Lógica Computacional (CC2003)

Nelma Moreira

Lógica Computacional 24

Conteúdo

1 Introdução à Programação em Lógica

1.1 Resolução SLD

Resolução para cláusulas de Horn

Seja $\mathcal{P}$ um programa (conjunto de cláusulas de Horn positivas) e G uma cláusula negativa (objectivo),

$$\leftarrow \beta_1, \ldots, \beta_n$$

$(= \forall \neg\beta_1 \vee \cdots \neg \vee \beta_n)$.

Temos que

$$\mathcal{P} \models \exists \beta_1 \ \wedge \ \cdots \ \wedge \ \beta_n \text{ sse } \not\models P \ \wedge \ G$$

Sendo a resolução integra, basta que

$$\mathcal{P} \cup \{G\} \vdash \mathbf{F}.$$

Mas então

$$\mathcal{P} \models (\beta_1 \ \wedge \ \cdots \ \wedge \ \beta_n)\theta$$

para alguma substituição θ.

Mais especificamente:

Resposta correcta

Uma substituição de variáveis θ em G é uma resposta correcta se $\mathcal{P} \models \forall(\neg G\theta)$ onde $\forall$ é o fecho universal das variáveis livres em $\neg G\theta$.

Exemplo 24.1. *Seja* P

$$\begin{array}{l}\mathsf{Double(0,0)} \leftarrow \\ \mathsf{Double(s(x),s(s(y)))} \leftarrow \mathsf{Double(x,y)}\end{array}$$

Então para $P \cup \{\leftarrow Double(x, s(y)\}$, $[s(0)/x, s(0)/y]$ *é uma resposta correcta.*

Resolução para cláusulas de Horn

Seja G um objectivo $\leftarrow \alpha_1, \ldots, \alpha_m, \ldots, \alpha_k$ e $\alpha \leftarrow \beta_1, \ldots, \beta_q$, uma cláusula C. G' é derivado por resolução de G e C, com unificador mais geral θ se:

1. α_m é o átomo seleccionado em G
2. θ é um *umg* de α_m e α
3. G' é $\leftarrow (\alpha_1, \ldots, \beta_1, \ldots, \beta_q, \ldots, \alpha_k)\theta$, isto é

$$\frac{\leftarrow \alpha_1, \ldots, \alpha_m, \ldots, \alpha_k \quad \alpha \leftarrow \beta_1, \ldots, \beta_q}{\leftarrow (\alpha_1, \ldots, \beta_1, \ldots, \beta_q, \ldots, \alpha_k)\theta}$$

4. G' diz-se a a resolvente de G e C.

Resolução para cláusulas de Horn

Exemplo 24.2. *Considera o seguinte programa* $\mathcal{P}$*:*

$$\begin{array}{lcl} q(x,y) & \leftarrow & p(x,y) \\ q(x,y) & \leftarrow & p(x,z), p(z,y) \\ p(b,a) & & \\ p(c,a) & & \\ p(d,b) & & \\ p(e,b) & & \end{array}$$

e o objectivo: $\leftarrow q(y,b), q(b,z)$.

Resolução para cláusulas de Horn

Exemplo 24.3. *Em cada passo escolher um literal que complemente a cabeça duma cláusula do programa (usar sempre variáveis novas para o programa!).*

1. *Escolher* $q(y,b)$ *e resolver com a primeira cláusula* ($q(x_1,y_1) \leftarrow p(x_1,y_1)$)*:* $\leftarrow p(y,b), q(b,z)$ *e a substituição* $[y/x_1, b/y_1]$
2. *Escolher* $p(y,b)$ *e resolver com a quinta cláusula (* $p(d,b)$*):* $\leftarrow q(b,z)$ *e substituição* $[d/y]$

3. *Resolver o literal que resta com a primeira cláusula* ($q(x_2, y_2) \leftarrow p(x_2, y_2)$)*:* $\leftarrow p(b, z)$

4. *Resolver o literal que resta com a terceira cláusula* ($p(b, a)$)*:* [] *e a substituição* $[a/z]$.

A resposta é $[d/y, a/z]$ *e* $\mathcal{P} \models q(d, b) \wedge (q(b, a)$*. E também,* $\mathcal{P} \models \exists y \exists z q(y, b) \wedge q(b, z)$ *(como se pretendia...).*

Regras de Computação e de Procura

Regra de Computação

Uma regra de computação é uma regra para determinar um literal do objectivo com o qual aplicar a regra de resolução (resolver).

Estratégia de Procura

Uma estratégia de procura é uma regra para determinar como escolher qual a cláusula do programa que se vai usar para resolver com o literal escolhido no objectivo.

Resolução SLD (Selectiva Linear para Claúsulas Definidas)

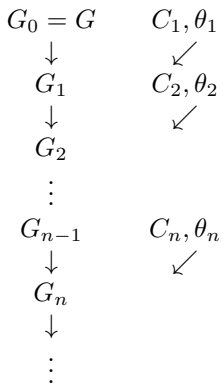
Seja $\mathcal{P}$ um programa definido, R uma regra de computação e G um objectivo.

Uma derivação por resolução-SLD é uma sequência de passos de resolução entre as cláusulas do objectivo e as de programa.

Seja $G_0 = G$. Dada a cláusula G_i, G_{i+1} é obtida selecionando um literal β_i de G_i, de acordo com R e escolhendo $C_i \in P$ (com variáveis novas) tal que a cabeça de C_i unifica com β_i com umg θ_i:

$$\begin{aligned} G_i &= \leftarrow \beta_1, \ldots, \beta_i, \beta_{i+1}, \ldots, \beta_n \\ C_i &= \alpha \leftarrow \alpha_1, \ldots, \alpha_k \\ \alpha\theta_i &= \beta_i\theta_i \\ G_{i+1} &= \leftarrow (\beta_1, \ldots, \alpha_1, \ldots, \alpha_k, \beta_{i+1}, \ldots, \beta_n)\theta_i \end{aligned}$$

Refutação SLD



Uma refutação SLD é uma derivação-SLD de []. Se $G_n = []$ dizemos que a refutação tem comprimento n.

Exemplo 24.4. *Seja P:*

$\mathsf{Max(x,y,x)} \leftarrow \mathsf{Less\_eq(y,x)}$
$\mathsf{Max(x,y,y)} \leftarrow \mathsf{Less\_eq(x,y)}$
$\mathsf{Less\_eq(0,x)} \leftarrow$
$\mathsf{Less\_eq(s(x),s(y))} \leftarrow \mathsf{Less\_eq(x,y)}$

Refutação *SLD*

e $G : \leftarrow Max(s(0), x, s(s(0)))$

Uma refutação para $P \cup \{G\}$ *de comprimento* $n = 3$ *é:*

$\leftarrow \mathsf{Max(s(0),x,s(s(0)))}$	$(\mathsf{Max}(x_1,y_1,y_1) \leftarrow \mathsf{Less\_eq}(x_1,y_1)$
	$\theta_1 = [\mathsf{s(0)}/x_1, \mathsf{s(s(0))}/x, \mathsf{s(s(0))}/y_1])$
$\leftarrow \mathsf{Less\_eq(s(0),s(s(0)))}$	$(\mathsf{Less\_eq(s}(x_2), \mathsf{s}(y_2)) \leftarrow \mathsf{Less\_eq}(x_2,y_2))$
	$\theta_2 = [\mathsf{0}/x_2, \mathsf{s(0)}/y_2])$
$\leftarrow \mathsf{Less\_eq(0,s(0))}$	$(\mathsf{Less\_eq(0,}x_3) \leftarrow, \theta_3 = [\mathsf{s(0)}/x_3])$
ϵ	

Árvore de derivação SLD

Dado um programa definido $\mathcal{P}$, uma regra de computação e um objectivo G, todas as derivações-SLD podem ser representadas por uma árvore tal que:

A raiz é etiquetada por G

Cada nó é etiquetado com uma cláusula objectivo G_i e cria-se um novo ramo para cada G_{i_j} que pode ser obtido resolvendo G_i com uma cláusula C_j cuja cabeça unifique com um literal de G_i.

As folhas se forem ϵ são designadas nós de sucesso (que correspondem a soluções), senão nós de insucesso (ou falha).

Árvore de derivação SLD

Os ramos da árvore podem então ser

de sucesso se terminarem numa folha de sucesso

de falha se terminarem numa folha de falha

infinitos se corresponderem a uma derivação infinita

Exemplo 24.5. *Determinar a árvore-SLD do exemplo 24.2, considerando como regra de computação a escolha do literal mais à esquerda.*

Um exemplo de um programa Prolog.

Exemplo 24.6. *Existem 5 casas de cores diferentes, habitadas por homens de nacionalidades diferentes, cada um dos quais tem um animal de estimação diferente, uma bebida preferida diferente e uma marca de cigarros diferente. E tais que:*

1. *o inglês vive na casa vermelha*
2. *o espanhol tem um cão*
3. *o dono da casa verde bebe café*
4. *o ucraniano bebe chá*
5. *a casa verde fica imediatamente à direita da casa cor de marfim*
6. *o fumador de Winston cria caracóis*
7. *o dono da casa amarela fuma Kools*
8. *o dono da casa do meio bebe leite*
9. *o norueguês mora na primeira casa da esquerda*
10. *o homem que fuma Chesterfields mora na casa ao lado do dono da raposa*
11. *o homem que fuma Kools mora ao lado do dono do cavalo*
12. *o homem que bebe sumo de laranja fuma Lucky Strike*
13. *o japonês fuma Parliaments*
14. *o norueguês vive ao lado da casa azul*

Quem é o dono da zebra? Quem prefere água?

Exercício 24.1. *Para cada uma das seguintes fórmulas encontre uma fórmula em forma clausal. Para tal:*

1. *Converte numa fórmula equivalente em forma normal prenexa.*
2. *Elimina os quantificadores existênciais introduzindo novos símbolos funcionais (*Skolemização*.*
3. *Converte a matriz da fórmula resultante para forma normal conjuntitiva.*
4. *Distribuí os quantificadores universais pelas conjunções, aplicando a regra:*

$$\forall x(\varphi \wedge \psi) \quad \longrightarrow \quad \forall x\varphi \wedge \forall x\psi$$

5. *Escreve a fórmula resultante em notação clausal.*

a) $\forall x P(x) \rightarrow \forall y Q(y)$;

b) $(\forall x P(x) \rightarrow \exists x Q(x)) \wedge \forall z(\neg P(z) \wedge Q(z))$;

c) $\forall y(\exists x(P(x,y) \rightarrow (Q(x) \vee Q(f(x)))))$;

d) $\exists x S(x,x) \vee \exists z(P(z) \wedge \forall x R(x,z))$

$\diamond$

Exercício 24.2. *Aplica o algoritmo de unificação de Robinson a cada um dos conjuntos S de expressões simples seguintes, indicando a resposta do algoritmo e ainda para cada iteração efectuada a substituição correspondente σ_i e os conjuntos D_i e $S\sigma_i$.*

1. $S = \{P(f(a), g(x)), P(y,y)\}$;
2. $S = \{P(a, x, h(g(z))), P(z, h(y), h(y))\}$;
3. $S = \{P(x,x), P(y, f(y))\}$;
4. $S = \{R(f(x, g(u))), R(f(g(u), g(z))\}$;
5. $S = \{P(x, f(z)), P(f(z), y), P(y,z)\}$

$\diamond$

Resolução

Para o último caso

- $\sigma_0 = \iota$
- $\mathsf{D_0} = \{\mathsf{x}, \mathsf{f(z)}, \mathsf{y}\}$, $\sigma_1 = [\mathsf{f(z)/x}]$ e $\mathsf{S}\sigma_1 = \{\mathsf{P(f(z), f(z)), P(f(z), y), P(y, z)}\}$
- $\mathsf{D_1} = \{\mathsf{f(z)}, \mathsf{y}\}$, $\sigma_2 = \sigma_1[\mathsf{f(z)/y}] = [\mathsf{f(z)/x, f(z)/y}]$ e $\mathsf{S}\sigma_2 = \{\mathsf{P(f(z), f(z)), P(f(z), f(z)), P(f(z), z)}\}$
- $\mathsf{D_2} = \{\mathsf{f(z)}, \mathsf{z}\}$, como z ocorre em $\mathsf{f(z)}$ o algoritmo termina indicando que S não é unificável.

Exercício 24.3. *Considera G o objectivo $\leftarrow \mathsf{P(x,x)}$ e o programa $\mathcal{P}$ seguinte:*

$$\begin{array}{l} \mathsf{P(a,b)} \leftarrow \\ \mathsf{P(x,z)} \leftarrow \mathsf{P(x,y)}, \mathsf{P(y,z)} \\ \mathsf{P(x,y)} \leftarrow \mathsf{P(y,x)} \end{array}$$

*Calcula uma refutação-**SLD** para $\mathcal{P} \cup \{G\}$ e a resposta calculada.* $\diamond$

Resolução

$\leftarrow \mathsf{P(x,x)}$	$\mathsf{P(x_1,z_1)} \leftarrow \mathsf{P(x_1,y_1)}, \mathsf{P(y_1,z_1)}$
	$\sigma_1 = [\mathsf{x/x_1, x/z_1}]$
$\leftarrow \mathsf{P(x,y_1)}, \mathsf{P(y_1,x)}$	$\mathsf{P(a,b)} \leftarrow$
	$\sigma_2 = [\mathsf{a/x/, b/y_1}]$
$\leftarrow \mathsf{P(b,a)}$	$\mathsf{P(x_2,y_2)} \leftarrow \mathsf{P(y_2,x_2)}$
	$\sigma_3 = [\mathsf{b/x_2/, a/y_2}]$
$\leftarrow \mathsf{P(a,b)}$	$\mathsf{P(a,b)} \leftarrow$
	$\sigma_4 = \iota$

ϵ

$\sigma_1\sigma_2\sigma_3\sigma_4 = [\mathsf{a/x_1, a/z_1, a/x, b/y_1, b/x_2, a/y_2}]$

Resposta calculada:$\sigma = [\mathsf{a/x}]$

Exercício 24.4. *Para o programa definido P e objectivo G (da forma $\leftarrow \beta$) descritos abaixo:*

- *Calcula uma refutação **SLD** para $P \cup \{G\}$, indicando as substituições em cada passo e a resposta calculada, no fim. Representa a refutação-SLD por uma árvore de derivação.*
- *Verifica a execução do programa com o objectivo em Prolog.*

1. $P:$ $Add(x,0,x) \leftarrow$
   $Add(x,s(y),s(z)) \leftarrow Add(x,y,z)$

   $G:$ $\leftarrow Add(s(0),x,s(s(y))$

2. $P:$ $Lenght(nil,0) \leftarrow$
   $Lenght(cons(x,y),s(z)) \leftarrow Lenght(y,z)$

   $G:$ $\leftarrow Lenght(x,s(s(y)))$

3. $P:$ $Append(cons(x,x_R),x_L,cons(x,x_{RL})) \leftarrow Append(x_R,x_L,x_{RL})$
   $Append(nil,x_L,x_L) \leftarrow$

   $G:$ $\leftarrow Append(cons(x,nil),cons(x,nil),y)$

4. $P:$ $Member(x,cons(x,y)) \leftarrow$
   $Member(x,cons(y,z)) \leftarrow Member(x,z)$

   $G:$ $\leftarrow Member(x,cons(y,cons(x,z)))$

$\diamond$