TÉCNICO LISBOA

Lógica para Programação

Exame de 2ª Época

12 de Julho de 2021

9:00–11:00

Nome: _____ Número: _____

- Esta prova, individual e sem consulta, tem **9** páginas com **11** perguntas. A cotação de cada pergunta está assinalada entre parêntesis.
- Escreva o seu número em todas as folhas da prova. O tamanho das respostas deve ser limitado ao espaço fornecido para cada questão. O corpo docente reserva-se o direito de não considerar a parte das respostas que excedam o espaço indicado.
- Pode responder utilizando lápis.
- Em cima da mesa devem apenas estar o enunciado, caneta ou lápis e borracha e cartão de aluno. Não é permitida a utilização de folhas de rascunho, telemóveis, calculadoras, etc.
- Boa sorte!

Pergunta	Cotação	Nota
1.	1.0	
2.	1.0	
3.	1.5	
4.	2.0	
5.	2.0	
6.	1.5	
7.	3.5	
8.	2.0	
9.	1.5	
10.	1.5	
11.	2.5	
Total	20.0	

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 Programação em Lógica é
   A. um conjunto de regras e objectivos.
   B. uma sequência de afirmações e regras.
   C. uma sequência afirmações e objectivos.
   D. um conjunto de afirmações e regras.

   Resposta: ___

   (b) Uma regra de procura
   A. recebe um programa e um objectivo e devolve uma cláusula determinada.
   B. recebe um programa e devolve um objectivo.
   C. recebe um objectivo e devolve um dos seus sub-objectivos.

   Resposta: ___

2. (1.0) Para cada uma das seguintes afirmações, indique a única resposta certa. Cada resposta correcta vale 0.5 valores e *cada resposta errada desconta 0.1 valores*. Considere os seguintes predicados/funções/constantes:

   $serieFavorita(x)$ é uma função que devolve a série favorita de x
   $Estudante(x)$ é um predicado que é verdade se x for um estudante
   $GostaDe(x, y)$ é um predicado que é verdade se x gosta de y
   Ana é uma constante
   $Passa(x, y)$ é um predicado que é verdade se x passa no canal y

   Indique a fórmula em Lógica de Primeira Ordem que melhor traduz as seguintes frases em Língua Natural:

   (a) *Todos os estudantes gostam da sua série favorita.*
   A: $\forall x[Estudante(x) \wedge GostaDe(x, serieFavorita(x))]$
   B: $\exists x[Estudante(x) \wedge GostaDe(x, serieFavorita(x))]$
   C: $\forall x[Estudante(x) \rightarrow GostaDe(x, serieFavorita(x))]$
   D: $\exists x[Estudante(x) \rightarrow GostaDe(x, serieFavorita(x))]$

   Resposta: ___

   (b) *A série favorita da Ana não passa na Netflix.*
   A: $\neg(Passa(x, Netflix) \wedge serieFavorita(Ana, x))$
   B: $\neg(Passa(serieFavorita(Ana), Netflix))$
   C: $\neg\exists x[passa(x, Netflix) \wedge serieFavorita(Ana, x)]$
   D: $\neg\exists x[passa(x, Netflix) \rightarrow serieFavorita(Ana, x)]$

   Resposta: ___

3. (1.5) Complete a seguinte tabela calculando o unificador mais geral (UMG) entre as fórmulas $P(a, f(f(y)), f(b))$ e $P(z, f(f(a)), f(x))$. Considere que P é um predicado, x, y e z são variáveis, a e b são constantes e f é uma função.

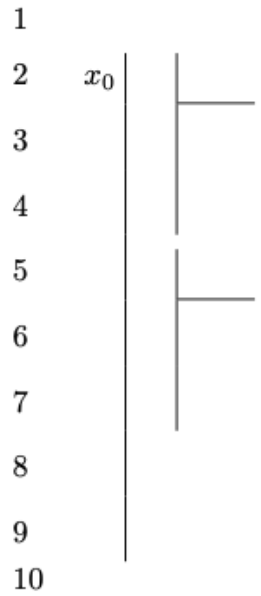
Conjunto de fórmulas bem formadas	Conjunto de desacordo	Substituição

UMG =
Preencha a tabela acima e indique explicitamente o UMG (caso exista) entre as fórmulas dadas.

4. (2.0) Demonstre o seguinte argumento, usando as regras de inferência 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):

$$\{\forall x[P(x)] \vee \forall x[Q(x)]\} \vdash \forall x[(P(x) \vee Q(x))]$$

Para tal, preencha o seguinte esquema de prova (pode usar outro esquema, se assim o entender):



5. (2.0) Preencha a seguinte tabela, tendo em conta a fórmula dada e a etapa da conversão para a forma clausal pedida.

Fórmula Original	Eliminação de	Fórmula resultante
$\exists z[A(z)] \wedge \forall x, y[\neg B(x) \vee \exists w C(y, x, w)]$	$\exists$	
$(P \rightarrow \neg Q) \wedge (P \rightarrow R) \wedge (P \rightarrow S)$	$\rightarrow$	
$(\neg P \vee \neg Q) \wedge (\neg P \vee R) \wedge (\neg P \vee S)$	$\wedge$	
$\forall z[A(z)] \wedge \exists x, y[\neg B(x) \vee \exists w C(y, x, w)]$	$\exists$	

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:

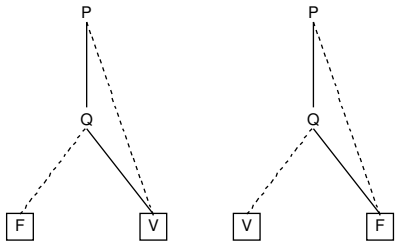
$I(a) = \diamond$
$I(b) = \square$
$I(c) = \odot$

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

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

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

7. (3.5) Considere as *fbfs* α e β, cujos OBDDs reduzidos são respectivamente



Utilize o algoritmo $aplica$, para determinar o OBDD reduzido da *fbf* $\alpha \rightarrow \beta$.

8. (1.0 + 0.5 + 0.5) Considera a fórmula $\{\{A, B, \neg C\}, \{\neg A, D\}, \{\neg B\}, \{C, D\}\}$.

   (a) Considere o algoritmo DP e a fórmula dada. Faça a distribuição por baldes e o seu processamento considerando a seguinte ordem:

   b_A:
   b_B:
   b_C:
   b_D:

   (b) (0.5) Das seguintes frases indique a única que **NÃO** é verdade:

   A: é possível obter uma testemunha em que D é verdadeiro e B é falso
   B: é possível obter uma testemunha em que D é verdadeiro e B é verdadeiro
   C: é possível obter uma testemunha em que D é verdadeiro e C é falso
   D: é possível obter uma testemunha em que D é verdadeiro e C é verdadeiro

   Resposta: ___

   (c) (0.5) Indique qual das seguintes interpretações representa uma testemunha:

   A: I(D) = F, I(C) = F, I(B) = F, I(A) = F
   B: I(D) = F, I(C) = F, I(B) = V, I(A) = V
   C: I(D) = V, I(C) = F, I(B) = F, I(A) = F
   D: I(D) = V, I(C) = V, I(B) = F, I(A) = F

   Resposta: ___

9. (1.5) Assuma definido o predicado `primo(N)` que é verdade se `N` for um número primo. Considere ainda o predicado `soPrimos(L1, L2)` que é verdade se `L1` for uma lista com inteiros e `L2` a lista obtida a partir de `L1` por eliminação dos inteiros não primos. No caso de não existir nenhum primo em `L1`, `L2` será a lista vazia. Por exemplo:

```
?- L1 = [1, 2, 8, 5, 4, 12], soPrimos(L1, L2).
L2 = [2, 5].
```

   (a) (0.75) Implemente o predicado `soPrimos(L1, L2)` recorrendo ao predicado `findall` (ou `bagof`, sem esquecer as suas especificidades). Sugestão: use o predicado `primo/2`.

   (b) (0.75) Implemente o predicado `soPrimos(L1, L2)` recorrendo ao predicado `include` (ou `exclude`). Sugestão: use o predicado `primo/2`.

10. (0.5 + 0.5 + 0.5) Considere o seguinte programa em Prolog:

C_1: `le(X, Y) :- estudante(X), livro(Y).`
C_2: `estudante(maria).`
C_3: `estudante(joao) :- fail.`
C_4: `estudante(ana).`
C_5: `livro(hungerGames).`

Supondo que vão sempre ser pedidas mais respostas enquanto tal for possível, qual a resposta do Prolog:

(a) ... ao objectivo `?- le(X, Y).`, considerando o programa anterior.

(b) ... ao objectivo `?- le(X, Y).`, considerando que C_3 é agora:
`estudante(joao) :- !, fail.`

(c) ... ao objectivo `?- le(ana, Y).`

11. (a) (1.5) Implemente o predicado `posicoes_diferentes/3`, tal que `posicoes_diferentes(L1, L2, Posicoes)`, em que todos os argumentos são listas, significa que `Posicoes` é a lista de posições das listas `L1` e `L2` cujos conteúdos são diferentes. Por exemplo,

```
?- posicoes_diferentes([a,b,c], [x,b,z], Posicoes).
Posicoes = [1, 3].

?- posicoes_diferentes([a,b,c,8], [x,b,z], Posicoes).
Posicoes = [1, 3].

?- posicoes_diferentes([a,b,c], [x,b,z,8], Posicoes).
Posicoes = [1, 3].

?- posicoes_diferentes([], [x,b,z,8], Posicoes).
Posicoes = [].

?- posicoes_diferentes([A,b,c], [X,b,z], Posicoes).
Posicoes = [1, 3].
```

Pode, ou não, usar meta-predicados. Fica ao seu critério.

(b) (1.0) Usando o predicado definido na alínea anterior, implemente o predicado `semelhantes/2`, tal que `semelhantes(L1, L2)`, em que `L1` e `L2` são duas listas com o mesmo comprimento, significa que o número de posições diferentes das duas listas é menor ou igual a 20% do comprimento das listas. Por exemplo,

```
?- semelhantes([a,b,c,d,e], [a,b,c,d,x]).
true.

?- semelhantes([a,b,c,d,e], [y,b,c,d,x]).
false.

?- semelhantes([], []).
true.
```