Inferência Lógica por Transformação Canônica

Guilherme Bittencourt

Instituto Nacional de Pesquisas Espaciais - INPE
Caixa Postal 515
12.201-970 - São José dos Campos - SP - Brasil
Tel.: (0123) 41 8977 r. 650
Fax: (0123) 21 8743
E-mail: inpelac@brfapesp.br

Resumo

Neste artigo, apresenta-se um método de inferência inerentemente paralelo para a lógica de primeira ordem, baseado na transformação entre formas normais canônicas. Introduz-se ainda um formalismo geométrico para representar cláusulas lógicas que permite a exploração do paralelismo do método de inferência através da representação da distributividade e da dominância entre cláusulas como fenômenos de propagação e interferência de ondas de informação em um espaço de abstrações.

Abstract

In this paper, we present an inherently parallel inference method for first-order logic, which is based on the transformation between canonical normal forms. We introduce also a geometric formalism to represent logical clauses that allows the exploitation of the parallelism of the method through the representation of distributivity and subsumption as propagation and interference phenomena of information waves in an abstraction space.

1. Introdução

A lógica tem um papel importante dentro da cultura ocidental. Suas aplicações vão desde a fundamentação teórica de diversas áreas da matemática (Russell e Whitehead, 1913) até a representação de conhecimentos em inteligência artificial (Nahl, 1983). A partir da introdução, por Robinson (1965), de um procedimento para demonstração automática de teoremas por computador baseado na teoria da resolução, a lógica passou a ser estudada também como método computacional para resolução de problemas. Desde então, diversas técnicas de representação de expressões lógicas, bem como diversas estratégias de controle para o processo de prova foram propostas (Stickel, 1986). Ainda hoje a área de prova automática de teoremas permanece bastante ativa, sendo objeto de diversas conferências internacionais.

O fato de ser possível associar uma semântica operacional a um procedimento de prova automática de teoremas (Van Emden e Kowalski, 1976) permitiu a introdução de linguagens de programação baseadas em lógica (e.g. Sterling e Shapiro, 1986). Estas linguagens, inicialmente restritas a laboratórios de

pesquisa, hoje são amplamente utilizadas, e seus aperfeiçoamentos e possíveis extensões são objetos de intensas pesquisas.

Uma dificuldade inerente a qualquer algoritmo de prova automática de teoremas é a chamada *explosão combinatória*, i.e. o tempo e o espaço necessários para a solução de um problema aumentam exponencialmente com o tamanho do problema. Esta dificuldade não está associada exclusivamente aos demonstradores automáticos, mas a todos os problemas NP completos. Uma solução proposta é a utilização de computadores paralelos (Rumelhart e McClelland, 1986).

A utilização de computadores paralelos exige que os algoritmos sejam redesenhados de maneira a utilizar eficientemente a capacidade de paralelismo da máquina. No caso da lógica, técnicas de prova automática como a *Resolução por Grafos* (Eisinger, 1986) foram adaptadas para o caso paralelo (Loganantharaj e Mueller, 1986), mas em geral os algoritmos de prova automática de teoremas não são facilmente paralelizáveis. Uma outra solução consiste em projetar algoritmos especiais para arquiteturas específicas (e.g. Chen e Fang, 1991; Martinez e Vidal, 1988).

No caso de linguagens de programação lógica, um grande esforço vem sendo desenvolvido para encontrar maneiras eficientes de tornar paralelo o algoritmo de execução de programas (e.g. Chasin de Kergommeaux e Robert, 1990; Ciepielewski et al., 1989).

Neste artigo, apresenta-se um método de inferência para a lógica de primeira ordem baseado na transformação de um conjunto de expressões lógicas de um tipo de forma canônica para outro. Este método de inferência é inerentemente paralelo, pois sua principal operação corresponde à aplicação da propriedade de distributividade dos operadores "e" e "ou". Esta operação, embora exponencial em tempo e espaço, tem um caráter local que facilita a paralelização (Socher, 1992). Além da aplicação da distributividade, o método utiliza intensivamente o teste de dominância ("subsumption") entre cláusulas.

Diferentemente do método de resolução, que atua sobre duas cláusulas, o método proposto atua de maneira global sobre um conjunto de expressões lógicas. Esta propriedade permite a generalização do conceito de estratégia de controle: o fato de não haver restrições quanto ao número de cláusulas a serem utilizadas para a inferência permite a utilização de critérios mais gerais de escolha das cláusulas.

Para explorar as possibilidades de paralelismo inerentes ao método proposto, introduz-se um formalismo geométrico para representar cláusulas lógicas. Neste formalismo, a distributividade pode ser realizada através da propagação e interferência de ondas de informação em um determinado espaço de abstrações, e a dominância pode ser determinada diretamente, pois no espaço de abstrações todas as cláusulas dominadas por

dada cláusula são representadas em um superespaço da representação da cláusula dominante.

O artigo é organizado da seguinte maneira: na Seção 2, apresenta-se o resultado lógico que sustenta o método de inferência proposto, define-se o algoritmo de inferência e demonstra-se seu funcionamento através de exemplos. Na Seção 3, introduz-se uma notação geométrica para cláusulas lógicas que permite explorar o paralelismo do método de inferência. Finalmente, na Seção 4, apresentam-se algumas conclusões e propostas para desenvolvimentos futuros.

2. Inferência por Transformação Dual

Nesta seção, apresenta-se o seguinte resultado: dado um conjunto de expressões em lógica de primeira ordem representado na forma clausal usual (conjunção de disjunções), a transformação de representação para a forma clausal dual (disjunção de conjunções) e de volta para a forma usual implica que o conjunto resultante contenha todas as conseqüências lógicas que poderiam ser geradas por resolução a partir do conjunto original.

O problema para a utilização deste resultado como método de inferência é o fato de que a transformação dual gera um grande número de cláusulas repetidas e dominadas. Deste modo, é necessário eliminar tais cláusulas antes de realizar a transformação inversa. Além disto, a transformação gera também cláusulas contraditórias; estas cláusulas podem ser eliminadas desde que as substituições que as tornam contraditórias sejam aplicadas às cláusulas restantes, antes da tranformação inversa. O enfoque adotado para tratar estes problemas, e a definição formal do método de inferência proposto, ilustrada através de exemplos, são apresentados abaixo.

Sistema Lógico S

Dados um conjunto de n predicados P, um conjunto de funções (e constantes) F, e um conjunto de variáveis X, define-se, da forma usual, um sistema lógico S.

Conjuntos de Cláusulas

Seja *Com* o conjunto de todas as cláusulas conjuntivas, e *Dis* o conjunto de as cláusulas disjuntivas, formadas com literais do sistema S. Quando o tipo de cláusula não for relevante ao contexto, utiliza-se o símbolo C para o conjunto de todas as cláusulas.v

Representações Canônicas

Dado um conjunto de fórmulas bem formadas (fbf) W do sistema lógico S, pode-se obter por skolemização duas representações canônicas, W_d — conjunção de disjunções — e W_c — disjunção de conjunções — formadas respectivamente por conjuntos de elementos de *Dis* e *Com*. Dado um conjunto W_d pode-se obter o conjunto W_c associado aplicando a propriedade de

distributividade dos operadores "e" ($\wedge$) e "ou" ($\vee$). Analogamente, pode-se obter W_d a partir de W_c.

É importante notar que na forma disjuntiva cada cláusula pode ter suas variáveis substituídas por novos símbolos de variável que não aparecem em nenhuma outra cláusula, pois nesta representação todas as cláusulas devem ser válidas independentemente. Já na forma conjuntiva o escopo das variáveis inclui todo o conjunto de cláusulas.

Dominância

A dominância ("subsumption") é definida na literatura para conjuntos do tipo W_d (conjunção de disjunções). Propõe-se uma extensão deste conceito para conjuntos do tipo W_c (disjunção de conjunções) e introduz-se o conceito de dominância interna para cláusulas disjuntivas e conjuntivas.

De acordo com a definição usual, diz-se que uma cláusula disjuntiva w_1 domina uma outra cláusula disjuntiva w_2 em um conjunto conjuntivo de cláusulas, se existe uma substituição θ, tal que:

$w_1 \bullet \theta \subset w_2$,

onde "$\bullet$" representa a aplicação de uma substituição, e $\subset$ é o conectivo "está contido em" da teoria dos conjuntos.

Por exemplo:

$w_1 = P(x) \vee Q(b)$ domina $w_2 = P(a) \vee Q(b) \vee R(y)$ com $\theta = ((x.a))$, pois
$\{P(x), Q(b)\} \bullet ((x.a)) = \{P(a), Q(b)\} \subset \{P(a), Q(b), R(y)\}$, e
$P(x) \vee Q(b) \Leftrightarrow (P(x) \vee Q(b)) \wedge (P(a) \vee Q(b) \vee R(y))$

Estendendo o conceito para conjuntos do tipo W_c, diz-se que uma cláusula conjuntiva w_1 domina uma outra cláusula conjuntiva w_2 em um conjunto disjuntivo de cláusulas, se existe uma substituição θ, tal que:

$w_1 \subset w_2 \bullet \theta$

Por exemplo:

$w_1 = P(a) \wedge Q(b)$ domina $w_2 = P(x) \wedge Q(b) \wedge R(y)$ com $\theta = ((x.a))$, pois
$\{P(a), Q(b)\} \subset \{P(x), Q(b), R(y)\} \bullet ((x.a)) = \{P(a), Q(b), R(y)\}$, e
$P(a) \wedge Q(b) \Leftrightarrow (P(a) \wedge Q(b)) \vee (P(x) \wedge Q(b) \wedge R(y))$

Para eliminar cláusulas dominadas em conjuntos do tipo W_c é necessário levar em conta a substituição envolvida, devido ao escopo global das variáveis. De todo modo, mesmo neste caso, não é possível eliminar qualquer uma dentre duas cláusulas que se dominem mutuamente.

Por exemplo:

$w_1 = P(x) \wedge P(a)$ não domina $w_2 = P(y) \wedge P(a)$, pois
$(\neg P(x) \wedge P(a)) \vee (\neg P(y) \wedge P(a)) \vee R(x,y) <\neq>$
$(\neg P(x) \wedge P(a)) \vee R(x,y)$

O conceito de dominância interna é introduzido a seguir. Diz-se que um literal l_1 de uma cláusula disjuntiva domina um literal l_2 da mesma cláusula disjuntiva, se existe uma substituição θ, tal que,

$l_1 = l_2 \bullet \theta$

Por exemplo:

$l_1 = P(a)$ domina $l_2 = P(x)$ com $\theta = ((x.a))$, pois
$P(a) = P(x) \bullet ((x.a)) = P(a)$, e
$P(a) \vee P(x) \Leftrightarrow P(a)$

Diz-se que um literal l_1 de uma cláusula conjuntiva domina um literal l_2 da mesma cláusula conjuntiva, se existe uma substituição θ, tal que,

$l_1 \bullet \theta = l_2$

Por exemplo:

$l_1 = P(x)$ domina $l_2 = P(a)$ com $\theta = ((x.a))$, pois
$P(x) \bullet ((x.a)) = P(a)$, e
$P(x) \wedge P(a) \Leftrightarrow P(x)$

Dado um conjunto de cláusulas W, define-se (W)' como um subconjunto de W que não contém nenhuma cláusula onde um literal domine outro, e nenhum par de cláusulas tal que uma domine a outra.

Contradições

Dado um conjunto W_d (respectivamente W_c), o conjunto $(W_c)'$ (respectivamente $(W_d)'$) correspondente pode apresentar cláusulas onde aparecem literais unificáveis com sinais contrários, e.g. $(\neg P(x),P(a),...)$, $(\neg P(x),P(f(a)),...)$, $(\neg P(x),P(y),...)$, $(\neg P(x),P(f(y)),...)$. Neste caso, a cláusula pode ser eliminada desde que a substituição resultante da unificação seja levada em conta. Em outras palavras, associada a cada cláusula deste tipo teremos um conjunto de substituições tal que cada uma representa uma maneira possível de eliminar a cláusula.

No caso de conjuntos do tipo W_c, o dual é conjuntivo, e o escopo das variáveis local às cláusulas. Neste caso, as cláusulas podem ser eliminadas diretamente: é a chamada eliminação de tautologias. No caso de conjuntos do tipo W_d, o dual é disjuntivo, e o escopo das variáveis global sobre conjunto de cláusulas. Neste caso é necessário analisar as combinações de fragmentos de substituição, resultantes de cada cláusula, que formam substituições válidas.

Deste modo, dado um conjunto W_d, ao conjunto $(W_c)'$ correspondente é associado um conjunto de substituições, formado por combinações válidas de fragmentos de substituição provenientes das cláusulas contraditórias. Cada substituição deste conjunto é associada a um conjunto de inferências possíveis. Chamaremos G o conjunto formado por estas substituições válidas, onde foram eliminadas as substituições contidas em outras substituições do conjunto.

É interessante notar que os fragmentos de substituição contêm toda a informação sobre as possibilidades de inferência representada através de uma estrutura matemática precisa: o conjunto das funções de X em T, onde X e T são respectivamente os conjuntos de variáveis e de termos do sistema lógico S. O estudo de métodos de manipulação deste tipo de estrutura é uma ativa área de pesquisa.

Inferência

A inferência é realizada através da aplicação de cada uma das substituições de G sobre o conjunto $(W_c)'$, seguida da eliminação das cláusulas contraditórias onde os literais de sinal contrário tenham se tornado idênticos. Caso o conjunto $(W_c)'$ torne-se vazio para alguma substituição em G, o conjunto original W_d é contraditório. Caso contrário, o conjunto

$$\cup_{\theta_i \in G} ((W_c)' \bullet \theta_i)_d$$

obtido por distributividade aplicada sobre a união dos conjuntos instanciados $(W_c)' \bullet \theta_i$ contém cláusulas que são conseqüências lógicas do conjunto original W_d.

O Teorema

Define-se $(W_d)^*$ como o conjunto que contém explicitamente todas as expressões que são conseqüências lógicas de um conjunto original W_d. Se $W_d = (W_d)^*$ o conjunto é dito fechado.

Define-se ainda a função:

$$\text{pulsar}(W_d) = (W_d \cup (\cup_{\theta_i \in G} (W_c)' \bullet \theta_i)_d)'$$

Teorema. Dado um conjunto de cláusulas W_d, propõe-se que:

$$((W_d)^*)' = \lim_{n \to \infty} \text{pulsar}^n(W_d). \blacklozenge$$

Uma linha de prova seria:

1. Mostrar que a função pulsar é correta: a partir de um conjunto de cláusulas válidas gera apenas cláusulas válidas.

2. Provar que dado, um conjunto de cláusulas válidas W_d, e um conjunto $V_d \subset W_d$, o conjunto $(W_d - V_d) \cup \text{pulsar}(V_d)$ contém apenas cláusulas válidas.

3. Mostrar que, limitada a conjuntos $V_d \subset W_d$, com $|V_d| = 2$, a função pulsar é equivalente à resolução.

4. Provar que, dado um conjunto W_d, $(R(W_d))' = \text{pulsar}(W_d)$, onde R é o operador que constrói o conjunto de todas as resoluções possíveis de um conjunto de cláusulas.

5. Mostrar que a eliminação de dominados a cada passo não impede que alguma cláusula de $((W_d)^*)'$ seja gerada. Este é o ponto mais sutil da prova pois depende das propriedades do conjunto $(W_d)^*$ e do operador $(\_)'$.

Provados os argumentos acima, conclui-se que a aplicação da função pulsar é uma generalização da resolução, e que o teorema é uma conseqüência de: $((W_d)^*)' = (R^n(W_d))'$.

Exemplos

Nos exemplos a seguir, gerados automaticamente por um programa Lisp que implementa o método proposto, apresenta-se o funcionamento da função pulsar mostrando diversos valores intermediários. Inicialmente, mostra-se o *Conjunto Original* de cláusulas W_d, a seguir o conjunto de *Fragmentos de Substituição*, a *Lista de Substituições* G, e o *Conjunto Dual* de cláusulas $(W_c)'$. Para cada *Substituição* da lista, apresenta-se o *Novo Conjunto* $((W_c)' \bullet \theta_i)_d$, e, finalmente, mostra-se o *Conjunto Dual Dual* $(W_d \cup (\cup_{\theta_i \in G} (W_c)' \bullet \theta_i)_d)'$.

Exemplo 1. O exemplo trivial.

```
Conjunto Original:
¬P(x1) ∧ P(a)
Fragmentos de Substituição:(((x1 . a)))
Lista de Substituições:(((x1 . a)))
Conjunto Dual:
[ ¬P(x1) ∧ P(a) ]
* Substituição: ((x1 . a))
  Novo Conjunto: ()
Conjunto Dual Dual: ()
```

Exemplo 2. Um exemplo mais complexo de conjunto insatisfazível.

```
Conjunto Original:
[ P(a) ∨ Q(x1) ] ∧ [ P(x2) ∨ ¬Q(x2) ] ∧ [ ¬P(x3) ∨ ¬Q(x3) ] ∧
[ ¬P(x4) ∨ Q(x4) ]
Fragmentos de Substituição:(((x1 . x2)) ((x1 . x3)) ((x4 . a))
                            ((x3 . a)) ((x3 . x4))
                            ((x2 . x4)) ((x2 . x3)))
Lista de Substituições :(((x1 . a) (x2 . a)(x3 . a) (x4 . a)))
Estado Dual:
[ P(x2) ∧ ¬P(x3) ] ∨ [ P(a) ∧ ¬Q(x2) ∧ ¬P(x3) ] ∨ [ P(a) ∧ ¬
Q(x2) ∧ ¬P(x4) ] ∨ [ Q(x1) ∧ ¬Q(x2) ]
* Substituição: ((x1 . a) (x2 . a) (x3 . a) (x4 . a))
  Novo Estado: ()
Estado Dual Dual: ()
```

Exemplo 3. Um exemplo de prova automática baseado na definição de "Avô". Observe-se como, além de deduzir que "Avo(a,c)", o sistema deduz também regras específicas para o caso, tais como "¬Pai(x2, a) ∨ Avo(x2, b)", i.e., "se a tiver um pai x2, então este pai será o avô de b, pois Pai(a,b)".

```
Conjunto Original:
[ ¬Pai(x2, x1) ∨ ¬Pai(x1, x3) ∨ Avo(x2, x3) ] ∧
Pai(a, b) ∧ Pai(b, c)
Fragmentos de Substituição:(((x1 . b) (x3 . c))
                            ((x1 . a) (x3 . b))
                            ((x2 . b) (x1 . c))
                            ((x2 . a) (x1 . b)))
Lista de Substituições :(((x2 . a) (x1 . b) (x3 . c))
                         ((x1 . a) (x3 . b))
                         ((x2 . b) (x1 . c)))
Estado Dual:
[ ¬Pai(x2, x1) ∧ Pai(a, b) ∧ Pai(b, c) ] ∨
[ ¬Pai(x1, x3) ∧ Pai(a, b) ∧ Pai(b, c) ] ∨
[ Avo(x2, x3) ∧ Pai(a, b) ∧ Pai(b, c) ]
* Substituição: ((x2 . a) (x1 . b) (x3 . c))
  Novo Estado: Avo(a, c) ∧ Pai(a, b) ∧ Pai(b, c)
* Substituição: ((x1 . a) (x3 . b))
  Novo Estado: [ ¬Pai(x2, a) ∨ Avo(x2, b) ] ∧
               Pai(a, b) ∧ Pai(b, c)
* Substituição: ((x2 . b) (x1 . c))
  Novo Estado: [ ¬Pai(c, x3) ∨ Avo(b, x3) ] ∧
               Pai(a, b) ∧ Pai(b, c)
Estado Dual Dual: [ ¬Pai(x2, x1) ∨ ¬Pai(x1, x3) ∨ Avo(x2,
x3) ] ∧ [ ¬Pai(c, x3) ∨ Avo(b, x3) ] ∧ [ ¬Pai(x2, a) ∨ Avo(x2,
b) ] ∧ Avo(a, c) ∧ Pai(a, b) ∧ Pai(b, c)
```

Exemplo 4. Um exemplo de como o sistema pode ser utilizado para programação em lógica. Apresenta-se a função "append" utilizada nas duas direções possíveis, para gerar a união de duas listas e para deduzir quais duas listas podem ser utilizadas para compor uma lista dada.

A. Resultado da união de duas listas (variável x6):

```
Conjunto Original:
App(nulo, x1, x1) ∧ [ ¬App(x2, x3, x5) ∨ App(Cons(x4, x2), x3,
Cons(x4, x5)) ] ∧ ¬App(Cons(a, nulo), Cons(b, nulo), x6)
Fragmentos de Substituição:(((x4 . a) (x2 . nulo)
                             (x3 . cons(b,nulo))
                             (x6 . cons(a,x5)))
                            ((x2 . nulo) (x1 . x3) (x3 . x5)))
Lista de Substituições :(((x6 . cons(a,x5)) (x2 . nulo)
                          (x4 . a) (x1 . cons(b,nulo))
                          (x3 . cons(b,nulo))
                          (x5 . cons(b,nulo))))
Estado Dual:
[ App(nulo, x1, x1) ∧ ¬App(x2, x3, x5) ] ∨ [ App(nulo, x1, x1)
∧ App(Cons(x4, x2), x3, Cons(x4, x5)) ∧
¬App(Cons(a, nulo), Cons(b, nulo), x6) ]
* Substituição: ((x6 . cons(a,x5)) (x2 . nulo) (x4 . a)
                 (x1 . cons(b,nulo)) (x3 . cons(b,nulo))
```

```
                (x5 . cons(b,nulo)))
  Novo Estado: ()
Estado Dual Dual: ()
```

B. Duas listas (variáveis x6 e x7) que geram uma lista dada:

```
Conjunto Original:
App(nulo, x1, x1) ∧ [ ¬App(x2, x3, x5) ∨ App(Cons(x4, x2), x3,
Cons(x4, x5)) ] ∧ ¬App(x6, x7, Cons(a, Cons(b, nulo)))
Fragmentos de Substituição:(((x6 . cons(x4,x2)) (x3 . x7)
                             (x4 . a) (x5 . cons(b,nulo)))
                            ((x6 . nulo) (x1 . x7)
                             (x7 . cons(a,cons(b,nulo))))
                            ((x2 . nulo) (x1 . x3)
                             (x3 . x5)))
Lista de Substituições :(((x6 . cons(x4 x2) (x2 . nulo)
                          (x1 . cons(b,nulo))
                          (x3 . cons(b,nulo))
                          (x5 . cons(b,nulo))
                          (x7 . cons(b,nulo)) (x4 . a))
                         ((x2 . nulo)
                          (x1 . cons(a,cons(b,nulo))
                         (x3 . cons(a,cons(b,nulo)))
                         (x5 . cons(a,cons(b,nulo)))
                         (x7 . cons(a,cons(b,nulo)))
                         (x6 . nulo)))
Estado Dual:
[ App(nulo, x1, x1) ∧ ¬App(x2, x3, x5) ] ∨ [ App(nulo, x1, x1)
∧ App(Cons(x4, x2), x3, Cons(x4, x5)) ∧ ¬App(x6, x7, Cons(a,
Cons(b, nulo))) ]
* Substituição: ((x6 . cons(x4,x2)) (x2 . nulo)
            (x1 . cons(b,nulo)) (x3 . cons(b,nulo))
            (x5 . cons(b,nulo)) (x7 . cons(b,nulo)) (x4 . a))
  Novo Estado: ()
* Substituição: ((x2 . nulo) (x1 . cons(a,cons(b,nulo)))
              (x3 . cons(a,cons(b,nulo)))
              (x5 . cons(a,cons(b,nulo)))
              (x7 . cons(a,cons(b,nulo))) (x6 . nulo))
  Novo Estado: ()
Estado Dual Dual: ()
```

Comentários

A função pulsar é definida completamente por quatro operadores:

* Transformação dual: $W_d \rightarrow W_c$ (respectivamente $W_c \rightarrow W_d$)

* Eliminação de dominância: $W \rightarrow (W)'$

* Busca de contradições potenciais

* Eliminação de contradições explícitas

O mecanismo de inferência descrito acima pode ser generalizado para lógicas não clássicas simplesmente através da definição dos operadores acima de acordo com a semântica da

lógica a ser considerada. Por exemplo, modificar a busca e eliminação de contradições, de maneira a tornar possível a existência de cláusulas contraditórias, permitiria a extensão do mecanismo para lógicas de mais de dois valores verdade (Belnap, 1977; Patel-Schneider, 1985).

A função pulsar estende o conceito de estratégia de prova: enquanto que ao utilizar a resolução é necessário escolher duas cláusulas dentre todo o conjunto, com a função pulsar é possível escolher um nθmero arbitrário de cláusulas, desde duas, quando a função simula a resolução e as estratégias usuais de busca são aplicáveis, até o conjunto todo, quando não haverá mais nenhum tipo de busca, todas as conseqüências lógicas sendo geradas simultaneamente.

As estratégias de resolução são definidas de maneira a escolher duas cláusulas promissoras para serem testadas, isto implica que apenas relações entre duas cláusulas podem ser utilizadas para guiar tais estratégias. Ao permitir um nθmero arbitrário de cláusulas, permite-se também que relações mais complexas, envolvendo mais do que duas cláusulas, sejam utilizadas para dirigir as estratégias de prova.

Dado um conjunto de variáveis X, uma substituição pode ser considerada como a definição de uma superfície em um espaço n dimensional, onde n = | X |. Dadas duas substituições, a mínima substituição equivalente pode ser obtida pela intersecção das duas superfícies associadas. Caso as duas superfícies sejam disjuntas, as substituições originais são contraditórias.

Para explorar esta interpretação espacial, introduz-se a seguir uma notação geométrica para cláusulas lógicas.

3. Notação Geométrica

O maior problema do método de inferência proposto na Seção 2 é o alto custo da aplicação da distributividade e do teste de dominância. Para contornar este problema, define-se nesta seção uma notação geométrica para cláusulas onde a distributividade pode ser realizada através de propagação e interferência de ondas de informação, e onde a dominância seja facilmente detectada devido ao fato de uma cláusula dominante sempre ser representada por um subespaço das representações de suas cláusulas dominadas.

Seja H o universo de Herbrand e B a base de Herbrand associados ao sistema lógico S. Define-se um mapeamento h, que leva cada tupla formada por elementos do espaço de Herbrand em um nθmero inteiro:

$h : H^* \rightarrow \mathbf{N}$, onde

$H^* = H \cup H^2 \cup H^3 \cup \ldots$

O mapeamento h permite que os predicados possam ser considerados sempre com aridade um, pois cada combinação de

argumentos corresponde a um único elemento de um "Espaço de Herbrand Generalizado" representado pelos números inteiros.

Define-se também uma ordem total entre os predicados:

$\Im : P \rightarrow \{1, \ldots, n\}$

Seja C o conjunto de todas as cláusulas definido acima. Define-se um operador sintático s que dada uma cláusula C e um predicado p retorna a subcláusula que contenha somente as ocorrências deste predicado.

$s : C \times P \rightarrow C_p$

$s(c,p) = c_p$

Seja ainda CG o conjunto de coordenadas generalizadas definido da seguinte maneira:

$CG = \{ (z_1, \ldots, z_n) \mid z_i \in F \}$, onde

$F = \{ z \mid z \in N \rightarrow C \}$, e

C é o conjunto dos nθmeros complexos.

O mapeamento entre o conjunto de cláusulas C e o conjunto CG é realizado através do mapeamento auxiliar f_0. Para representar cláusulas contendo variáveis e funções este mapeamento é definido da seguinte maneira:

$f_0 : C \times P \rightarrow (N \rightarrow C)$

Para $c \in C$, $p \in P$ e $k \in H^*$, $g \in F$, e x uma variável:

$f_0(\Lambda,p) = \underline{0}$, onde $\underline{0}(n) = 0$ para $n \in N$,

$f_0(c,p) = \Sigma_{i=1,\ldots,m} z_i$

onde $s(c,p) = L_1 \cup \ldots \cup L_m$, e para $i = 1, \ldots, m$,

se $L_i = p(k)$ então $z_i = h(k)\ \delta_{h(k)}$,

se $L_i = \neg p(k)$ então $z_i = -h(k)\ \delta_{h(k)}$, onde

δ_i é a função delta centrada em i, e

se $L_i = p(x)$ então $z_i = f_p$,

se $L_i = \neg p(x)$ então $z_i = -f_p$,

se $L_i = p(g(x))$ então $z_i = f_p \circ g$,

se $L_i = \neg p(g(x))$ então $z_i = -f_p \circ g$, onde

$f_p(j) = \delta_j$, para $j \in N$.

Finalmente, define-se o mapeamento f entre C e CG.

$f : C \rightarrow CG$

$f(c) = (f_0(c,p_1), \ldots, f_0(c,p_n))$

Definida esta notação, a semântica correspondente pode ser determinada a partir da semântica da lógica de primeira ordem (Tarski, 1956).

Define-se o mapeamento f^* sobre o conjunto de coordenadas generalizadas, que a cada coordenada generalizada associa um conjunto de funções cujo domínio é o conjunto de coordenadas convencionais n dimensionais, e cuja imagem é o conjunto dos nθmeros complexos.

$f^* : CG \rightarrow (\mathbf{N}^n \rightarrow \mathbf{C})$, onde

$f^*((z_1, \ldots, z_n)) = \varphi_c$, onde

$\varphi_c : \mathbf{N}^n \rightarrow \mathbf{C}$

$\varphi_c((l_1, \ldots, l_n)) = \prod_i z_i(l_i)$, para $l_i \neq 0$

A cada elemento c do conjunto C (ou do conjunto CG) corresponde uma cláusula lógica. A interpretação associada a uma função do conjunto $(\mathbf{N}^n \rightarrow \mathbf{C})$, associada a este elemento, depende do tipo de cláusula. Para cláusulas conjuntivas os valores da função nos pontos fora dos eixos representam conjunções, e para cláusulas disjuntivas estes valores representam disjunções.

A cada cláusula c de C é associada uma *partícula*. Uma partícula é definida pela função de φ_c, à qual é incorporada uma dimensão temporal. Esta função contém toda a informação possível de ser obtida sobre a partícula, e pode ser interpretada como a amplitude de probabilidade da partícula associada a uma cláusula c estar em um determinado ponto **r** de $\mathbf{N}^n$ em um instante t.

$\varphi_c : \mathbf{N}^n \times \mathbf{T} \rightarrow \mathbf{C}$, onde **T** corresponde ao tempo.

A partir da função φ_c define-se a probabilidade dP(**r**,t) da partícula estar, no tempo t, em um elemento de volume d**r**:

$dP(\mathbf{r},t) = | \varphi_c(\mathbf{r},t) |^2 \, d\mathbf{r}$

Define-se um *estado* E_W como o conjunto de funções φ_c associadas às coordenadas generalizadas $c \in W$, onde W é um conjunto de coordenadas generalizadas dado. De acordo com esta definição, um estado é um conjunto de partículas.

A evolução temporal de um estado é definida intuitivamente da seguinte forma: a cada instante um determinado conjunto de partículas, por exemplo as partículas cuja "energia" ultrapasse um certo limiar, é escolhido. Este conjunto emite ondas de informação de tal maneira que o

resultado de sua interferência é a criação de novas partículas que correspondem à representação dual do conjunto de partículas escolhido. Pela própria definição do espaço de abstrações, a propagação sempre se realiza perpendicularmente aos eixos representando predicados com uma nova partícula sendo criada quando um conjunto adequado de ondas se encontram num determinado ponto.

No escopo deste artigo, o conceito de "energia" associado a uma partícula é deixado indefinido. De qualquer modo, este conceito deve integrar, além de aspectos lógicos, aspectos heurísticos associados ao tipo de domínio de aplicação. Em Bittencourt (1993) é discutida a possibilidade de associar o conceito de energia a uma medida de adequação de um organismo cognitivo atuando em um ambiente governado pelos princípios da teoria da evolução.

Para eliminar as cláusulas dominadas basta que todas as partículas enviem uma onda de eliminação em direção ao seu superespaço. Cada partícula que receber uma destas ondas será eliminada, restando apenas as partículas que correspondem a cláusulas que não são dominadas por nenhuma outra do conjunto. Estas partículas, por sua vez, emitem ondas cuja interferência cria um novo conjunto de partículas, desta vez do mesmo tipo das partículas do estado inicial, e que traduzido em termos de cláusulas lógicas, corresponde às cláusulas resultantes da aplicação da função "pulsar" sobre as cláusulas associadas às partículas escolhidas inicialmente.

Limitando-se o número de predicados, este processo pode ser implementado facilmente em uma arquitetura do tipo hipercubo, pois as comunicações entre partículas são totalmente locais, cada partícula necessitando apenas das informações locais do campo de informações determinado pelo estado atual para definir seu comportamento.

4. Conclusão

Apresentou-se um método de inferência para lógica de primeira ordem baseado em transformações entre formas canônicas duais. Introduziu-se ainda uma notação geométrica para as cláusulas capaz de explorar o paralelismo inerente ao método de inferência proposto. Uma versão seqüencial do método de inferência foi implementada utilizando a linguagem Common Lisp (Steele Jr., 1984) em uma estação de trabalho SUN.

O modelo para a lógica apresentado é parte de uma proposta mais ambiciosa para a modelagem de processos cognitivos (Bittencourt, 1991, 1992 e 1993). Nesta proposta o modelo lógico dedutivo é utilizado como um mecanismo de inferência ao qual está associada uma memória e um mecanismo de interação com o ambiente externo baseado em redes neurais.

O comportamento pulsante do método de inferência proposto e a estrutura ondulatória do formalismo introduzido para modelá-lo apresenta uma certa analogia com propostas de modelos biológicos para a capacidade cognitiva humana (Pribram, 1971).

No futuro, pretende-se prosseguir com o desenvolvimento formal desta analogia, utilizando, por exemplo, as ferramentas matemáticas desenvolvidas na mecânica quântica para tratar estados de partículas.

Outra direção de pesquisa consiste na integração do mecanismo de inferência proposto com uma interface com o mundo real através de redes neurais. Basicamente as redes neurais serviriam para classificar as situações percebidas no mundo. As conclusões obtidas por inferência seriam utilizadas como realimentação tanto para otimizar a classificação como para realizar previsões de seqüências de eventos.

Referências

BELNAP, N.D., *A Useful Four-Valued Logic*. In "J.M. Dunn fand G. Epstein (Editors), Modern Uses of Multiple-Valued Logics", D. Reidel Pub. Co., 1977.

BITTENCOURT, G. *A Connectionist-Symbolic Model for Cognition*. In "Springer Verlag Lecture Notes in Artificial Intelligence, Proceedings of the 1993 International Symposium on Methodologies for Intelligent Systems", Trondheim, Norway, June 15-18, 1993.

BITTENCOURT, G. *Seleção Natural e Física de Partículas: Uma direção de pesquisa em Inteligência Artificial*. 8o. Simpósio Brasileiro de Inteligência Artificial, pp. 211-216, Universidade de Brasília, Brasília, 18 a 21 de novembro de 1991 (INPE - 5335 - PRE/1734, dezembro de 1991).

BITTENCOURT, G. *Um modelo paralelo para a lógica baseado em propagação de ondas*. XV Congresso Nacional de Matemática Aplicada e Computacional (CNMAC), São Carlos, SP, 07 a 10 de setembro de 1992 (INPE - 5430 - PRE/1764, 1992).

CHASSIN DE KERGOMMEAUX, J. and ROBERT, P., *An Abstract Machine to Implement OR-And Parallel Prolog Efficiently*. The Journal of Logic Programming, Vol. 8, No. 3, pp. 249-264, May 1990.

CHEN, W.-T. and FANG, M.-Y., *An Efficient Procedure for Theorem Proving in Propositional Logic on Vector Computers*. Parallel Computing (North-Holland). Vol. 17, pp. 983-5, 1991.

CIEPIELEWSKI, A., HARIDI, S. and HAUSMAN, B., *Or-Parallel Prolog on Shared Memory Multiprocessors*. The Journal of Logic Programming, Vol. 7, No. 2, pp. 125-147, September 1989.

CLARK, K. and STICKEL, S., *Predicate Logic: A Calculus for Deriving Programs*. Proceedings of IJCAI 5, pp. 419-420, 1977.

DAHL, V., *Logic Programming as Representation of Knowledge*. IEEE Computer, pp. 106-111, October 1983.

EISINGER,N., *What You Always Wanted to Know about Clause Graph Resolution*. Proceedings of the 8th ICAD, LNCS No. 230, pp. 316-336, 1986.

GANANTHARAJ, R. and MUELLER, R.A., *Parallel Theorem Proving with Connection Graphs*. Proceedings of the 8th ICAD, LNCS No. 230, pp. 337-352, 1986.

MARTINEZ, T.R. and VIDAL, J.J., *Adaptative Parallel Logic Networks*. Journal of Parallel and Distributed Computing, Vol. 5, No. 1, pp. 26-58, February 1988.

NEEL-SCHNEIDER, P.F., *A Decidable First-Order Logic for Knowledge Representation*. Proceedings of IJCAI 9, pp. 455-458, 1985.

PRIBRAM, K., *Language of the Brain: Experimental Paradoxes and Principles in Neuropsychology*. Englewood Cliffs, N.J., Prentice Hall, 1971.

ROBINSON, J.A., *A Machine-Oriented Logic Based on the Resolution Principle*. Journal of the ACM, Vol. 12, No. 1, pp. 23-41, January 1965.

RUMELHART, D.E. and McCLELLAND, J. (Editors), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. M.I.T. Press, Cambridge, MA, 1986.

RUSSELL, B. and WHITEHEAD, A.N., *Principia Mathematica*. Vol. I, Cambridge U.P., 1913.

SCHER, R., *Optimizing the Clausal Normal Form Transformation*. Journal of Automated Reasoning, Vol. 7, No. 3, pp. 325-336, February 1992.

STEELE Jr., G.L., *Common LISP*. Digital Press, Burlington, 1984.

STERLING, L. and SHAPIRO, E., *The Art of Prolog*. M.I.T. Press Series in Logic Programming, Cambridge, MA, London, England, 1986.

STICKEL, M.E., *An Introduction to Automated Deduction*. In "Fundamentals of Artificial Intelligence", LNCS No. 232, Springer-Verlag, Berlin Heidlberg New York Tokyo, 75, 132, 1986.

TARSKI, A., *The Concept of Truth in Formalized Languages*. In "A. Tarski (Editor), Logic Semantics and Mathematics, Oxford, Clarendon Press, 1956.

VAN EMDEN, M.H. and KOWALSKI, R.A., *The Semantics of Predicate Logic as a Programming Language*. Journal of the ACM, Vol. 23, No. 4, pp. 733-742, October 1976.