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

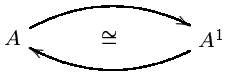
4.º Ano da LMCC (7008N2) + LESI (5308P3)
Ano Lectivo de 2003/04

Exame (época de recurso) — 21 de Julho de 2004
09h30
Sala 2204

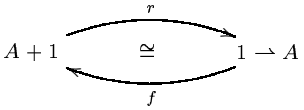
NB: *Esta prova consta de 7 alíneas todas com a mesma cotação.*

PROVA SEM CONSULTA (2 horas)

Questão 1 O isomorfismo



estende-se a funções parciais finitas ("mappings") da forma seguinte:



Quer dizer, qualquer "mapping" com no máximo um elemento no domínio "esconde" um apontador. Prove o isomorfismo e adivinhe f e r, escrevendo-as em notação VDM-SL.

Questão 2 Considere o seguinte fragmento de VDM-SL envolvendo duas *representações* s e r:

```
s(m) == if m={|->} then []
        else r(dom m) ^ s({ a |-> m(a)-1 | a in set dom m & m(a) > 1});
r(s) == if s = {} then []
        else let e in set s in [e] ^ r(s \ {e});
```

Identifique leis de *refinamento* de dados correspondentes a essas representações e especifique, em VDM-SL, as respectivas *abstracções*.

Questão 3 A informação constante dos *registos prediais* duma Conservatória de Registo Predial (CRP) reparte-se por várias séries de livros de acordo com os bens registados e as relações jurídicas entre eles. Só nos vamos aqui preocupar com as séries "B" e "C". Na primeira, cada entrada descreve um prédio, *e.g.* um edifício, um terreno, *etc.* No chamado *regime de propriedade horizontal* a informação de cada prédio consta sobretudo da descrição das suas fracções, hipotecas e prédio origem. A situação típica é a seguinte: um empreiteiro hipoteca um terreno (prédio origem) para poder suportar financeiramente a construção de um edifício de n-andares (fracções). Os andares acabam por ser comprados por particulares que normalmente recorrem a empréstimos que lhes são facultados mediante novas hipotecas, agora sobre as fracções em causa. Todas as hipotecas são registadas em livros da série "C".

Considere então a especificação seguinte da estrutura de base para gestão de uma CRP, onde *IdPre (código de prédio)*, *Text (texto sumário)*, *IdFra (identificador de fracção)* e *IdHip (referência a uma hipoteca)* são tipos primitivos:

```
CRP :: B: LivrosB
       C: LivrosC ;
LivrosB :: map IdPre to PreInfo;
LivrosC :: map IdHip to HipInfo;
PreInfo :: desc:       Text    -- PreInfo : informação de prédio
```

```
          hipotecas : set of IdHip
          frac:        Fraccoes
          origem:     [IdPre] -- prédio origem
          valor:      real ;
Fraccoes  :: map IdFra to FraInfo;
HipInfo :: credor:   Text      -- de uma hipoteca
          montante: real;
FraInfo :: desc:        Text    -- FraInfo : informação de fracção
          hipotecas:    set of IdHip
          percentagem: real    -- percentagem da fracção
          rend:         real ;  -- rendimento colectável
IdFra = token;
IdPre = token;
IdHip = token;
Text = seq of char;
```

1. Escreva em VDM-SL o seguinte invariante sobre CRP:

   uma fracção nunca é 0% do valor total do prédio e todas as fracções do mesmo prédio prefazem 100%.

2. Calcule expeditamente (isto é, sem justificar com leis de refinamento) uma implementação relacional de `CRP`.

Questão 4 Recorde a lei de Fokkinga que permite intercombinar duas (ou mais) definições mutuamente recursivas:

$$\left\{\begin{array}{l} f \cdot in = h \cdot \mathsf{F}\,\langle f, g\rangle \\ g \cdot in = k \cdot \mathsf{F}\,\langle f, g\rangle \end{array}\right. \quad \equiv \quad \langle f, g\rangle = (\!|\langle h, k\rangle|\!) \tag{1}$$

É o caso dos predicados par/ímpar ($even, odd$) que se seguem

```
even: nat -> bool
even(n) == if n=0 then true else odd(n-1);

odd : nat -> bool
odd (n) == if n=0 then false else even(n-1);
```

dos quais se obtém (em VDM-SL):

```
even(n) == h(n).#1;
odd(n)  == h(n).#2;
h(n) = if n=0 then mk_(true,false) else mk_(h(n-1).#2,h(n-1).#1);
```

Complete o processo de cálculo dessa solução algorítmica que a seguir se sugere (onde $in = [\underline{0}, suc]$):

$$\left\{\begin{array}{rcl} even \cdot in & = & [\underline{\mathrm{T}}, odd] \\ odd \cdot in & = & [\underline{\mathrm{F}}, even] \end{array}\right.$$

$\equiv$ { ..... }

$$\left\{\begin{array}{rcl} even \cdot in & = & [\underline{\mathrm{T}}, \pi_2] \cdot (id + \langle even, odd\rangle) \\ odd \cdot in & = & [\underline{\mathrm{F}}, \pi_1] \cdot (id + \langle even, odd\rangle) \end{array}\right.$$

$\equiv$ { lei de Fokkinga acima (1) }

$$\langle even, odd\rangle \quad = \quad (\!|\langle [\underline{\mathrm{T}}, \pi_2], [\underline{\mathrm{F}}, \pi_1]\rangle|\!)$$

$\equiv$ { ..... }

$$\langle even, odd\rangle \quad = \quad (\!|[\langle \underline{\mathrm{T}}, \underline{\mathrm{F}}\rangle, swap]|\!)$$

$\equiv$ { ..... }

$$\begin{array}{rcl} even & = & \pi_1 \cdot (\!|[\langle \underline{\mathrm{T}}, \underline{\mathrm{F}}\rangle, swap]|\!) \\ odd & = & \pi_2 \cdot (\!|[\langle \underline{\mathrm{T}}, \underline{\mathrm{F}}\rangle, swap]|\!) \end{array}$$

$\equiv$ { ..... }

$$\left\{\begin{array}{rcl} even & = & \pi_1 \cdot h \\ odd & = & \pi_2 \cdot h \\ h \cdot in & = & [\langle \underline{\mathrm{T}}, \underline{\mathrm{F}}\rangle, swap] \cdot (id + h) \end{array}\right.$$

$$
\begin{array}{ll}
\equiv & \{ \quad \ldots\ldots \} \\
& \left\{ \begin{array}{lcl} even & = & \pi_1 \cdot h \\ odd & = & \pi_2 \cdot h \\ h & = & [\langle \underline{\mathrm{T}}, \underline{\mathrm{F}} \rangle, swap \cdot h] \cdot [\underline{0}, suc]^\circ \end{array} \right. \\
\equiv & \{ \quad \ldots\ldots \} \\
& \left\{ \begin{array}{lcl} even & = & \pi_1 \cdot h \\ odd & = & \pi_2 \cdot h \\ h & = & [\langle \underline{\mathrm{T}}, \underline{\mathrm{F}} \rangle, swap \cdot h] \cdot (i_1 \cdot \underline{0}^\circ \cup i_2 \cdot suc^\circ) \end{array} \right. \\
\equiv & \{ \quad \ldots\ldots \} \\
& \left\{ \begin{array}{lcl} even & = & \pi_1 \cdot h \\ odd & = & \pi_2 \cdot h \\ h & = & ([\langle \underline{\mathrm{T}}, \underline{\mathrm{F}} \rangle, swap \cdot h] \cdot i_1 \cdot \underline{0}^\circ) \cup ([\langle \underline{\mathrm{T}}, \underline{\mathrm{F}} \rangle, swap \cdot h] \cdot i_2 \cdot suc^\circ) \end{array} \right. \\
\equiv & \{ \quad \ldots\ldots \} \\
& \left\{ \begin{array}{lcl} even & = & \pi_1 \cdot h \\ odd & = & \pi_2 \cdot h \\ h & = & (\underline{(\underline{\mathrm{T}}, \underline{\mathrm{F}})} \cdot \underline{0}^\circ) \cup (swap \cdot h \cdot pred) \end{array} \right.
\end{array}
$$

Questão 5 Recorde a lei de refinamento algorítmico

$$S \vdash f \quad \equiv \quad f \cdot dom\ S \subseteq S \tag{2}$$

Comece por completar o cálculo da sua versão com variáveis,

$$
\begin{array}{ll}
& S \vdash f \\
\equiv & \{ \text{ definição (2) } \} \\
& \ldots\ldots \\
\equiv & \{ \text{ shunting (Galois) } \} \\
& \ldots\ldots \\
\equiv & \{ \text{ passagem a “pointwise” } \} \\
& y(dom\ S)x \quad \Rightarrow \quad y(f^\circ \cdot S)x \\
\equiv & \{ \quad \ldots\ldots \} \\
& y = x \wedge x \in dom\ S \quad \Rightarrow \quad (f\ y)\ S\ x \\
\equiv & \{ \ y := x \ \} \\
& x \in dom\ S \quad \Rightarrow \quad (f\ x)\ S\ x
\end{array}
$$

aplicando de seguida essa versão à verificação do facto $S \vdash id$ onde S é a especificação que se segue, escrita em VDM-SL:

```
S(n: real) r: real
pre n > 1
post r*r + 2*n*n = 3*n*r;
```

Será id a única implementação functional de S? Justifique informalmente.

Questão 6 Estudou-se nesta disciplina que, *sob certas condições*, toda a função f expressável sob a forma

$$
\begin{array}{rcl}
\overline{f} & : & B \longrightarrow A \longrightarrow M \\
\overline{f}\ b & \stackrel{\text{def}}{=} & p \rightarrow u, \theta \cdot \langle d_2 \cdot d_1, (\overline{f}\ b) \cdot e \rangle
\end{array} \tag{3}
$$

é convertível num ciclo “while”. Considere o seguinte fragmento de notação VDM++:

```
public filt : (nat -> bool) * seq of nat ==> seq of nat
  filt(p,n) ==
    (dcl n': seq of nat := n,
         r : seq of nat := [];
     while not n'=[] do
           ( r  := r ^ if p(hd n') then [hd n'] else [];
             n' := tl n'
           );
     return r
    );
```

Mostre como derivar este ciclo a partir do esquema (3). Identifique *p*, *e*, etc. incluindo na sua resposta as *condições de aplicação* da referida lei de refinamento algorítmico. Escreva f em notação VDM-SL.
