Licenciatura Engenharia Informática e de Computadores

Lógica para Programação

Solução do Segundo Teste

11 de Junho de 2012

18:30–20:00

Nome: _____ Número: _____

1. **(2.0)** Para cada uma das seguintes questões, indique se é verdadeira ou falsa. Cada resposta certa vale 0.5 valores e *cada resposta errada desconta 0.2 valores*.

   (a) Uma interpretação é uma função que tem como domínio as entidades da conceptualização e como contradomínio as entidades da linguagem.
   Resposta:
   Falsa

   (b) A resolução SLD assenta numa estratégia de resolução linear.
   Resposta:
   Verdadeira

   (c) Para usar resolução com as cláusulas de Horn, um dos resolventes tem de ser necessariamente um objectivo.
   Resposta:
   Falsa

   (d) Uma função de selecção permite escolher o literal de uma cláusula objectivo como candidato na aplicação do princípio da resolução.
   Resposta:
   Verdadeira

2. **(2.0)** Escolha a *única* resposta *correcta* para as seguintes questões. Cada resposta certa vale 1 valor e *cada resposta errada desconta 0.4 valores*.

   (a) Seja $s_1 = \{f(a)/x, f(y)/y, y/z\}$ e $s_2 = \{b/x, z/y, g(x)/z, b/w\}$. Considerando que x, y, z e w são variáveis, o valor de $s_1 \circ s_2$ é dado por:
   A. $\{f(a)/x, f(z)/y, b/w\}$
   B. $\{f(a)/x, f(b)/y, b/w\}$
   C. $\{f(a)/x, f(z)/y\}$
   D. $\{f(a)/x, f(x)/y, y/z, b/x, z/y, g(x)/z, b/w\}$
   Resposta:
   A.

   (b) Dizem-se *cláusulas determinadas*
   A. as regras e os objectivos;
   B. as regras e os factos;
   C. os objectivos e os factos;
   D. as regras, os objectivos e os factos.

Resposta:

B

3. Considere os seguintes predicados

$$Nasceu(x, y) = x \text{ nasceu no país } y$$

$$Joga\_Seleccão(x, y) = x \text{ joga na selecção do país } y$$

$$Igual(x, y) = x \text{ é igual a } y,$$

as constantes $Portugal$, $Rui\_Patricio$ e $Cristiano\_Ronaldo$, e a função

$$posicão\_de(x) = \text{a posição de } x.$$

Represente em Lógica de Primeira Ordem as seguintes frases:

(a) **(0.5)** O Cristiano Ronaldo e o Rui Patrício não jogam na mesma posição.

Resposta:

$\neg Igual(posicão\_de(Cristiano\_Ronaldo), posicão\_de(Rui\_Patricio))$

(b) **(0.5)** Nem todos os jogadores da selecção portuguesa nasceram em Portugal.

Resposta:

$\exists x[Joga\_Seleccão(x, Portugal) \wedge \neg nasceu(x, Portugal)]$

4. **(1.5)** Usando dedução natural, prove que a seguinte *fbf* é um teorema. Esta *fbf* corresponde a uma das regras de De Morgan para quantificadores, também conhecidas por segundas leis de De Morgan.

$$\neg\exists x[P(x)] \leftrightarrow \forall x[\neg P(x)]$$

Resposta:

1	$\neg\exists x[P(x)]$	Hyp
2	x_0 | $P(x_0)$	Hyp
3	$\exists x[P(x)]$	$\exists$I, 2
4	$\neg\exists x[P(x)]$	Rei, 1
5	$\neg P(x_0)$	$\neg$I, (1, (2, 3))
6	$\forall x[\neg P(x)]$	$\forall$I, (2, 5)
7	$\neg\exists x[P(x)] \rightarrow \forall x[\neg P(x)]$	$\rightarrow$I, (1, 6)
8	$\forall x[\neg P(x)]$	Hyp
9	$\exists x[P(x)]$	Hyp
10	x_0 | $P(x_0)$	Hyp
11	$\forall x[\neg P(x)]$	Hyp
12	$\neg P(x_0)$	$\forall$E, 11
13	$P(x_0)$	Rei, 10
14	$\neg\forall x[\neg P(x)]$	$\neg$I, (11, (12, 13))
15	$\neg\forall x[\neg P(x)]$	$\exists$E, (9, (10, 14))
16	$\forall x[\neg P(x)]$	Rei, 8
17	$\neg\exists x[P(x)]$	$\neg$I, (9, (15, 16))
18	$\forall x[\neg P(x)] \rightarrow \neg\exists x[P(x)]$	$\rightarrow$I, (8, 17)
19	$\neg\exists x[P(x)] \leftrightarrow \forall x[\neg P(x)]$	$\leftrightarrow$I, (7, 18)

5. **(1.0)** Transforme a seguinte *fbf* da lógica de primeira ordem em forma clausal.

$$\exists x[A(x)] \wedge \forall x, y[B(x) \rightarrow (\exists w[C(y, x, w)] \wedge \exists z[D(z, x)])] \wedge \exists x[E(x) \vee F(x)]$$

Resposta:

(a) Eliminação de $\rightarrow$:
$\exists x[A(x)] \wedge \forall x, y[\neg B(x) \vee (\exists w[C(y, x, w)] \wedge \exists z[D(z, x)])] \wedge \exists x[E(x) \vee F(x)]$

(b) Redução do domínio de $\neg$: já está.

(c) Normalização de variáveis:
$\exists x[A(x)] \wedge \forall z, y[\neg B(z) \vee \exists w[C(y, z, w)] \wedge \exists r[D(r, z)])] \wedge (\exists s[E(s) \vee F(s)]$

(d) Eliminação de $\exists$:
$A(sk_1) \wedge \forall z, y[\neg B(z) \vee (C(y, z, skf_1(z, y)) \wedge D(skf_2(z, y), z))] \wedge (E(sk_2) \vee F(sk_2))$

(e) Mover $\forall$ para esquerda:
$\forall z, y[A(sk_1) \wedge (\neg B(z) \vee (C(y, z, skf_1(z, y)) \wedge D(skf_2(z, y), z))) \wedge (E(sk_2) \vee F(sk_2))]$

(f) Eliminação de $\forall$:
$A(sk_1) \wedge (\neg B(z) \vee (C(y, z, skf_1(z, y)) \wedge D(skf_2(z, y), z))) \wedge (E(sk_2) \vee F(sk_2))$

(g) Obtenção da forma conjuntiva normal:
$A(sk_1) \wedge (\neg B(z) \vee C(y, z, skf_1(z, y))) \wedge (\neg B(z) \vee D(skf_2(z, y), z)) \wedge (E(sk_2) \vee F(sk_2))$

(h) Eliminação de $\wedge$:
$\{A(sk_1), \neg B(z) \vee C(y, z, skf_1(z, y)), \neg B(z) \vee D(skf_2(z, y), z), (E(sk_2) \vee F(sk_2))\}$

(i) Eliminação de $\vee$:
$\{\{A(sk_1)\}, \{\neg B(z), C(y, z, skf_1(z, y))\}, \{\neg B(z), D(skf_2(z, y), z)\}, \{E(sk_2), F(sk_2)\}\}$

6. **(1.0)** Sabendo que {{$\neg F(x)$, $G(x)$}, {$\neg G(y)$, $H(x)$}, {$F(a)$}, {$\neg H(a)$}} é a forma clausal da fórmula $\neg\alpha$, recorrendo ao que aprendeu relativamente ao Universo de Herbrand, prove que α é um teorema.

   Resposta:

   Para provar que α é um teorema há que provar que a sua negação não é satisfazível. Ora de acordo com o teorema de Herbrand, um conjunto de cláusulas não é satisfazível se e só se um conjunto finito de instâncias das suas cláusulas não for satisfazível. Assim sendo, uma solução passaria por transformar $\neg\alpha$ num conjunto de cláusulas (o conjunto dado) e, de seguida, provar que existe um conjunto finito das suas instâncias que não é satisfazível. Tendo em conta a forma clausal dada, como {{$\neg F(a)$, $G(a)$}, {$\neg G(a)$, $H(a)$}, {$F(a)$}, {$\neg H(a)$}} não é satisfazível está concluída a demonstração.

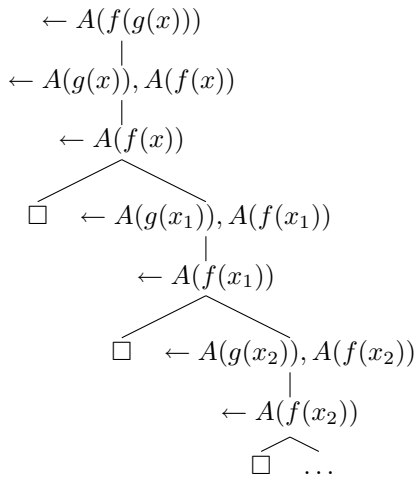
7. **(1.0)** Considerando o programa em lógica

   $A(f(g(x))) \leftarrow A(g(x)), A(f(x))$
   $A(f(0)) \leftarrow$
   $A(g(y)) \leftarrow$

   e o objectivo $\leftarrow A(f(g(x)))$, desenhe a árvore SLD parcial usando a função de selecção que escolha para unificar o primeiro literal do objectivo. A árvore SLD parcial que se pretende deve conter no máximo três nós bem sucedidos. Indique as respostas representadas nesta árvore.

   Resposta:

   A árvore SLD:



   As respostas correspondentes são $x = 0$, $x = g(0)$ e $x = g(g(0))$.

8. Implemente em PROLOG os seguintes predicados:

   (a) **(1.5)** `nth(Pos, Lista, Elem)` o qual afirma que `Elem` é o elemento que ocupa a posição `Pos` da lista `Lista`. Por exemplo:

```
?- nth(6, [5, 3, 2, 4, 6, 8],  Elem).
Elem = 8.
?- nth(6, [a, c, f, d, a, b], a).
false.
?- nth(10, [5, 3, 2, 4, 6, 8], Num).
```

```
false.
```

Resposta:

```
nth(1, [Elem|_], Elem1) :-
                !, Elem = Elem1.
nth(Pos, [_|Tail], Elem) :-
                Pos_1 is Pos - 1,
                nth(Pos_1, Tail, Elem).
```

(b) **(1.0)** `menorPosLista(Pos, Comp, Lista)` o qual afirma que o elemento que ocupa a posição `Pos` em `Lista` é menor que `Comp`. Deve usar o predicado `nth` definido anteriormente (se não o implementou, suponha-o definido). Por exemplo:

```
?- menorPosLista(6, 11, [5, 4, 5, 8, 9, 10]).
true.
?- menorPosLista(6, 5, [5, 4, 5, 8, 9, 10]).
false.
?- menorPosLista(10, 5, [5, 4, 5, 8, 9, 10]).
false.
```

Resposta:

```
menorPosLista(Pos, Comp, Lista) :-
           nth(Pos, Lista, Elem),
           Elem < Comp.
```

9. Considere o seguinte programa em PROLOG:

```
serie(grimm).
serie(galáctica).
serie('csi NY').
canalTV(axn).
canalTV(mov).
policial('csi NY').
passa1(S, C) :- !, serie(S), canalTV(C).
passa2(S, C) :- serie(S), !, canalTV(C).
passa3(S, C) :- serie(S), canalTV(C), !.
gosta1('Alberto', S) :- serie(S), not(policial(S)).
gosta2('Alberto', S) :- not(policial(S)), serie(S).
```

(a) **(1.5)** Considerando que escreveu ";" as vezes necessárias, indique todos os valores devolvidos para cada um dos objectivos `passa1(X, Y)`,`passa2(X, Y)` e `passa3(X, Y)`. Justifique a sua resposta.

Resposta:

```
?- passa1(X, Y).
X = grimm,
Y = axn ;
X = grimm,
Y = mov ;
X = 'galáctica',
Y = axn ;
X = 'galáctica',
Y = mov ;
```

```
X = 'csi NY',
Y = axn ;
X = 'csi NY',
Y = mov.

?- passa2(X, Y).
X = grimm,
Y = axn ;
X = grimm,
Y = mov.

?- passa3(X, Y).
X = grimm,
Y = axn.
```

No primeiro caso o corte impediria apenas unificações com outro `passa1` (caso existisse); no segundo caso, as alternativas à primeira série encontrada já não são exploradas; no último caso são cortadas também as alternativas ao primeiro canalTV.

(b) **(1.0)** Considerando que escreveu "`;`" as vezes necessárias, indique todos os valores devolvidos para cada um dos objectivos `gosta1('Alberto', S)` e `gosta2('Alberto', S)`. Justifique a sua resposta.
Resposta:

```
?- gosta1('Alberto', S).
S = grimm ;
S = 'galáctica' ;
false.
?- gosta2('Alberto', S).
false.
```

No primeiro caso apenas `S = 'csi NY'` não é solução, dado que o Alberto não gosta de policiais e surge na base de conhecimento que esta série é um policial. No segundo caso, dado que surge em primeiro lugar o literal negado, este é resolvido não tendo a variável instanciada. Ora como existe um policial, `not(policial(S))` vai ser avaliado como `false`, logo é este o resultado devolvido.

10. Considere o projecto que implementou este ano em LP.

(a) **(1.0)** Suponha que em vez de um tabuleiro com 9 posições, tem em mãos um tabuleiro com 4 posições – `(top, left)`, `(top, right)`, `(bottom, left)`, `(bottom, right)` – e que qualquer tipo de peça pode ser usado. Defina o predicado `coloca/4`, em que, tal como no projecto, o seu primeiro argumento indica a peça em jogo, o segundo e o terceiro a posição ocupada no tabuleiro (linha e coluna, respectivamente); o último argumento representa o tabuleiro em questão. Assuma também, tal como no projecto, que o tabuleiro é representado por uma lista cujas posições correspondem à da matriz, percorrendo-a da esquerda para a direita e de cima para baixo.

Resposta:

```
coloca(Peca, top, left, [Peca, _, _, _]).
coloca(Peca, top, right, [_, Peca, _, _]).
coloca(Peca, bottom, left, [_, _, Peca, _]).
coloca(Peca, bottom, right, [_, _, _, Peca]).
```

(b) **(1.0)** Considere agora o predicado `linhaDuplaVertical/3`, em que o primeiro argumento representa a peça a colocar, o segundo a linha onde deve ser colocada (`top` ou `bottom`) e, o terceiro, o tabuleiro. Implemente este predicado em PROLOG, tendo em conta a matriz de quatro posições.
Resposta:

```
linhaDuplaVertical(Peca, Linha, Tabuleiro) :-
        coloca(Peca, Linha, _, Tabuleiro).
```

(c) **(1.0)** Considere agora a versão negada do predicado anterior que indica onde não deve ser colocada a peça no tabuleiro (`linhaDuplaVerticalNeg/3`). Ao contrário do predicado definido no projecto, assuma que este recebe na segunda posição a linha onde a peça não deve ser colocada (`top` ou `bottom`) e não uma lista com posições onde a peça não pode ser colocada. Implemente este predicado tendo em conta o tabuleiro de 4 posições. Deve usar o predicado `linhaDuplaVertical/3`.

Resposta:

```
linhaDuplaVerticalNeg(Peca, top, Tabuleiro) :-
        linhaDuplaVertical(Peca, bottom, Tabuleiro).
linhaDuplaVerticalNeg(Peca, bottom, Tabuleiro) :-
        linhaDuplaVertical(Peca, top, Tabuleiro).
```

(d) **(1.0)** Imagine a pista `pares/2` que recebe uma peça e um tabuleiro e indica que esta pode ser colocada numa das posições pares desse tabuleiro (isto é, posição 2, 4, 6, etc. da lista que representa o tabuleiro). Supondo o tabuleiro de 4 posições, implemente este predicado em PROLOG.
Resposta:

```
pares(Peca, Tabuleiro) :-
           coloca(peca, top, right, Tabuleiro);
           coloca(peca, bottom, right, Tabuleiro).
```

11. **(1.5)** Determine o unificador mais geral para o seguinte conjunto de *fbfs*. Apresente todos os passos intermédios.

$$\Delta = \{P(a, f(x), g(z)), P(x, f(a), y)\}$$

Resposta:

Conjunto	Conjunto de desacordo	Substituição
$\{P(a, f(x), g(z)), P(x, f(a), y)\}$	$\{a, x\}$	$\{a/x\}$
$\{P(a, f(a), g(z)), P(a, f(a), y)\}$	$\{g(z), y\}$	$\{g(z)/y\}$
$\{P(a, f(a), g(z))\}$		

O unificador mais geral é $\{a/x, g(z)/y\}$.