TÉCNICO LISBOA

Lógica para Programação

Solução da Repescagem do Segundo Teste

27 de Junho de 2013

11:30–13:00

1. **(1.0)** 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) A resolução SLD assenta numa estratégia de resolução linear.

   Resposta:

   Verdadeira

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

   Resposta:

   Verdadeira

2. **(1.0)** Escolha a *única* resposta *correcta* para as seguintes questões. Cada resposta certa vale 0.5 valores e *cada resposta errada desconta 0.2 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. **(1.0)** No contexto da programação em lógica diga o que é uma resposta correcta de um programa a um objectivo.

   Resposta:

   Sendo Δ um programa e α um objectivo, uma substituição s para as variáveis de α diz-se uma resposta correcta de Δ ao objectivo α se $\Delta \models (\alpha \cdot s)$.

4. **(1.0)** Considere a seguinte *fbf* $(B \vee (A \wedge \neg A)) \wedge (B \rightarrow C) \wedge \neg C$. Aplique o algoritmo DP recorrendo a baldes e usando a ordem alfabética. Se a fórmula for satisfazível indique uma testemunha.

   Resposta:

   A *fbf* apresentada corresponde à seguinte *fbf* na forma clausal: $\{\{A, B\}, \{\neg A, B\}, \{\neg B, C\}, \{\neg C\}\}$.

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

   Como foi derivada a cláusula vazia, conclui-se que a fórmula é não satisfazível.

5. Considere que $Carro(x)$ significa que x é um carro, $Diferente(x, y)$ siginifica que x é diferente de y e $Maior(x, y)$ significa que x é maior que y. Represente em Lógica de Primeira Ordem as seguintes proposições:

   (a) **(0.5)** Existem pelo menos dois carros.
   Resposta:
   $\exists x[Carro(x) \wedge \exists y[Carro(y) \wedge Diferente(x, y)]]$

   (b) **(0.5)** Existe um carro que é maior que todos os outros carros.
   Resposta:
   $\exists x[Carro(x) \wedge \forall y[Carro(y) \wedge Diferente(x, y) \rightarrow Maior(x, y)]]$

6. Considere a seguinte *fbf*:

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

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

   (b) **(0.5)** Qual o domínio do primeiro quantificador existencial ($\exists z$) existente na *fbf* anterior.
   Resposta:
   $P(x, f(z)) \rightarrow \forall y[Q(y) \rightarrow \neg\exists w[R(g(y), w)]]$

   (c) **(1.0)** Converta a *fbf* anterior para a forma clausal, indicando *todos* os passos realizados.
   Resposta:
   $\forall x[\exists z[\neg P(x, f(z)) \vee \forall y[\neg Q(y) \vee \neg\exists w[R(g(y), w)]]]]$ (El. do símbolo $\rightarrow$)
   $\forall x[\neg P(x, f(s_1(x))) \vee \forall y[\neg Q(y) \vee \neg(R(g(y), s_2(x, y)))]]$ (El. dos quantificadores existenciais)
   $\neg P(x, f(s_1(x))) \vee (\neg Q(y) \vee \neg(R(g(y), s_2(x, y))))$ (El. dos quantificadores universais)
   $\{\{\neg P(x, f(s_1(x))), \neg Q(y), \neg R(g(y), s_2(x, y))\}\}$(El. do símbolo $\vee$)

7. Usando dedução natural, prove que as seguintes *fbfs* são teoremas. Use apenas as regras básicas.

   (a) **(1.0)**

$$P(a) \rightarrow \neg\forall x[\neg P(x)]$$

   Resposta:

1		$P(a)$	Hip
2		$\quad \forall x[\neg P(x)]$	Hip
3		$\quad P(a)$	Rei, 1
4		$\quad \neg P(a)$	E$\forall$, 2
5		$\neg\forall x[\neg P(x)]$	I$\neg$, (2, (3, 4))
6	$P(a) \rightarrow \neg\forall x[\neg P(x)]$		I$\rightarrow$, (1, 5)

(b) **(1.0)**

$$\forall x[P(x) \vee \neg P(x)]$$

Resposta:

1	x_0		$\neg(P(x_0) \vee \neg P(x_0))$	Hip
2			$\quad P(x_0)$	Hip
3			$\quad P(x_0) \vee \neg P(x_0)$	I$\vee$, 2
4			$\quad \neg(P(x_0) \vee \neg P(x_0))$	Rei, 1
5			$\neg P(x_0)$	I$\neg$, (2, (3, 4))
6			$P(x_0) \vee \neg P(x_0)$	I$\vee$, 5
7			$\neg(P(x_0) \vee \neg P(x_0))$	Rep, 1
8		$\neg\neg(P(x_0) \vee \neg P(x_0))$		I$\neg$, (1, (6, 7))
9		$P(x_0) \vee \neg P(x_0)$		E$\neg$, 8
10	$\forall x[P(x) \vee \neg P(x)]$			I$\forall$, (1, 9)

8. **(1.0)** Recorrendo ao Universo de Herbrand, mostre que o conjunto de cláusulas $\{\{P(x)\}, \{\neg P(x), Q(y, y)\}, \{\neg Q(f(x), f(x))\}, \{R(f(x))\}\}$ é não satisfazível.

   Resposta:

   De acordo com o teorema de Herbrand, um conjunto de cláusulas é não satisfazível se e só se um conjunto finito de instâncias das suas cláusulas não for satisfazível. Como $\{\{P(a)\}, \{\neg P(a), Q(f(a), f(a))\}, \{\neg Q(f(a), f(a))\}\}$ é não satisfazível, então o conjunto dado também é não satisfazível.

9. Considere o programa, definido através do seguinte conjunto de cláusulas de Horn:

   $A(x, y) \leftarrow B(x, y), C(x, y), D(y)$

   $B(a, b)$

   $B(b, a)$

   $C(x, y) \leftarrow B(y, x)$

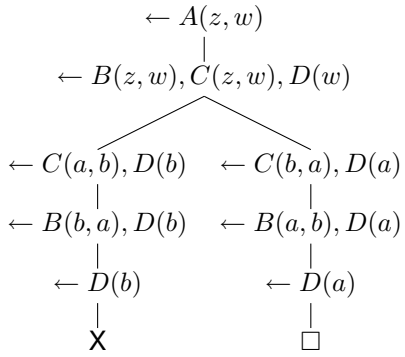
   $C(x, x) \leftarrow B(x, x), D(b)$

   $D(a)$

   $D(c)$

(a) **(1.0)** Usando uma árvore de resolução SLD e uma função de selecção que escolhe o primeiro literal do objectivo para unificar, indique explicitamente todas as soluções para o objectivo $\leftarrow A(z, w)$.

Resposta:



As respostas seriam: $\{b/z, a/w\}$.

(b) **(1.0)** Sem alterar o programa anterior, crie um objectivo de modo a que resposta do programa a esse objectivo seja $\{b/x_1\}$. Justifique.

Resposta:

A resposta do programa ao objectivo $\leftarrow B(x_1, a)$ seria $\{b/x_1\}$ e obtém-se por unificação com a terceira cláusula do programa, $B(b, a)$.

10. Considere o seguinte programa em PROLOG:

```
f(X, Y) :- xpto(X), blabla(Y).
h(X, Y) :- xpto(X), !, blabla(Y).
j(X, Y) :- xpto(X), blabla(Y), !.
xpto(a).
xpto(b).
xpto(c).
blabla(a1).
blabla(a2).
```

(a) **(1.0)** Diga qual a resposta do PROLOG aos seguintes objectivos:

```
?- f(X, Y).
?- h(X, Y).
?- j(X, Y).
```

Resposta:

```
?- f(X, Y).
X = a,
Y = a1 ;
X = a,
Y = a2 ;
X = b,
Y = a1 ;
X = b,
Y = a2 ;
X = c,
Y = a1 ;
X = c,
Y = a2.
```

```
?- h(X, Y)
X = a,
Y = a1 ;
X = a,
Y = a2.

?- j(X, Y).
X = a,
Y = a1.
```

(b) **(1.0)** Adicione, no fim do programa anterior, a cláusula
`h(X, _) :- not(xpto(X)).`
Qual seria agora a resposta do PROLOG ao objectivo `h(X, Y)`? (Justifique a sua resposta)
Resposta:

A resposta é a mesma que a obtida pelo objectivo `h(X, Y)` antes da introdução da nova cláusula, pois o a nova cláusula não é chamada devido ao corte existente na cláusula original.

11. Considere o predicado `substitui(Lista1, X, Y, Lista2)` que indica que `Lista2` é a lista obtida substituindo *todas* as ocorrências do inteiro `X` por `Y` (`Y` não tem de ser um inteiro) na lista `Lista1` (por exemplo, verifica-se `substitui([1, 2, 1], 1, batata, [batata, 2, batata])`).

(a) **(1.0)** Implemente em PROLOG o predicado `substitui`.
Resposta:

```
substitui([], _, _, []).
substitui([H|T], E1, E2, [E2|L2]) :-
            H == E1, !, substitui(T, E1, E2, L2).
substitui([H|T], E1, E2, [H|L2]) :-
            H =\= E1, substitui(T, E1, E2, L2).
```

(b) **(1.0)** Altere o programa anterior de modo a que apenas a primeira ocorrência de `X` na lista `Lista1` seja substituída por `Y`.
Resposta:

```
substitui([], _, _, []).
substitui([H|T], E1, E2, [E2|T]) :-
          H == E1, !.
substitui([H|T], E1, E2, [H|L2]) :-
          H =\= E1, substitui(T, E1, E2, L2).
```

12. Considere o projeto que implementou este ano em LP. Na resolução das alíneas que se seguem pode usar predicados que implementou no projeto, mas se usar algum predicado pré-definido do PROLOG, tem que explicar a sua funcionalidade.

(a) **(1.0)** Suponha que em vez de 6 suspeitos, podia ter um número variável de suspeitos. Implemente o predicado `esquerda(S1, S2, Suspeitos)` que indica que o suspeito `S1` está à esquerda do suspeito `S2` na lista de suspeitos `Suspeitos` (provavelmente o que terá feito no seu projeto).
Resposta:

```
esquerda(S1, S2, [S1 | R]) :- existe(S2, R).
esquerda(S1, S2,  [_| Resto]) :- esquerda(S1, S2, Resto).
```

(b) **(1.5)** Usando os predicados `esquerda` e `direita`, ambos de três argumentos, tal como definidos no projecto, implemente o predicado `naoEntreNovo(S1, S2, S3, Suspeitos)`, que indica que o suspeito `S2` não está entre os suspeitos `S1` e `S3` na lista `Suspeitos`, podendo existir mais do que uma posição entre `S1` e `S3`.

Resposta:

```
naoEntreNovo(S1, S2, S3, Suspeitos) :-
     esquerda(S1, S3, Suspeitos),
     (esquerda(S2, S1, Suspeitos);
      direita(S2, S3, Suspeitos)).

naoEntreNovo(S1, S2, S3, Suspeitos) :-
     esquerda(S3, S1, Suspeitos),
     (esquerda(S2, S3, Suspeitos);
      direita(S2, S1, Suspeitos)).
```

(c) **(1.5)** Suponha que está definido o predicado `idade(S, I)` que é verdadeiro se a idade do suspeito `S` for `I`. Usando o predicado `idade`, implemente o predicado `crescente(Suspeitos)` que é verdadeiro se na solução dada pela lista `Suspeitos`, as idades dos suspeitos estiverem ordenadas por ordem estritamente crescente.

Resposta:

```
crescente([X]).
crescente([S1, S2 | Resto]) :- idade(S1, I1),
       idade(S2, I2), I1 < I2, crescente([S2 | Resto]).
```