TÉCNICO LISBOA

Lógica para Programação

Solução do Exame de 2ª Época

5 de Julho de 2018

9:00–11:00

1. (1.0) Para cada uma das seguintes questões, escolha a única alternativa correcta. Cada resposta correcta vale 0.5 valores e *cada resposta errada desconta 0.2 valores*.

   (a) O conjunto das *fbfs* não satisfazíveis está contido no conjunto das contradições.
   Resposta: ___

   Resposta:

   V

   (b) Se $(\{\}, \alpha)$ é um argumento válido, então α é uma contradição.
   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
   $Diferente\_de(x, y)$: se x é diferente de y

   Represente em Lógica de Primeira Ordem as seguintes proposições:

   (a) *Se o Dart for amigo do Dustin, todos os amigos do Dustin gostam do Dart*

   Resposta:

   $Amigo(Dart, Dustin) \rightarrow \forall x(Amigo(x, Dustin) \rightarrow Gosta\_de(x, Dart))$

   (b) *Existem pelo menos dois amigos do Dustin que não gostam do Dart.*

   Resposta:

   $\exists x \exists y[Amigo(x, Dustin) \wedge Amigo(y, Dustin) \wedge Diferente(x, y) \wedge \neg Gosta\_de(x, Dart) \wedge \neg Gosta\_de(y, Dart)]$

3. (2.0) Demonstre que

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

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	$\forall x[S(x) \to P(x)]$	Prem
2	$\forall x[S(x) \to \neg P(x)]$	Prem
3	$\quad\vert\ \forall x[R(x) \wedge S(x)]$	Hip
4	$\quad\vert\ R(a) \wedge S(a)$	E$\forall$, 3
5	$\quad\vert\ S(a)$	E$\wedge$, 4
6	$\quad\vert\ \forall x[S(x) \to P(x)]$	Rei, 1
7	$\quad\vert\ \forall x[S(x) \to \neg P(x)]$	Rei, 2
8	$\quad\vert\ S(a) \to P(a)$	E$\forall$, 6
9	$\quad\vert\ S(a) \to \neg P(a)$	E$\forall$, 7
10	$\quad\vert\ P(a)$	E$\to$, (5, 8)
11	$\quad\vert\ \neg P(a)$	E$\to$, (5, 9)
12	$\neg \forall x[R(x) \wedge S(x)]$	I$\neg$, (3, (10, 11))

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

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

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

Unificador mais geral (se existir):

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

5. (2.0) Demonstre o seguinte argumento

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

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

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

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

6. (1.0) Considere a conceptualização (D, F, R) em que:

$D = \{\diamond, \square, \odot\}$
$F = \{\}$
$R = \{\ldots\}$.

Considere a interpretação I: $\{a, b, c, P, S\} \mapsto D \cup F \cup R$, tal que:

$I(a) = \diamond$
$I(b) = \square$
$I(c) = \odot$

Preencha a tabela abaixo, de forma a que a interpretação I seja um modelo do conjunto de *fbfs*

$$\Delta = \{\forall x[\neg S(x)], \forall x[P(x) \rightarrow S(x)]\}.$$

$I(P)$	
$I(S)$	

Resposta:

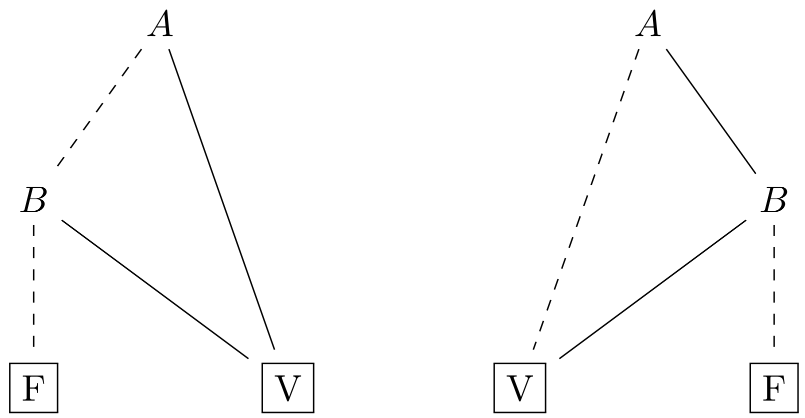
$I(P)$	$\{\}$
$I(S)$	$\{\}$

7. (1.5) Explique como pode obter os modelos de uma fbf a partir do OBDD que representa essa fbf.

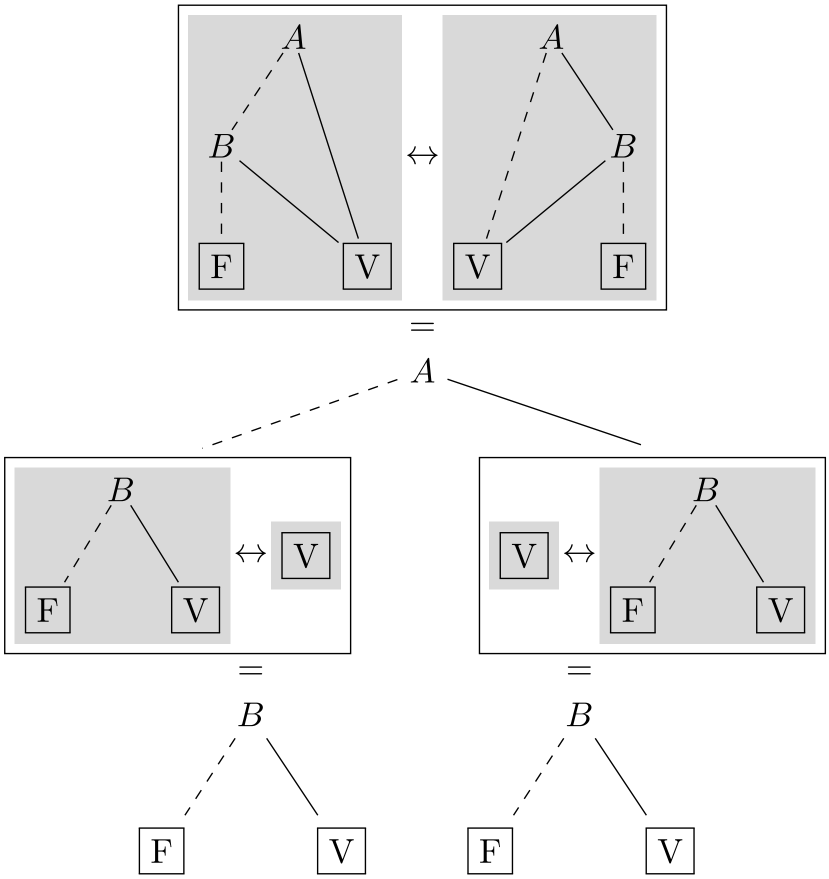
Resposta:

Os modelos de uma fbf podem ser obtidos a partir de um OBDD através dos caminhos que têm origem na raiz do BDD e terminam na folha V. A interpretação de cada símbolo proposicional depende do tipo de arco, sendo V para o arco cheio e F para o arco tracejado.

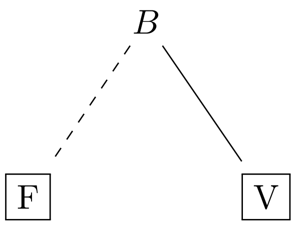
8. (2.0) Considere os dois OBDDs reduzidos que representam as fbfs $A \vee B$ e $A \to B$. Usando o algoritmo aplica, obtenha o OBDD reduzido para $(A \vee B) \leftrightarrow (A \to B)$. Recorde que $(\alpha \leftrightarrow \beta) \equiv (\alpha \to \beta) \wedge (\beta \to \alpha)$.



Resposta:



Após a compactação obtemos o BDD



9. (2.5) Use o algoritmo DP (usando a ordem alfabética) para provar que

   $\{A \to B, B \to C\} \models (A \to C)$.

   Sugestão: use o teorema da refutação.

   Resposta:

   Usando o teorema de refutação, provaremos que o conjunto de fbfs $\{A \to B, B \to C, \neg(A \to C)\}$ não é satisfazível. Logo, fica provado que $\{A \to B, B \to C\} \models (A \to C)$.
   1. Conversão para a forma clausal

   $\Delta = \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}\}$

   2. Aplicação do algoritmo DP seguindo a ordem alfabética

$\exists A(\Delta) = \{\{B\}, \{\neg B, C\}, \{\neg C\}\}$
$\exists B(\exists A(\Delta)) = \{\{C\}, \{\neg C\}\}$
$\exists C(\exists B(\exists A(\Delta))) = \{\{\}\}$

Como o último conjunto contém a cláusula vazia, podemos concluir que o conjunto $\{A \to B, B \to C, \neg(A \to C)\}$ não é satisfazível.

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

C_1: `criatura_fofinha(X) :- criatura(X), not(come_gatos(X)).`
C_2: `criatura(dart).`
C_3: `criatura(pikachu).`
C_4: `criatura(catzilla).`
C_5: `come_gatos(dart).`

Supondo que vão sempre ser pedidas mais respostas, enquanto tal for possível, qual a resposta do Prolog ao objectivo `?- criatura_fofinha(X)`...

(a) ... considerando o programa anterior.
Resposta:
```
X = pikachu;
X = catzilla.
```
(b) ... considerando que C_1 é agora:
`criatura_fofinha(X) :- criatura(X), !, not(come_gatos(X)).`

Resposta:
```
false
```
(c) ... considerando que a ordem das cláusula é agora C_1 C_3 C_4 C_2 C_5 e supondo que C_1 é a cláusula definida na alínea (b).
Resposta:
```
X = pikachu.
```

11. Considere a implementação do predicado `sufixo`, em que `sufixo(L1, L2)` é verdade se a lista L1 é um sufixo da lista L2:

```
sufixo(L, L).
sufixo(L1, [ _ | L2]) :- sufixo(L1, L2).
```

por exemplo:

```
?- sufixo(X, [8, 3, 4]).
X = [8, 3, 4] ;
X = [3, 4] ;
X = [4] ;
X = [] ;
false.
```

(a) (0.75) Implemente o predicado `sufixoFixo`, em que `sufixoFixo(L1, N, L2)` é verdade se a lista L1 é o sufixo da lista L2 de tamanho N.
Resposta:

```
sufixoFixo(L1, N, L2) :- sufixo(L1, L2), length(L1, N).
```

(b) (0.75) Altere a condição de paragem do predicado `sufixo`, de modo a que uma lista não seja considerada sufixo de si própria.

Resposta:

Nova condição de paragem: `sufixo(L, [_ | L]).`

12. No contexto do projecto, considere definidos os seguintes predicados:

- `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, se `Posicoes` for a lista `[(4,3), (2,2), (4,5), (2,3), (2,7), (1,3), (2,2), (1,5), (3,3)]`, temos

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

- `totais_colunas/3` faz o mesmo que `totais_linhas/3`, mas para as colunas.
- `propaga/3`, em que `propaga(Puz, Posicao, Posicoes_propagadas)` é o predicado definido no enunciado do projecto.
- `subset/2`, tal que `subset(L1,L2)`, em que `L1` e `L2` são listas, significa que todos os elementos de `L1` pertencem a `L2`.

(a) (1.5) Implemente o predicado `verifica_propaga/2`, tal que `verifica_propaga(Puz, Posicoes)`, em que `Puz` é um puzzle e `Posicoes` é uma lista de posições, significa que a lista resultante de propagar qualquer posição de `Posicoes` está contida em `Posicoes`.

Resposta:

```
verifica_propaga(Puz, Posicoes) :-
    verifica_propaga(Puz, Posicoes, Posicoes).

verifica_propaga(_, _,[]) :-!.

verifica_propaga(Puz, Posicoes, [Pos | R]) :-
    propaga(Puz, Pos, Pos_propagadas),
    subset(Pos_propagadas, Posicoes),
    verifica_propaga(Puz, Posicoes, R).
```

(b) (1.0) Usando o predicado definido na alínea anterior, implemente o predicado `verifica_solucao/2`, tal que `verifica_solucao(Puz, Posicoes)`, em que `Puz` é um puzzle e `Posicoes` é uma lista de posições, significa que `Posicoes` é uma solução para o puzzle `Puz`.

Recorda-se que um puzzle é representado por uma lista de 3 elementos, sendo o primeiro uma lista de termómetros, o segundo a lista dos totais por linha, e o terceiro a lista dos totais por coluna.

Resposta:

```
verifica_solucao([Terms, Totais_linhas, Totais_colunas],
                 Posicoes) :-
    length(Totais_linhas, Dim),
    totais_linhas(Dim, Posicoes, Totais_linhas),
    totais_colunas(Dim, Posicoes, Totais_colunas),
    verifica_propaga([Terms, Totais_linhas, Totais_colunas],
                     Posicoes).
```