

Cartão: 170622

Nome: Maurício Leite Dau

Primeira Verificação de Aproveitamento (Prova #1) (61/68)

A prova é individual e sem consulta. Leia atentamente toda a prova e, após, responda as questões. Se tiver alguma dúvida, você pode consultar o professor nos primeiros 15 minutos. Decorridos os primeiros 15 minutos, a interpretação das questões passa a fazer parte da prova. Faça as suposições necessárias e explique-as como parte da resolução da questão. É permitido resolver a prova utilizando lápis, mas provas feitas a lápis não poderão sofrer revisão da correção. Aparelhos eletrônicos não podem ser utilizados durante a prova.

– Questões –

- 1.** *[2.5 pontos, 20 minutos]* Os dois principais paradigmas de computação são o imperativo (linguagens procedurais e orientadas a objetos) e o declarativo (linguagens funcionais e lógicas). Na abordagem imperativa, dado um determinado problema, o foco do programa é o que o computador deve fazer para se atingir uma solução. Na declarativa, o foco do programa é em como o computador deveria fazer para resolver o problema. Nesta questão, vamos considerar a **Sequência Lucas** de números positivos, definida por:

$$L_n = \begin{cases} 2 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ L_{n-1} + L_{n-2} & \text{if } n > 1. \end{cases}$$

A tabela abaixo mostra o início da sequência de números Lucas. A linha “Posição” indica qual a posição do número Lucas na sequência, começando por zero. A linha “Número Lucas” indica o número Lucas na posição correspondente.

Posição	0	1	2	3	4	5	6	7	8	9	10	...
Número Lucas	2	1	3	4	7	11	18	29	47	76	123	...

Com o objetivo de constrastar esses diferentes paradigmas de computação, escreva uma versão imperativa, uma funcional e uma versão em linguagem lógica de um programa que calcule o número Lucas correspondente a uma determinada posição. Por exemplo, `lucas(5)` deve retornar 11 e `lucas(10)` deve retornar 123.

Não se preocupe muito com a sintaxe, mas sim em utilizar construções que possibilitem salientar as características e especificidades de cada paradigma. Com base nos programas implementados, discorra sobre as semelhanças e diferenças entre fundamentos, formalismos e capacidades de abstração de cada paradigma.

- 2.** *[1.5 pontos, 10 minutos]* Uma linguagem de programação qualquer é capaz de expressar qualquer programa de computador? Justifique a sua resposta para classes de problemas diferentes como, por exemplo, multiplicação de matrizes, sistemas operacionais e geração de gráficos estatísticos. Relacione a sua justificativa com os critérios de legibilidade, redigibilidade e confiabilidade de linguagens de programação.

- 3.** *[1.5 pontos, 15 minutos]* O cálculo Lambda (λ) é a base da programação funcional, onde as funções com mais de um parâmetro são transformadas em uma série de funções, cada uma recebendo no máximo um parâmetro. Duas operações são fundamentais no processo de resolução do cálculo Lambda: Redução- β e Redução- α . Qual o objetivo de cada uma dessas operações? Na sua resposta, defina e utilize os conceitos de variáveis ligadas e variáveis livres.

4. [2 pontos, 15 minutos] Reduza ao máximo as seguintes expressões Lambda utilizando as operações de Redução- β e Redução- α do Cálculo Lambda. A resposta para cada expressão deve ter todos os passos de redução. Indique, em cada um desses passos de redução, qual das operações é aplicada e sobre qual variável esta operação está sendo feita.

1. $(\lambda z \cdot z)(\lambda y \cdot y y)(\lambda x \cdot x a)$
2. $(\lambda z \cdot z)(\lambda z \cdot z z)(\lambda z \cdot z y)$
3. $(\lambda x \cdot (\lambda y \cdot (x y)) y) z$
4. $((\lambda x \cdot \lambda y \cdot (x y))(\lambda y \cdot y)) w$
5. $(\lambda f. \lambda g. \lambda h. fg(h h))(\lambda x. \lambda y. x)h(\lambda x. x x)$

5. [1 pontos, 10 minutos] Considerando a base de conhecimento na linguagem lógica Prolog, listada abaixo, mostre os passos do mecanismo de resolução do motor de inferência para a consulta `capital_estado (X, rs)`.

```

1 cidade(floripa).
2 cidade(portoalegre).
3 estado(sc).
4 estado(rs).
5 capital(portoalegre, rs).
6 capital(floripa, sc).
7 capital_estado(X,Y) :- estado(Y), cidade(X), capital(X, Y).
```

6. [0.5 pontos, 5 minutos] Qual o significado do operador de corte, simbolizado pelo símbolo de exclamação `!`, da linguagem de programação Prolog? Qual o seu impacto no processo de resolução do motor de inferência?

7. [1 pontos, 10 minutos] Qual seria o resultado da consulta `capital_estado (X, rs)`. se a linha 7 da base de conhecimento da questão 5. fosse a regra de inferência listada abaixo? Explique a razão do resultado detalhando os passos do mecanismo de resolução do Prolog, justificando dessa forma a sua resposta.

```
1 capital_estado(X,Y) :- estado(Y), cidade(X), !, capital(X, Y).
```

1) LINGUAGEM ~~FUNCIONAL~~^{FUNCIONAL} (SINTAXE PRÓXIMA A ML)

FUN LUCAS $\lambda Q = 2 \mid$

LUCAS $\mid = 1 \mid$

LUCAS $N = \text{LUCAS}(N-1) + \text{WCAS}(N-2);$

C_{0,5}

LINGUAGEM LÓGICA (SINTAXE PRÓXIMA A PROLOG)

~~WCAS(N-1), WCAS(N-2).~~ LUCAS(0,2).

~~WCAS(N-1), WCAS(N-2).~~ WCAS(1,1).

~~WCAS(N-1), WCAS(N-2).~~ LUCAS(N,X) :- ~~WCAS(N-1,P), WCAS(N-2,Q), X is P+Q,~~

LUCAS(N,X) :- ~~WCAS(N-1,P), WCAS(N-2,Q), X is P+Q,~~

C_{0,3}

IMPERATIVA (C-LIKE)

```
INT LUCAS(INT N) {
    IF (N == 0)
        RETURN 2;
    IF (N == 1)
        RETURN 1;
    RETURN (WCAS(N-1) + WCAS(N-2));
}
```

C_{0,5}

NA VERSÃO IMPERATIVA FICA CLARO QUE O PROGRAMADOR "EXPLICOU" AO COMPUTADOR COMO FAZER O CÁLCULO DO NÚMERO WCAS, ENQUANTO QUE NAS OUTRAS DUAS FOI DECLARADO COMO O RESULTADO SE PARECE, OU SEJA, EM QUais REGRAS O RESULTADO DEVE SE ENCAIXAR. PODEMOS PERCEBER QUE NA IMPLEMENTAÇÃO IMPERATIVA O TIPO DO DADO OPERADO É MUITO IMPORTANTE, TANTO QUE APARECE COMO PRIMEIRO ELEMENTO DO CÓDIGO, ENQUANTO QUE NAS OUTRAS DUAS A TIPIAGEM NÃO É TÃO VISÍVEL, MESMO A FUNCIONAL SENDO FORTEMENTE TIPIADA.

O ALGORITMO IMPLEMENTADO É UMA REPRESENTAÇÃO DE UMA FUNÇÃO MATEMÁTICA, O QUE FOI TRANSMITIDO PI FUNCIONAL TRANSPARENTEMENTE, ENQUANTO QUE NA LINGUAGEM LÓGICA EXISTE UM POUCO MAIS DE DIFICULDADE DE LIDAR COM O "ARMazenamento" DOS NÚMEROS, MAS QUE A IMPLEMENTAÇÃO DE REGRAS MATEMÁTICAS É MUITO SIMPLES, FICANDO BEM SEMELHANTE A FUNCIONAL.

A CAPACIDADE DE ABSTRAÇÃO DA IMPERATIVA É Mais REDUZIDA, CONTUDO PODEMOS INFLUENCiar E DECIDIR SOBRE A MANEIRA COMO O CÁLCULO SERÁ REALIZADO.

2,1

2/4

3) As reduções β tem como objetivo a aplicação de valores ou regras numa função; ou seja, é a aplicação dos parâmetros de uma função nela mesma. Exemplo:

$$(\lambda x. x+x) 2 \quad \text{C}$$

Neste caso temos a função $\lambda x. x+x$ que recebe o valor 2 p/ ser aplicado na variável x , que aparece duas vezes na função e, com isso, está ligada nessa função. A aplicação da redução β produz o seguinte efeito:

$$\begin{array}{l} \lambda x. x+x \rightarrow 2+2 \rightarrow 4 \\ x=2 \end{array} \quad \text{C}$$

Poderemos aplicar reduções β com funções p/ funções, exemplo: C

$$((\lambda x. x+x)(\lambda y. yy))$$

Neste caso vamos aplicar na variável x da 1ª função o valor $\lambda y. yy$. Olhando p/ a 1ª função percebemos que não há referência a variável y , neste caso a variável y está desligada na primeira função. Assim, podemos aplicar a redução β nessa função, resultando em:

$$(\lambda y. yy) + (\lambda y. yy) \quad \text{C}$$

As reduções α tem como objetivo resolver a situação de quando queremos aplicar uma redução β à uma função com as variáveis da redução ligadas, como no caso:

$$((\lambda x. xy)y) : \text{A variável } y \text{ está ligada na função.} \quad \text{C}$$

Sendo assim, aplicamos a redução α, que funciona como uma renomeação, para habilitarmos a redução β

$$(\lambda x. xy)y : \alpha$$

$$(\lambda x. x\alpha)y : \beta$$

\boxed{ya}

1,5

AS

4)

$$\textcircled{1} \quad (\lambda z \cdot z)(\lambda y \cdot yy)(\lambda x \cdot xa) : \beta y + \lambda x \cdot xa$$

$$(\lambda z \cdot z)(\lambda x \cdot xa)(\lambda x \cdot xa) : \alpha x + b$$

$$(\lambda z \cdot z)(\lambda b \cdot ba)(\lambda x \cdot xa) : \beta b + \lambda x \cdot xa$$

$$(\lambda z \cdot z)(\lambda x \cdot xaa) : \beta z + \lambda x \cdot xaa$$

$\lambda x \cdot xaa$ aa

$$\textcircled{2} \quad (\lambda z \cdot z)(\lambda z \cdot zz)(\lambda z \cdot zy) : \alpha z + a$$

$$(\lambda z \cdot z)(\lambda a \cdot aa)(\lambda z \cdot zy) : \beta a + \lambda z \cdot zy$$

$$(\lambda z \cdot z)(\lambda z \cdot zy)(\lambda z \cdot zy) : \alpha z + b$$

$$(\lambda z \cdot z)(\lambda b \cdot by)(\lambda z \cdot zy) : \beta z + \lambda z \cdot zy$$

$$(\lambda z \cdot z)(\lambda z \cdot zyy) : \alpha z + c$$

$$(\lambda c \cdot c)(\lambda z \cdot zyy) : \beta c + \lambda z \cdot zyy$$

$\lambda z \cdot zyy$ yy

$$\textcircled{3} \quad (\lambda x \cdot (\lambda y \cdot (xy))y)z : \beta x + z \quad \alpha \quad \beta$$

$$(\lambda y \cdot (zy))y : \alpha y + a \quad \alpha \quad a$$

$$(\lambda a \cdot (za))y : \beta a + y \quad \beta \quad a$$

(zy) c

$$\textcircled{4} \quad (((\lambda x \cdot \lambda y \cdot (xy))(\lambda y \cdot y))w) : \alpha y + a$$

$$(((\lambda x \cdot \lambda a \cdot (xa))(\lambda y \cdot y))w) : \beta x + \lambda y \cdot y$$

$$((\lambda a \cdot (\lambda y \cdot ya))w) : \beta a + w$$

$\lambda y \cdot yw$ w

CONTINUAÇÃO DA 4)

$$\textcircled{5} \quad (\lambda f. \lambda g. \lambda h. f g (h h)) (\lambda x. \lambda y. x) h (\lambda x. x x) \quad \cancel{B \vdash f + (\lambda x. \lambda y. x) h (\lambda x. x x)} \\ \cancel{\alpha h = a}$$

 ~~$\lambda x. \lambda y. x$~~

$$(\lambda f. \lambda g. \lambda a. f g (a a)) (\lambda x. \lambda y. x) h (\lambda x. x x) \quad \cancel{B \vdash f + (\lambda x. \lambda y. x) h (\lambda x. x x)}$$

$$\lambda g. \lambda a. (\lambda x. \lambda y. x) h (\lambda x. x x) g (a a)$$

 h

(0,4)

QUESTÃO

 $\textcircled{5}$ CAPITAL-ESTADO (x, RS) $y = \text{RS}$ $\textcircled{2}$

ESTADO (RS)

TRUE

FAVOR

TRUE

ESTADO (SC)

TRUE

ESTADO (RS)

TRUE

 $\text{CIDADE}(x)$ $\textcircled{4}$

CIDADE (FURNAS)

TRUE

CIDADE (PORTO ALEGRE)

TRUE

 $\textcircled{6}$

CAPITAL (FURNAS, RS)

FALSE

CAPITAL (PORTO ALEGRE, RS)

TRUE

CAPITAL (PORTO ALEGRE, RS)

TRUE

 $\textcircled{3} \quad x = \text{PORTO ALEGRE}; \text{false.}$ ~~nao ja esta unificado~~

INICIALMENTE O MOTOR PROCUROU TODOS OS ESTADOS QUE CASASSEM COM RS, ACABOU E CONTINUOU POR PROCURAR TODAS AS CIDADES, DE POSSE DESSES DADOS, PROCUROU CASAR OS VALORES DE x COM OS DE y DE NOVO A ACHAR RAMOS DA ÁRVORE COM VALORES TRUE.

No ~~PASSO~~ $\textcircled{5}$ É DA TRUE COM A PRÓXIMA CIDADE(x), MAS QUANDO REALIZA O CASAMENTO EM CAPITAL(x, y) ESSE RAMO FICOU COM VALOR FALSE.

(0,8)

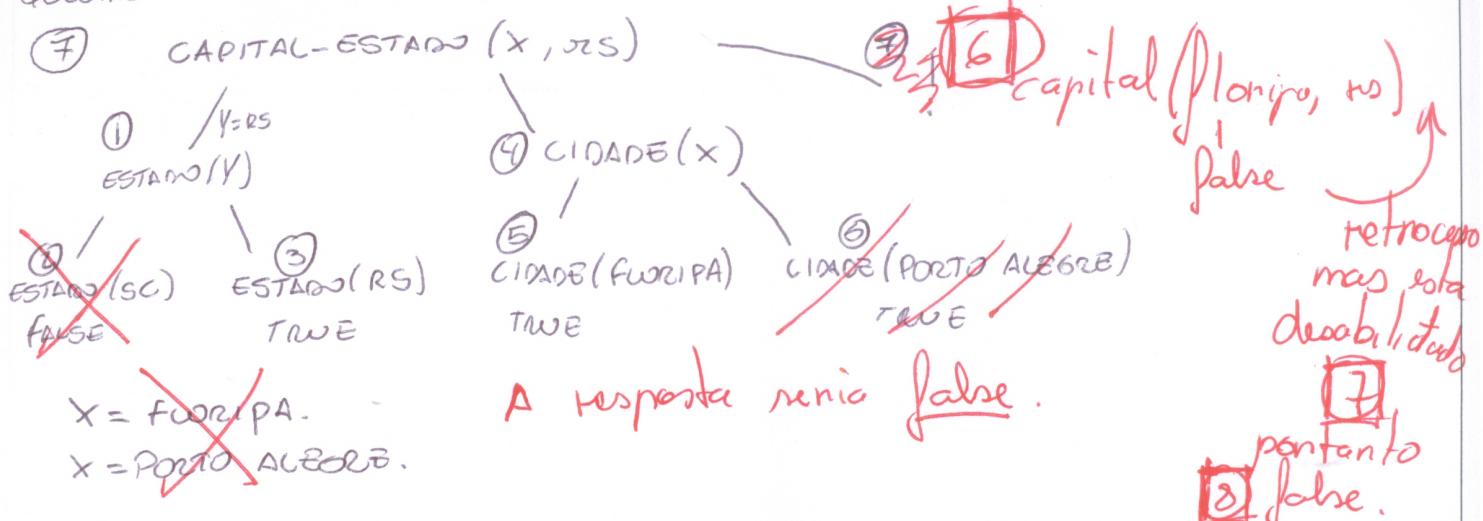
QUESTÃO

$\textcircled{6}$ O MECANISMO DE RESOLUÇÃO DO MOTOR DE INFERNÉCIA ~~REALIZA~~ REALIZA UMA PESQUISA EM ÁRVORE E, QUANDO ENCONTRA O OPERADOR DE CORTE, NÃO PROCURA MAIS RESULTADOS NAQUELE RAMO DA ÁRVORE, OU SEJA, O OPERADOR !CORTA/PARA A PESQUISA, NUMA REGIÃO DA ÁRVORE.

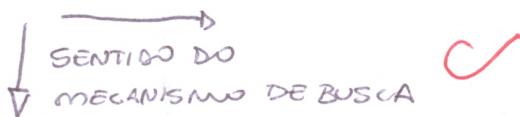
O USO DESSE OPERADOR RESULTA NUM MAIOR DESEMPENHO DO MOTOR DE BUSCA, POIS EVITA DE ANALISAR RAMOS DESNECESSÁRIOS DA ÁRVORE.

(0,5)

QUESTÃO



O MECANISMO DE INFERNÊNCIA DO PROLOG REALIZA A ANÁLISE NA ÁRVORE DA ESQUERDA P/ A DIREITA E DE CIMA P/ BAIXO:



Com isso ele realiza os passos de ① à ⑥ tentando casar dois valores p/ X e, em seguida, corta o resto da árvore e, consequentemente, o resto da análise é devolvo os valores casados até o momento.

QUESTÃO

2 Sim, se considerarmos os frameworks e máquinas virtuais que a linguagem possa depender como fazendo parte da mesma. Sejam assim ~~temos~~ linguagens com diferentes características, que as tornam mais úteis dependendo da classe de problemas a resolver. Podemos resolver problemas de utilizar mais linhas, em outras mais bibliotecas. No caso de problemas ligados a sistemas operacionais procuramos mais confiabilidade por parte das linguagens, por se tratar de uma área crítica e até transformar a legibilidade e reutilizabilidade por mais opções de controle de memória e processos, como no caso do C para SOs.

Contudo, em aplicações científicas estamos mais interessados em provar teorias ou analisar comportamentos, assim podemos abrir mão de vantagens para termos mais reutilizabilidade e legibilidade, como no caso das linguagens funcionais com problemas científicos.

Assim, ~~passando~~ ~~uma~~ uma linguagem de prog. qualquer é capaz de expressar qualquer programa de computador, mas as condições de legibilidade, reutilizabilidade e confiabilidade vão se alterar bruscamente de acordo com a classe de problemas e a linguagem utilizada.