Lógica Computacional

LCC

2006/2007 (Recurso)

Grupo I

1. Enuncie o teorema da *Eliminação do Corte*. Justifique a sua importância no âmbito da demonstração automática.
2. *O* método de Tableaux *pode ser entendido como uma operacionalização do processo de construção de árvores de sequentes*. Justifique esta afirmação e derive as regras do método de Tableaux a partir das correspondentes regras do cálculo de sequentes.
3. Exiba modelos que validem e refutem a seguinte fórmula proposicional. Justifique a sua resposta.

$$(p \Rightarrow q) \Rightarrow q \Rightarrow p$$

Grupo II

Considere a seguinte fórmula proposicional:

$$\varphi \doteq (p \Rightarrow \neg q) \Rightarrow \neg(p \wedge q)$$

1. Construa BDD da fórmula φ:
   (a) Por redução da árvore de decisão binária;
   (b) De forma estrutural (*bottom up*).
2. Mostre como poderia aplicar o algoritmo *Davis-Putnam* para verificar a validade da formula φ.
   (a) Quais os passos envolvidos na verificação.
   (b) Concretize esses passos.

Grupo III

1. Construa a árvore de derivação (no cálculo de sequentes) da seguinte fórmula de primeira ordem:

$$(\forall x.P(x)) \Rightarrow (\exists y.Q(y)) \Rightarrow (\exists w.\forall z.(P(z) \wedge Q(w))$$

2. Calcule os unificadores mais gerais para os seguintes pares de termos (se existirem). Justifique a sua resposta.
   - $\mathsf{f}(x, \mathsf{g}(x), y), \mathsf{f}(z, u, \mathsf{g}(u))$
   - $\mathsf{f}(x, y), \mathsf{f}(y, \mathsf{g}(x))$
3. Prove, usando o processo de Skolemização e a resolução proposicional, a inconsistência da seguinte fórmula (a é uma constante):

$$\exists_x \forall_y ((Q(y, x) \Rightarrow \neg R(y, y)) \wedge Q(\mathsf{a}, x) \wedge R(\mathsf{a}, \mathsf{a})).$$

4. Use a resolução de primeira ordem para mostrar que da assunção $\forall x.\neg gosta(x, ana) \Rightarrow gosta(ana, x)$ se pode inferir que $gosta(ana, ana)$.

Lógica Computacional

Formulário

Regras de construção dos *Tableaux*

- Regras α:

α	α_1	α_2
$+(X \wedge Y)$	$+X$	$+Y$
$-(X \vee Y)$	$-X$	$-Y$
$-(X \Rightarrow Y)$	$+X$	$-Y$

- Regras β:

β	β_1, β_2
$-(X \wedge Y)$	$-X$, $-Y$
$+(X \vee Y)$	$+X$, $+Y$
$+(X \Rightarrow Y)$	$-X$, $+Y$
$+(\neg X)$	$-X$
$-(\neg X)$	$+X$

Cálculo de Sequentes

$$\text{(Corte)}\ \frac{\Gamma, C \vdash \Delta \qquad \Gamma' \vdash C, \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \qquad \text{(Ax)}\ \frac{}{\Gamma, A \vdash A, \Delta}$$

$$\text{(D }\wedge\text{)}\ \frac{\Gamma \vdash P, \Delta \qquad \Gamma \vdash Q, \Delta}{\Gamma \vdash P \wedge Q, \Delta} \qquad \text{(D }\neg\text{)}\ \frac{\Gamma, P \vdash \Delta}{\Gamma \vdash \neg P, \Delta}$$

$$\text{(D }\vee\text{)}\ \frac{\Gamma \vdash P, Q, \Delta}{\Gamma \vdash P \vee Q, \Delta} \qquad \text{(D }\Rightarrow\text{)}\ \frac{\Gamma, P \vdash Q, \Delta}{\Gamma \vdash P \Rightarrow Q, \Delta}$$

$$\text{(E }\wedge\text{)}\ \frac{\Gamma, P, Q \vdash \Delta}{\Gamma, P \wedge Q \vdash \Delta} \qquad \text{(E }\neg\text{)}\ \frac{\Gamma \vdash P, \Delta}{\Gamma, \neg P \vdash \Delta}$$

$$\text{(E }\vee\text{)}\ \frac{\Gamma, P \vdash \Delta \qquad \Gamma, Q \vdash \Delta}{\Gamma, P \vee Q \vdash \Delta} \qquad \text{(E }\Rightarrow\text{)}\ \frac{\Gamma, Q \vdash \Delta \qquad \Gamma \vdash P, \Delta}{\Gamma, P \Rightarrow Q \vdash \Delta}$$

$$\text{(E }\forall\text{)}\ \frac{\Gamma, \forall X.\varphi, \varphi[t/X] \vdash \Delta}{\Gamma, \forall X.\varphi \vdash \Delta} \qquad \text{(E }\exists\text{)}\ \frac{\Gamma, \varphi[A/X] \vdash \Delta}{\Gamma, \exists X.\varphi \vdash \Delta}\ \text{sendo } A \text{ uma var. nova}$$

$$\text{(D }\forall\text{)}\ \frac{\Gamma \vdash \varphi[A/X], \Delta}{\Gamma \vdash \forall X.\varphi, \Delta}\ \text{sendo } A \text{ uma var. nova} \qquad \text{(D }\exists\text{)}\ \frac{\Gamma \vdash \exists X.\varphi, \varphi[t/X], \Delta}{\Gamma \vdash \exists X.\varphi, \Delta}$$

Lógica Computacional

LCC

2006/2007 (Recurso)

PROLOG

1. Considere a seguinte base de conhecimento de um programa *Prolog*:

```
q(0,X,Y) :- p(X), !, p(Y).
q(_,X,b) :- p(X).
q(_,c,Y) :- p(Y), !.
p(a). p(b).
```

   (a) Quais são (todas) as soluções admissíveis para os seguintes objectivos:

      i. `q(1,X,Y).`

      ii. `q(0,X,Y).`

   (b) Apresente as *árvores de derivação* para os objectivos da alínea anterior.

2. Defina o predicado `posFirst(+Elem, +List, ?Pos)` que, dado um elemento `Elem` e uma lista `List`, determine a primeira posição onde ocorre `Elem` em `List` (0 se não ocorrer).

3. Uma forma simples de descrever um algorito de ordenação é: *encontrar uma permutação da lista dada que esteja ordenada*. Essa descrição pode ser directamente codificada em *Prolog* usando a estratégia *generate-and-test*. Codifique o predicado `slowSort/2` que implemente essa estratégia.