TÉCNICO LISBOA

Lógica para Programação

Solução do Exame de 1ª Época

15 de Junho de 2018

15:00–17:00

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) O conjunto das tautologias está contido no conjunto das *fbfs* satisfazíveis.
   Resposta: ___
   Resposta:
   V

   (b) Todos os argumentos válidos têm uma conclusão verdadeira.
   Resposta: ___
   Resposta:
   F

2. (1.0) Considere as constantes $Dustin$ e $Dart$ e os seguintes predicados:

   $Amigo(x, y)$: se x é amigo de y
   $Gosta\_de(x, y)$: se x gosta de y

   Represente em Lógica de Primeira Ordem as seguintes proposições:

   (a) *Se o Dustin gosta do Dart então o Dart é seu amigo*
   Resposta:
   $Gosta\_de(Dustin, Dart) \rightarrow Amigo(Dart, Dustin)$

   (b) *Nem todos os amigos de Dustin gostam de Dart.*
   Resposta:
   $\exists x[Amigo(x, Dustin) \wedge \neg Gosta\_de(x, Dart)]$

3. (1.5) Considere o seguinte conjunto de *fbfs* (em que x, y e z são variáveis, a é uma constante e f é uma função):

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

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

Unificador mais geral (se existir):
$\{x/y\} \circ \{f(x)/z\} \circ \{a/x\} = \{x/y, f(x)/z\} \circ \{a/x\} = \{a/y, f(a)/z, a/x\}$

4. (2.0) Demonstre que

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

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		$\forall x[P(x) \to R(x)]$	Prem
2		$\forall x[R(x) \to Q(x)]$	Prem
3		$\exists x[P(x)]$	Prem
4	x_0	$P(x_0)$	Hip
5		$\forall x[P(x) \to R(x)]$	Rei, 1
6		$P(x_0) \to R(x_0)$	E$\forall$, 5
7		$R(x_0)$	E$\to$, (4, 6)
8		$\forall x[R(x) \to Q(x)]$	Rei, 2
9		$R(x_0) \to Q(x_0)$	E$\forall$, 8
10		$Q(x_0)$	E$\to$, (7, 9)
11		$P(x_0)$	Rep, 4
12		$Q(x_0) \wedge P(x_0)$	I$\wedge$, (10, 11)
13		$\exists x[Q(x) \wedge P(x)]$	I$\exists$, 12
14		$\exists x[Q(x) \wedge P(x)]$	E$\exists$, (3, (4, 13))

5. (2.0) Demonstre o seguinte argumento

$$\{\forall x[P(x) \to \exists y[Q(y,x)]], P(a)\} \vdash \exists x[Q(x,a)]$$

usando resolução unitária e linear, fazendo uma prova por refutação.

Resposta:

- *Forma clausal das premissas e da negação da conclusão:*
  - $\forall x[P(x) \rightarrow \exists y[Q(y,x)]]$
    $\forall x[\neg P(x) \vee \exists y[Q(y,x)]]$
    $\forall x[\neg P(x) \vee Q(f(x),x)]$ (em que f é uma função de Skolem)
    $\neg P(x) \vee Q(f(x),x)$
    $\{\neg P(x), Q(f(x),x)\}$
  - $P(a)$
    $\{P(a)\}$
  - $\neg\exists x[Q(x,a)]$
    $\forall x[\neg Q(x,a)]$
    $\neg Q(x,a)$
    $\{\neg Q(x,a)\}$

- *Prova:*

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

6. Seja $\alpha = ((P \wedge Q) \rightarrow R) \vee ((\neg P \vee \neg Q) \rightarrow R)$.

   (a) (0.5) Complete a seguinte tabela de verdade:

P	Q	R	$P \wedge Q$	$(P \wedge Q) \rightarrow R$	$\neg P \vee \neg Q$	$(\neg P \vee \neg Q) \rightarrow R$	α
V	V	V			F	V	
V	V	F	V	F		V	
V	F	V		V	V	V	
V	F	F			V	F	
F	V	V	F	V		V	
F	V	F	F		V	F	
F	F	V	F	V	V	V	
F	F	F	F			F	

   (b) (0.5) Qual o menor conjunto Δ tal que $\Delta \models \alpha$? Justifique a sua resposta.

Resposta:

(a)

P	Q	R	$P \wedge Q$	$(P \wedge Q) \rightarrow R$	$\neg P \vee \neg Q$	$(\neg P \vee \neg Q) \rightarrow R$	α
V	V	V	V	V	F	V	V
V	V	F	V	F	F	V	V
V	F	V	F	V	V	V	V
V	F	F	F	V	V	F	V
F	V	V	F	V	V	V	V
F	V	F	F	V	V	F	V
F	F	V	F	V	V	V	V
F	F	F	F	V	V	F	V

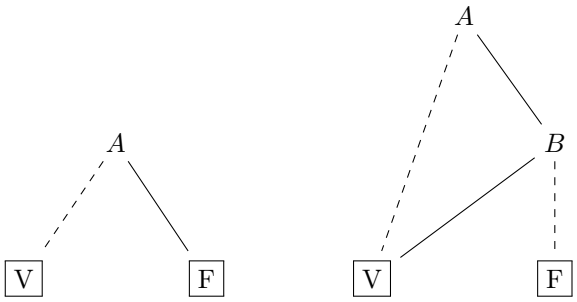
(b) Uma vez que α é uma tautologia, α é consequência semântica de qualquer conjunto de *fbfs* . Logo, $\Delta = \{\}$.

7. (1.0) No contexto de BDDs, as ordenações $\Omega_1 = [A, B, D]$ e $\Omega_2 = [A, C, D, B]$ são compatíveis? Justifique a sua resposta.

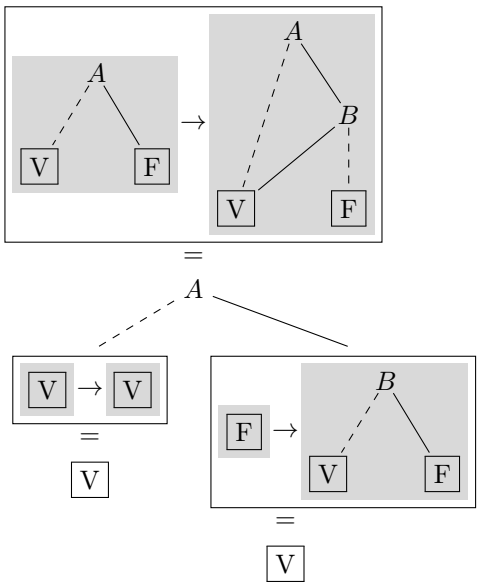
   Resposta:

   As ordenações Ω_1 e Ω_2 são compatíveis se nunca se verifica que, dados dois símbolos proposicionais x e y, x ocorre antes de y em Ω_1 e y ocorre antes de x em Ω_2. Assim, as ordenações Ω_1 e Ω_2 não são compatíveis porque para Ω_1 se verifica $B \prec D$ e para Ω_2 se verifica $D \prec B$.

8. (2.0) Considere os dois OBDDs reduzidos que representam as fbfs $\neg A$ e $A \rightarrow B$. Usando o algoritmo aplica, obtenha o OBDD reduzido para $\neg A \rightarrow (A \rightarrow B)$. O que pode concluir?



   Resposta:



   Após a compactação obtemos a folha $\boxed{V}$ pelo que podemos concluir que a fbf $\neg A \rightarrow (A \rightarrow B)$ é uma tautologia.

9. (3.0) Usando o algoritmo de propagação de marcas, prove que a fbf

   $(A \wedge (A \rightarrow B)) \rightarrow B$

é uma tautologia. Sugestão: Prove que a negação da fbf não é satisfazível.

Resposta:

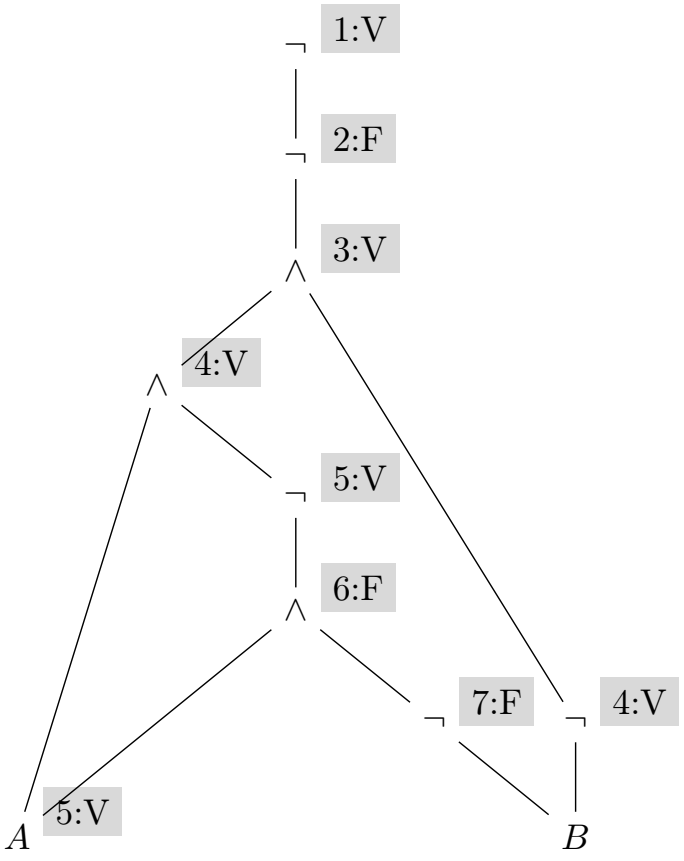
1. Negação da fbf

$\neg((A \wedge (A \rightarrow B)) \rightarrow B)$

2. Eliminação do símbolo $\rightarrow$ $((\alpha \rightarrow \beta) \equiv \neg(\alpha \wedge \neg\beta))$

$\neg\neg((A \wedge \neg(A \wedge \neg B)) \wedge \neg B)$

3. Aplicação do algoritmo de propagação de marcas



A aplicação do algoritmo de propagação de marcas origina duas marcas com valores contraditórios para o nó com o rótulo B: uma proveniente da marca 7:F e outra proveniente da marca 4:F. Conclui-se assim que a negação da fbf original é contraditória e consequentemente a fbf original é uma tautologia.

10. (1.5) Considere o seguinte programa em Prolog:

```
criatura_fofinha_1(X) :- criatura(X), not(come_gatos(X)).
criatura_fofinha_2(X) :- criatura(X), !, not(come_gatos(X)).
criatura_fofinha_3(X) :- not(come_gatos(X)), criatura(X).
criatura(pikachu).
criatura(dart).
criatura(catzilla).
```

```
come_gatos(dart).
```

Qual a resposta do Prolog aos seguintes objectivos (suponha que vai pedindo mais respostas, enquanto tal for possível):

(a) ?- `criatura_fofinha_1(X).`
Resposta:
```
X = pikachu;
X = catzilla.
```

(b) ?- `criatura_fofinha_2(X).`
Resposta:
```
X = pikachu.
```

(c) ?- `criatura_fofinha_3(X).`
Resposta:
```
false.
```

11. (a) (0.75) Complete a implementação do predicado `nth(Pos, Lista, Elem)` em que `Elem` é o elemento que ocupa a posição `Pos` da lista `Lista`.

```
/* Escreva aqui a condição de paragem em falta */



nth(Pos, [_|Tail], Elem) :-
                Pos_1 is Pos - 1,
                nth(Pos_1, Tail, Elem).
```

Resposta:

```
nth(1, [Elem|_], Elem) :- !.
nth(Pos, [_|Tail], Elem) :-
               Pos_1 is Pos - 1,
               nth(Pos_1, Tail, Elem).
```

(b) (0.75) Considere agora o predicado `xpto(Pos, Num, Lista)`, definido como se segue:

```
xpto(Pos, Num, Lista) :-
       nth(Pos, Lista, Elem),
       Elem < Num.
```

Escolha a única resposta correcta para as seguintes questões. Cada resposta certa vale 0.25 valores e cada resposta errada desconta 0.05 valores.

i. A resposta ao objectivo `?- xpto(6, 11, [5, 4, 5, 8, 9, 10]).` é:
A. false.
B. true.
C. X = 10.
D. Nenhuma das anteriores.
Resposta:

ii. A resposta ao objectivo `?- xpto(6, 5, [5, 4, 5, 8, 9, 10]).` é:
A. false.
B. true.
C. X = 11.
D. Nenhuma das anteriores.
Resposta:

iii. A resposta ao objectivo `?- xpto(X, 11, [5, 4, 5, 8, 9, 10]).` é:
A. false.
B. true.
C. X = 1.
D. Nenhuma das anteriores.
Resposta:

Resposta:

B, A, C.

12. (a) (1.0) No contexto do projecto, implemente o predicado `total_linha/3`, tal que `total_linha(Lin, Posicoes, Total)`, em que `Lin` é um número de linha e `Posicoes` é uma lista de posições, significa que `Total` é o número de posições de `Posicoes` pertencentes à linha `Lin`. Por exemplo, se `Posicoes` for a lista `[(4,3), (2,2), (4,5), (2,3), (2,7), (1,3), (2,2), (1,5), (3,3)]`, temos

```
 ?- ..., total_linha(2, Posicoes, Total).
Total = 4.

?- ..., total_linha(5, Posicoes, Total).
Total = 0.
```

Resposta:

```
total_linha(_, [], 0) :- !.

total_linha(Lin, [(Lin,_) | R], Total) :-
    total_linha(Lin, R, Total_R),
    Total is Total_R + 1,
    !.

total_linha(Lin, [_ | R], Total) :-
    total_linha(Lin, R, Total).
```

ou

```
total_linha(Lin, Posicoes, Total) :-
    findall((Lin,Col), member((Lin,Col),Posicoes),Posicoes_Lin),
    length(Posicoes_Lin, Total).
```

(b) (1.5) Usando o predicado definido na alínea anterior, implemente o predicado `totais_linhas/3`, tal que `totais_linhas(Dim, Posicoes, Totais)`, em que `Dim` é a dimensão de um puzzle e `Posicoes` é uma lista de posições, significa que `Totais` é a lista de totais por linha da lista `Posicoes`. Por exemplo, sendo `Posicoes` a lista da alínea anterior, temos

```
?- ..., totais_linhas(5, Posicoes, Totais).
...,
Totais = [2, 4, 1, 2, 0].
```

Resposta:

```
totais_linhas(Dim, Posicoes, Totais) :-
    totais_linhas(Posicoes, Totais, Dim, []).

totais_linhas(_, Totais, 0, Totais) :-  !.

totais_linhas(Posicoes, Totais, Lin, Totais_parciais) :-
    total_linha(Lin, Posicoes, Total_Lin),
    N_Lin is Lin - 1,
    totais_linhas(Posicoes, Totais, N_Lin,
                  [Total_Lin | Totais_parciais]).
```