

Marco A. Casanova* e Maria Emilia M. T. Walter**

*Centro Científico da IBM
**Departamento de Matemática da UnB e DHS/DITEC/SERPRO

SUMÁRIO

Este trabalho apresenta uma variação do método de eliminação de modelos como uma alternativa para a construção de sistemas para Programação em Lógica. A variação de Eliminação de Modelos escolhida possui como pontos interessantes o fato de ser linear de entrada, não utilizar fatoração e, apesar destas restrições, ser completa. Mais importante ainda, a variação escolhida leva a procedimentos de refutação que, mesmo aceitando cláusulas genéricas, podem ser implementados de forma muito semelhante aos interpretadores para Prolog.

1. INTRODUÇÃO

Eliminação de modelos constitui uma família de métodos de refutação que oferecem alternativas interessantes para a construção de sistemas para Programação em Lógica. A versão de eliminação de modelos escolhida neste trabalho é linear de entrada, não utiliza fatoração e, apesar destas restrições, mantém completude. Para isto usa um artifício: as cláusulas derivadas mantêm os literais resolvidos, que são posteriormente tratados por regras especiais. O método incorpora ainda uma forma restritiva de seleção de literais e um filtro para eliminar cláusulas derivadas que não são necessárias para a obtenção da cláusula vazia. O resultado são árvores de refutação mais compactas e, conseqüentemente, procedimentos de refutação mais eficientes.

A versão escolhida, descrita nas seções 2 e 3, é uma variação da original, introduzida em Loveland [1968] e posteriormente elaborada em Loveland [1969a, 1969b]. Ela é chamada de eliminação de modelos fraca em Loveland [1978, Sec. 3.6]. Uma outra versão, admitindo fatoração e baseada em um critério ainda mais restritivo para eliminação de cláusulas, aparece em Loveland [1978, Sec 3.6]. A Resolução-SL de Kowalski e Kuehner [1971] também pode ser incluída na lista de variações de eliminação de modelos.

A versão aqui adotada leva à construção de procedimentos de refutação, discutida na seção 4, que generalizam os interpretadores Prolog em dois sentidos. Primeiro, quando utilizados com cláusulas definidas (cláusulas de Prolog básico), comportam-se exatamente como o interpretador Prolog padrão. Segundo, a forma de implementação destes procedimentos

de refutação, mesmo trabalhando com cláusulas genéricas, é bastante semelhante e quase tão eficiente quanto os interpretadores Prolog.

Uma implementação da versão original de eliminação de modelos e de algumas variações é descrita em Fleisig et alii [1974]. Uma proposta mais recente de implementação é esboçada em Stickel [1984]. A discussão da seção 4 difere destas por explorar em detalhe quais modificações devem ser feitas na implementação usual dos interpretadores Prolog para transformá-los em procedimentos de refutação por eliminação de modelos.

2. ELIMINAÇÃO DE MODELOS

Eliminação de modelos trabalha com seqüências de literais, resolvidos ou não, chamadas de cadeias para não confundir com as cláusulas comumente usadas. Os colchetes esquerdo e direito, "[" e "]", agora incluídos nos alfabetos de primeira ordem, marcarão os literais resolvidos.

Mais precisamente, um literal resolvido (ou um R-literal) é uma expressão da forma [L], onde L é um literal, e um elemento é um literal ou um R-literal. Uma cadeia elementar é uma seqüência (possivelmente vazia) de literais e uma cadeia é uma seqüência não vazia de elementos. Cada cadeia C representa, por convenção, o fecho universal da disjunção dos seus literais, no sentido de que uma estrutura satisfaz C se e somente se satisfizer a fórmula que C representa. Logo, os R-literais de uma cadeia não influenciam o seu significado.

Denotaremos uma cadeia justapondo os elementos que a compõem e usaremos "[]" para representar a cadeia vazia (que por definição é elementar). Assim "p(x)q(x,y)r(y)" é uma cadeia elementar e "p(f(z)) [q(f(z),a)] r(a)" é uma cadeia onde o segundo elemento é um R-literal.

Os vários métodos de eliminação de modelos baseiam-se em um sistema formal contendo três regras de inferência, cuja descrição por sua vez depende de certas definições auxiliares, introduzidas a seguir. 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 ¬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 de $\{[L'], [L"]\}$. Dois literais são canceláveis se e somente se forem canceláveis por alguma substituição θ .

Sejam A' e A" cadeias e β uma renomeação para A" em presença 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" β 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' θ com o literal L' θ transformado em um R-literal.

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.

O sistema formal de eliminação de modelos, EM, contera então três regras de inferência, extensão (EX), redução (RD) e contração (CN), definidas como:

EX: se A' e A'' são cadeias tais que
o elemento mais à esquerda de A' é um literal
e A é uma extensão de A' por A''
derive A de A' e A''

RD: se A' é uma cadeia
tal que o elemento mais à esquerda de A' é um literal
e A é uma redução de A'
derive A de A'

CN: se A' é uma cadeia
tal que o elemento mais à esquerda de A' é um R-literal
e A é a contração de A'
derive A de A'

As definições de dedução e de refutação neste sistema formal são semelhantes às definições correspondentes para o sistema formal da resolução, exceto que existem três regras. O sistema formal da eliminação de modelos, como o sistema da resolução, é refutacionalmente correto e completo (para uma demonstração, veja Loveland [1969a]).

Uma dedução linear de entrada em EM é uma dedução em que cada nova cadeia é derivada da imediatamente anterior, a cadeia-pai, e de uma cadeia de entrada, a cadeia auxiliar, no caso de uma aplicação da regra da extensão.

O método da eliminação de modelos ^{fraco} é o par $\bar{M}=(EM, D)$, onde D é o conjunto das deduições lineares de entrada em EM. Combinando as restrições sobre a aplicação de redução e extensão com as restrições sobre as deduições, as principais características do método são então:

cadeia-pai:

- ° sempre a última cadeia derivada (linear)
- ° sem fatoração
- ° literal escolhido é sempre o mais à esquerda

cadeia auxiliar (para extensão):

- ° sempre uma cadeia de entrada (linear de entrada)
- ° sem fatoração

* Literal escolhido não é fixado a priori

É possível incorporar também filtros que eliminam cadeias derivadas sem perder a correção e completude do método, pontos que não serão discutidos neste texto (veja Loveland [1978]).

Terminaremos esta seção com exemplos do uso do método de eliminação de modelos. Seja S o seguinte conjunto de cadeias elementares:

1. $p(b)p(x)r(x)$
2. $\neg p(a)$
3. $\neg r(a)$
4. $\neg p(b)$

Uma refutação a partir de S começando em (4) seria:

1. $p(b)p(x)r(x)$
2. $\neg p(a)$
3. $\neg r(a)$
4. $\neg p(b)$
5. $p(x)r(x)[\neg p(b)]$. 1a
6. $[p(a)r(a)[\neg p(b)]]$. 2a
7. $r(a)[\neg p(b)]$. contração
8. $[r(a)[\neg p(b)]]$. 3a
9. \square . contração

onde a anotação "1a" na linha (5) indica que (5) é o resultado de estender (4) por (1), selecionando-se o primeiro literal de (1), e analogamente para as anotações nas linhas (6) e (8).

Uma dedução a partir de S começando em (4) seria:

1. $p(b)p(x)r(x)$
2. $\neg p(a)$
3. $\neg r(a)$
4. $\neg p(b)$
5. $p(x)r(x)[\neg p(b)]$. 1a
6. $r(b)[\neg p(b)]$. redução

Note que a dedução é bloqueada neste ponto pois o literal mais à esquerda de (6) não é cancelável com nenhum literal de uma cadeia de entrada ou com um R-literal de (6).

note sobre filtragem

3. ÁRVORES DE RESOLUÇÃO PARA ELIMINAÇÃO DE MODELOS

Da mesma forma que os procedimentos de refutação baseados no método de resolução linear, os procedimentos baseados em eliminação de modelos devem ser entendidos como algoritmos para construção de florestas de refutação. Estas florestas são definidas de forma semelhante às florestas para resolução linear, apenas considerando-se as restrições adicionais do método de eliminação de modelos, principalmente o fato do método ser linear de entrada e não utilizar fatoração.

Cômo no método de resolução linear, podemos provar que um conjunto S de cadeias elementares é insatisfatível se e somente se existe uma árvore A na floresta de refutação por eliminação de modelos para S tal que A tem um ramo de sucesso, ou seja, um ramo terminando em um nó rotulado com a cadeia vazia.

Portanto, um procedimento de refutação baseado no método de eliminação de modelos reduz-se a um algoritmo que recebe como entrada um conjunto S de cláusulas e progressivamente constrói a floresta de refutação por eliminação de modelos para S em busca de um nó de sucesso. Porém, procedimentos de refutação baseados no método de eliminação de modelos serão em geral mais eficientes pois, neste caso, as florestas de refutação tenderão a ser menores do que no caso de resolução linear. De fato, a seleção do mesmo literal em todas as deduções a partir da mesma cadeia (o literal mais à esquerda da cadeia), a inexistência de fatoração e o uso de cadeias auxiliares retiradas apenas do conjunto de entrada reduzem o tamanho da árvore de refutação, quando comparado com a árvore equivalente para resolução linear.

Dadas uma lista finita C de cadeias e uma cadeia D , a árvore de refutação canônica para C e D , é a árvore de refutação construída da seguinte forma:

1. a raiz é rotulada por D ;
2. os filhos de cada nó n são gerados da seguinte forma:
 - a. o primeiro filho, se possível, deve ser gerado por redução da cadeia que rotula n ;
 - b. os outros filhos, se possível, devem ser gerados estendendo-se a cadeia que rotula n pelas cadeias de entrada na ordem dada em C e consideram-se todos os literais de uma cadeia, na ordem dada, antes de tentar a próxima.

É possível provar que, se C for satisfatível, então $C \cup \{D\}$ é insatisfatível se e somente se a árvore de refutação canônica para C e D tem um ramo de sucesso. Portanto, um procedimento de refutação necessitará construir apenas a árvore de refutação canônica para C e D .

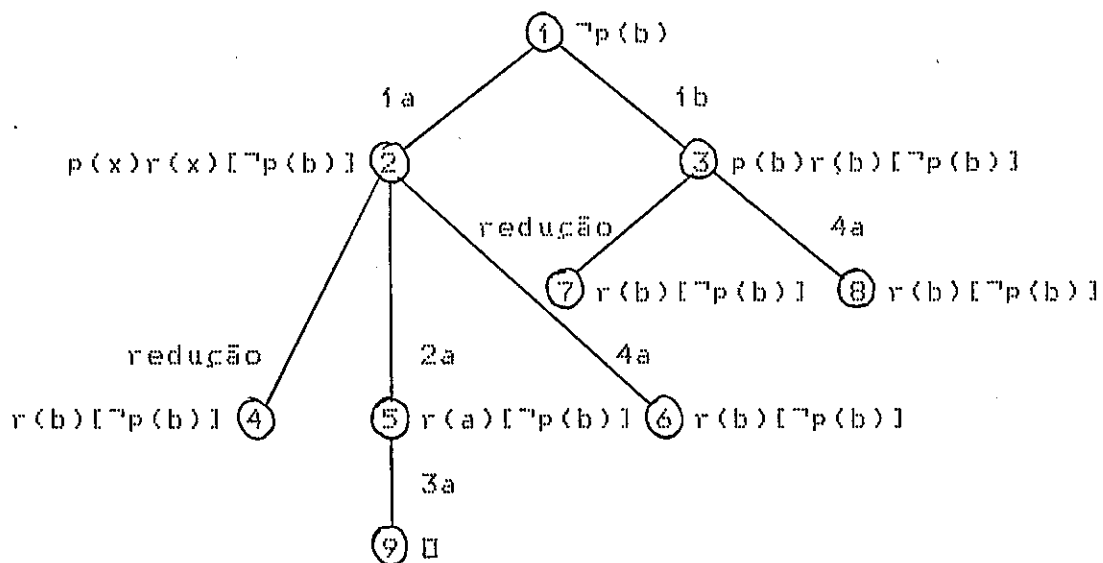
Seja C a seguinte lista de cadeias elementares:

1. $p(b)p(x)r(x)$
2. $\neg p(a)$
3. $\neg r(a)$

e D a cadeia abaixo:

4. $\neg p(b)$

A árvore de refutação canônica para C e D seria então:



Para facilitar o entendimento da árvore, rotulamos as arestas de acordo com a mesma convenção usada para anotar as derivações. Em detalhe temos, onde R_i denota a cadeia que rotula o nó i :

1. R_1 é a cadeia inicial em (1);
2. R_2 é a extensão de R_1 por (1), escolhendo o primeiro literal de (1);
3. R_3 é a extensão de R_1 também por (1), mas escolhendo o segundo literal de (1);
4. R_4 é a redução de R_2 , seguida de contração;
5. R_5 é a extensão de R_2 por (2), seguida de contração;
6. R_6 é a extensão de R_2 por (4), seguida de contração;
7. R_7 é a redução de R_3 , seguida de contração;
8. R_8 é a extensão de R_3 por (4), seguida de contração;
9. R_9 é a extensão de R_5 por (3), seguida de contração;

Descrição do Procedimento de Refutação

4. CONSIDERAÇÕES SOBRE IMPLEMENTAÇÃO

Esta seção esboça um procedimento de refutação baseado no método de eliminação de modelos cuja implementação segue essencialmente a técnica usada nos interpretadores Prolog. A entrada para o procedimento será um programa em lógica expresso por uma lista finita C de cadeias e uma consulta expressa por uma cadeia D . O procedimento basicamente simulará, em uma pilha, o caminhamento em preordem da árvore de refutação canônica para C e D .

Antes de apresentar os campos de um elemento da pilha, note que podemos identificar univocamente cada literal dentro de um programa C por um par (C,L) onde C indica o número de ordem de uma cadeia de entrada em C e L indica o número de ordem do literal na cadeia. Utilizaremos ainda "a" para indicar o primeiro literal de uma cadeia, "b" para o segundo e assim por diante.

Um elemento E_i da pilha terá os seguintes campos:

- CHAM é um par (C,L) que, junto com as substituições no campo AMB do pai de E_i , representa um R-literal;
- PROC é um par (C',L') que, junto com as substituições no campo AMB, representa o literal de uma cadeia de entrada cancelado com aquele representado por CHAM;
- RLIT é uma lista de literais da cadeia apontada por C' , representados também pelo seu número de ordem em C' , que indica os literais de C' já cancelados;
- PROX é um par (C'',L'') que indica o próximo literal L'' de uma cadeia C'' a ser considerado para extensão;
- PAI aponta para o pai de E_i dentro da pilha;
- AMB, REFAZ definem as substituições que compõem o ambiente de E_i , de acordo com a técnica usual de compartilhamento de estruturas (veja Bruynooghe [1982] e van Emden [1984], por exemplo).

Dada uma pilha P , para simular a aplicação de uma regra de inferência, o procedimento caminha pelos ponteiros do campo PAI, a partir do topo de P , até localizar o primeiro elemento E_j de P cujo campo RLIT não contém todos os literais da cadeia C referenciada no campo PROC. O literal L a ser considerado para redução ou extensão é o literal mais à esquerda da cadeia C não referenciado no campo RLIT de E_j , com as substituições indicadas no ambiente de E_j . O elemento E_j será o elemento ativo e o literal L o literal ativo de P .

A aplicação da regra de contração é simulada considerando-se o elemento ativo da pilha como pai do próximo elemento a ser adicionado na pilha. Uma aplicação da regra de redução é simulada basicamente tentando-se cancelar o literal ativo com literais representados por campos CHAM de elementos da pilha acessíveis a partir do elemento ativo via o campo PAI. A aplicação de extensão é implementada, por sua vez, tentando-se cancelar o literal ativo com literais das cadeias de entrada, na ordem em que aparecem no programa C .

O exemplo a seguir ilustra como a pilha será criada, ignorando os campos PROX, AMB e REFAZ por simplicidade. Suponha que o programa C seja:

1. $p(b)p(x)r(x)$
2. $\neg p(a)$
3. $\neg r(a)$

Suponha que a consulta D seja expressa pela cadeia

4. $\neg p(b)$

Mostraremos como o procedimento simula na pilha o caminhamento em preordem da árvore de refutação canônica para C e D , que é aquela apresentada no final da seção 3.

O primeiro passo do caminhamento consiste em gerar um nó rotulado com (4). Este passo é simulado na pilha da seguinte forma, observando-se que os elementos da pilha serão numerados E1, E2, etc...:

	CHAM	PROC	RLIT	PAI
E1	-	(4,a)	∅	-

O segundo passo do caminhamento será gerar um nó rotulado por:

5. $p(x)r(x)[\neg p(b)]$

resultante da extensão de (4) por (1), utilizando o primeiro literal de (1).

O procedimento toma a decisão de aplicar extensão a (4) da seguinte forma. Como o campo RLIT de E1 é vazio e o campo PROC aponta para a cadeia (4), o literal ativo neste ponto é " $\neg p(b)$ "; o primeiro literal de (4). Mas, como E1 é o único elemento da pilha, não é possível tentar redução. Logo o procedimento tenta extensão, procurando o primeiro literal do programa C que pode ser cancelado com " $\neg p(b)$ ", que neste caso é o literal (1,a). A simulação da extensão de (4) por (1), para esta seleção de literais, resultará em:

	CHAM	PROC	RLIT	PAI
E1	-	(4,a)	a	-
E2	(4,a)	(1,a)	a	E1

O campo CHAM de E2 representa um R-literal, no caso, " $[\neg p(b)]$ ". Na verdade, os campos CHAM armazenam os R-literais sem custo adicional, eliminando assim uma das desvantagens aparentes de eliminação de modelos.

O campo PROC de E2 indica o literal L' de uma cadeia C' utilizado na aplicação da regra da extensão, no caso, " $p(b)$ ".

O campo RLIT de E2 indica o conjunto de literais de C' já cancelados ao longo da dedução, que neste ponto reduz-se ao primeiro literal de (1). Observe que o campo RLIT de E1 agora indica o conjunto {a} pois o primeiro literal da cadeia (4) agora foi cancelado. Note ainda que RLIT poderia ser dispensado pois é computável a partir dos campos CHAM dos outros elementos da pilha. Porém, preferiu-se evitar esta computação armazenando RLIT explicitamente.

A tentativa de aplicar redução a cada novo elemento gerado é a componente principal do custo adicional de tempo e a manutenção de RLIT é o único custo adicional de espaço do procedimento, quando comparado com o procedimento padrão de PROLOG operando sobre cláusulas definidas. De fato, quando a entrada é um programa Prolog puro, redução torna-se desnecessária e o campo RLIT redundante, pois todas as cadeias derivadas possuem apenas literais negados e todas as cadeias do programa possuem exatamente um literal positivo.

O passo seguinte será a geração de um nó rotulado com:

6. $r(b)[\neg p(b)]$

resultante da redução de (5), seguida de contração.

O procedimento toma a decisão de aplicar redução da seguinte forma. Como, para o elemento E2, PROC=(1,a) e RLIT=(a), o literal ativo neste ponto é "p(x)", o literal mais à esquerda de (1) não referenciado em RLIT. Mas o R-literal "[$\neg p(b)$]", representado no campo CHAM de E1, pode ser cancelado com "p(x)". Logo, é possível aplicar redução. A simulação desta redução na pilha resulta em:

	CHAM	PROC	RLIT	PAI
E1	-	(4,a)	a	-
E2	(4,a)	(1,a)	a,b	E1
E3	(1,b)	-	\emptyset	E2

Note que a redução é caracterizada pela existência de um valor nulo no campo PROC de E3 e por ajustes no campo RLIT de E2, o pai de E3.

A contração será efetivada quando o procedimento selecionar o elemento e o literal ativos. Neste caso, como E3 foi adicionado por redução, o elemento ativo é E2 e o literal ativo é "r(b)", resultante de aplicar ao literal (1,c) as substituições do ambiente de E2 (não mostradas no exemplo). Logo, o pai do próximo elemento da pilha será E2.

O próximo passo será retornar ao nó rotulado por (5), já que não é possível aplicar qualquer regra de inferência a (6).

O procedimento toma esta decisão da seguinte forma. Observe inicialmente que o literal ativo, "r(b)", não pode ser cancelado com nenhum literal de uma cadeia de entrada e com nenhum R-literal representado pelos campos CHAM dos elementos da pilha acessíveis a E3 pelo campo PAI. Logo, o procedimento retrocede para o nó anterior. O efeito na pilha será então:

	CHAM	PROC	RLIT	PAI
E1	-	(4,a)	a	-
E2	(4,a)	(1,a)	a	E1

Note que a lista RLIT de E2 voltou ao estado anterior.

O passo seguinte será a geração de um nó rotulado com:

7. $r(a)[\neg p(b)]$

resultante da extensão de (5) por (2), seguida de contração.

O procedimento toma a decisão de aplicar extensão da seguinte forma. Primeiro, note que o literal ativo neste ponto é "p(x)", representado por (1,b) no ambiente de E2. Como redução já foi tentada, o procedimento tenta agora expansão, utilizando como cadeia auxiliar a primeira cadeia que ocorre no programa que contém um literal cancelável com "p(x)" e selecionando o primeiro literal desta cadeia que pode ser cancelado com

"p(x)". Neste caso, o literal selecionado será (2,a). A pilha é então alterada para:

	CHAM	PROC	RLIT	PAI
E1	-	(4,a)	a	-
E2	(4,a)	(1,a)	a,b	E1
E3	(1,b)	(2,a)	a	E2

A contração será efetivada quando o procedimento selecionar o elemento e o literal ativos. Neste caso, como o campo RLIT de E3 contém todos os literais da cadeia (2), o que não ocorre com o campo RLIT de E2, o elemento ativo é E2 e o literal ativo é "r(a)", resultante de aplicar ao literal (1,c) as devidas substituições (não mostradas no exemplo).

O passo seguinte será a geração de um nó rotulado com:

8. □

resultante da extensão de (7) por (3), seguida de contração.

O procedimento toma esta decisão da seguinte forma. Como visto, o literal ativo neste ponto é "r(a)". Porém, o procedimento não pode aplicar redução, pois "r(a)" não pode ser cancelado com nenhum R-literal representado pelos campos CHAM dos elementos da pilha acessíveis a E3 pelo campo PAI. O procedimento tenta então aplicar extensão, utilizando como cadeia auxiliar a primeira cadeia que ocorre no programa que contém um literal cancelável com "r(a)" e selecionando o primeiro literal desta cadeia que pode ser cancelado com "r(a)". Neste caso, o literal selecionado será (3,a).

A pilha é então alterada para:

	CHAM	PROC	RLIT	PAI
E1	-	(4,a)	a	-
E2	(4,a)	(1,a)	a,b,c	E1
E3	(1,b)	(2,a)	a	E2
E4	(1,c)	(3,a)	a	E2

Note que o campo PAI de E4 aponta para E2, refletindo assim a contração efetuada para gerar (7). Note ainda que o campo RLIT de E2 foi alterado para (a,b,c), refletindo o fato de que o literal "r(x)" da cadeia (1) foi cancelado.

Como não existem mais literais a resolver nas cadeias indicadas nos campos PROC dos elementos da pilha, o procedimento pára, indicando que a cadeia vazia foi obtida.

Pode-se provar que se o procedimento acima pára então o conjunto inicial de cadeias é insatisfável. É possível também modificar o procedimento para computar respostas às consultas (veja Walter [1986]). Naturalmente, como o procedimento simula um caminhamento em profundidade pela árvore de refutação, ele poderá não parar em certos casos apesar do conjunto inicial de cadeias ser insatisfável. Note também que o procedimento de refutação simula apenas a construção da

árvore de refutação canônica para o programa C e a consulta D, o que é suficiente se C for satisfável.

6. CONCLUSÕES

A escolha judiciousa de uma variação de eliminação de modelos, conforme descrito na seção 2, conduz à implementação de procedimentos de refutação com duas características importantes. Primeiro, diferentemente do procedimento padrão de Prolog, tais procedimentos trabalham com cláusulas genéricas, e não apenas com cláusulas definidas. Em outras palavras, isto significa que os literais negativos em uma cláusula recebem o significado clássico, não sendo tratados pela regra de negação por falha, conforme descrito em Clark [1978] ou em Lloyd [1984]. Segundo, estes procedimentos terão desempenho comparável à do interpretador padrão de Prolog, pois o custo adicional de tempo reduz-se à tentativa de aplicar a regra de redução sempre que um novo elemento é adicionado à pilha e o custo adicional de espaço resume-se à manutenção da lista de literais RLIT.

A primeira característica é particularmente importante pois permite reservar o símbolo "¬" para denotar a negação clássica e utilizar um outro símbolo, por exemplo "NÃO", para forçar a aplicação da regra da negação por falha. Assim, uma fórmula da forma "NÃO(p(t1,...,tn))" seria utilizada nos casos em que p é um símbolo predicativo que admite uma interpretação de acordo com a hipótese do mundo fechado (Reiter [1978]). Os literais negados, por outro lado, continuariam a ser tratados de acordo com as regras usuais de eliminação de modelos.

REFERÊNCIAS BIBLIOGRÁFICAS

- Bruynooghe, M. [1982]. "The Memory Management of PROLOG Implementations", em *Logic Programming*, K.L. Clark e S.-A. Tärnlund, Academic Press.
- Clark, K.L. [1978]. "Negation as Failure", em *Logic and Databases*, H. Gallaire e J. Minker (eds.), Plenum Press.
- Fleisig, S., D. Loveland, A.K. Smiley III e D.L. Yarmush [1969]. "An implementation of the model elimination proof procedure", *J. ACM* 21:1 (jan. 1974), 124-139.
- Kowalski, R. e D. Kuehner [1971]. "Linear resolution with selection function", *Artificial Intelligence* 2, 227-260.
- Lloyd, J.W. [1984]. *Foundations of Logic Programming*, Springer-Verlag.
- Loveland, D.W. [1968]. "Mechanical theorem-proving by model elimination", *J. ACM* 15:2 (abril 1968), 236-251.
- Loveland, D.W. [1969a]. "A simplified format for the model elimination theorem-proving procedure", *J. ACM* 16:3 (julho 1969), 349-363.
- Loveland, D.W. [1969b]. "Theorem-provers combining model elimination and resolution", em *Machine Intelligence* 4, B.

- Meltzer e D. Michie (eds.), Edinburgh U. Press, Edinburgh, 73-86.
- Loveland, D.W. [1970]. "A linear format for resolution" em SYMPOSIUM ON AUTOMATIC DEMONSTRATION. Lecture Notes in Mathematics 125, Springer-Verlag, 147-163.
- Loveland, D.W. [1972]. "A unifying view of some linear Herbrand procedures", J. ACM 19:2 (abril 1972), 366-384.
- Loveland, D.W. [1978]. Automated Theorem Proving: a Logical Basis, North-Holland Publishing Company, Amsterdam.
- Reiter, R. [1978]. "On Closed World Databases", em Logic and Databases, H. Gallaire and J. Minker (eds.), Plenum Press.
- Stickel, M.E. [1984]. "A PROLOG technology theorem prover", New Generation Computing 2, 371-383.
- Walter, M.E.M.T. [1986]. "Procedimentos de Refutação por Eliminação de Modelos" (em preparação).
- van Emden, M.H. [1984]. "An Interpreting Algorithm for Prolog Programs", em Implementations of ERLILOG, J.A. Campbell (ed.), Ellis Horwood, Ltd.