E_0, \dots, E_k)$ uma EMF-refutação com defaults a partir de $\Delta = (P, D)$ e de um conjunto de cadeias elementares A . Suponha que C seja o conjunto de todos os consequentes dos defaults retornados por E_k . Seja π uma substituição das variáveis livres em C por constantes novas e distintas entre si (os "indivíduos típicos"). Para obter respostas envolvendo apenas "indivíduos típicos", substitua o teste de consistência por um teste de consistência por falha, ou seja, um teste verificando se a floresta de EMF-refutação (sem defaults) para $P \cup C\pi$ é finita e não possui uma árvore com uma folha rotulada com a cláusula vazia.

Para obter respostas envolvendo "indivíduos típicos" e todos (ou apenas alguns) "indivíduos atípicos", execute inicialmente o teste de consistência por falha para $P \cup C\pi$. Se o teste for satisfeito, então a resposta baseada em indivíduos típicos é de fato correta. Em seguida, compute as respostas (sem defaults) de C a P : cada resposta identificará possivelmente um "indivíduo atípico".

No exemplo corrente, temos $P = \{\text{pássaro}(z), \neg \text{voa}(\text{pinguim}), \neg \text{voa}(\text{ema}), \text{amarelo}(\text{sabiá})\}$ e $C = \{\text{voa}(x)\}$. Logo, as únicas respostas EMF-computadas (sem defaults) de C a P são (recorde que as resposta vêm negadas): $\{\text{voa}(\text{pinguim})\}$ e $\{\text{voa}(\text{ema})\}$, que caracterizam "pinguim" e "ema" como "indivíduos atípicos".

Para concluir, observamos que um tratamento exaustivo da noção de elemento típico pode ser encontrado em Silva [1988].

5. CONCLUSÕES

Eliminação de modelos fraca oferece uma alternativa interessante para construção de sistemas para Programação em Lógica, dado que trabalha com classes de programa e consulta mais genéricas do que as comumente adotados. A seção 3 oferece os fundamentos teóricos para tais sistemas, estabelecendo a correção e completude do método no que concerne à computação de respostas. Já a seção 4 esboça uma extensão do método para incorporar uma forma de defaults. Neste caso, a própria noção de resposta precisa de uma revisão apropriada para aumentar a capacidade expressiva do enfoque.

Salientamos ainda que a definição de consulta e resposta adotada, sempre na negativa, é conveniente apenas para simplificar o desenvolvimento da metateoria. Ela pode ser facilmente adaptada para trabalhar na forma não-negada, possibilitando uma leitura mais intuitiva.

Finalmente, observamos que a descrição de um sistema para Programação em Lógica baseado em eliminação de modelos pode ser encontrada em Guerreiro [1988].

AGRADECIMENTOS:

Agradecemos ao Prof. Tarcísio Pequeno pela leitura cuidadosa de uma versão preliminar do material contido na seção 4 e também várias discussões informais sobre o assunto.

REFERÊNCIAS BIBLIOGRÁFICAS

- Casanova, M.A., F.A.C. Giorno e A.L. Furtado [1987]. *Programação em Lógica e a Linguagem Prolog*, Ed. Blucher.
- Clark, K.L. [1978]. "Negation as Failure", em *Logic and Databases*, H. Gallaire e J. Minker (eds.), Plenum Press.
- Guerreiro, R.A.T., A. Silva e M.A.Casanova [1988]. "Programação em cláusulas genéricas utilizando o método de eliminação de modelos" (em preparação).
- Lloyd, J.W. [1984]. *Foundations of Logic Programming*, Springer-Verlag.
- Loveland, D.W. [1968]. "Mechanical theorem-proving by model elimination", *Journal of the ACM* 15:2, 236-251.
- Loveland, D.W. [1969]. "A simplified format for the model elimination theorem-proving procedure", *Journal of the ACM* 16:3, 349-363.
- Loveland, D.W. [1978]. *Automated Theorem Proving: a Logical Basis*, North-Holland Publishing Company, Amsterdam.
- Luckham, D.W. e N.J. Nilsson [1971]. "Extracting Information from Resolution Trees", *Artificial Intelligence* 2, 27-54.
- Reiter, R. [1980]. "A logic for default reasoning", *Artificial Intelligence* 13, 81-132.
- Silva, A. [1988]. "Sistemas para Programação em Lógica baseados em Eliminação de Modelos e Defaults" (tese de mestrado em preparação, a ser submetida ao Departamento de Informática da PUC/RJ).