TÉCNICO LISBOA

Lógica para Programação

Solução do Exame de 1ª Época

16 de Junho de 2017

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) Se 2 argumentos são ambos válidos ou inválidos, então têm a mesma forma.
   Resposta: ___
   Resposta:
   F

   (b) Se $\{\alpha_1, \ldots, \alpha_n\} \models \beta$, então o OBDD reduzido da *fbf* $\alpha_1 \wedge \ldots \wedge \alpha_n \wedge \neg\beta$ é a folha $\boxed{F}$.
   Resposta: ___
   Resposta:
   V

2. (1.0) Considere a constante $Deadpool$ e os seguintes predicados:

   $HeroiMarvel(x)$: x é um herói da Marvel
   $Gosta\_de(x, y)$: x gosta de y

   Represente em Lógica de Primeira Ordem as seguintes proposições:

   (a) *Todos os heróis da Marvel gostam do Deadpool.*
   Resposta:
   $\forall x[HeroiMarvel(x) \rightarrow Gosta\_de(x, Deadpool)]$

   (b) *Todos os heróis da Marvel gostam do Deadpool, mas o Deadpool não gosta de nenhum herói da Marvel.*
   Resposta:
   $\forall x[HeroiMarvel(x) \rightarrow (Gosta\_de(x, Deadpool) \wedge \neg Gosta\_de(Deadpool, x))]$

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

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

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

Unificador mais geral (se existir): não existe.

4. (2.0) Demonstre que

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

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

5. (2.0) Demonstre o seguinte argumento

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

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

Resposta:

- *Forma clausal das premissas e da negação da conclusão:*
  $\{\{\neg P(x_1), R(x_1)\}, \{P(x_2), R(x_2))\}, \{\neg R(a)\}\}$

- *Prova:*

1	$\{\neg P(x_1), R(x_1)\}$	Prem
2	$\{P(x_2), R(x_2)\}$	Prem
3	$\{\neg R(a)\}$	Prem
4	$\{\neg P(a)\}$	Res, (1,3), $_{\{a/x_1\}}$
5	$\{R(a)\}$	Res, (2,4), $_{\{a/x_2\}}$
6	$\{\}$	Res, (3,5), $_{\epsilon}$

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

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

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

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

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

   $$\Delta = \{\neg P(a), \neg P(b), S(a) \vee P(a), P(a) \rightarrow S(b)\}.$$

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

   Resposta:

$I(P)$	$\{\}$
$I(S)$	$\{(\diamond), (\square)\}$

   ou

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

7. (2.0) Considere a fbf $\neg((P \vee Q) \wedge \neg R)$ e a ordem $P \prec Q \prec R$. Obtenha o seu OBDD reduzido, por aplicação dos algoritmos *reduz* e *compacta* à respectiva árvore de decisão.

   Resposta:

   Tabela de verdade (não é obrigatória a sua apresentação, no entanto a mesma facilita a resolução do exercício):

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

Atribuição de marcadores:

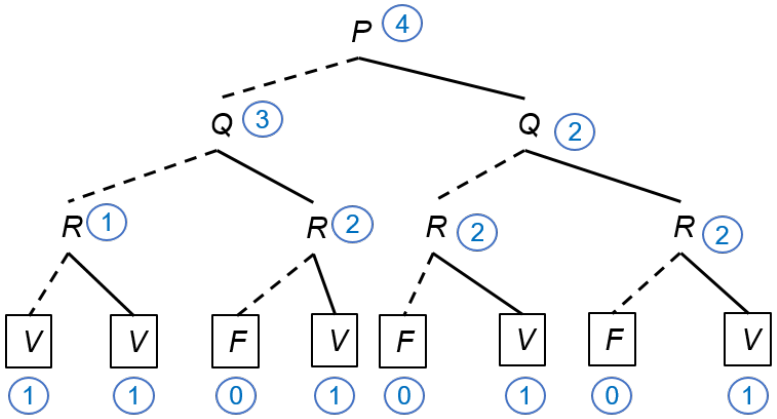
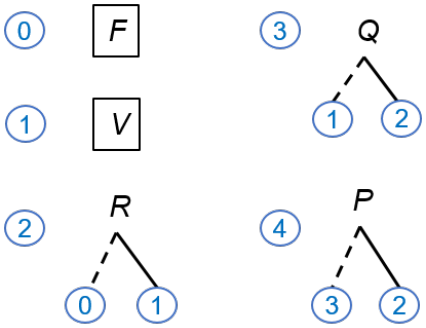
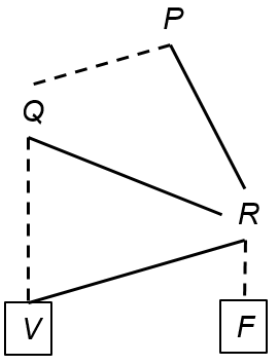


Tabela associativa:

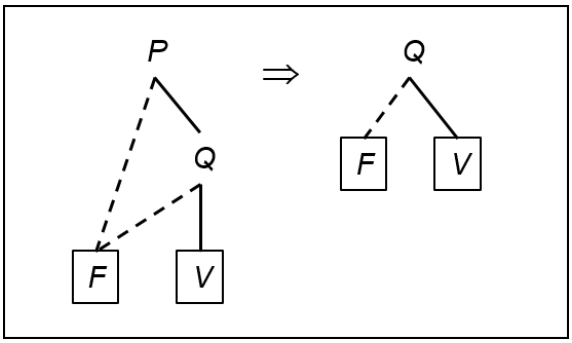


Compactação:



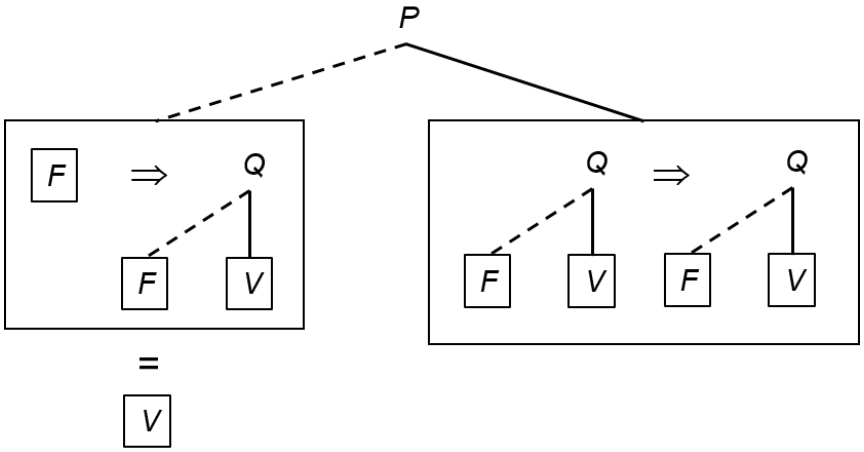
8. (2.0) Considere a relação de ordem $P \prec Q$.

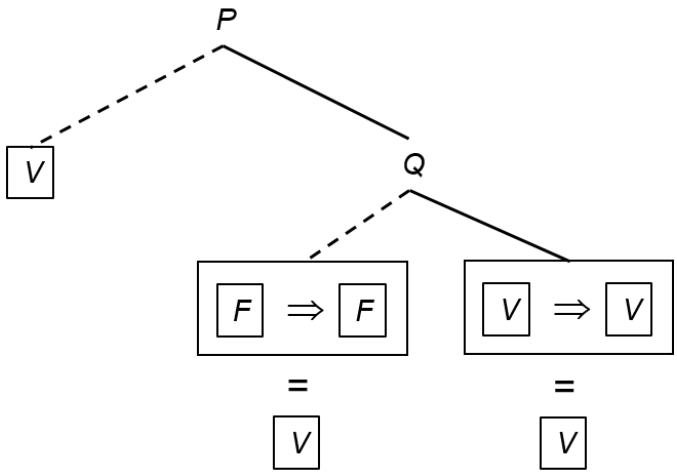
(a) (1.5) Utilize o algoritmo aplica para obter o OBDD reduzido da seguinte *fbf*.



Resposta:

Aplica:





Atribuição de marcadores:

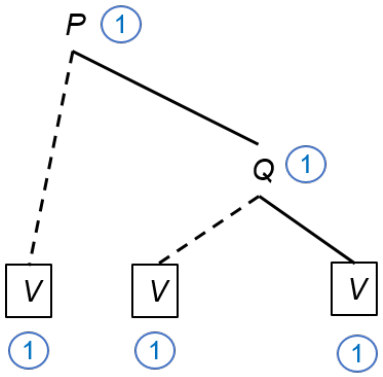


Tabela associativa:

Compactação:

(b) (0.5) Com base no resultado da alinea anterior, o que pode concluir acerca da sua satisfazibilidade?
Resposta:
Podemos concluir que a fbf é uma tautologia.

9. (2.0) Considere o conjunto de cláusulas

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

e o algoritmo de Davis-Putnam (DP).

(a) (1.5) Introduza a informação em falta resultante da aplicação do algoritmo DP.

$\exists A(\Delta) =$
$\exists B(\exists A(\Delta)) =$
$\exists C(\exists B(\exists A(\Delta))) =$

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

(b) (0.5) O que pode concluir em relação à satisfazibilidade de Δ após a aplicação do algoritmo DP? Justifique.
Resposta:

O conjunto de cláusulas Δ é não satisfazível porque é derivada a cláusula vazia ({}) como resultado da aplicação do algoritmo DP.

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

```
heroi(capitaoAmerica).
% heroi(deadpool) :- !.
heroi(homemDeFerro).
voa(homemDeFerro).
heroi_pes_na_terra_1(X) :- heroi(X), not(voa(X)).
heroi_pes_na_terra_2(X) :- not(voa(X)), heroi(X).
```

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

(a) ?- `heroi_pes_na_terra_1(X).`
Resposta:
```
X = capitaoAmerica ;
false.
```

(b) ?- `heroi_pes_na_terra_2(X).`
Resposta:
```
false.
```

(c) ?- `heroi_pes_na_terra_1(X).` (mas agora supondo que a segunda cláusula não está comentada)
Resposta:
```
X = capitaoAmerica ;
X = deadpool.
```

11. (a) (1.0) Defina o predicado `listaMarvel(L1, L2)` que significa que a lista `L2` contém os elementos de `L1` que são heróis da Marvel. Suponha definido o predicado `heroiMarvel(X)` que significa que `X` é um herói da Marvel.
Resposta:

```
listaMarvel([], []) :- !.
listaMarvel([ H | T], [H | L]) :-
    heroiMarvel(H), !,
    listaMarvel(T, L).
listaMarvel([ H | T], L) :-
    \+ heroiMarvel(H),
    listaMarvel(T, L).
```

(b) (0.5) Supondo definido o predicado `subtract(L, L1, L2)` em que L2 é a lista resultante de retirar todos os elementos da lista L1 da lista L, defina o predicado `listaOutros(L1, L2)` que significa que a lista `L2` contém os elementos de `L1` que não são heróis da Marvel.
Resposta:
```
listaOutros(L1, L2) :- listaMarvel(L1, L3), subtract(L1, L3, L2).
```

12. (a) (1.5) No contexto do projecto, implemente o predicado `acrescenta_num_elementos_posicao/3`, tal que `acrescenta_num_elementos_posicao(Puz, Pos, N_Puz)`

significa que `N_Puz` é o puzzle resultante de modificar o conteúdo da posição `Pos` do puzzle `Puz` da seguinte forma: o novo conteúdo é uma lista de 2 elementos, em que o primeiro elemento é o número de elementos do conteúdo original, e o segundo elemento, é o conteúdo original. Por exemplo, se o conteúdo original for `[1,2,5]`, o novo conteúdo será `[3,[1,2,5]]`. Sugestão: utilize os predicados `puzzle_ref(Puz,Pos,Cont)` e `puzzle_muda(Puz,Pos,Cont,N_Puz)`.

Resposta:

```
acrescenta_num_elementos_posicao(Puz,Pos,N_Puz) :-
    puzzle_ref(Puz,Pos,Cont),
    length(Cont,Num_elementos),
    puzzle_muda(Puz,Pos,[Num_elementos,Cont],N_Puz).
```

(b) (1.0) Usando o predicado definido na alínea anterior, implemente o predicado `acrescenta_num_elementos_puzzle/2`, tal que
`acrescenta_num_elementos_puzzle(Puz, N_Puz)`
significa que `N_Puz` é o puzzle resultante de aplicar o predicado `acrescenta_num_elementos_posicao/3`
a todas as posições do puzzle `Puz`. Por exemplo, sendo `Puz` o puzzle

```
[[[3],[2,4],[1],[2,4]],
[[1,2,4],[1,2,4],[3,4],[2,3,4]],
[[1,4],[1,3,4],[2],[1,3,4]],
[[1,2,4],[1,2,3,4],[3,4],[1,3,4]]],
```

teríamos

```
?- ..., acrescenta_num_elementos_puzzle(Puz,N_Puz), escreve(N_Puz).
[[[1,[3]],[2,[2,4]],[1,[1]],[2,[2,4]]],
[[3,[1,2,4]],[3,[1,2,4]],[2,[3,4]],[3,[2,3,4]]],
[[2,[1,4]],[3,[1,3,4]],[1,[2]],[3,[1,3,4]]],
[[3,[1,2,4]],[4,[1,2,3,4]],[2,[3,4]],[3,[1,3,4]]]]
Puz = [[[3], [2, 4], [1], .]]],
N_Puz = [[[1, [3]], [2, [2, 4]], [1, [1]...]
```

Sugestão: utilize os predicados `todas_posicoes(Posicoes)` e `percorre_muda_Puz(Puz,Accao,Posicoes,N_Puz)`.

Resposta:

```
acrescenta_num_elementos_puzzle(Puz,N_Puz) :-
    todas_posicoes(Todas),
    percorre_muda_Puz(Puz,acresenta_num_elementos_posicao,Todas,N_Puz).
```