Lógica Computacional

LCC

2006/2007 (Época Especial)

Grupo I

1. Mostre a validade da seguinte proposição: *Se φ (na FNC) contém um par de cláusulas $(\alpha \vee p)$ e $(\beta \vee \neg p)$ então φ é semanticamente equivalente a $\varphi \wedge (\alpha \vee \beta)$.*

2. Apresente as vantagens do cálculo de sequentes em relação à dedução natural.

3. *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.

4. Usando o *algoritmo de resolução de Robinson*, verifique se o seguinte conjunto de cláusulas (na FNC) é inconsistente:
$$\{\{a, b, \neg c\}, \{\neg a, c\}, \{\neg b, e\}\}$$

Grupo II

Considere a seguinte fórmula proposicional:

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

1. Construa BDD da fórmula φ de forma estrutural (*bottom-up*).

2. Determine a FNC e a FND da fórmula apresentada. Será que se pode a validade ou inconsistência de φ por simples inspecção "visual" dessas formas normais? Justifique.

3. Aplique o algoritmo *Davis-Putnam* para estabelecer a validade da formula φ.

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. Prove, usando o processo de Skolemização e a resolução proposicional, a inconsistência da seguinte fórmula (a é uma constante):
$$\exists_x \forall_y ((Q(y, x) \Rightarrow \neg R(y, y)) \wedge Q(\mathsf{a}, x) \wedge R(\mathsf{a}, \mathsf{a})).$$

3. Considere a teoria $\Gamma = \{\forall_{x,y}.elem(x, cons(x, y)), \forall_{x,y,z}.elem(x, y) \Rightarrow elem(x, cons(z, y))\}$. Usando a resolução, demonstre que: $\Gamma \models \exists_x elem(x, cons(a, cons(b, c)))$. Indique explicitamente as unificações necessárias.

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

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

$$\text{(Ax)}\ \frac{}{\Gamma, A \vdash A, \Delta}$$

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

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

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

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

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

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

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

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

$$\text{(E }\forall\text{)}\ \frac{\Gamma, \forall X.\varphi, \varphi[t/X] \vdash \Delta}{\Gamma, \forall X.\varphi \vdash \Delta}$$

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

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

$$\text{(D }\exists\text{)}\ \frac{\Gamma \vdash \exists X.\varphi, \varphi[t/X], \Delta}{\Gamma \vdash \exists X.\varphi, \Delta}$$

PROLOG

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

```
not(X) :- X, !, fail.
not(X).
q(X,Y) :- not(p(X)), !.
q(X,Y) :- p(Y).
p(a). p(b).
```

   (a) Considere o seguinte objectivo: `q(X,X)`.

      i. Quais as soluções retornadas pelo interpretador de *Prolog* à questão apresentada.

      ii. Construa a *árvores de derivação* respectiva.

   (b) A *negação por falha* pode não se comportar como a *negação lógica* quando estão envolvidas variáveis nos predicados. Mostre um exemplo onde seja possível observar este fenómeno.

2. O seguinte programa Prolog constroi a árvore de fraccionamento de Davis Putnam pelo próprio processo de prova. Neste programa, as formas clausais são representadas como listas de listas de literais.

```
inconsistente([]) :- fail.
inconsistente(P) :- member([],P).
inconsistente(P) :- fracciona(P,P1,P2),
                    inconsistente(P1), inconsistente(P2).
```

   Defina o predicado `fracciona/3` (juntamente com os predicados auxiliares que entender necessários). Pode utilizar funções da biblioteca standard (e.g. `member`).