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 (2.ª chamada) — 3 de Julho de 2004
09h30
Sala 2209

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

PROVA SEM CONSULTA (2 horas)

Questão 1 O cálculo de refinamento de estruturas de dados estudado nesta disciplina baseia-se em inequações da forma

$$A \underset{F}{\overset{R}{\rightleftarrows}} B \qquad (\leq)$$

onde F se diz uma relação de *abstracção* e R uma relação de *representação* tais que

$$F \cdot R \quad = \quad id \tag{1}$$

Apresente justificações detalhadas para os passos da seguinte prova de que, sempre que R é uma função r, então (1) implica que r seja injectiva:

$$\begin{array}{ll}
 & F \cdot r \quad = \quad id \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & F \cdot r \subseteq id \quad \wedge \quad id \subseteq F \cdot r \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & r \subseteq F \setminus id \quad \wedge \quad id \cdot r^{\circ} \subseteq F \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & r^{\circ} \subseteq F \quad \wedge \quad r \subseteq F \setminus id \\
\Rightarrow & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & r^{\circ} \cdot r \quad \subseteq \quad F \cdot (F \setminus id) \\
\Rightarrow & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & r^{\circ} \cdot r \quad \subseteq \quad id
\end{array}$$

Questão 2 Na modelação formal, em VDM-SL, de um sistema de reserva de lugares numa rede de transportes (eg. comboio, camionete ou outros) entende-se por *linha* uma sequência de paragens, ou estações,

```
Line    = seq of Station;
```

e por uma *reserva* um segmento de uma linha (eg. da segunda à quinta paragem),

```
Reservation :: line        : Line
               origin      : nat1
               destination : nat1
inv x == x.origin < x.destination;
```

convencionando-se que, numa reserva `mk_Reservation(l,i,j)`, o ocupante entra na `i`-ésima estação da linha e sai na `j`-ésima (quer dizer, o lugar já está vago na estação `j`).

O modelo toma ainda como primitivos os tipos que descrevem estações, paragens ou apeadeiros,

```
Station = token;
```

os identificadores do meio de transporte em si

```
TransId = token;
```

os números de lugar,

```
SeatNo = token;
```

e os códigos de reserva de lugar:

```
ResId = token;
```

Para cada comboio (camionete, etc), regista-se a sua rota (as sucessivas estações onde pára) e o total de lugares disponíveis:

```
TransInfo :: route : Line
             seats : set of SeatNo;
```

O sistema de reservas é então modelado por duas funções parciais finitas:

```
System :: trains : map TransId to TransInfo
          res    : map ResId to Reservation;
```

1. Suponha que alguém questiona o modelo adoptado para `Reservation` e propõe como alternativa

   ```
   Reservation' :: line     : Line
                   origin   : nat1
                   occupies : nat;
   ```

   onde `occupies` indica o número de estações (a partir de `origin`) em que o lugar permanece ocupado. A questão é: qual dos modelos `Reservation` ou `Reservation'` é mais abstracto? Isto é, qual desses modelos deve substituir X e Y em

   $$X \overset{R}{\underset{F}{\leq}} Y \tag{2}$$

   Defina F e R após ter feito a sua escolha.

2. Qualquer um desses modelos não garante que se possam reservar lugares para estações além das que a linha dá acesso. Escreva esse invariante em falta para o caso de se preferir `Reservation'` a `Reservation`.

3. Calcule, usando as leis de refinamento estudadas nesta disciplina, uma implementação relacional de `System`. Indique as relações de abstracção/representação que justificam três (à sua escolha) dos passos do cálculo efectuado.

Questão 3 Seja dado o seguinte esboço da especificação (muito simplificada!) do módulo de gestão de processos de um sistema operativo:

```
OSystem :: P: Proc         -- hierarquia de processos
           T: Time;        -- instante actual
Proc    :: I: InfP         -- informação sobre processo
           D: [Proc];      -- processo ''pai''
InfP    :: PRI: nat        -- prioridade do processo
           TTY: [String]   -- terminal associado (se processo interactivo)
           COMMAND: Job    -- tarefa correntemente em execução
           START: nat      -- instante de criação do processo
           TIME: nat       -- tempo de CPU consumido até ao momento
           D: Dir;         -- directoria corrente
```

onde todos os tipos não declarados são considerados atómicos.

1. Especifique em VDM-SL o invariante seguinte sobre a espécie $Proc$:

   > Numa hierarquia $p \in Proc$ de processos, qualquer processo que descenda de outro deverá ser tal que o seu instante de criação seja posterior ao instante de criação do primeiro

2. Numa implementação de `OSystem` será muito provável a representação do tipo recursivo `Proc` (hierarquia de processos) em formato tabular textual, tal como acontece em Linux, por exemplo:

   ```
   PID   PPID PRI  NI   VSZ  RSS WCHAN  STAT TTY   TIME COMMAND
   1153  1136  15   0  4616 1472 wait4  S    pts/0 0:00 bash
   1240  1199  15   0  8476 2972 schedu S    pts/1 0:01 vim
   242      1  15   0 16164 7440 schedu S    ?     0:01 /usr/libexe
   ```

Calcule expeditamente (isto é, sem justificar com leis de refinamento) a representação tabular de $Proc$ e identifique, no modelo obtido, os atributos PID (identificador de processo) e $PPID$ (identificador de processo pai) do exemplo acima. Sendo necessária, no passo de desrecursivação de `Proc`, a formulação da relação de pertença estrutural $\in_{\mathsf{F}}$ associada ao functor F tal que $Proc \cong \mathsf{F}\, Proc$, calcule essa relação $\in_{\mathsf{F}}$.

Questão 4 Estudou-se nesta disciplina que, *sob certas condições*, toda a função f expressável sob a forma

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

onde

$$\begin{array}{rcl} p & : & A \longrightarrow \mathsf{Bool} \\ e & : & A \longrightarrow A \\ d_1 & : & A \longrightarrow B \\ d_2 & : & B \longrightarrow M \\ \theta & : & M \times M \longrightarrow M \\ u & : & 1 \longrightarrow M \end{array}$$

é convertível num ciclo "while". Que f está a ser implementada pelo ciclo seguinte,

```
public f : nat ==> nat
  f(n) ==
    (dcl n': nat := n,
         r : nat := 0;
     while not n'=0 do
           ( r  := r + 2*n'-1;
             n' := n'-1
           );
     return r
    );
```

escrito em notação VDM++? Identifique p, e, etc. e escreva f segundo (3). Mostre ainda que f está nas *condições de aplicação* da referida lei de refinamento algorítmico.
