TÉCNICO LISBOA

Lógica para Programação

Solução do Segundo Teste

11 de Junho de 2013

11:30–13:00

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 0.2 valores.

   (a) Na conversão para a forma clausal 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 da lógica 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. **(1.0)** Considere a seguinte fórmula na forma clausal $\{\{A, B\}, \{\neg A, B\}, \{\neg B, C\}\}$. Aplique o algoritmo DP recorrendo a baldes e usando a ordem alfabética. Se a fórmula for satisfazível indique uma testemunha.

   Resposta:

   b_A: $\{A, B\}, \{\neg A, B\}$
   b_B: $\{\neg B, C\}$, $\{B\}$
   b_C: $\{C\}$

   Uma testemunha poderá ser $I(A) = V$, $I(B) = V$ e $I(C) = V$ (o valor de A também poderá ser F).

3. Considere que $M(x)$ significa que x é membro do Clube Aventura, $E(x)$ significa que x é um esquiador, $A(x)$ significa que x é um alpinista e $G(x, y)$ significa que x gosta de y.

   Utilize lógica de primeira ordem para representar as seguintes proposições:

   (a) **(0.5)** Todos os membros do Clube Aventura são esquiadores ou alpinistas (mas não ambos).

   Resposta:

   $\forall_x[M(x) \rightarrow ((E(x) \wedge \neg A(x)) \vee (\neg E(x) \wedge A(x)))]$

   (b) **(0.5)** A Ana gosta de tudo o que o Pedro não gosta e o Pedro gosta de tudo o que a Ana não gosta.

   Resposta:

   $\forall_y[G(Ana, y) \leftrightarrow \neg G(Pedro, y)]$ ou
   $\forall_y[(G(Ana, y) \rightarrow \neg G(Pedro, y)) \wedge (\neg G(Ana, y) \rightarrow G(Pedro, x))]$

4. 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 $\vee$)

5. **(1.5)** Usando dedução natural, prove que a seguinte *fbf* é um teorema. Use apenas as regras básicas e a definição de equivalência. 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	$\neg\exists x[P(x)]$	Hip
2	x_0 | $P(x_0)$	Hip
3	$\exists x[P(x)]$	I$\exists$, 2
4	$\neg\exists x[P(x)]$	Rei, 1
5	$\neg P(x_0)$	I$\neg$, (1, (2, 3))
6	$\forall x[\neg P(x)]$	I$\forall$, (2, 5)
7	$\neg\exists x[P(x)] \rightarrow \forall x[\neg P(x)]$	I$\rightarrow$, (1, 6)
8	$\forall x[\neg P(x)]$	Hip
9	$\exists x[P(x)]$	Hip
10	x_0 | $P(x_0)$	Hip
11	$\forall x[\neg P(x)]$	Hip
12	$\neg P(x_0)$	E$\forall$, 11
13	$P(x_0)$	Rei, 10
14	$\neg\forall x[\neg P(x)]$	I$\neg$, (11, (12, 13))
15	$\neg\forall x[\neg P(x)]$	E$\exists$, (9, (10, 14))
16	$\forall x[\neg P(x)]$	Rei, 8
17	$\neg\exists x[P(x)]$	I$\neg$, (9, (15, 16))
18	$\forall x[\neg P(x)] \rightarrow \neg\exists x[P(x)]$	I$\rightarrow$, (8, 17)
19	$\neg\exists x[P(x)] \leftrightarrow \forall x[\neg P(x)]$	I$\leftrightarrow$, (7, 18)

6. **(1.5)** Determine o unificador mais geral para o seguinte conjunto de *fbfs*. Considere que a é uma constante e que x, y, z e w são variáveis. Apresente *todos* os passos intermédios.

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

Resposta:

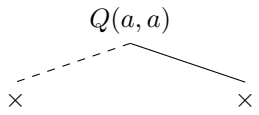
Conjunto	Conjunto de desacordo	Substituição
$\{P(f(x), a), P(y, w), P(f(z), z)\}$	$\{f(x), y, f(z)\}$	$\{f(x)/y\}$
$\{P(f(x), a), P(f(x), w), P(f(z), z)\}$	$\{x, z\}$	$\{z/x\}$
$\{P(f(z), a), P(f(z), w), P(f(z), z)\}$	$\{a, w, z\}$	$\{a/w\}$
$\{P(f(z), a), P(f(z), z)\}$	$\{a, z\}$	$\{a/z\}$
$\{P(f(a),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

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

7. **(1.0)** Considere o conjunto de cláusulas $\{\{Q(a, x)\}, \{\neg Q(y, z)\}\}$. Indique qual a base de Herbrand para este conjunto. Mostre que o conjunto não é satisfazível desenhando a respectiva árvore de decisão fechada.

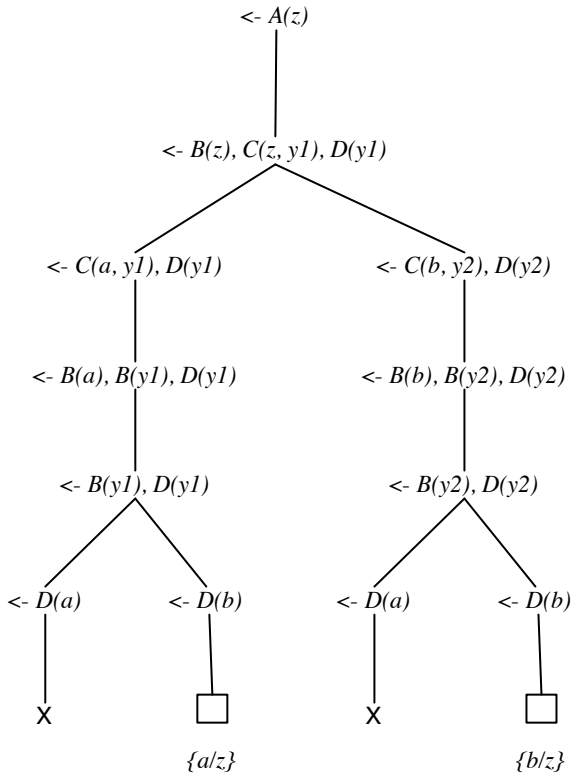
Resposta:

A base de Herbrand deste conjunto de cláusulas é $\{Q(a, a)\}$. A respectiva árvore de decisão fechada é a seguinte:



8. **(1.0)** Usando a resolução SLD e uma função de selecção que escolhe o primeiro literal do objectivo para unificar (podendo usar a regra de procura que entender), indique explicitamente numa árvore SLD todas as soluções para o objectivo $\leftarrow A(z)$.
$A(x) \leftarrow B(x), C(x, y), D(y)$
$C(x, y) \leftarrow B(x), B(y)$
$B(a) \leftarrow$
$B(b) \leftarrow$
$D(b) \leftarrow$
$D(c) \leftarrow$

Resposta:



9. Considere o seguinte programa em PROLOG:

```
serie(grimm).
serie(galactica).
serie(csi).
canalTV(axn).
canalTV(mov).
policial(csi).
passa(S, C) :- serie(S), canalTV(C).
gosta(alberto, S) :- not(policial(S)), serie(S).
```

(a) **(1.0)** Usando o corte implemente `passa1/2` e `passa2/2`, definidos como `passa(S, C)` (isto é, em função de `serie(S)` e `canalTV(C)`), mas de modo a que os valores devolvidos para os objectivos `passa1(X, Y)` e `passa2(X, Y)` sejam, respectivamente:

```
?- passa1(X, Y).
X = grimm,
Y = axn.

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

Resposta:

```
passa1(S, C) :- serie(S), canalTV(C), !.
passa2(S, C) :- serie(S), !, canalTV(C).
```

(b) **(1.0)** Justificando a sua resposta, indique a resposta do PROLOG ao objectivo

```
?- gosta(alberto, S).
```

Resposta:

A resposta é `false`.

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 predicado `xpto(X, L, Y)`, implementado como se segue:

```
xpto(1, [X|_], X) :- !.
xpto(N, [_|Tail], X) :-
                N_1 is N - 1,
                xpto(N_1, Tail, X).
```

(a) **(1.0)** Que funcionalidade implementa o predicado `xpto(X, L, Y)` e o que devolvem os seguintes objectivos?

```
?- xpto(3, [5, 7, 12, 4, 6, 8],  X).

?- xpto(2, [a, c, f, d, a, b], c).

?- xpto(3, [5, 3, 2, 4, 6, 8], 5).
```

Resposta:

O predicado `xpto(X, L, Y)` é verdadeiro se o elemento `Y` ocupar a posição `X` da lista `L`, e os objectivos devolvem, respectivamente, `X = 12`, `true` e `false`.

(b) **(1.0)** Implemente o predicado `maiorPosLista(Pos, Num, Lista)` que significa que o elemento que ocupa a posição `Pos` em `Lista` é maior que `Num`. Se quiser, pode usar o predicado definido anteriormente. Exemplo:

```
?- maiorPosLista(5, 6, [5, 4, 5, 8, 9, 10]).
true.
?- maiorPosLista(6, 5, [5, 4, 5, 8, 9, 2]).
false.
```

Resposta:

```
maiorPosLista(Pos, Num, Lista) :-
            xpto(Pos, Lista, Elem),
            Elem > Num.
```

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

    (a) **(1.0)** Suponha que em vez de 6 suspeitos, podia ter um número variável de suspeitos. Implemente o predicado `existe(S1, Suspeitos)` que indica que o suspeito `S1` existe na lista de suspeitos `Suspeitos` (provavelmente o que terá feito no seu projecto).
    Resposta:

    ```
    existe(S1, [S1 | _]).
    existe(S1, [_| Resto]) :- existe(S1, Resto).
    ```

    (b) **(1.5)** Considere agora o predicado `naoLado(S1, S2, Suspeitos)`, que indica que o suspeito `S1` não está na posição imediatamente anterior ou posterior do suspeito `S2` na lista `Suspeitos` (note que os suspeitos devem existir na lista e ser únicos). Implemente este predicado em PROLOG, tendo em conta a situação anterior em que o número de suspeitos é variável.
    Resposta:

    ```
    naoLado(S1, S2, [S1, Y|  Resto]) :-
         Y \= S2, existe(S2, Resto).
    naoLado(S1, S2, [S2, Y | Resto]) :-
         Y \= S1, existe(S1, Resto).
    naoLado(S1, S2, [X | Resto]) :-
         X \= S1, X \= S2, naoLado(S1, S2, Resto).
    ```

    (c) **(1.5)** Implemente o predicado `entreDif(S1, S2, S3, Suspeitos)` que indica que o suspeito `S2` está entre os suspeitos `S1` e `S3`, ou então está ao lado de um suspeito que junta as características de `S1` e de `S3` (dito de outro modo, `S1` e `S3` seriam o mesmo suspeito). Pode usar predicados definidos no projecto.
    Resposta:

    ```
    entreDif(S1, S2, S3, Suspeitos) :-
         lado(S1, S2), lado(S2, S3).
    ```