TÉCNICO LISBOA

Lógica para Programação

Solução do Exame de 2ª Época

30 de Junho de 2016

1. (1.0) Para cada uma das seguintes afirmações, diga se é verdadeira (V) ou falsa (F). Cada resposta correcta vale 0.5 valores e *cada resposta errada desconta 0.2 valores*.

   (a) Se o número 2 é ímpar, então todos os elefantes que sabem programar são azuis.
   Resposta: ___
   Resposta:
   V

   (b) O seguinte argumento é válido:

   Numa lógica completa todos os argumentos demonstráveis são válidos
   ∴ Numa lógica completa nenhum argumento não válido é demonstrável

   Resposta: ___
   Resposta:
   V

2. Considere os seguintes predicados:

$$Inteiro(x) = x \text{ é um número inteiro}$$
$$Maior(x, y) = x \text{ é maior que } y$$
$$Suc(x, y) = y \text{ é o sucessor de } x$$
$$Entre(x, y, z) = y \text{ está entre } x \text{ e } z \ (x < y < z)$$

   Represente em Lógica de Primeira Ordem as seguintes proposições:

   (a) (0.5) Se um inteiro n está entre 2 inteiros n_1 e n_2, então n_2 é maior do que n_1.

   (b) (0.5) O sucessor de qualquer inteiro par é um inteiro ímpar.

   Resposta:

   (a) $\forall x, y, z[(Inteiro(x) \wedge Inteiro(y) \wedge Inteiro(z) \wedge Entre(x, y, z)) \rightarrow Maior(z, x)]$

   (b) $\forall x, y[(Inteiro(x) \wedge Par(x) \wedge Suc(x, y)) \rightarrow (Inteiro(y) \wedge Ímpar(y))]$

3. (2.0) Demonstre o seguinte teorema

$$\exists x[P(x) \wedge Q(x)] \rightarrow (\exists x[P(x)] \wedge \exists x[Q(x)])$$

usando o sistema dedutivo da Lógica de Primeira Ordem (apenas pode usar as regras de premissa, hipótese, repetição, reiteração, e as regras de introdução e eliminação de cada um dos símbolos lógicos).

Resposta:

1	$\exists x[P(x) \wedge Q(x)]$	Hip
2	$x_0 \mid P(x_0) \wedge Q(x_0)$	Hip
3	$P(x_0)$	E$\wedge$, 2
4	$Q(x_0)$	E$\wedge$, 2
5	$\exists x[P(x)]$	I$\exists$, 3
6	$\exists x[Q(x)]$	I$\exists$, 4
7	$\exists x[P(x)] \wedge \exists x[Q(x)]$	I$\wedge$, (5, 6)
8	$\exists x[P(x)] \wedge \exists x[Q(x)]$	E$\exists$, (1, (2, 7))
9	$\exists x[P(x) \wedge Q(x)] \rightarrow (\exists x[P(x)] \wedge \exists x[Q(x)])$	I$\rightarrow$, (1, 8)

4. (1.5) Considere o seguinte conjunto de *fbfs* (em que x, y e z são variáveis e a é uma constante)

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

Preencha as linhas necessárias da seguinte tabela, de forma a seguir o algoritmo de unificação para determinar se as *fbfs* são unificáveis. Em caso afirmativo, indique o unificador mais geral; caso contrário, indique que as *fbfs* não são unificáveis.

Conjunto de fbfs	**Conjunto de desacordo**	**Substituição**

Unificador mais geral (se existir):

Resposta:

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

Unificador mais geral (se existir):

$$\{x/y\} \circ \{f(z)/x\} \circ \{a/z\} = \{f(z)/y, f(z)/x\} \circ \{a/z\} = \{f(a)/y, f(a)/x, a/z\}$$

5. (2.0) Demonstre que

$$\{\forall x[P(x) \to R(x)] \vee \forall x[Q(x) \to R(x)]\} \vdash \forall x[(P(x) \wedge Q(x)) \to R(x)]$$

usando resolução, e fazendo uma prova por refutação.

Resposta:

Vamos fazer uma prova por refutação:

- *Passagem à forma clausal:*
  $(\forall x[P(x) \to R(x)] \vee \forall x[Q(x) \to R(x)]) \wedge \neg(\forall x[(P(x) \wedge Q(x)) \to R(x)])$
  $(\forall x[\neg P(x) \vee R(x)] \vee \forall x[\neg Q(x) \vee R(x)]) \wedge \neg(\forall x[\neg(P(x) \wedge Q(x)) \vee R(x)])$
  $(\forall x[\neg P(x) \vee R(x)] \vee \forall x[\neg Q(x) \vee R(x)]) \wedge \exists x[\neg(\neg(P(x) \wedge Q(x)) \vee R(x)]$
  $(\forall x[\neg P(x) \vee R(x)] \vee \forall x[\neg Q(x) \vee R(x)]) \wedge \exists x[\neg\neg(P(x) \wedge Q(x)) \wedge \neg R(x)]$
  $(\forall x[\neg P(x) \vee R(x)] \vee \forall x[\neg Q(x) \vee R(x)]) \wedge \exists x[P(x) \wedge Q(x) \wedge \neg R(x)]$
  $(\forall x[\neg P(x) \vee R(x)] \vee \forall y[\neg Q(y) \vee R(y)]) \wedge \exists z[P(z) \wedge Q(z) \wedge \neg R(z)]$
  $(\forall x[\neg P(x) \vee R(x)] \vee \forall y[\neg Q(y) \vee R(y)]) \wedge P(a) \wedge Q(a) \wedge \neg R(a)$
  (em que a é uma constante de Skolem)
  $(\neg P(x) \vee R(x) \vee \neg Q(y) \vee R(y)) \wedge P(a) \wedge Q(a) \wedge \neg R(a)$

  $\{\{\neg P(x), R(x), \neg Q(y), R(y)\}, \{P(a)\}, \{Q(a)\}, \{\neg R(a))\}\}$

- *Prova:*

1	$\{\neg P(x), R(x), \neg Q(y), R(y)\}$	Prem
2	$\{P(a)\}$	Prem
3	$\{Q(a)\}$	Prem
4	$\{\neg R(a)\}$	Prem
5	$\{R(a), \neg Q(y), R(y)\}$	Res, (1,2), $\{a/x\}$
6	$\{R(a)\}$	Res, (3,5), $\{a/y\}$
7	$\{\}$	Res, (4, 6), $\{\}$

6. (1.0) Demonstre que

$$\{\forall x[P(x) \to R(x)] \vee \forall x[Q(x) \to R(x)]\} \models \forall x[(P(x) \wedge Q(x)) \to R(x)]$$

usando o método de Herbrand. Sugestão: use a passagem à forma clausal da questão 5.

Resposta:

Provaremos que o conjunto

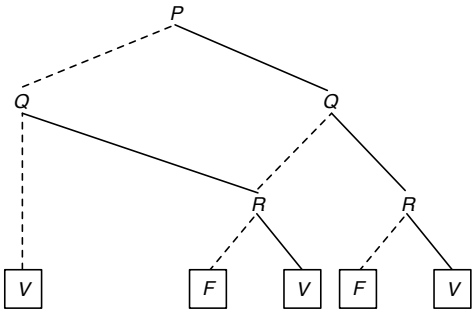
$$\{\{\neg P(x), R(x), \neg Q(y), R(y)\}, \{P(a)\}, \{Q(a)\}, \{\neg R(a))\}\}$$

não é satisfazível, mostrando que um conjunto finito de instâncias das cláusulas do conjunto não é satisfazível. O conjunto

$$\{\{\neg P(a), R(a), \neg Q(a)\}, \{P(a)\}, \{Q(a)\}, \{\neg R(a))\}\}$$

não é satisfazível, pois qualquer interpretação que satisfaça as 3 últimas cláusulas não satisfaz a primeira.

7. Considere que o OBDD abaixo, designado por $OBDD_\alpha$, representa a *fbf* α:



(a) (1.0) Aplique o algoritmo $reduz$ ao $OBDD_{\alpha}$.

Resposta:

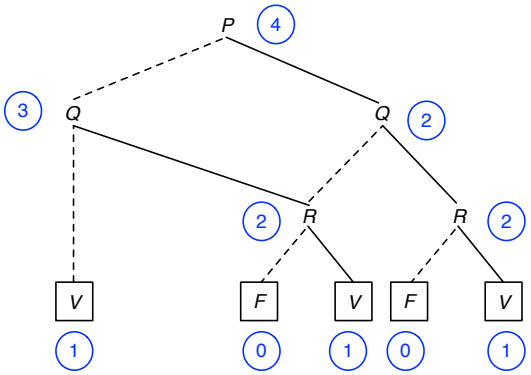
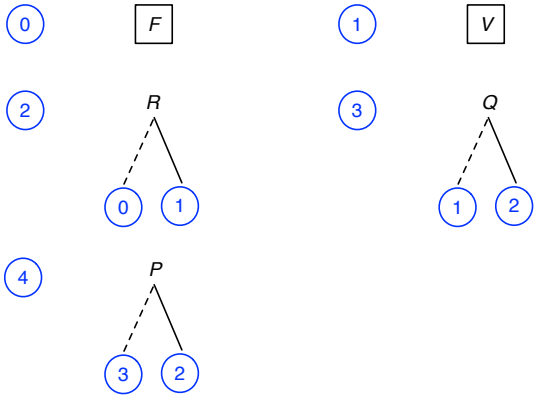
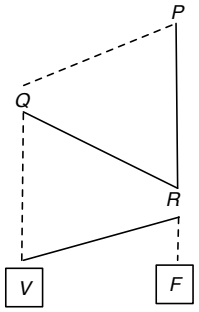


Tabela associativa:

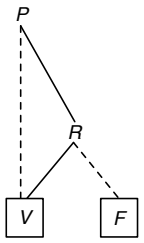


(b) (1.0) Aplique o algoritmo $compacta$ ao resultado obtido na alínea anterior.

Resposta:

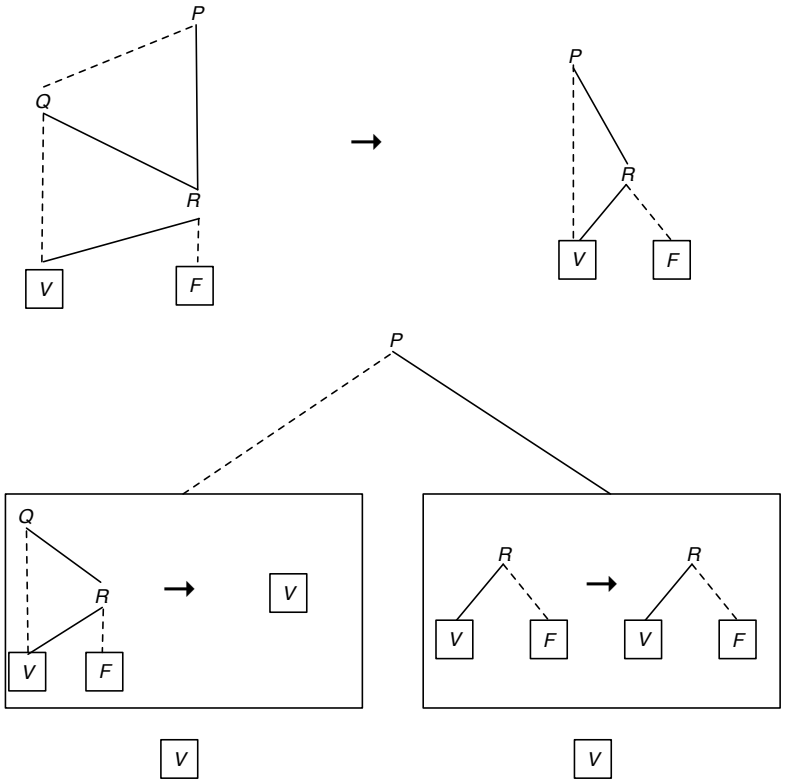


(c) (1.0) Considere que o OBDD abaixo, designado por $OBDD_\beta$, representa a *fbf* β:



Determine o OBDD resultante de $aplica(\rightarrow, OBDD_\alpha, OBDD_\beta)$. O que pode concluir sobre a *fbf* $\alpha \rightarrow \beta$? Justifique a sua resposta.

Resposta:



Todas as folhas são $\boxed{V}$, logo o OBDD reduzido consiste na folha $\boxed{V}$.
A *fbf* $\alpha \rightarrow \beta$ é uma tautologia, porque o seu OBDD reduzido é a folha $\boxed{V}$.

8. (2.0) Considere o seguinte conjunto de *fbfs* :

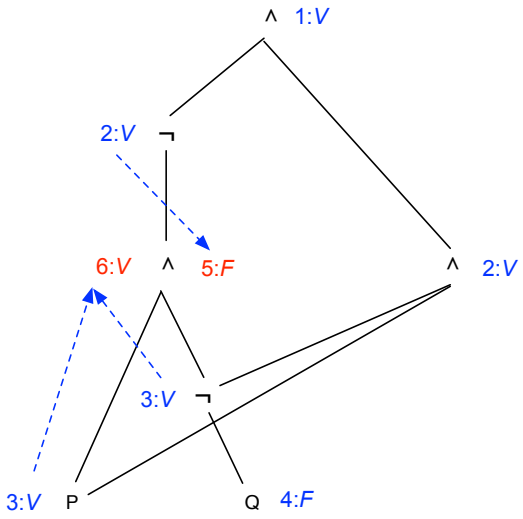
$$\{P \rightarrow Q, P, \neg Q\}$$

Use o algoritmo de propagação de marcas para determinar se o conjunto é satisfazível. Em caso afirmativo apresente uma testemunha.

Resposta:

Eliminação da implicação:

$$\neg(P \wedge \neg Q) \wedge (P \wedge \neg Q)$$



A propagação das marcas no sentido ascendente leva a uma contradição, logo o conjunto dado não é satisfazível.

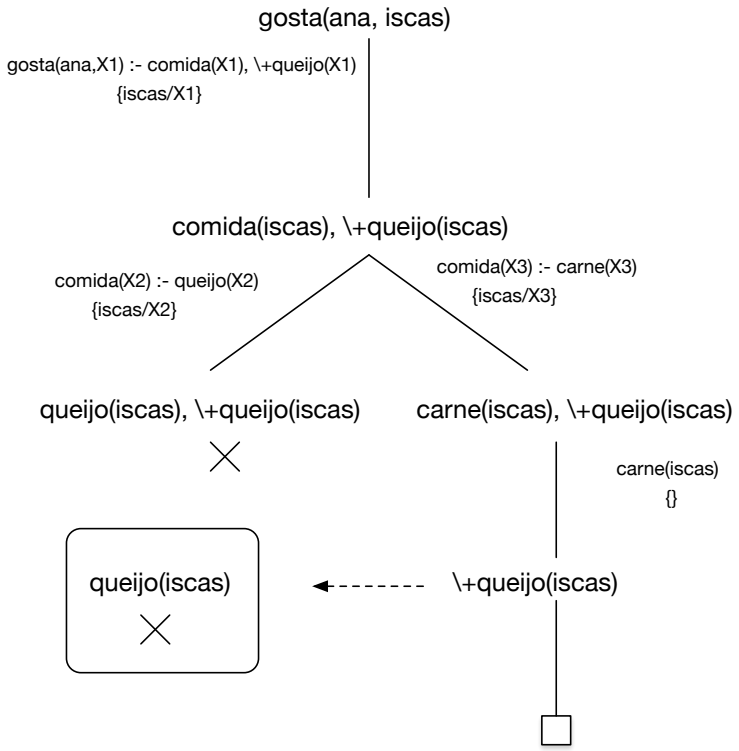
9. Considere o seguinte programa em PROLOG :

```
gosta(ana, X) :-
    comida(X),
    \+ queijo(X).

queijo(camembert).
queijo(chevre).

carne(iscas).
carne(lombo).

comida(X) :- queijo(X).
comida(X) :- carne(X).
```

(a) (1.0) Quais as respostas do PROLOG ao objectivo `?- gosta(ana, iscas)`? Desenhe a árvore SLD gerada, indicando em cada ramo a cláusula usada e respectiva substituição.

Resposta:

A resposta do PROLOG ao objectivo `?- gosta(ana, iscas)` é `true`.



(b) (0.5) Quais as respostas do PROLOG ao objectivo `?- comida(X), \+ carne(X)`?
Resposta:

```
X = camembert
X = chevre
```

(c) (0.5) Quais as respostas do PROLOG ao objectivo `?- \+ queijo(X), comida(X)`?
Resposta:

```
false
```

10. (a) (1.0) Defina o predicado `substitui/4` tal que `substitui(X, Y, L1, L2)` tem o valor verdadeiro se a lista `L2` corresponder à lista `L1` onde as ocorrências de `X` são substituídas por Y. Por exemplo, `substitui(2, 4, [1, 2, 3], [1, 4, 3])` tem o valor verdadeiro. (Assuma que L1 é uma lista de inteiros.)
Resposta:

```
substitui(_, _, [], []).
substitui(X, Y, [X|T1], [Y|T2]) :- substitui(X, Y, T1, T2).
substitui(X, Y, [H|T1], [H|T2]) :- H \= X, substitui(X, Y, T1, T2).
```

(b) (1.0) Considere o predicado xpto/4 definido abaixo.

```
xpto(X,Y,L1,L3) :- substitui(X, Y, L1, L2),
                   substitui(Y, X, L2, L3).
```

Diga que é devolvido pela avaliação que se segue:

```
?- xpto(6, 8, [6, 3, 8], L).
```

Resposta:

```
 L = [6, 3, 6].
```

11. No contexto do projecto:

    (a) (1.5) Defina o predicado `dir/3`, tal que `dir(Pos1, Pos2, Dir)` significa que, para ir da posição `Pos1` para a posição `Pos2`, sendo `Pos1` e `Pos2` duas posições adjacentes, é necessário fazer um movimento na direcção `Dir`. Se `Pos1` e `Pos2` não forem adjacentes, `dir(Pos1, Pos2, Dir)` é falso. Por exemplo:

```
?- dir((1,1), (1, 2), Dir).
Dir = d.

?- dir((1, 2), (2, 2), Dir).
Dir = b.

?- dir((1,2), (1,4), D).
false.
```

Resposta:

```
dir((L1, C1), (L2, C2), Dir) :-
    Dif_Linhas is L2 - L1,
    Dif_Colunas is C2 - C1,
    member(((Dif_Linhas, Dif_Colunas), Dir),
           [((1, 0), b), ((-1, 0), c), ((0, 1), d), ((0, -1), e)]),
    !.
```

    (b) (1.5) Usando o predicado `dir` da alínea anterior, escreva o predicado `adjacentes/1`, tal que `adjacentes(Lst_Pos)`, em que `Lst_Pos` é uma lista não vazia de posições, significa que quaisquer duas posições seguidas na lista são adjacentes. Por exemplo,

```
?- adjacentes([(1,1), (2,1), (3,1), (3,2), (3,3)]).
true.

?- adjacentes([(1,1), (2,2), (3,1), (3,2), (3,3)]).
false.
```

Resposta:

```
adjacentes([_]) :- !.
adjacentes([P1 , P2 | R]):-
    dir(P1, P2, _),
    adjacentes([P2 | R]).
```