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

4.º Ano da LMCC (7008N2) + LESI (5308P3)
Ano Lectivo de 2005/06

Exame (época especial) — 9 de Outubro de 2006
16h00
Sala 2205

NB: *Esta prova consta de **8** alíneas todas com a mesma cotação.*

PROVA SEM CONSULTA (2 horas)

Questão 1 Mostre que o isomorfismo

$$(A \rightharpoonup B)^C \quad \cong \quad (C \times A) \rightharpoonup B \tag{1}$$

decorre de outras propriedades do cálculo de estruturas de dados que se estudou nesta disciplina.

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});
```

1. Mostre que S é um hilomorfismo relacional identificando, na definição dada, as relações $Divide$ e $Conquer$ do diagrama

$$\begin{tikzcd} A \rightharpoonup B \arrow[r, "Divide"] \arrow[d, "S"'] & 1 + A^\star \times (A \rightharpoonup B) \arrow[d, "id+id\times S"] \\ A^\star & 1 + A^\star \times A^\star \arrow[l, "Conquer"] \end{tikzcd}$$

2. Identifique leis de *refinamento* de dados correspondentes às representações R e S e especifique, em VDM-SL, as respectivas *abstracções*.

Questão 3 De acordo com o modelo de gestão de uma dada empresa de informática, os programadores e outros recursos humanos ($HResource$) apresentam, todas as semanas, um 'time card' que indica o número de horas que dedicaram a cada tarefa/projecto em curso. Esta informação é depois cruzada com a estimativa inicial do esforço para cada tarefa (medido em "homens*hora"), o que vem a permitir monitorar o andamento dos trabalhos e os desvios entre o real e o previsto.

Apresenta-se a seguir um fragmento da especificação formal do modelo de dados da aplicação que regista os 'time card's, escrita em sintaxe VDM-SL:

```
types

Db = map HRId to HResource;                 -- HRId identifies human resources
HResource :: profile: Txt
             tcards: map Week to Effort;     -- total effort per week
Effort = map Task to Hours;                 -- hours spent per task
Task :: project: PId
        subtask: TId;
Hours = nat1;
Txt   = seq of char;
PId   = seq of char;
TId   = seq of char;
Week  = nat1;                               -- there are 52 weeks in one year
```

1. Especifique — sobre o modelo dado acima — a função que calcula, para um dado identificador de projecto pid, o número total de horas que foram necessárias para o concluir.
2. Com recurso às leis de refinamento de dados que foram estudadas nesta disciplina, calcule o esquema de uma base de dados em SQL que represente o modelo dado, escrevendo-o como um tipo de dados em VDM-SL. Apresente o respectivo processo de cálculo e indique as leis que foram aplicadas em cada passo ou o respectivo par de funções de *abstracção/representação*.

Questão 4 Como sabe, dado um tipo de dados C com pelo menos um habitante c, faz sentido definir a função constante $C \xleftarrow{\underline{c}} A$, que é polimórfica em A. Mostre que o teorema grátis da função $\underline{c}$ se reduz a

$$\langle \forall\ R\ ::\ R \subseteq \mathsf{ker}\,\underline{c} \rangle \tag{2}$$

Questão 5 Nesta disciplina foi estudada uma lei de refinamento de estruturas de dados recursivas, que as permite representar de forma não recursiva com recurso a "apontadores". Enuncie essa lei e explique, por palavras suas, porque é que se calcula, no contexto da aplicação dessa lei a casos concretos, a relação de pertença associada ao functor de base da definição recursiva de que se parte.

Questão 6 Verifique se a função f definida no fragmento de VDM-SL que se segue,

```
types
BTree = [Node];
Node :: item: int left: BTree right: BTree;

functions
f : int * BTree -> bool
f(i,t) == cases t:
           nil            -> false,
           mk_Node(x,l,r) -> if x = i then true else
                             if (i < x) then f(i,l) else f(i,r)
          end;
```

está em condições de ser transformada num ciclo-while. Justifique adequadamente a sua resposta identificando eventuais lei de cálculo que tenha utilizado.

Anexo–Algumas leis de cálculo que podem ser úteis

Hilomorfismos:

$$[\![R, S]\!]^\circ = [\![S^\circ, R^\circ]\!] \tag{3}$$

$$V \cdot [\![S, R]\!] = [\![T, R]\!] \quad \Leftarrow \quad V \cdot S = T \cdot (\mathsf{F}\, V) \tag{4}$$

$$[\![S, R]\!] \cdot V = [\![S, U]\!] \quad \Leftarrow \quad R \cdot V = \mathsf{F}\, V \cdot U \tag{5}$$

$$[\![T, U]\!] \subseteq [\![R, S]\!] \quad \Leftarrow \quad T \subseteq R \,\wedge\, U \subseteq S \tag{6}$$

$$[\![R, S]\!] \subseteq T \quad \Leftarrow \quad R \cdot \mathsf{F}\, T \cdot S \subseteq T \tag{7}$$

Recursividade múltipla:

$$\left\{ \begin{array}{l} f \cdot in = h \cdot \mathsf{F}\,\langle f, g \rangle \\ g \cdot in = k \cdot \mathsf{F}\,\langle f, g \rangle \end{array} \right. \quad \equiv \quad \langle f, g \rangle = (\!|\langle h, k \rangle|\!) \tag{8}$$

$$\langle (\!|i|\!), (\!|j|\!) \rangle = (\!|(i \times j) \cdot \langle \mathsf{F}\,\pi_1, \mathsf{F}\,\pi_2 \rangle|\!) \tag{9}$$

Factorização iterativa: para θ associativa, tem-se

$$\begin{array}{lcl} \langle \mu\, f \;::\; p \rightarrow b \;,\; \theta \cdot \langle d, f \cdot e \rangle \rangle & = & p \rightarrow b \;,\; \theta \cdot (id \times b) \cdot w \cdot \langle d, e \rangle \\ & & onde \\ & & w = \underline{while}\ (\neg \cdot p \cdot \pi_2)\ \underline{do}\ \langle \theta \cdot (id \times d), e \cdot \pi_2 \rangle \end{array} \tag{10}$$

Sendo (θ, u) um monoide, tem-se

$$\langle \mu\, f \;::\; p \rightarrow \underline{u} \;,\; \theta \cdot \langle d, f \cdot e \rangle \rangle \quad = \quad \pi_1 \cdot w \cdot \langle \underline{u}, id \rangle \tag{11}$$

onde w é o mesmo que em (10).

Para S simples, tem-se:

$$S \cdot R \subseteq T \quad \equiv \quad (\delta\, S) \cdot R \quad \subseteq \quad S^\circ \cdot T \tag{12}$$

$$R \cdot S^\circ \subseteq T \quad \equiv \quad R \cdot \delta\, S \quad \subseteq \quad T \cdot S \tag{13}$$