TÉCNICO LISBOA

Lógica para Programação

Primeiro Exame

12 de Junho de 2015

09:00–11:00

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) Numa lógica não completa, nenhum argumento válido é demonstrável.
   Resposta: ______
   Resposta:
   Falsa

   (b) Se (Δ, α) é um contra-argumento para o argumento (Δ', α'), então os dois argumentos têm a mesma forma.
   Resposta: ______
   Resposta:
   Verdadeira

   (c) As ordenações para BDDs $[P,\ Q,\ S]$ e $[R,\ P,\ T,\ Q,\ S]$ são compatíveis.
   Resposta: ______
   Resposta:
   Verdadeira

   (d) Uma regra de procura permite escolher um literal de uma cláusula objectivo como candidato na aplicação do princípio da resolução.
   Resposta: ______
   Resposta:
   Falsa

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: ______
   Resposta:
   A.

(b) No PROLOG, o predicado da unificação (=):

A. Tem sucesso apenas se os dois termos forem iguais.
B. Avalia a expressão do lado direito e unifica com a expressão do lado esquerdo.
C. Avalia a expressão do lado esquerdo e unifica com a expressão do lado direito.
D. Tem sucesso se os dois termos forem unificáveis.

Resposta: ______

Resposta:

D

3. Considere que α, β, γ e δ são proposições. Sabendo que o argumento

α
β
$\therefore\ \neg\gamma$

é válido, diga, justificando, o que se pode concluir sobre a validade dos seguintes argumentos (válido, não válido, ou não se pode concluir nada):

(a) **(0.5)** α
β
$\therefore\ \gamma \rightarrow \delta$

Resposta:

Da validade do argumento original podemos concluir que é impossível ter α e β verdadeiras e γ verdadeira. Podemos assim concluir que sendo α e β verdadeiras o antecedente da implicação $\gamma \rightarrow \delta$ é sempre falso, pelo que a implicação é sempre verdadeira. Logo o argumento é válido.

(b) **(0.5)** α
β
γ
$\therefore\ \delta$

Resposta:

Da validade do argumento original podemos concluir que é impossível ter α e β verdadeiras e γ verdadeira. Podemos assim concluir que é impossível ter α, β e γ verdadeiras e δ falsa, logo o argumento é válido.

4. **(1.0)** Indique qual o resultado que se obtém ao passar a seguinte *fbf* para a forma clausal, eliminando eventuais cláusulas subordinadas.

$$(P \rightarrow \neg R) \rightarrow \neg(Q \rightarrow \neg R)$$

Resposta:

$$(P \rightarrow \neg R) \rightarrow \neg(Q \rightarrow \neg R)$$

- Eliminação do símbolo $\rightarrow$

$$\neg(\neg P \vee \neg R) \vee \neg(\neg Q \vee \neg R)$$

- Redução do domínio do símbolo $\neg$

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

$$(P \wedge R) \vee (Q \wedge R)$$

- Obtenção da forma conjuntiva normal

$$((P \wedge R) \vee Q) \wedge ((P \wedge R) \vee R)$$

$$(P \vee Q) \wedge (R \vee Q) \wedge (P \vee R) \wedge (R \vee R)$$

- Eliminação do símbolo $\wedge$

$$\{(P \vee Q), (R \vee Q), (P \vee R), (R \vee R)\}$$

- Eliminação do símbolo $\vee$

$$\{\{P, Q\}, \{R, Q\}, \{P, R\}, \{R\}\}$$

- Eliminação das cláusulas subordinadas

$$\{\{P, Q\}, \{R\}\}$$

5. **(1.5)** Demonstre o seguinte teorema usando o sistema de dedução natural da lógica proposicional (apenas pode utilizar as regras Prem, Rep, Reit e introdução e eliminação de cada uma das conectivas):

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

Resposta:

1	$\mid\ (P \rightarrow Q) \wedge (P \rightarrow \neg Q)$	Hip
2	$\mid\ P \rightarrow Q$	E$\wedge$, 1
3	$\mid\ P \rightarrow \neg Q$	E$\wedge$, 1
4	$\mid\ \mid\ P$	Hip
5	$\mid\ \mid\ P \rightarrow Q$	Rei, 2
6	$\mid\ \mid\ Q$	E$\rightarrow$, (4, 5)
7	$\mid\ \mid\ P \rightarrow \neg Q$	Rei, 3
8	$\mid\ \mid\ \neg Q$	E$\rightarrow$, (4, 7)
9	$\mid\ \neg P$	I$\neg$, (4, (6, 8)
10	$((P \rightarrow Q) \wedge (P \rightarrow \neg Q)) \rightarrow \neg P$	I$\rightarrow$, (1, 9)

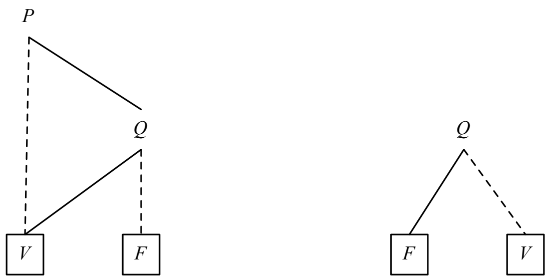
6. **(1.5)** Demonstre o seguinte teorema usando o sistema de dedução natural da lógica de primeira ordem (apenas pode utilizar as regras Prem, Rep, Reit e introdução e eliminação de cada uma das conectivas e quantificadores):

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

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)

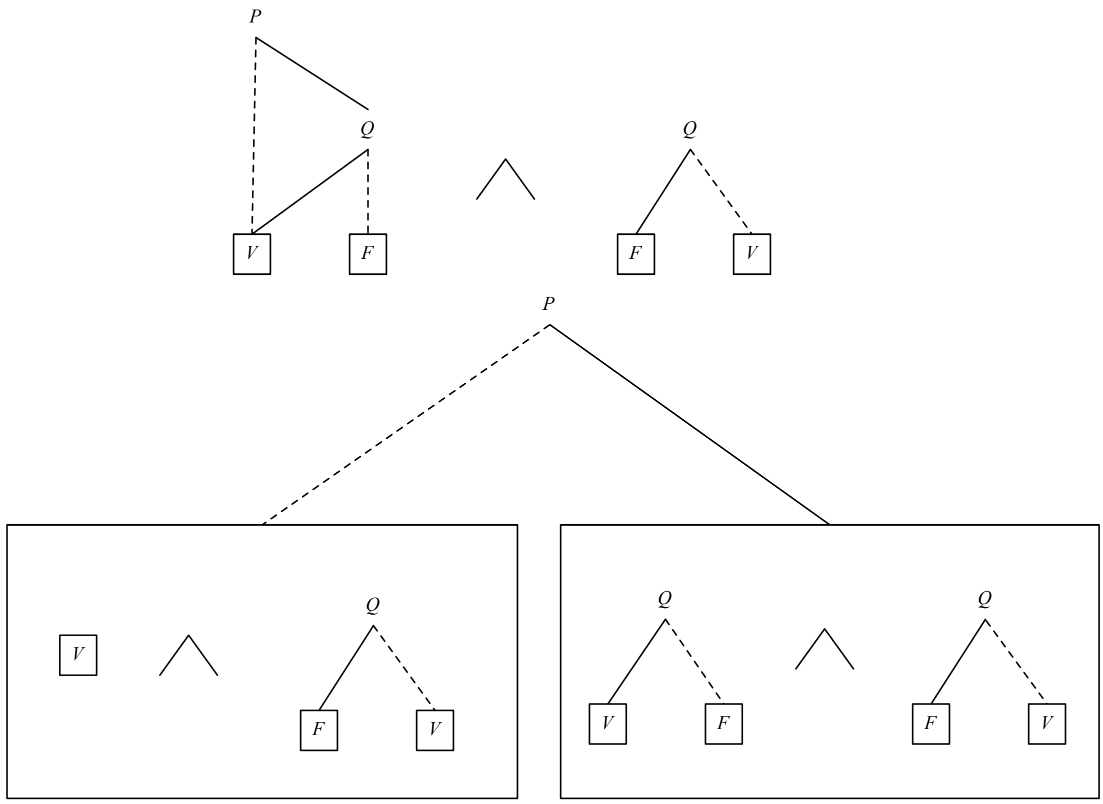
7. Considere as *fbfs* $P \rightarrow Q$ e $\neg Q$ cujos OBDDs reduzidos são, respectivamente:

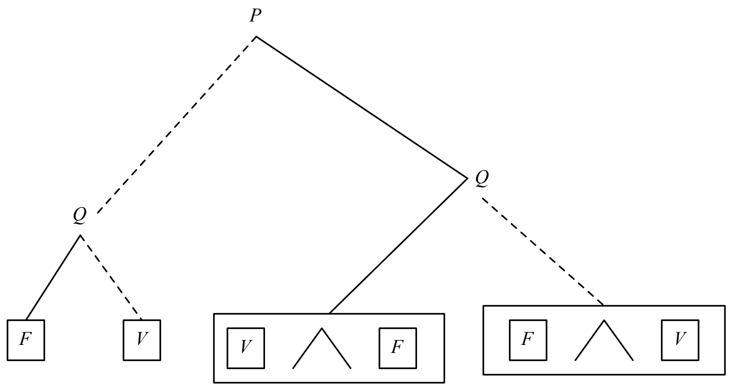


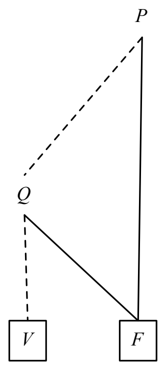
(a) **(1.5)** Utilizando o algoritmo *aplica*, determine o OBDD reduzido da *fbf*

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

Resposta:







(b) **(0.5)** O resultado obtido na alínea anterior permite concluir que $\{P \to Q, \neg Q\} \models \neg P$? Justifique a sua resposta.
Resposta:
Sim. Com efeito, o OBDD anterior permite concluir que a *fbf* $(P \to Q) \wedge Q$ tem um modelo: $M1(P) = F$ e $M1(Q) = F$. Neste modelo, a *fbf* $\neg P$ é verdadeira.

8. Considere os predicados $Tokra(x)$ (x é um *Tokra*), $Pai(x, y)$ (x é pai de y) e $Mae(x, y)$ (x é mãe de y). Represente, em Lógica de Primeira Ordem, as seguintes proposições:

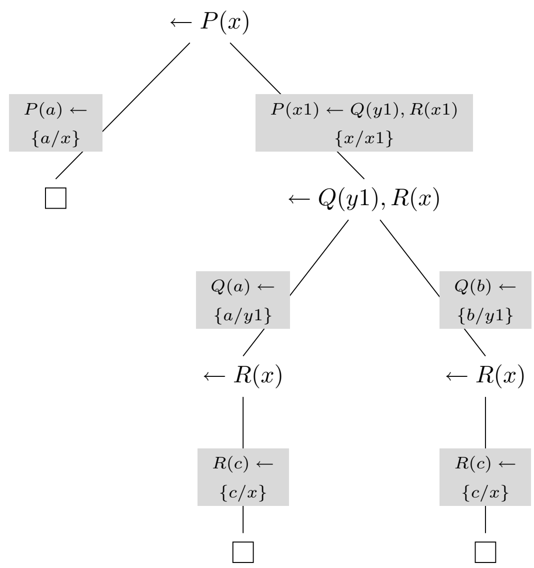
   (a) **(0.5)** Qualquer *Tokra* tem um pai e também uma mãe que é a mesma para todos os *Tokras*.
   Resposta:
   $\exists z[\forall x[Tokra(x) \to \exists y[Pai(y, x)] \wedge Mae(z, x)]]$

   (b) **(0.5)** Existe um *Tokra* que não é pai de ninguém.
   Resposta:
   $\exists x[Tokra(x) \wedge \neg \exists y[Pai(x, y)]]$

9. **(2.0)** 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 $\leftarrow P(x)$. (Em cada ramo da árvore indique a cláusula e substituição respectivas.)

   $P(a).$
   $P(x) \leftarrow Q(y), R(x).$
   $Q(a).$
   $Q(b).$
   $R(c).$

   Resposta:



As soluções são: $\{a/x\}$ e $\{c/x\}$.

10. **(1.5)** Considere a *fbf* $\{\{\neg A, B, C\}, \{\neg B, C, \neg D\}, \{\neg C, D\}, \{\neg A, \neg B, C\}, \{D\}, \{B, \neg C\}\}$ e o algoritmo DP implementado recorrendo a baldes.

    Complete a tabela que se segue:

Baldes	Cláusulas originais	Resolventes
b_A		
b_B		
b_C		
b_D		

Conclui que a *fbf* é satisfazível ou não satisfazível? Justifique. No caso de a *fbf* ser satisfazível indique uma testemunha.

Resposta:

Baldes	Cláusulas originais	Resolventes
b_A	$\{\neg A, B, C\}$ $\{\neg A, \neg B, C\}$	
b_B	$\{\neg B, C, \neg D\}$ $\{B, \neg C\}$	
b_C	$\{\neg C, D\}$	
b_D	$\{D\}$	

A *fbf* é satisfazível. Testemunha: $I(A) = I(B) = I(C) = F$, $I(D) = V$.

11. **(1.0)** Considere o predicado `parte_lista/4` tal que `parte_lista(L, N, L1, L2)` significa que `L` é uma lista, `N` é um inteiro, `L1` é a lista que contém os primeiros `N` elementos de `L` e `L2` é a lista que contém os restantes elementos de `L`.

    Por exemplo:

```
?- parte_lista([a,b,c,d,e,f,g,h,i,k],3,L1,L2).
L1 = [a,b,c]
L2 = [d,e,f,g,h,i,k]
```

Complete o código que se segue:

```
parte_lista(L, 0,    ,    ).
parte_lista([X|Xs], N, [X|Ys], Zs) :-
```

Resposta:

```
parte_lista(L,0,[],L).
parte_lista([X|Xs],N,[X|Ys],Zs) :- N > 0,
                                   N1 is N - 1,
                                   parte_lista(Xs,N1,Ys,Zs).
```

12. **(1.0)** Considere o predicado `comprime_lista/2` tal que `comprime_lista(L1, L2)` significa que `L1` é uma lista eventualmente com elementos consecutivos repetidos e `L2` é uma lista com os mesmos elementos de `L1` mas sem repetições consecutivas.

    Por exemplo:

```
?- comprime_lista([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X).
X = [a,b,c,a,d,e].
```

Complete o código que se segue:

```
comprime_lista([],    ).
comprime_lista([X],    ).
comprime_lista([X,X|Xs],Zs) :-
comprime_lista([X,Y|Ys],[X|Zs]) :-
```

Resposta:

```
comprime_lista([],[]).
comprime_lista([X],[X]).
comprime_lista([X,X|Xs],Zs) :- comprime_lista([X|Xs],Zs).
comprime_lista([X,Y|Ys],[X|Zs]) :- X \= Y,
                                   comprime_lista([Y|Ys],Zs).
```

13. **(1.0)** Considere o seguinte programa em PROLOG:

```
serie('Game of Thrones').
serie(gal·ctica).
serie('csi NY').
canalTV(axn).
canalTV(mov):-!.
canalTV(fox).
passa1(S,C) :- !, serie(S), canalTV(C).
passa2(S,C) :- serie(S), !, canalTV(C).
passa3(S,C) :- serie(S), canalTV(C), !.
```

Indique todos os valores devolvidos para os objectivos `passa1(X,Y)`,`passa2(X,Y)` e `passa3(X,Y)`.

Resposta:

```
?-passa1(X,Y).
X='Game of Thrones',
Y=axn;
X='Game of Thrones',
Y=mov;
X=gal·ctica,
Y=axn;
X=gal·ctica,
Y=mov;
X='csi NY',
Y=axn;
X='csi NY',
Y=mov.

?-passa2(X,Y).
X='Game of Thrones',
Y=axn;
X='Game of Thrones',
y=mov.

?-passa3(X,Y).
X='Game of Thrones',
Y=axn.
```

14. **(1.5)** A heurística n-MaxSwap para o 8-puzzle baseia-se na suposição de que se pode mover qualquer peça do 8-puzzle para o espaço vazio num único salto. Tendo esta ideia em consideração a fórmula de cálculo para esta heurística é bastante simples, bastando somar o custo de todas as peças onde:

   - Se a peça está na posição final, **custo = 0**, pois não é necessário nenhum salto para colocar esta peça na posição final.
   - Se a posição final da peça está vazia no estado actual, **custo = 1**, é necessário apenas um salto da peça para a posição actualmente vazia.
   - Se a posição final da peça não estiver vazia no estado actual, **custo = 2**, pois são necessários dois saltos, um para esvaziar a posição final, e outro para mover a peça pretendida para lá.

   Implemente o predicado `nMaxSwap(C, CObjectivo, H)`, onde `C` corresponde à configuração actual, `CObjectivo` à configuração final pretendida, e `H` ao valor heurístico, de modo a implementar a heurística descrita acima. Por exemplo:

   ```
   ?-nMaxSwap([1,2,3,4,5,6,0,8,7],[1,2,3,4,5,6,7,8,0],H).
   H=1.
   ```

   Sugestão: a implementação desta heurística poderá ser semelhante à heurística da distância de hamming, comparando as pecas da configuração final com a configuração actual.

   Resposta:

```
nMaxSwap([], [], 0).
nMaxSwap([P|C], [P|CObjectivo], H):- !, nMaxSwap(C, CObjectivo, H).
nMaxSwap([_|C], [0|CObjectivo], H):- !, nMaxSwap(C, CObjectivo, H).
nMaxSwap([0|C], [_|CObjectivo], H):-
                          !, nMaxSwap(C, CObjectivo, H_aux),
                          H is H_aux + 1.
nMaxSwap([_|C], [_|CObjectivo], H):-
                          nMaxSwap(C, CObjectivo, H_aux),
                          H is H_aux + 2.
```