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

4.º Ano da LMCC (7007N2) + LESI (5307P6)
Ano Lectivo de 2000/01

Exame (1.ª chamada) — 10 de Janeiro de 2001
09h30
Salas 2204 e 2205

NB: *Esta prova consta de **10** alíneas que valem, cada uma, 2 valores.*

PROVA SEM CONSULTA ($3\ horas$)

Questão 1 Identifique que operadores sobre conjuntos/funções parciais finitas são especificados pelos seguintes morfismos de conjuntos (justifique, expandindo cada morfismo para notação VDM-SL):

$$\{|[\underline{0}, succ \cdot \pi_2]|\} \quad (1)$$

$$\{|ins \cdot (id + \pi_2 \times id)|\} \quad (2)$$

$$\cup \cdot \langle (\setminus) \cdot (id \times dom), \pi_2 \rangle \quad (3)$$

Questão 2 Considere a seguinte definição indutiva da estrutura de dados A^+ (*listas não vazias*):

$$A^+ \quad \cong \quad A + A \times A^+$$

1. Defina, em VDM-SL, A^+ e duas funções paramétricas que realizem catamorfismos e anamorfismos deste tipo (assuma $A = int$). Apoie a sua resposta com diagramas elucidativos.
2. Acrescente à primeira alínea um invariante que force a sua definição a modelar apenas listas crescentes, assumindo uma ordem total $<$ sobre A.

Questão 3 Considere o seguinte excerto de VDM-SL no qual se definem duas funções sobre a estrutura de dados *árvore binária de inteiros* ($BTreeInt$):

```
types

  Node = ...;
  BTreeInt = ...;

functions

  f : BTreeInt -> int
  f(bt) == if bt = nil then 0
           else let i = bt.Info,
                    lt = bt.LTree,
```

```
                rt = bt.RTree
            in i + f(lt) + f(rt);

g : BTreeInt -> int
g(bt) == if bt = nil then 0
         else let lt = bt.LTree,
                  rt = bt.RTree
              in 1 + g(lt) + g(rt);
```

1. Complete as "..." nas definições de `BTreeInt` e `Node`.
2. Reescreva, em notação *point-free*, as funções `f` e `g` como catamorfimos de `BTreeInt` e explique o significado da função $\alpha = div \cdot \langle f, g \rangle$.

Questão 4 Considere, em VDM-SL, a expressão genérica

```
{ f(i) |-> g(i) | i in set s }
```

onde `f` e `g` são funções arbitrárias e `s` é um subconjunto da intersecção dos domínios de `f` e de `g`. Indique em quais das situações que abaixo se descrevem essa expressão tem problemas de definição e porquê:

- O conjunto `s` tem no máximo 1 elemento.
- A função `f` é constante, isto é `f(i) == k` qualquer que seja `i`.
- O facto `forall i,j in set s & i <> j => f(i) <> f(j)` verifica-se.

Questão 5 Indique através de contra-exemplos quais das seguintes igualdades não são válidas em VDM-SL, onde `f` é uma função parcial finita arbitrária:

```
dom f <-:  f = f        (4)
     <:  f = |->        (5)
  f munion f = f        (6)
      f = f ++ f        (7)
```

Questão 6 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. Cada tarefa tem um número estimado de esforço, medido em "homens*hora", o que permite ao departamento de gestão de projectos e recursos humanos monitorar o andamento dos trabalhos e os desvios entre o real e o previsto.

Apresenta-se a seguir um fragmento da especificação formal da aplicação que gere este processo, escrita em VDM-SL:

```
types

Db = map HRId to HResource;

HResource :: profile: seq of char
```

```
            tcards: map Week to Effort;

Effort = map Task to Hours;

Task :: project: PId
        subtask: TId;

Hours = nat1;
HRId = seq of char;
PId = seq of char;
TId = seq of char;
Week = nat1;
```

1. No contexto deste fragmento, indique quais das seguintes expressões estão bem definidas e, para essas, qual o resultado em cada caso:

```
is_HResource(mk_Task("",""))                              (8)
mk_Task(["n","m"])                                        (9)
dom(mk_HResource("developer",{|->{|->}}).tcards)          (10)
is_Db({|->})                                              (11)
```

2. Analise a seguinte especificação (incompleta) de uma função sobre `Db`:

```
aFunction: ............. -> ......
aFunction(db) ==
 dunion { let tc = db(r).tcards
          in { w | w in set dom tc } | r in set dom db };
```

   Preencha as reticências na declaração de `aFunction` e explique por palavras suas o significado desta função, isto é, qual a informação que `aFunction` extrai do seu argumento.

3. Sendo `WeekMax` uma constante indicando o número máximo de horas de trabalho possíveis por semana, complete a seguinte formulação de um invariante sobre `Db` que garante que nenhum programador declarou um número de horas por semana superior a esse valor:

```
inv db == forall hid in set dom db &
              let hr = db(hid)
              in ......................
```
