TÉCNICO LISBOA

Lógica para Programação

Solução do Exame de 2ª Época

6 de Julho de 2017

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) Um programa em Prolog é uma sequência de
      A. regras e objectivos.
      B. afirmações e regras.
      C. afirmações e objectivos.

   Resposta: ___

   Resposta:

   B

   (b) Uma função de selecção
      A. recebe um programa e devolve uma afirmação.
      B. recebe um objectivo e devolve uma regra.
      C. recebe um objectivo e devolve um dos seus sub-objectivos.

   Resposta: ___

   Resposta:

   C

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) (0.5) *Nenhum herói da Marvel gosta do Deadpool*

   Resposta:

   $\forall x[HeroiMarvel(x) \rightarrow \neg Gosta\_de(x, Deadpool)]$

   (b) (0.5) *Todos os heróis da Marvel gostam de um herói da Marvel*

   Resposta:

   $\forall x[HeroiMarvel(x) \rightarrow \exists y[HeroiMarvel(y) \land Gosta\_de(x, y)]]$

3. (2.0) Demonstre o seguinte argumento

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

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

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

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

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

Unificador mais geral (se existir):

$\{x/y\} \circ \{a/x\} = \{a/y, a/x\}$

5. (2.0) Demonstre o seguinte teorema

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

usando resolução.

Resposta:

Para provar o teorema teremos de fazer uma prova por refutação:

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

- *Prova:*

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

6. (1.5) 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*

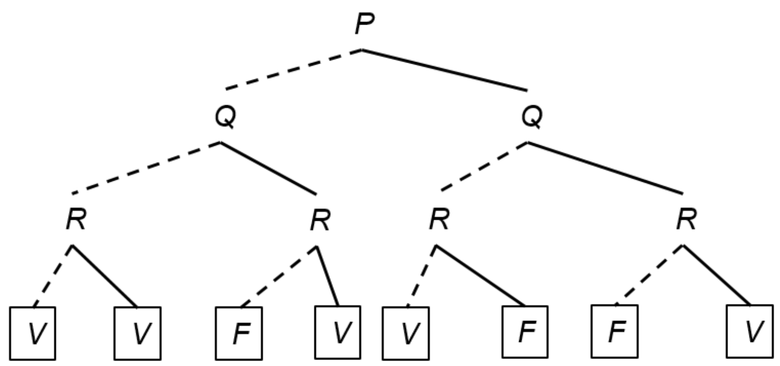
$$\Delta = \{P(c), P(a), \neg P(b), \forall x, y[S(x, y) \leftrightarrow x = a]\}.$$

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

Resposta:

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

7. (3.0) Considere o seguinte BDD:

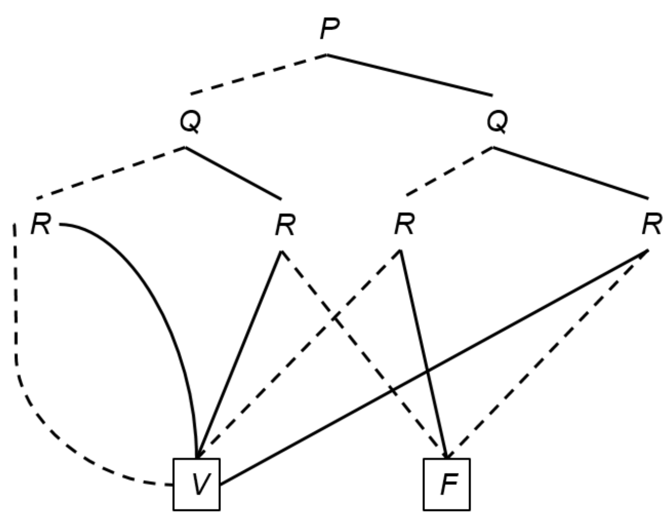


(a) (2.0) Obtenha o BDD reduzido correspondente, por aplicação das transformações aplicáveis em BDDs:

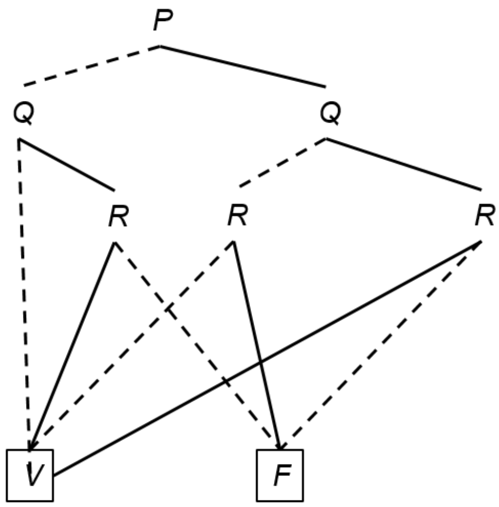
- R1 Remoção de folhas duplicadas.
- R2 Remoção de testes redundantes.
- R3 Remoção de nós redundantes.

Resposta:

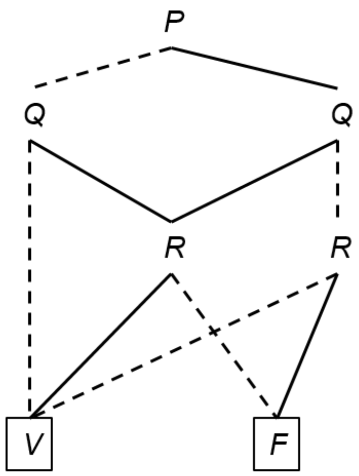
R1:



R2:



R3:

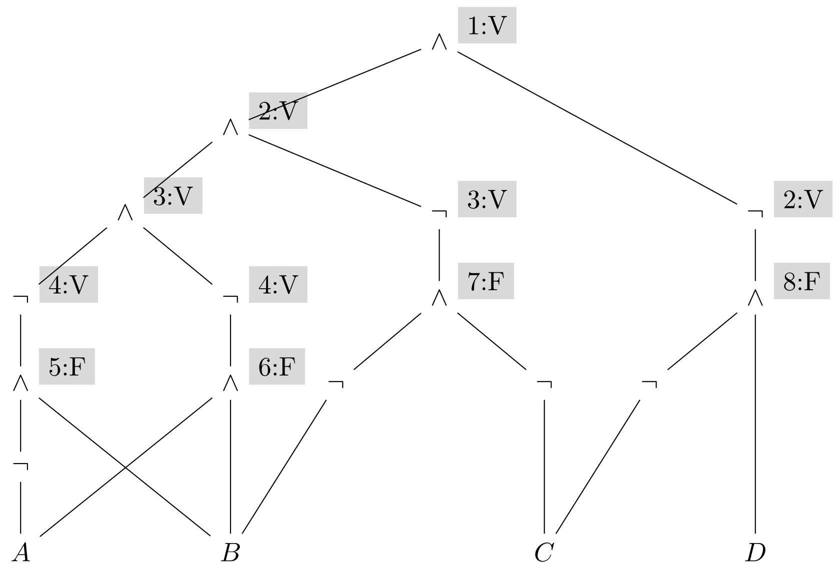


(b) (1.0) Quais as interpretações que **não** satisfazem a *fbf* representada pelo BDD?

Resposta:

$I(P) = V, I(Q) = F, I(R) = V$
$I(P) = V, I(Q) = V, I(R) = F$
$I(P) = F, I(Q) = V, I(R) = F$

8. (2.0) Considere o seguinte DAG ao qual foi aplicado o algoritmo de propagação de marcas.



(a) (1.0) Introduza na tabela que se segue as marcas propagadas como resultado da aplicação do algoritmo de teste de nós. Por exemplo, a primeira linha representa o cenário em que a marca F é temporariamente atribuída ao nó com rótulo C. (Se existirem nós sem marcas coloque 'X' na posição respectiva.)

A	B	C	D
		F	
V			
F			

Resposta:

A	B	C	D
V/F	V	F	F
V	F	V	X
F	F	V	X

(b) (0.5) Analisando o conteúdo da primeira linha da tabela anterior, o que pode concluir quanto ao nó com rótulo C? Justifique.

Resposta:

Como é encontrada uma contradição após a atribuição da marca F ao nó com rótulo C, podemos concluir que o nó com rótulo C tem de ser marcado com V.

(c) (0.5) Analisando o conteúdo das duas últimas linhas da tabela anterior, o que pode concluir quanto aos nós com rótulos B e C? Justifique.

Resposta:

Comparando as marcas obtidas nos dois testes do nó com rótulo A, podemos passar a permanentes as marcas temporárias comuns aos dois testes, ou seja, a marca F para o nó com rótulo B e a marca V para o nó com rótulo C.

9. (2.0) Considere as seguintes cláusulas em Prolog:

C1: `remRep([], []).`
C2: `remRep(L, L).`
C3: `remRep(L, []).`

C4: `remRep(L, _).`
C5: `remRep([ H | T], L) :- member(H, T), remRep(T, L).`
C6: `remRep([ H | T], L) :- !, member(H, T), remRep(T, L).`
C7: `remRep([ H | T], L) :- member(H, T), !, remRep(T, L).`
C8: `remRep([ H | T], L) :- member(H, T), remRep(T, L), !.`
C9: `remRep([ H | T], L) :- not(member(H, T)), remRep(T, L), !.`
C10: `remRep([ H | T], [ H | L]) :- remRep(T, L).`

(a) (1.0) Suponha que se quer o seguinte comportamento para `remRep(L1, L2)`: `L2` é a lista que resulta de `L1` tendo sido eliminados de `L1` os elementos repetidos. Por exemplo, queremos que se verifique:

```
?- remRep([1, 1, a, a, b, a, c], X).
X = [1, b, a, c].
```

Das cláusulas dadas, escolha e indique três para constituir um programa que implemente o predicado com o comportamento desejado.

Resposta:

C1, C7, C10 ou C1, C8, C10

(b) (0.5) Considerando um programa constituído pelas cláusulas C1, C6, C7 e C10, qual o resultado de ? - `remRep([1, 1, a, a, b, a, c], L)` (suponha que vão sendo pedidas respostas, enquanto for possível).

Resposta:

false

(c) (0.5) Considerando um programa constituído pelas cláusulas C1, C9 e C10, qual o resultado de ? - `remRep([1, 1, a, a, b, a, c], L)` (suponha que vão sendo pedidas respostas, enquanto for possível).

Resposta:

X = [1, a, a].

10. (1.5) Implemente o predicado

```
alcunha(ListaHerois, Nome, Alcunha, ListaAlcunhaHerois)
```

em que `ListaAlcunhaHerois` é a lista obtida substituindo TODAS as ocorrências do nome `Nome` por `Alcunha` na lista `ListaHerois` (por exemplo, verifica-se `alcunha([capitaoAmerica, homemAranha, hulk, homemAranha], homemAranha, webHead, [capitaoAmerica, webHead, hulk, webHead])`). Use o corte de modo a optimizar a execução do seu programa.

Resposta:

```
alcunha([], _, _, []).
alcunha([H | T], N1, N2, [N2 | L]) :-
            H = N1, !, alcunha(T, N1, N2, L).
alcunha([H | T], N1, N2, [H | L]) :-
            H \= N1, alcunha(T, N1, N2, L).
```

11. (a) (1.5) No contexto do projecto, implemente o predicado `retira_pares_posicao/3`, tal que `retira_pares_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 o resultado de retirar os elementos pares do conteúdo original. Por exemplo, se o conteúdo original for `[1,2,5]`, o novo conteúdo será `[1,5]`.

    Sugestão: utilize os predicados `puzzle_ref(Puz,Pos,Cont)`, `puzzle_muda(Puz,Pos,Cont,N_Puz)` e `exclude(Predicado,Lst1,Lst2)`.

    Resposta:

```
retira_pares_posicao(Puz,Pos,N_Puz) :-
    puzzle_ref(Puz,Pos,Cont),
    exclude(par,Cont,Cont_sem_pares),
    puzzle_muda(Puz,Pos,Cont_sem_pares,N_Puz).

par(X) :- X mod 2 =:= 0.
```

(b) (1.0) Usando o predicado definido na alínea anterior, implemente o predicado `retira_pares_puzzle/2`, tal que `retira_pares_puzzle(Puz, N_Puz)` significa que `N_Puz` é o puzzle resultante de aplicar o predicado `retira_pares_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

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

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

Resposta:

```
retira_pares_puzzle(Puz,N_Puz) :-
    todas_posicoes(Todas),
    percorre_muda_Puz(Puz,retira_pares_posicao,Todas,N_Puz).
```