Licenciatura Engenharia Informática e de Computadores

Lógica para Programação

Resolução do Segundo Teste

15 de Junho de 2011

09:00–10:30

Nome: ______________________________ Número: __________

1. **(3.0)** Para cada uma das seguintes questões, indique se é verdadeira ou falsa. NOTA: Uma resposta correcta vale 0.5 valores e uma resposta errada desconta 0.3 valores.

   (a) Na conversão para a fórma clausal normal de uma *fbf* em lógica de primeira ordem, a eliminação do quantificador existencial consiste em substituir todas as variáveis quantificadas existencialmente por uma constante de Skolem.
   Resposta:
   F

   (b) Um conjunto de cláusulas Δ é não satisfazível se e só se um conjunto finito de instâncias fechadas de cláusulas de Δ é não satisfazível.
   Resposta:
   V

   (c) Uma cláusula de Horn é uma cláusula que contém, no máximo, um literal negativo.
   Resposta:
   F

   (d) A programação em lógica combina a representação de um subconjunto de fórmulas de primeira ordem com uma estratégia de resolução.
   Resposta:
   V

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

   (f) O PROLOG não permite que o mesmo símbolo de predicado seja utilizado com diferentes números de argumentos.
   Resposta:
   F

2. **(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 $\{a/x, g(z)/y\}$.

3. Considere a seguinte *fbf*:

$$\forall x[P(x, f(x)) \rightarrow \exists y[Q(y) \rightarrow \neg R(g(y), x)]]$$

   (a) **(0.5)** Indique *todos* os termos existentes na *fbf* anterior.
   Resposta:
   x, $f(x)$, y, $g(y)$

   (b) **(0.5)** Indique todas as *fbfs* atómicas existentes na *fbf* anterior.
   Resposta:
   $P(x, f(x))$, $Q(y)$, $R(g(y), x)$

   (c) **(1.0)** Converta a *fbf* anterior para a forma clausal, indicando todos os passos realizados.
   Resposta:
   $\forall x[\neg P(x, f(x)) \vee \exists y[\neg Q(y) \vee \neg R(g(y), x)]]$ (El. do símbolo $\rightarrow$)
   $\forall x[\neg P(x, f(x)) \vee (\neg Q(s(x)) \vee \neg R(g(s(x)), x))]$ (El. dos quantificadores existenciais)
   $\neg P(x, f(x)) \vee (\neg Q(s(x)) \vee \neg R(g(s(x)), x))$ (El. dos quantificadores universais)
   $\{\neg P(x, f(x)) \vee \neg Q(s(x)) \vee \neg R(g(s(x)), x))\}$(El. do símbolo $\wedge$)
   $\{\{\neg P(x, f(x)), \neg Q(s(x)), \neg R(g(s(x)), x))\}\}$(El. do símbolo $\vee$)

4. Considerando definidos os predicados *Super-heroi(x)* (que afirma que x é um super-herói) e *Tem(x, y)* (que afirma que x tem a propriedade y), bem como as constantes *Batman* e *super-poderes*, represente em Lógica de Primeira Ordem as seguintes frases:

   (a) **(0.5)** O Batman é um super-herói.
   Resposta:
   Super-heroi(Batman)

   (b) **(0.5)** Nem todos os super-heróis têm super-poderes.
   Resposta:
   $\exists$*x [Super-herói(x)* $\wedge$ $\neg$*Tem(x, super-poderes)]* ou
   $\neg\forall$ *x [Super-herói(x)* $\rightarrow$ *tem(x, super-poderes)]*

5. **(1.5)** Usando o sistema de dedução natural, demonstre o seguinte teorema

$$\forall x[(\exists y[A(x, y)] \wedge \forall z[A(x, z) \rightarrow B(z)]) \rightarrow \exists y[A(x, y) \wedge B(y)]]$$

Resposta:

1	x_0	$\exists y[A(x_0,y)] \wedge \forall z[A(x_0,z) \rightarrow B(z)]$	Hip
2		$\exists y[A(x_0,y)]$	$E\wedge$, 1
3		$\forall z[A(x_0,z) \rightarrow B(z)]$	$E\wedge$, 2
4		y_0 $A(x_0,y_0)$	Hip
5		$\forall z[A(x_0,z) \rightarrow B(z)]$	Rei, 3
6		$A(x_0,y_0) \rightarrow B(y_0)]$	$E\forall$, 5
7		$A(x_0,y_0)$	Rep, 4
8		$A(x_0,y_0) \rightarrow B(y_0)]$	Rep, 6
9		$B(y_0)]$	$E \rightarrow$, (7,8)
10		$A(x_0,y_0)$	Rep, 4
11		$B(y_0)]$	Rep, 9
12		$A(x_0,y_0) \wedge B(y_0)$	$I\wedge$, (10,11)
13		$\exists y[A(x_0,y) \wedge B(y)]$	$I\exists$, 12
14		$\exists y[A(x_0,y) \wedge B(y)]$	$E\exists$, (2,(4,13))
15		$(\exists y[A(x_0,y)] \wedge \forall z[A(x_0,z) \rightarrow B(z)]) \rightarrow \exists y[A(x_0,y) \wedge B(y)]$	$I \rightarrow$, (1,14)
16		$\forall x[(\exists y[A(x,y)] \wedge \forall z[A(x,z) \rightarrow B(z)]) \rightarrow \exists y[A(x,y) \wedge B(y)]]$	$I\forall$, (1,15)

6. **(1.0)** Recorrendo ao que aprendeu relativamente ao Universo de Herbrand, prove que $\neg\forall x[F(x)] \rightarrow \exists x[\neg F(x)]$ é um teorema.

   Resposta:

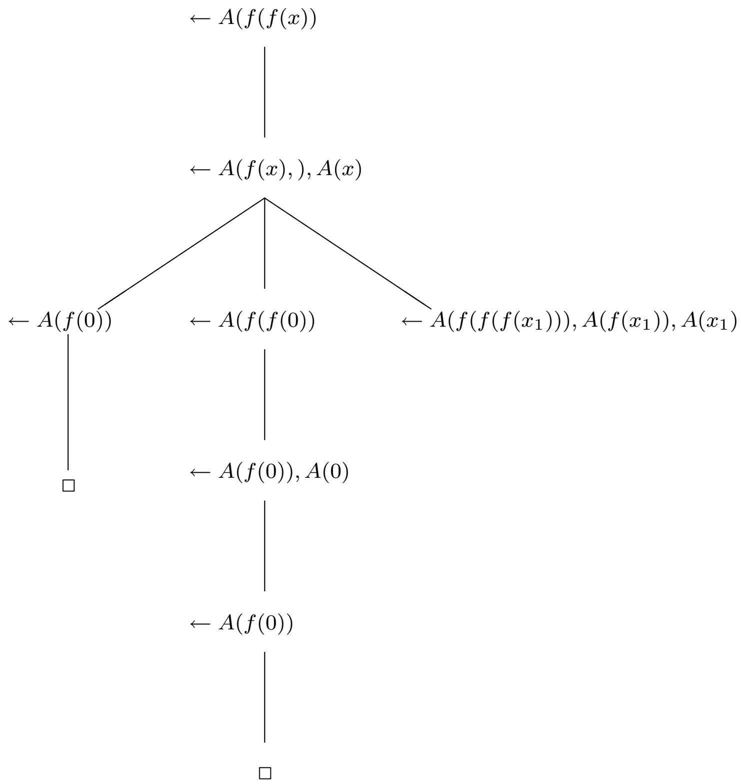
   Para provar que $\neg\forall x[F(x)] \rightarrow \exists x[\neg F(x)]$ é um teorema há que provar que $\neg(\neg\forall x[F(x)] \rightarrow \exists x[\neg F(x)])$ não é satisfazível. Ora de acordo com o teorema de Herbrand, um conjunto de cláusulas não é satisfazível se e só se um conjunto finito de instâncias das suas cláusulas não é satisfazível. Assim sendo, uma solução passa por transformar $\neg(\neg\forall x[F(x)] \rightarrow \exists x[\neg F(x)])$ num conjunto de cláusulas e, de seguida, provar que existe um conjunto finito das suas instâncias que não é satisfazível. A forma clausal resultante é $\{\{\neg F(a)\}, \{F(y)\}\}$ e como $\{\{\neg F(a)\}, \{F(a)\}\}$ não é satisfazível está concluída a demonstração.

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

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

   e o objectivo $\leftarrow A(f(f(x)))$, desenhe a árvore SLD parcial usando a função de selecção que escolha para unificar o último literal do objectivo. A árvore SLD parcial que se pretende deve conter duas soluções e 9 nós (ou menos, se não for possível ter 9 nós).

Resposta:



8. Implemente em PROLOG os seguintes predicados:

   (a) **(1.0)** `num_ocorrencias(Elem, Lista, Num)`, que afirma que `Elem` é um elemento que aparece *exactamente* `Num` vezes na lista `Lista`. Por exemplo,

```
?- num_ocorrencias(a, [a, b, a, a, a, c], Num).
Num = 4 ;
false.
?- num_ocorrencias(a, [a, b, a, a, a, c], 2).
false.
?- num_ocorrencias(a, [a, b, a, a, a, c], 4).
true
```

   Resposta:

```
num_ocorrencias(_, [], 0).
num_ocorrencias(Elem, [Elem | Resto], Num) :- !,
     num_ocorrencias(Elem, Resto, Num1),
     Num is Num1 + 1.
num_ocorrencias(Elem, [Y | Resto], Num) :-
     Y \= Elem,
     num_ocorrencias(Elem, Resto, Num).
```

   (b) **(1.5)** `listaPares(Lista1, Lista2)` que afirma que a `Lista2` contém todos os elementos da `Lista1` que *aparecem exactamente duas vezes*. Sugestões: use o predicado da alínea anterior e assuma que está definido o predicado `remove(Elem, Lista1, Lista2)`, o qual afirma que a `Lista2` é o resultado de eliminar de `Lista1` todos os elementos `Elem`. Por exemplo,

```
listaPares([a, b, a, a, b, c, c, d, c, d, e], L).
L = [b, d] ;
false.
```

Resposta:

```
listaPares([], []).

listaPares([Elem|Lista1], [Elem|Lista2]) :-
     num_ocorrencias(Elem, [Elem|Lista1], 2), !,
     remove(Elem, Lista1, Lista3),
     listaPares(Lista3, Lista2).

listaPares([Elem|Lista1], Lista2) :-
     remove(Elem, Lista1, Lista3),
     listaPares(Lista3, Lista2).
```

9. **(1.0)** Considere as seguintas regras com excepções:

   - Normalmente os adultos trabalham nos dias úteis, a não ser que tenham uma justificação para não o fazer.
   - Se uma pessoa está doente num dia, tem uma justificação para não trabalhar nesse dia, a não ser que esteja apenas constipada.

   Escreva regras em PROLOG que traduzam as regras acima.

   Resposta:

```
trabalha(P,D) :- pessoa(P), diautil(D), \+ justificacao(P,D).

justificacao(P,D) :- doente(P,D), \+ constipado(P,D).
```

10. **(1.5)** Considere o seguinte programa em PROLOG. Indique todas as respostas do programa ao objectivo `p(X,Y,Z)`.

```
p(X,Y,Z) :- q(X,Y,Z).
p(5,5,5).
q(X,Y,Z) :- r(X,Y), !, s(Z).
r(X,Y) :- t(X),s(Y),!.
r(1,2).
s(0).
s(1).
t(2).
t(3).
```

   Resposta:

```
X = 2, Y= 0, Z=0 ;
X = 2, Y= 0, Z=1 ;
X = 5, Y= 5, Z=5
```

11. Considerando os seguinte predicados definidos no projecto deste ano:

   - `pessoa(P_id, Nome_pessoa, Ano_nascimento, Ano_morte)`

- `actividade(A_id, Nome_actividade)`
- `filme(F_id, Nome, Ano_estreia, Lugar_top_250)`
- `participa(P_id, F_id, A_id)`
- `oscar(O_id, A_id, Tipo_oscar)`
- `nomeada(P_id, F_id, A_id, Ganhou?)`

(a) **(1.5)** Escreva em PROLOG o predicado `filmes(P_id, A_id, Lista)` que afirma que a `Lista` contém os nomes dos filmes em que participou a pessoa `P_id` com a actividade `A_id`.

Resposta:

```
filmes(P_id, A_id, Lista) :-
            findall(Nome_filme,
                        (participa(P_id, F_id, A_id),
                         filme(F_id, Nome_filme, _, _)),
                         Lista).
```

(b) **(1.0)** Escreva em PROLOG o predicado `filmes(P_id, Lista)` que afirma que `Lista` contém os nomes dos filmes em que participou a pessoa `P_id`, independentemente da actividade desempenhada. Sugestão: recorra ao predicado anterior.

Resposta:

```
filmes(P_id, Lista) :- filmes(P_id, _, Lista).
```

(c) **(1.5)** Escreva em PROLOG o predicado `nomeacao(P_id, A_id, Lista)` que afirma que `Lista` contém pares cujo primeiro elemento é o nome do filme para o qual a pessoa foi nomeada com a actividade `A_id` e o segundo elemento é o tipo de Oscar em causa.

Resposta:

```
nomeacao(P_id, A_id, Lista) :-
          oscar(_, A_id, Tipo_oscar),
          findall((Nome_filme, Tipo_oscar),
                      (nomeada(P_id, F_id, A_id, _),
                       filme(F_id, Nome_filme, _, _)),
                       Lista).
```