

**Métodos Formais de Programação II +  
Opção I - Métodos Formais de Programação II**

4.º Ano da LMCC (7008N2) + LESI (5308P3)  
Ano Lectivo de 2000/2001

Exame (época de Recurso) — 21 de Setembro 2001  
09h30  
Sala 1301

**NB:** Esta prova consta de **10** alíneas que valem, cada uma, 2 valores. Para sua consulta, encontra anexa a esta prova a listagem de algumas leis de cálculo estudadas na disciplina.

PROVA SEM CONSULTA (3 horas)

**Questão 1** Uma das operações conhecidas sobre listas é a inversão:

```
invl[@A] : seq of @A -> seq of @A
invl(l) == if l = [] then l else invl[@A](tl l) ^ [hd l] ;
```

1. Calcule a definição de *invl*, dada acima em VDM-SL, a partir do seguinte catamorfismo:

$$invl \stackrel{\text{def}}{=} \underbrace{([[], ^ \cdot swap \cdot (singl \times id)]]}_g \quad (1)$$

2. Converta para notação com variáveis a propriedade

$$invl \cdot ^ = ^ \cdot (invl \times invl) \cdot swap \quad (2)$$

e complete as igualdades seguintes por forma a exprimirem também propriedades válidas:

$$invl \cdot singl = \dots \quad (3)$$

$$^ \cdot (\dots) = cons \quad (4)$$

3. Complete as justificações da seguinte prova por **reflexão-cata** (32) da propriedade involutiva de *invl*:

$$\begin{aligned} & invl \cdot invl = id \\ \equiv & \quad \{ \text{reflexão-cata (32) e (1)} \} \\ & invl \cdot ([g]) = ([in]) \\ \Leftarrow & \quad \{ \dots \} \\ & invl \cdot g = in \cdot (id + id \times invl) \\ \equiv & \quad \{ \text{expansão de } g \text{ (1)} \} \\ & invl \cdot ([[], ^ \cdot swap \cdot (singl \times id)]) = in \cdot (id + id \times invl) \\ \equiv & \quad \{ \dots \} \\ & [invl \cdot [ ], invl \cdot ^ \cdot swap \cdot (singl \times id)] = in \cdot (id + id \times invl) \\ \equiv & \quad \{ \dots \} \\ & ([[], ^ \cdot (invl \times invl) \cdot swap \cdot swap \cdot (singl \times id)]) = in \cdot (id + id \times invl) \\ \equiv & \quad \{ \dots \} \\ & ([[], ^ \cdot (invl \cdot singl \times invl)]) = in \cdot (id + id \times invl) \\ \equiv & \quad \{ \dots \} \\ & ([[], ^ \cdot (singl \times invl)]) = in \cdot (id + id \times invl) \\ \equiv & \quad \{ \dots \} \\ & ([[], cons \cdot (id \times invl)]) = in \cdot (id + id \times invl) \\ \equiv & \quad \{ \dots \} \\ & ([[], cons] \cdot (id + id \times invl)) = in \cdot (id + id \times invl) \\ \equiv & \quad \{ \dots \} \\ & V \end{aligned}$$

**Questão 2** O modelo de dados em VDM-SL que se apresenta a seguir estende o “sistema de contas bancárias” estudado nas aulas práticas com informação adicional, referente aos clientes do banco:

```
types

BAMS' :: bams: BAMS
      people: map AccHolder to AccHinfo;

BAMS = map AccId to Account;

Account :: B: Amount;
         H: set of AccHolder

AccHinfo :: name: seq of char
         address: seq of char;

AccId     = seq of char;
AccHolder = seq of char;
Amount    = int;
```

1. Escreva em sintaxe VDM-SL um invariante sobre BAMS' que garanta que *todas as pessoas registadas no sistema são clientes, isto é, têm pelo menos uma conta aberta*.
2. O modelo BAMS' é instância do esquema genérico de dados  $(A \rightarrow B \times \mathcal{P}C) \times (C \rightarrow D)$ , de onde se parte no diagrama

$$(A \rightarrow B \times \mathcal{P}C) \times (C \rightarrow D) \xrightarrow{f_1} (A \rightarrow \mathcal{P}C) \times \mathcal{P}C \xrightarrow{f_2} \mathcal{P}(\mathcal{P}C) \times \mathcal{P}C \xrightarrow{f_3} \mathcal{P}C \times \mathcal{P}C \xrightarrow{f_4} 2 \quad (5)$$

$\text{inv}$

para registar o processo de definição de um invariante  $(A \rightarrow B \times \mathcal{P}C) \times (C \rightarrow D) \xrightarrow{\text{inv}} 2$  cujo objectivo é garantir a integridade referencial entre as duas funções finitas em jogo no que diz respeito ao domínio de chaves  $C$ . Assim,  $\text{inv}(x, y)$  deverá garantir que todos os  $C$ s que ocorrem em  $x$  estão no domínio de  $y$ .

Identifique as funções  $f_1$  a  $f_4$ , justificando a sua resposta.

3. Suponha que BAMS' é submetido a um processo de migração de dados  $\text{BAMS}' \xrightarrow{m} \text{NEWBAMS}$  que injecta a sua informação num novo sistema NEWBAMS, de acordo com os passos  $m_1$  a  $m_3$  seguintes:

$$\begin{array}{c} (A \rightarrow B \times \mathcal{P}C) \times (C \rightarrow D) \\ \downarrow m_1 \\ (A \rightarrow B) \times (A \rightarrow \mathcal{P}C) \times (C \rightarrow D) \\ \downarrow m_2 = id \times f \times id \\ (A \rightarrow B) \times (C \rightarrow \mathcal{P}A) \times (C \rightarrow D) \\ \downarrow m_3 \\ (A \rightarrow B) \times (C \rightarrow (\mathcal{P}A \times D)) \end{array}$$

Exprima  $f$  com base nas funções *discollect*, *swap* e suas inversas.

4. Defina em VDM-SL o tipo NEWBAMS a partir de BAMS' e do resultado do processo de cálculo acima.
5. Mostre através de contra-exemplos que  $f$  e  $m_3$  não são isomorfismos, dando lugar à perda de informação. Identifique a partir daí quais os riscos que se correm na migração, indicando que dados é que se perdem (ou podem perder).

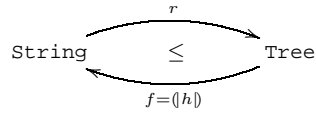
**Questão 3** Considere dois tipos de dados em VDM-SL: ‘strings’ não vazios,

```
String = seq of char
inv s == s <> [];
```

e árvores binárias de caracteres:

```
Tree = char | Node ;
Node :: left : Tree
      right : Tree;
```

Alguns algoritmos de processamento de ‘strings’ tornam-se mais eficientes se se usar *Tree* como representação de *String*:

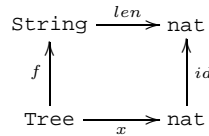


onde

$$h = [singl, ^] \quad (6)$$

Por exemplo, o ‘string’ “abc” pode ser representado tanto pela árvore  $\text{mk\_Node}(\text{"a"}, \text{mk\_Node}(\text{"b"}, \text{"c"}))$  como pela árvore  $\text{mk\_Node}(\text{mk\_Node}(\text{"a"}, \text{"b"}), \text{"c"})$ .

1. O diagrama



descreve o processo de refinamento da operação *len* quando *String* é representado por *Tree*, sendo *x* a função a calcular. Complete o seguinte processo de cálculo de *x*:

$$\begin{aligned} id \cdot x &= len \cdot f \\ \equiv & \quad \{ \text{definição de } x \text{ e de } f \text{ como catamorfismos (a incógnita passa a ser } \alpha) \} \\ id \cdot \langle \alpha \rangle &= len \cdot \langle h \rangle \\ \equiv & \quad \{ \dots \} \\ \langle \alpha \rangle &= len \cdot \langle h \rangle \\ \Leftarrow & \quad \{ \text{fusão-cata (33)} \} \\ & \dots \\ \equiv & \quad \{ \text{definição de } h \text{ e desdobramento de } \alpha \text{ num ‘either’} \} \\ & \dots \\ \equiv & \quad \{ \text{fusão-+ (19) e absorção-+ (20)} \} \\ & \dots \\ \equiv & \quad \{ \text{igualdade estrutural de ‘eithers’} \} \\ & \dots \\ \equiv & \quad \{ \dots \} \\ & \left\{ \begin{array}{l} \alpha_1 = \underline{1} \\ \alpha_2 = + \end{array} \right. \end{aligned}$$

2. Converta a função  $x = \langle [\underline{1}, +] \rangle$  calculada na alínea anterior para notação VDM-SL com variáveis, sem recurso a funções de ordem superior.

## Anexo

### COMPOSIÇÃO

$$\text{Natural-id} \quad f \cdot id = id \cdot f = f \quad (7)$$

$$\text{Associatividade} \quad (f \cdot g) \cdot h = f \cdot (g \cdot h) \quad (8)$$

### PRODUTO

$$\text{Universal-}\times \quad k = \langle f, g \rangle \Leftrightarrow \left\{ \begin{array}{l} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{array} \right. \quad (9)$$

$$\text{Cancelamento-}\times \quad \pi_1 \cdot \langle f, g \rangle = f \quad , \quad \pi_2 \cdot \langle f, g \rangle = g \quad (10)$$

$$\text{Reflexão-}\times \quad \langle \pi_1, \pi_2 \rangle = id_{A \times B} \quad (11)$$

$$\text{Fusão-}\times \quad \langle g, h \rangle \cdot f = \langle g \cdot f, h \cdot f \rangle \quad (12)$$

$$\text{Absorção-}\times \quad (i \times j) \cdot \langle g, h \rangle = \langle i \cdot g, j \cdot h \rangle \quad (13)$$

$$\text{Functor-}\times \quad (g \cdot h) \times (i \cdot j) = (g \times i) \cdot (h \times j) \quad (14)$$

$$\text{Functor-id-}\times \quad id_A \times id_B = id_{A \times B} \quad (15)$$

## COPRODUTO

$$\begin{aligned}
\text{Universal-+} \quad k = [f, g] &\Leftrightarrow \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases} & (16) \\
\text{Cancelamento-+} \quad [g, h] \cdot i_1 = g \quad , \quad [g, h] \cdot i_2 = h & & (17) \\
\text{Reflexão-+} \quad [i_1, i_2] &= id_{A+B} & (18) \\
\text{Fusão-+} \quad f \cdot [g, h] &= [f \cdot g, f \cdot h] & (19) \\
\text{Absorção-+} \quad [g, h] \cdot (i + j) &= [g \cdot i, h \cdot j] & (20) \\
\text{Functor-+} \quad (g \cdot h) + (i \cdot j) &= (g + i) \cdot (h + j) & (21) \\
\text{Functor-id-+} \quad id_A + id_B &= id_{A+B} & (22)
\end{aligned}$$

## EXPONENCIAÇÃO

$$\begin{aligned}
\text{Universal} \quad k = \overline{f} &\Leftrightarrow f = ap \cdot (k \times id) & (23) \\
\text{Cancelamento} \quad f &= ap \cdot (\overline{f} \times id) & (24) \\
\text{Reflexão} \quad \overline{ap} &= id_{B^A} & (25) \\
\text{Fusão} \quad \overline{g \cdot (f \times id)} &= \overline{g} \cdot f & (26) \\
\text{Absorção} \quad f^A \cdot \overline{g} &= \overline{f \cdot g} & (27) \\
\text{Functor} \quad (g \cdot h)^A &= g^A \cdot h^A & (28) \\
\text{Functor-id} \quad id^A &= id & (29)
\end{aligned}$$

## INDUÇÃO

$$\begin{aligned}
\text{Universal-cata} \quad k = \langle \alpha \rangle &\Leftrightarrow k \cdot in = \alpha \cdot (F k) & (30) \\
\text{Cancelamento-cata} \quad \langle \alpha \rangle \cdot in &= \alpha \cdot F \langle \alpha \rangle & (31) \\
\text{Reflexão-cata} \quad \langle in \rangle &= id_{\top} & (32) \\
\text{Fusão-cata} \quad f \cdot \langle \alpha \rangle = \langle \beta \rangle &\text{ if } f \cdot \alpha = \beta \cdot (F f) & (33) \\
\text{Absorção-cata} \quad \langle g \rangle \cdot \top f &= \langle g \cdot B(f, id) \rangle & (34)
\end{aligned}$$

## FUNCTORES

$$\begin{aligned}
\text{Functor-F} \quad F(g \cdot h) &= (F g) \cdot (F h) & (35) \\
\text{Functor-id-F} \quad F id_A &= id_{(F A)} & (36) \\
\text{“Teorema grátis” de } g & \quad (G f) \cdot g = g \cdot (F f) & (37)
\end{aligned}$$

## MISC.

$$\begin{aligned}
\text{Lei da troca} \quad [\langle f, g \rangle, \langle h, k \rangle] &= \langle [f, h], [g, k] \rangle & (38) \\
\text{Lei de “banana-split”} \quad \langle \langle i \rangle, \langle j \rangle \rangle &= \langle (i \times j) \cdot \langle F \pi_1, F \pi_2 \rangle \rangle & (39)
\end{aligned}$$