Um Algoritmo para a Unificação na Lógica de Ordem Superior

Artêmio Ludwig
Departamento de Informática
Universidade Federal de Viçosa
36570-000 Viçosa (MG) – Brasil

Wagner C. do Amaral
Faculdade de Engenharia Elétrica – C.P. 6101
Universidade Estadual de Campinas – UNICAMP
13081-000 Campinas (SP) – Brasil

20 julho 1992

Resumo **A implementação de interpretadores para a linguagem da Lógica de Ordem Superior (LOS) ainda conta com dois desafios: (a) tornar a linguagem mais *amigável* permitindo estimular sua adoção e (b) implementar interpretadores cujo desempenho não sacrifique sua usabilidade.**

Este trabalho é resultado da prototipação de um interpretador para a linguagem da LOS chamado λ-Prolog, no qual são tomadas apenas as sentenças definidas positivas. Nela a unificação é um dos procedimentos mais importantes e apresenta complexidade elevada. Seu processamento é razão de depauperação na qualidade do sistema. Partindo-se dos algoritmos SIMPL e MATCH já conhecidos, propõe-se um algoritmo alternativo para MATCH com indicações de melhor desempenho e com uma interpretação mais facilitada do mecanismo da unificação.

Palavras-Chave: Unificação, Match, Lógica de Ordem Superior.

1 Introdução

Excluindo a parte de inicialização, que inclui a conferência gramatical das sentenças, a formação da base de conhecimento e a tipificação de constantes e variáveis, a implementação do interpretador para a LOS tem procedimentos similares àqueles do interpretador para a primeira ordem. Algumas partes são específicas deste sistema. Evidenciam-se dois aspectos:

1. A unificação que na LOS não apresenta um *mgu*.
2. O retroencadeamento conta com mais uma razão para acontecer.

A complexidade da unificação constitui-se num desestímulo para avanços no uso da linguagem da LOS, quer por falta de seu entendimento quer por suas dificuldades computacionais.

Aborda-se, na sequência , o algoritmo da unificação especificado em [1] e [4] , (Fig.1) apresentado com mais detalhes por [1]. Os mesmos autores formalizam a linguagem sobre a

*Trabalho parcialmente financiado pela Fundação de Amparo à Pesquisa de Minas Gerais.

qual o algoritmo foi proposto e trabalha-se com a notação e as fórmulas definidas positivas especificadas em [4]. Em particular busca-se analisar a *imitação* e a *projeção*. Espera-se que o leitor esteja familiarizado com a terminologia usada.

Para o estudo de MATCH e a proposição de novos algoritmos considera-se que o primeiro par da esquerda do conjunto discórdia sempre é tomado para dar sequência ao processo de unificação. Adota-se a caracterização de "unificador parcial" para a composição dos unificadores obtidos até determinado nó da árvore de unificação, chamado ainda de unificador para o ramo. O unificador que permite a passagem de um nó para outro é chamado de unificador do arco. Usa-se ainda os símbolos $\bar{w}$ e $\bar{x}$ para representar as sequências $w_1 \cdots w_r$ e $x_1 \cdots x_n$, respectivamente.

Na próxima seção são descritas algumas características do algoritmo MATCH. Três lemas são enunciados e constituem a base para o método de obtenção dos unificadores "principais" que são assunto para a seção 3. A seção 4 apresenta a combinação de unificadores que é uma proposta para se obter os demais unificadores da árvore de unificação de MATCH. Os resultados de algumas aplicações dos algoritmos propostos são vistos na seção 5 e as conclusões encontram-se na seção 6.

2 Características Gerais

Considera-se $\langle F_1, F_2 \rangle$ um par de discórdia com duas fórmulas de mesmo tipo, uma flexível e outra rígida assim descrito:

$$\langle F_1, F_2 \rangle = \langle\, f\, A_1 \cdots A_r \;,\; C\, B_1 \cdots B_s \rangle$$

Os três lemas que se seguem, cujas demonstrações podem ser vistas em [2], usam como suporte a árvore de unificação gerada por MATCH. Os algoritmos SIMPL e MATCH extraídos de [3] estão colocados na Fig. 1.

Lema 2.1 *Uma sequência de imitações aplicadas a qualquer par de discórdia*

$$\langle f A_1 \cdots A_r, C\, B_1 \cdots B_s \rangle$$

produz um ramo de sucesso cujo unificador parcial é da forma

$$\theta_n = f/\lambda \bar{w}\; C\, B_1 \cdots B_n\, (h_{n+1}\bar{w}) \cdots (h_s \bar{w})$$

e o número total de arcos np(f) para atingir este nó da árvore de unificação é definido por:

- $np(f) = 1$ *se* $r = 0$,
- $np(f) = 1 + \sum_{i=1}^{r} np(A_i)$ *se* $r > 0$.

O lema não considera os pares flexível-flexível que podem surgir na árvore de unificação. O algoritmo MATCH está proposto apenas para pares flexível-rígido e seu resultado consiste de um unificador onde o conjunto D pode ser vazio, F (de falso , caso em que não houve unificação) ou contém apenas pares flexível- flexível. Nosso método obterá sempre como resultado um unificador com $D = \emptyset$ ou $D = F$.

Na projeção, segundo o algoritmo MATCH, quando um dos argumentos de F_1 é funcional,

Definição da Rotina SIMPL: Seja o conjunto de discórdia D constituido de pares de fórmulas em sua forma normal de unificação (FNU).

1) Se D = $\emptyset$ então SIMPL(D) = $\emptyset$.

2) Se D = $\{\langle F_1, F_2\rangle\}$ e
   a) F_1 é flexível então SIMPL(D) = D; caso contrário
   b) Se F_2 é flexível então SIMPL(D) = $\{\langle F_2, F_1\rangle\}$;
   c) De outro modo F_1 e F_2 são ambos rígidos. Sejam $\lambda\bar{x}(C^1A^1\cdots A^r)$ e $\lambda\bar{x}(C^2B^1\cdots B^s)$ as formas normais de unificação para F_1 e F_2. Se $C^1 \neq C^2$ SIMPL(D) = F, caso contrario SIMPL(D) = SIMPL($\{\langle\lambda\bar{x}.A^i, \lambda\bar{x}.B^i\rangle$, $1 \leq i \leq r\}$)

3) De outro modo D tem no mínimo dois membros. Seja D = $\{\langle F^i, G^i\rangle\}$, $1 \leq i \leq n\}$,
   a) Se SIMPL($\{\langle F^i, G^i\rangle\}$) = F para algum i então SIMPL(D) = F
   b) Senão SIMPL(D) = $\bigcup_{i=1}^{n}$ SIMPL($\{\langle F^i, G^i\rangle\}$).

Definição da rotina MATCH: Sejam V um conjunto de variáveis, F_1 e F_2 duas fórmulas na FNU $\lambda\bar{x}(f\,A^1\cdots A^r)$ e $\lambda\bar{x}(C\,B^1\cdots B^s)$, de mesmo tipo, flexível e rígida, respectivamente. Além disto, seja $\alpha_1 \rightarrow \cdots \rightarrow \alpha_r \rightarrow \beta$ o tipo de f e para $1 \leq i \leq r$, seja w_i uma variável do tipo α_i

i) Se C é uma variável (isto é C aparece em $\bar{x}$), então IMIT(F_1, F_2, V) = $\emptyset$; Caso contrário (C é constante) sejam $h^i \notin$ V $\cup$ $\{\bar{w}\}$ variáveis de tipos adequados para $1 \leq i \leq s$, isto é, do mesmo tipo de $B^1\cdots B^s$ então IMIT(F_1, F_2, V) = $\{\langle f, \lambda\bar{w}(C(h^1\bar{w})\cdots(h^t\bar{w}))\rangle\}$.

ii) Para $1 \leq i \leq r$, se α_i não for da forma $\beta_1 \rightarrow \cdots \rightarrow \beta_t \rightarrow \beta$ então $PROJ_i(F_1, F_2,$ V) = $\emptyset$; Caso contrário, sejam $h^i \notin V \cup \{\bar{w}\}$ variáveis de tipos adequados para $1 \leq i \leq t$, então $PROJ_i(F_1, F_2, V) = \{\langle f, \lambda\bar{w}(w_i\,(h^1\,\bar{w})\cdots(h^t\,\bar{w}))\rangle\}$.

iii) MATCH(F_1, F_2, V) = IMIT(F_1, F_2, V) $\cup$ $\bigcup_{1\leq i\leq r} PROJ_i(F_1, F_2, V)$

Figura 1: Definição das rotinas SIMPL e MATCH [3]

o unificador gerado para o arco é

$$\lambda\bar{w}.w_i(h_1^i\bar{w}\cdots h_t^i\bar{w})$$

onde t é o número de variáveis ligadas por λ no argumento A_i projetado.

Este unificador existirá se houver unificação entre A_i aplicado a

$$\xi = ((f/h_1^k)F_1, (f/h_2^k)F_1, \cdots, (f/h_t^k)F_1)$$

e F_2, isto é, A_i aplicado a uma série de t argumentos $(f/h_i^k)F_1$, tantos quantas forem as variáveis de ligação de F_1. A expressão $(f/h_i^k)F_1$ representa F_1 com f substituido por h_i^k. Doravante representa-se esta aplicação por $\xi(A_i)$. Para argumentos A_i não funcionais, não haverá variáveis de ligação e $\xi(A_i) = A_i$.

Sujeito pois a condição de que exista um ou mais unificadores para $\langle \xi(A_i), F_2 \rangle$ o unificador parcial inicial obtido por projeção é então

$$\{\ f/\lambda\bar{w}\ C[\ w_i(h_i^i w_1 \cdots w_o) \cdots (h_i^1 w_1 \cdots w_o)\ ]\ \}.$$

representando a projeção do i-ésimo argumento de f.

Lema 2.2 *Uma projeção de A_i de F_1 sobre F_2, após imitações sucessivas sobre os k-1 A_i anteriores, ($k \geq 2$), gera unificadores parciais que têm a forma:*

$$\{\ \phi(f/\lambda\bar{w}\ C\ B_1 \cdots B_{k-1}\ (w_i\ (h_1^k\bar{w}) \cdots (h_i^k\bar{w}) \cdots (h_s\bar{w})) \cup \phi\ \},$$

em que $\phi = unif(\langle \xi(A_i), B_k \rangle)$, unif representa um dos conjuntos de unificadores para o par, k a posição que $(w_i(h_1^k\bar{w}) \cdots (h_i^k\bar{w}))$ ocupou como argumento na substituição para f e t, o número de variáveis ligadas de B_k, não iguais à sua cabeça.

A segunda parte do conjunto pode ser vazia ou ser um conjunto de substituições, indicando se houve sucesso na projeção.

Lema 2.3 *Qualquer unificador parcial θ_n, obtido por meio do uso dos lemas 1 e 2, depois de obtidos os unificadores ϕ, pode ser transformado em um unificador de $\langle F_1, F_2 \rangle$ pela substituição de cada termo cuja cabeça é h_i, ($1 \leq i \leq r$) existente no unificador parcial por seu correspondente $\theta_n(B_i)$. O termo $\theta_n(B_i)$ representa a aplicação do unificador parcial a B_i.*

Este lema é consequência da possibilidade de se efetuar tantas imitações quantas necessário após qualquer nó, até que não haja mais termos h_i.

3 Otenção dos Unificadores "Principais"

Como consequência dos lemas anteriores pode-se ter, a partir da primeira imitação, unificadores para f que são a fórmula F_2 abstraida de $\lambda\bar{w}$ com qualquer $w_i[\cdots]$ substituindo um dos argumentos de F_2.

Método similar ao usado para derivar os lemas 2.1, 2.2 e 2.3 permite mostrar que os termos $w_i[\cdots]$ podem substituir qualquer argumento i de $\lambda\bar{w}\ F_2$ tornando-se um unificador para o par, independente da profundidade do argumento dentro do unificador parcial.

Os lemas anteriores e suas extensões levam à obtenção dos seguintes unificadores para f:

1. $\lambda\bar{w}\ F_2$ no qual não há termos $w_i[\cdots]$ como argumentos.
2. $\lambda\bar{w}\ C\ \phi(B_i \cdots w_i[\cdots h_j^k \cdots] \cdots B_o)$ com apenas um $w_i[\cdots h_j^k \cdots]$ substituindo um dos argumentos de F_2 , ($1 \leq j \leq t$) e ϕ = { outras substituições obtidas}.
3. $\lambda\bar{w}\ C\ \phi(B_i \cdots\cdots B_o)$ com apenas um $w_i[\cdots]$ substituindo recursivamente um dos argumentos de B_i e ϕ = {outras substituições obtidas}.

O universo dos unificadores de $\langle F_1, F_2 \rangle$ pode conter ainda qualquer combinação dos unificadores acima listados.

Para melhor entendimento considere-se ϕ como condicionador para a existência de um dos unificadores listados em 2 e 3 acima. Ou seja, ϕ estipula uma lista de substituições necessárias para que exista o unificador.

Como pode haver $w_i[\cdots h_i^k \cdots]$ em diversas posições, a árvore de unificação permite observar que se existirem dois unificadores com $w_i[\cdots h_i^k \cdots]$ em posições diferentes então pode-se obter um unificador que contenha os dois $w_i[\cdots h_i^k \cdots]$ desde que seja possível uma combinação entre os unificadores ϕ correspondentes à unificação dos pares $\langle A_i, B_k \rangle$. Em outras palavras, se os condicionadores puderem funcionar simultaneamente. O conceito de "unificador mais geral" está sendo usado nesta "acomodação". Dois unificadores podem ser combinados se houver um unificador mais geral que os absorva.

A proposição a seguir indica a condição de existência de um unificador combinado.

Proposição 3.1 *Sejam os unificadores entre* $\langle A_i, B_k \rangle$ *e* $\langle A_j, B_l \rangle$, $(l \neq k)$ *representados por*

$$\phi_1 = \{v_1/tv_1, v_2/tv_2 \cdots v_m/tv_m \} \text{ e } \phi_2 = \{u_1/tu_1, u_2/tu_2 \cdots u_l/tu_l\}.$$

Seja ainda definida a pseudo-composição dos dois unificadores como $\phi_1 \diamond \phi_2$, *que é a composição sujeita a não eliminação das substituições* u_i/tu_i *quando* $u_i = v_j$ *para qualquer* i *e* j. *Estes pares de substituição devem ser substituidos por uma substituição mais geral formada a partir das duas. Se isto for possível então tem-se o unificador*

$$\{\phi_2(\phi_1(f/\lambda\bar{w}\ C\ B_i \cdots w_i[\cdots h_i^k \cdots] \cdots w_j[\cdots h_i^l \cdots] \cdots B_n)) \cup \phi_1 \diamond \phi_2 \}.$$

Usa-se na proposição o conceito de que ϕ_1 e ϕ_2 são condições para que existam os unificadores $w_i[\cdots h_i \cdots]$ nas posições l e k respectivamente. Subentende-se também que a pseudo-composição é comutativa pois representa apenas a união dos dois unificadores.

A proposição indica que basta obter os unificadores parciais ϕ_1 e ϕ_2 e testar se os mesmos se "combinam".

Considere-se os ramos da árvore de unificação cujo primeiro arco é uma projeção. A intenção com estas projeções iniciais é verificar a possibilidade de se unificar pares de discórdia da forma $\langle A_i, F_2 \rangle$.

Obedecidas as condições de tipo, os primeiros unificadores parciais serão:

$$f/\lambda\bar{w}.w_i(h_1\bar{w}) \cdots (h_t\bar{w})$$

onde t tem o significado a ele atribuído anteriormente.

O primeiro conjunto discórdia resultante destas projeções é (já descrito no lema 3) constituída de t pares de discórdia da forma $\langle (h_i^k A_1 \cdots A_r), B_i \rangle$

O procedimento de obtenção dos unificadores para estas projeções segue recursivamente os conceitos estabelecidos nos lemas anteriores. No entanto, nestes casos não pode haver combinação pois a substituição para f tem uma única possibilidade para $w_i[\cdots h_i^k \cdots]$

Tem-se agora condições de visualisar um algoritmo geral para a unificação do par $\langle F_1, F_2 \rangle$.

Define-se a seguinte estrutura de árvore para a fórmula F_2: O nó raiz é rotulado por F_2. Recursivamente os nós filhos são os argumentos do nó pai. Os nós são enumerados da seguinte forma. A raiz (F_2) tem número 0 e é sub-entendida como argumento 0 (zero). Os nós subsequentes herdam a numeração do nó pai seguida do número sequencial do argumento que eles representam.

Desta forma o número 031 para o nó representa que ele é o primeiro argumento do terceiro argumento da raiz. Em cada nó, o número que o segue indica o número de abstrações daquele termo. Para cada arco, a existência de um rótulo i indica que seu nó está ligado à i-ésima abstração, sequencialmente até aquele arco da árvore.

Considere dois valores i e j, onde i representa o índice de um argumento A_i e j o número de um nó qualquer da estrutura de F_2. São unificadores para o par $\langle F_1, F_2 \rangle$, gerados por imitação ou projeção, para $(m \geq 0)$

1. Para $1 \leq i \leq r$ e $\forall\, j$ ($j \in$ {nós da árvore F_2 }) os unificadores ij são
   $\{\phi(f/\lambda \bar{w}\ C\ B_1 \cdots w_i[\cdots h_i^j \cdots] \cdots B_s) \cup unif_m(\langle A_i,$ termo do nó j $\rangle)\ \}$
   O índice j representa qual termo da fórmula F_2 deve ser substituido por $w_i[\cdots h_i^j \cdots]$ e ϕ representa $unif_m$. Se i = 0, então $f/\lambda \bar{w}\ CB_1 \cdots B_s$ é o unificador ij.

2. Para $1 \leq i \leq r$ e j=0, o unificador ij é
   $\{\phi(f/\lambda \bar{w}.w_i(h_1^k A_1 \cdots A_r) \cdots (h_t^k A_1 \cdots A_r)) \cup unif_m(\langle A_i, F_2 \rangle\ \}$
   em que t é o número de variáveis ligadas em A_i e ϕ representa $unif_m(\langle A_i, F_2 \rangle)$

Figura 2: Algoritmo ADF para obtenção dos unificadores "principais", que são o resultado de imitações e de i ($0 \leq i \leq 1$) projeções.

A Fig. 2 contém o algoritmo que obtém os unificadores nos quais apenas um arco foi obtido por meio de projeção.Ele utiliza a estrutura de árvore proposta e os obtém por meio da unificação dos argumentos de F_1 com os nodos da árvore de F_2.

Nas duas condições do algoritmo da figura 2, a união com $unif$ resulta numa nova aplicação do algoritmo, como será visto adiante. Não existirá o unificador ij se não houver unificação do par sujeito a $unif$. Este algoritmo é obtido pela aplicação direta do lema 3.

4 Combinação de Unificadores

A idéia sobre a qual se elabora o conceito de combinação de unificadores parte da seguinte observação que já foi exposta na propriedade 2. Sejam

$$\phi_1 = g/\lambda w_1 w_2\ C\ w_1\ 2 \text{ e } \phi_2 = g/\lambda w_1 w_2\ C\ 3\ w_2$$

dois unificadores para um par de discórdia. Neste caso $g/\lambda w_1 w_2 C\ w_1\ w_2$ também é um unificador para o par. Intuitivamente percebe-se que o primeiro argumento de F_2 é 3 e o segundo é 2.

Em um grande número de casos o unificador é um conjunto de substituições com as variáveis livres do par de discórdia à esquerda da substituição. Dois unificadores, para se combinarem precisam ter os unificadores de suas variáveis livres combinados.

O algoritmo COMB (Fig. 3) é usado para testar se duas substituições se combinam gerando uma terceira. Nota-se que se em ξ_1 não houver $w_i[\cdots h_i^b \cdots]$ não há combinação.

Para se combinar dois unificadores considerando-os como conjuntos de substituições para as variáveis livres de F_1 e F_2 observa-se se as substituições com a mesma variável livre à esquerda se combinam. O algoritmo COMBT (Fig. 4) foi proposto com esta finalidade .

Obtidos os unificadores principais, os demais unificadores do par de discórdia são todas

Sejam ξ_1 e ξ_2 duas substituições par uma variável livre g.

Em ξ_1 localize as posições onde há $w_i[\cdots h_i^k \cdots]$ na substituição para g (exceto no conjunto de variáveis de ligação). Use para isso a estrutura de árvore, por exemplo.

1. COMB := $\emptyset$.
2. Copie ξ_2 para ξ_3.
3. Para cada $w_i[\cdots h_i^k \cdots]$ existente no nó ijkl... da árvore da substituição para g de ξ_1

   3.1 Verifique se os nós anteriores do ramo (formado pela sequência de nós i, ij, ijk, ...($i \geq 0$) de ξ_3) não estão ocupados por algum $w_i[\cdots h_i^k \cdots]$, $(1 \leq i \leq r)$.

   3.2 Se estiver, a combinação fracassa. COMB := $\emptyset$. Vá para 4. Caso contrário acrescente $w_i[\cdots h_i^k \cdots]$ na posição ijkl... da correspondente substituição de ξ_3. COMB := ξ_3.
4. Fim {COMB é a combinação entre ξ_1 e ξ_2.}

Figura 3: Algoritmo COMB para cobinação de duas substituições para a mesma variável livre gerando um unificador mais geral.

Sejam ϕ_1 e ϕ_2 unificadores ou conjuntos de substituições.

1. COMBT := $\emptyset$.
2. $\phi_3 := \emptyset$.
3. Para cada substituição $\xi = y/t_{\phi_i}$, y pertencente às variáveis livres, $\xi \in \phi_i$, $(1 \leq i \leq 2)$,
   (a) Se não existir uma substituição y/t_{ϕ_j} em $\phi_j, j \neq i$, $\phi_3 := \phi_3 \cup \{y/t_{\phi_i}\}$.
   (b) Se $t_{\phi_1} = t_{\phi_2}$ então $\phi_3 := \phi_3 \cup \{y/t_{\phi_1}\}$.
   (c) Senão se $t_{\phi_1} \neq t_{\phi_2}$ e $COMB(\xi_{\phi_1}, \xi_{\phi_2}) = \emptyset$ então $COMBT(\phi_1, \phi_2) := \emptyset$. Va para Fim.
   Caso contrário $\phi_3 := \phi_3 \cup \{COMB(\xi_{\phi_1}, \xi_{\phi_2})\}$.
4. Fim.

Figura 4: Algoritmo COMBT para combinação de dois unificadores para um par de discórdia de $\langle F_1, F_2 \rangle$.

as combinações entre os unificadores eles. Define-se na figura 5 o algoritmo UNIF($\langle F_1, F_2 \rangle$) que obtém todos os unificadores para o par

A consequência final da aplicação do algoritmo UNIF, após a aplicação de ADF é obter um conjunto de unificadores para o par. Mostrando-se que: (1) todos os unificadores finais θ

Seja P(ADF) o conjunto potência de ADF($\langle F_1, F_2 \rangle$).

Para cada elemento de P(ADF), combine seus elementos da seguinte maneira. Seja um elemento γ de P(ADF) = $\{\theta_1 \cdots \theta_n\}$

1. n = 0 então UNIF($\langle F_1, F_2 \rangle$:= UNIF($\langle F_1, F_2 \rangle \cup \gamma$.
2. Se n = 1 então UNIF($\langle F_1, F_2 \rangle$) := UNIF($\langle F_1, F_2 \rangle$) $\cup$ θ_1
3. Se n = 2 então UNIF($\langle F_1, F_2 \rangle$) := UNIF($\langle F_1, F_2 \rangle$) $\cup$ $COMBT(\theta_1, \theta_2)$
4. Se n$\geq$ 3 então UNIF($\langle F_1, F_2 \rangle$) := UNIF($\langle F_1, F_2 \rangle$) $\cup$ CC onde CC é obtido por
   CC := $\emptyset$,
   Para i = 1 até n CC :=$COMBT(CC, \theta_i)$.

Figura 5: Algoritmo de Unificação geral.

da árvore de unificação que contêm apenas um argumento da forma $w_i[\cdots h_i^k \cdots]$ são gerados pelo algoritmo ADF e são os unificadores chamados de "principais", e que (2) os demais são combinações daqueles obtidos em 1, sendo a combinação definida pelo algoritmo COMBT anteriormente definido, chega-se a conclusão de que UNIF gera os mesmos unificadores que MATCH.

5 Aplicações dos Algoritmos

Utiliza-se a seguir um exemplo tirado de [3] para estabelecer, ainda que sobre alguns aspectos apenas, um contraste entre a utilização da combinação dos algoritmos MATCH e SIMPL e o algoritmo UNIF, aqui proposto. Naquele exemplo tinha-se o seguinte par $\langle F_1, F_2 \rangle$, rígido-rígido

{((mapfun F_1 (cons X L_1) (cons (F_1 X) L_2)),
(mapfun F_2 (cons 1 (cons 2 nil))
(cons (g 1 1) (cons (g 1 2) nil))))}

que por meio da rotina SIMPL gerou o seguinte conjunto discórdia:

$\{\langle F_1, F_2 \rangle, \langle X, 1 \rangle, \langle L_1, (cons\ 2\ nil) \rangle, \langle (F_1\ X), (g\ 1\ 1) \rangle, \langle L_2, (cons\ (g\ 1\ 2)\ nil) \rangle \}$

Os pares flexível-rígido do conjunto são:

$\langle X, 1 \rangle$, $\langle L_1, (cons\ 2\ nil) \rangle$, $\langle (F_1\ X), (g\ 1\ 1) \rangle$ e $\langle L_2, (cons\ (g\ 1\ 2)\ nil) \rangle$.

Apenas o terceiro par não tem ADF unitário. Os demais não têm combinações e UNIF = ADF com os unificadores

$X/1$, $L_1/(cons\ 2\ nil)$ e $L_2/(cons\ (g\ 1\ 2)\ nil)$

respectivamente. Para se encontrar estes unificadores não houve qualquer criação de termos. Apenas verificou-se a adequação de tipo. Geração de variáveis h_i não foi necessária, tampouco β-reduções.

O terceiro par foi alterado para $\langle (F_1\ 1), (g\ 1\ 1)\rangle$. Nele é gritante a vantagem de UNIF pois bastou a comparação de $\langle 1, 1\rangle$ e posteriormente $\langle 1, 1\rangle$ para se obter os três unificadores principais que são

$$\{\ F_1/\lambda w_1(g\ 1\ 1),\ F_1/\lambda w_1(g\ 1\ w_1),\ F_1/\lambda w_1(g\ w_1\ 1)\}$$

e a combinação gerou o quarto unificador que é $\{\ F_1/\lambda w_1(g\ w_1\ w_1)$.

Apesar de não representar uma regra geral, esta unificação não usa β-reduções, geração de novas variáveis livres, consulta a tipos para geração de variáveis, controle de estrutura de árvore para posterior composição dos unificadores parciais, e outros mecanismos necessários para a geração do unificador.

A unificação é feita apenas para os unificadores chamados de "principais". Os demais são conseguidos por meio de mecanismos de combinação e dispensam toda a mecânica de pesquisa incluida em MATCH. O número máximo de argumentos principais é função do número de argumentos de F^2, em qualquer profundidade, e pode ser melhor visto pela estrutura de árvore proposta. Em muitos casos ele não atinge a metade do número total de unificadores.

Como um exemplo, a unificação do par $\langle f\ 1\ (T\ 2)\ (y\ 2), C\ 1\ (P\ 1)\ (y\ 2)\rangle$ produs 12 unificadores principais. O número total de unificadores está em torno de 50, dependendo da forma como se processam os termos flexível-flexível.

6 Conclusões

Os exemplos precedentes representam um sinal de que o algoritmo UNIF usado para a unificação de pares de discórdia deve ser analisado como uma alternativa para MATCH. Ele aponta para uma melhora significativa de desempenho. Embora testes exaustivos devem ainda ser feitos antes de quaisquer conclusões, nossas observações preliminares mostram que existem vantagens que devem ser exploradas.

Além da redução representada pela diminuição do número de unificadores pesquisados, o próprio tabalho de busca destes unificadores é consideravelmente reduzido no algoritmo UNIF. Este fato pode ser observado porque

- β-reduções são quase desnecessárias.
- Geração de novas variáveis livres só ocorrem quando há abstração de algum argumento de F^1.
- Não existe repetição de segmentos na árvore de unificação.
- A rotina SIMPL é simplificada

Restritas a observações empíricas preliminares, estas vantagens conferem ao algoritmo a característica promissora indicada no inicio desta seção. Recomenda-se no entanto uma formalização teórica mais profunda que conduza a propriedades como consistência e completude.

O aprofundamento da análise deste método de unificação sinaliza que mesmo a grande preocupação com tipos existente em MATCH poderia ser significativamente reduzida. A

confirmação desta conjectura pode representar um passo a mais sobre um dos parâmetros que influenciam negativamente sobre o desempenho dos interpretadores.

7 Bibliografia

[1] G. P. Huet, "A Unification Algorithm for Typed Lambda-Calculus," *Theoretical Computer Science*, vol 1, pp 27-57, 1975.

[2] A. Ludwig, "Um Interpretador para o λ-Prolog", Tese de Doutorado, Universidade Estadual de Campinas, SP, *(to appear)*.

[3] G. Nadathur, *A Higher Order Logic as The Basis for Logic Programing.* PhD Dissertation, University of Pennsylvania, 1987.

[4] G. Nadathur and D. Miller, "Higher-Order Horn Clause," *JACM*, 37:4, pp. 777-814, 1990.