Lógica Computacional (CC2003) Lógica e Programação (CC216)

Nelma Moreira

Lógica Computacional 25

Conteúdo

1 Introdução à Programação em Lógica

1.1 Fórmulas de Horn

Fórmulas de Horn para Lógica proposicional

Uma *fórmula de Horn* é uma fórmula em forma normal conjuntiva em que em cada disjunção (cláusula) existe no máximo um literal positivo.

$$p \wedge \neg q \wedge (q \vee \neg p)$$
$$(\neg p \vee \neg q \vee \neg s \vee p) \wedge (\neg q \vee \neg r \vee p) \wedge (\neg p \vee \neg s \vee s)$$
$$(\neg p \vee \neg q \vee \neg s) \wedge (\neg q \vee \neg r \vee p) \wedge s$$

Numa fórmula de Horn, as disjunções $\neg p_1 \vee \ldots \vee \neg p_n \vee p$ também se podem escrever como

$$(p_1 \wedge \ldots \wedge p_n) \rightarrow p$$

ou se p não existe (ou é **F**):

$$(p_1 \wedge \ldots \wedge p_n) \rightarrow \mathbf{F}$$

ou se os p_i não existem:

$$\mathbf{V} \rightarrow p$$

Fórmulas de Horn

Nem todas as fórmulas têm uma fórmula de Horn equivalente!

Mas,
Para determinar se uma fórmula de Horn da lógica proposicional é **satisfazível** podemos usar um algoritmo eficiente.

Satisfazibilidade duma fórmula de Horn

Considera a fórmula $p \wedge \neg q \wedge (q \vee \neg p)$:

- começar por colocar numa linha as variáveis proposicionais que ocorrem na fórmula e colocar a fórmula. Ex:

p	q	$p \wedge \neg q \wedge (q \vee \neg p)$

- se alguma das variáveis proposicionais é um dos elementos da conjunção atribuir o valor **V** a essa variável . Ex:

p	q	$p \wedge \neg q \wedge (q \vee \neg p)$
V		

Satisfazibilidade duma fórmula de Horn

- Com essa informação preencher a tabela como se tivesse a construir a tabela de verdade (para essa linha), analisando cada disjunção para determinar se, para ela ser verdadeira, se pode determinar mais valores para as variáveis proposicionais:

p	q	p	$\wedge$	$\neg$	q	$\wedge$	$(q$	$\vee$	$\neg$	$p)$
V									**F**	

- Neste caso, q tem de ser **V** e então isso pode ser acrescentado:

p	q	p	$\wedge$	$\neg$	q	$\wedge$	$(q$	$\vee$	$\neg$	$p)$
V	**V**								**F**	

Satisfazibilidade duma fórmula de Horn

- E voltando a repetir este passo, obtém-se:

p	q	p	$\wedge$	$\neg$	q	$\wedge$	$(q$	$\vee$	$\neg$	$p)$
V	**V**			**F**					**F**	

  Continuar até mais nada poder ser acrescentado.

- se no passo anterior se atribuir **F** a um dos elementos da conjunção, a fórmula também fica com o valor **F** e não é satisfazível. Caso contrário podemos atribuir à fórmula o valor **V** se atribuirmos **F** às restantes variáveis proposicionais.

No exemplo que estamos a considerar, a fórmula tem o valor **F** e portanto não é satisfazível.

Fórmulas de Horn para Lógica de primeira ordem

Uma cláusula é uma fórmula de Horn se tem no máximo um literal positivo.

Uma fórmula (ou cláusula) de Horn é positiva se tem um literal positivo:

$$\forall x_1 \ldots \forall x_s(\alpha_1 \vee \neg\beta_1 \vee \ldots \vee \neg\beta_n)$$
$$\alpha_1 \leftarrow \beta_1, \ldots, \beta_n$$

α_1 é a a cabeça da cláusula e $\beta_1, \ldots, \beta_n$ o corpo da cláusula. Se o corpo é vazio a cláusula diz-se unitária.

Uma cláusula de Horn é negativa (objectivo) se não tem literal positivo:

$$\forall x_1 \ldots \forall x_s(\neg\beta_1 \vee \ldots \vee \neg\beta_n)$$
$$\neg\exists x_1 \ldots \exists x_s(\beta_1 \wedge \ldots \wedge \beta_n)$$
$$\leftarrow \beta_1, \ldots, \beta_n$$

A cláusula vazia representa-se por ϵ corresponde a uma contradição (**F**) e é uma cláusula sem cabeça nem corpo.

Fórmulas de Horn

Exemplo 25.1. *Das seguintes fórmulas, indica quais são cláusulas de Horn, positivas ou negativas e escreve-as na notação clausal:*

- $\forall x \forall y \forall z(P(x,z) \vee Q(x,y) \vee \neg Q(x,z))$ *Não Horn* $P(x,z), Q(x,y) \leftarrow Q(x,z)$
- $\forall x \forall y \forall z(P(x,z) \vee \neg Q(x,y) \vee \neg Q(x,z))$*positiva* $P(x,z) \leftarrow Q(x,y), Q(x,z)$
- $\forall x \forall z P(x,z)$*unitária* $P(x,z) \leftarrow$
- $\forall x \forall y \forall z(\neg P(x,z) \vee \neg Q(x,y))$*negativa* $\leftarrow P(x,z), Q(x,y)$

Proposição 25.1. *Seja $\mathcal{A} = (A, \cdot^{\mathcal{A}})$ uma estrutura de $\mathcal{L}$ e ϕ a cláusula $\alpha_1, \ldots, \alpha_k \leftarrow \beta_1, \ldots, \beta_n$, com variáveis $x_1, \ldots x_s$. $\mathcal{A}$ satisfaz ϕ, $\mathcal{A} \models \phi$, se e só se para todos os $a_1, \ldots, a_s \in A$ existe um literal $\lambda \in \{\alpha_1, \ldots, \alpha_k, \neg\beta_1, \ldots, \neg\beta_n\}$ tal que $A \models_s \lambda$ para $s(x_i) = a_i$, $1 \leq i \leq s$*

Demonstração. Resulta directamente da definição de cláusula e da relação $\models_s$. □

Programa definido

Um programa definido é um conjunto finito de cláusulas de Horn positivas.

Num programa o conjunto de cláusulas com cabeças com mesmo símbolo de predicado P, chama-se a definição de P.

Exemplo 25.2. *Considera a linguagem de* 1^{a} *ordem* $\mathcal{L}_{ss}$ *sem igualdade com* $\mathcal{F}_0 = \{0, \mathsf{nil}\}$, $\mathcal{F}_1 = \{\mathsf{s}\}$, $\mathcal{F}_2 = \{\mathsf{cons}\}$, $\mathcal{R}_1 = \{\mathsf{sorted}\}$, $\mathcal{R}_2 = \{\mathsf{slowsort}, \mathsf{perm}, \mathsf{less\_eq}\}$ *e* $\mathcal{R}_3 = \{\mathsf{delete}\}$.

O programa seguinte dada uma sequência de inteiros permuta os seus elementos até estarem ordenados!:

$\mathsf{slowsort}(\mathsf{x}, \mathsf{y}) \leftarrow \mathsf{sorted}(\mathsf{y}), \mathsf{perm}(\mathsf{x}, \mathsf{y})$
$\mathsf{sorted}(\mathsf{nil}) \leftarrow$
$\mathsf{sorted}(\mathsf{cons}(\mathsf{x}, \mathsf{nil})) \leftarrow$
$\mathsf{sorted}(\mathsf{cons}(\mathsf{x}, \mathsf{cons}(\mathsf{y}, \mathsf{z}))) \leftarrow \mathsf{less\_eq}(\mathsf{x}, \mathsf{y}), \mathsf{sorted}(\mathsf{cons}(\mathsf{y}, \mathsf{z}))$
$\mathsf{perm}(\mathsf{nil}, \mathsf{nil}) \leftarrow$
$\mathsf{perm}(\mathsf{cons}(\mathsf{x}, \mathsf{y}), \mathsf{cons}(\mathsf{u}, \mathsf{v})) \leftarrow \mathsf{delete}(\mathsf{u}, \mathsf{cons}(\mathsf{x}, \mathsf{y}), \mathsf{z}), \mathsf{perm}(\mathsf{z}, \mathsf{v})$

$\mathsf{delete}(\mathsf{x}, \mathsf{cons}(\mathsf{x}, \mathsf{y}), \mathsf{y}) \leftarrow$
$\mathsf{delete}(\mathsf{x}, \mathsf{cons}(\mathsf{y}, \mathsf{z}), \mathsf{cons}(\mathsf{y}, \mathsf{w})) \leftarrow \mathsf{delete}(\mathsf{x}, \mathsf{z}, \mathsf{w})$
$\mathsf{less\_eq}(0, \mathsf{x})$
$\mathsf{less\_eq}(\mathsf{s}(\mathsf{x}), \mathsf{s}(\mathsf{y})) \leftarrow \mathsf{less\_eq}(\mathsf{x}, \mathsf{y})$

- um inteiro n é representado pelo termo $\mathsf{s}(\mathsf{s}(...\mathsf{s}(0)))$ com n símbolos funcionais s. Ex: $\mathsf{s}(\mathsf{s}(\mathsf{s}(0)))$ representa 3
- uma lista de inteiros é representada por termos $\mathsf{cons}(\mathsf{x}, \mathsf{y})$ onde x é um inteiro e y é uma lista. A lista vazia é representada por nil Ex: $\mathsf{cons}(\mathsf{s}(\mathsf{s}(0)), \mathsf{cons}(\mathsf{s}(0), \mathsf{cons}(\mathsf{s}(\mathsf{s}(\mathsf{s}(0))), \mathsf{nil})$ representa a lista $[2, 1, 3]$
- $\mathsf{slowsort}$ dada uma lista x, verifica se está ordenada (sorted), senão permuta (perm)...

Resolução para cláusulas de Horn

Seja $\mathcal{P}$ um programa (conjunto de cláusulas de Horn positivas) e G uma cláusula negativa (objectivo), $\leftarrow \beta_1, \ldots, \beta_n$.

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

Mas então

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

para alguma substituição θ.

1.2 Resolução SLD

Resolução para cláusulas de Horn

Exemplo 25.3. *Considera o seguinte programa $\mathcal{P}$:*

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

Resolução para cláusulas de Horn

Exemplo 25.4. *Considera o objectivo: $\leftarrow q(y,b), q(b,z)$. Em cada passo escolher um literal que complemente a cabeça duma cláusula do programa (usar 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:*
   $\leftarrow q(b,z)$ *e substituição* $[d/y]$
3. *Resolver o literal que resta com a primeira cláusula:*
   $\leftarrow p(b,z)$
4. *Resolver o literal que resta com a terceira cláusula:* [] *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 c. 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:

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

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

Exemplo 25.5. *Seja P:*

Refutação *SLD*

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

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

Exemplo 25.6. *Uma refutação para* $P \cup \{G\}$ *de comprimento* $n = 3$ *é:*

$\leftarrow \mathsf{Max(s(0),x,s(s(0)))}$	$(\mathsf{Max}(x_1, y_1, y_1) \leftarrow \mathsf{Less\_eq}(x_1, y_1)$
	$\theta_1 = [\mathsf{s(0)}/x_1, \mathsf{s(s(0))}/x, \mathsf{s(s(0))}/y_1])$
$\leftarrow \mathsf{Less\_eq(s(0),s(s(0)))}$	$(\mathsf{Less\_eq(s(x_2),s(y_2))} \leftarrow \mathsf{Less\_eq}(x_2, y_2))$
	$\theta_2 = [\mathsf{0}/x_2, \mathsf{s(0)}/y_2])$
$\leftarrow \mathsf{Less\_eq(0,s(0))}$	$(\mathsf{Less\_eq(0,x_3)} \leftarrow, \theta_3 = [\mathsf{s(0)}/x_3])$
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 25.7. *Determinar a árvore-SLD do exemplo 25.3, considerando como regra de computação a escolha do literal mais à esquerda.*