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

4.º Ano de LMCC (7008N2) + LESI (5308P3)
Ano Lectivo de 2006/07

Exame (época especial) — 11 de Setembro 2007
17H00
Sala 2206

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

PROVA SEM CONSULTA (2 horas)

Questão 1 Partindo da seguinte especificação da operação da divisão inteira entre naturais,

$$\begin{array}{l} Idiv(a : I\!N_0, b : I\!N)\ r : I\!N_0 \\ \textbf{post}\ \langle \forall\ q\ ::\ q \leq r \equiv b \times q \leq a \rangle \end{array}$$

um colega seu refinou-a em

$$\begin{array}{l} Idiv_1(a : I\!N_0, b : I\!N)\ r : I\!N_0 \\ \textbf{post}\ \ b \times r \leq a\ \wedge\ b \times (r+1) > a \end{array}$$

antes de passar à derivação da respectiva implementação algorítmica.

1. Mostre que, de facto, post-$Idiv(r, a, b) \ \Rightarrow\ $ post-$Idiv_1(r, a, b)$, para quaisquer naturais a, b, r.
2. Será a prova da implicação da alínea anterior suficiente para que $Idiv \vdash Idiv_1$ se verifique? Justifique informalmente.

Questão 2 Considere os tipos de estruturas de dados *"apontador para* `struct`*"* ($A \times B + 1$) and *"apontador dentro de* `struct`*"* ($(A+1) \times B$). A questão é: qual destes padrões representa o outro?

Numa questão de um exame anterior desta disciplina pediu-se para provar a injectividade da representação de

$$\begin{tikzcd} A \times B + 1 \arrow[rr, bend left, "R = [i_1 \times id\ , \langle i_2, !^\circ \rangle]"] & \leq & (A+1) \times B \arrow[ll, bend left, "f"] \end{tikzcd} \tag{1}$$

Prove agora que essa mesma relação é inteira.
Sugestão: recorra à lei (9).

Questão 3 Um dos casos de estudo de refinamento de dados estudados nesta disciplina foi o da representação de colecções finitas de dados, de tipo genérico `Collection = set of Data`, por tabelas de *hashing*,

```
HTable = map Location to set of Data
inv HT == forall k in set dom HT &
          HT(k) <> {} and forall d in set HT(k) & hash(d) = k;
Location = nat;
```

refinamento esse estabelecido pela função de representação

```
repf : Collection ->  HTable
repf(S) == { hash(x) |->  { d | d in set S & hash(d) = hash(x) } | x in set S };
```

assumindo fixa uma dada função de *hashing* `hash : Data -> Location`.

1. Sendo $hash$ desejavelmente uma função não injectiva, surgem dúvidas quanto à correcção da definição acima, já que a compreensão pode não garantir a simplicidade do 'mapping' resultado.

   Como a verificação estática das VDMTools não nos pode ajudar, precisamos de "fazer contas". Para isso, vamos aplicar a transformada-PF à definição VDM dada que foi estudada nas aulas desta disciplina,

   $$\lceil repf\ S \rceil \quad = \quad \Lambda(\lceil S \rceil \cdot \mathit{ker}\, hash) \cdot \lceil S \rceil \cdot hash^{\circ} \tag{2}$$

   (onde, como sabe, Λ é o operador de transposição (6) e $\lceil S \rceil$ designa a co-reflexiva associada ao conjunto S) e mostrar que $\lceil repf\ S \rceil$ é sempre simples, qualquer que seja S. Chega-se rapidamente ao seguinte:

   $$\begin{array}{cl} & \lceil repf\ S \rceil \text{ é simples} \\ \Leftarrow & \{ \text{ omitem-se os passos até aqui } \} \\ & hash \cdot \lceil S \rceil \cdot hash^{\circ} \cdot hash \cdot \lceil S \rceil \subseteq hash \end{array}$$

   Complete a demonstração a partir daí, isto é, mostre que a inclusão acima se verifica sempre.
   Sugestão: tire partido do facto de $hash \cdot \lceil S \rceil$ ser simples.

2. Conjecture uma abstracção $absf$ para $repf$, escrevendo-a em notação VDM-SL e dê um exemplo concreto de invertibilidade, isto é, S tal que $absf(repf\ S) = S$. (**NB:** S deverá ser estritamente maior que a colecção vazia; calcule $H = repf\ S$ e depois $absf\ H$; tem total liberdade para inventar a função $hash$.)

Questão 4 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`), os identificadores do meio de transporte em si (`TransId`), os números de lugar (`SeatNo`) e os códigos de reserva de lugar (`ResId`). 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;
```

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 5 A lei que estudou

$$\begin{tikzcd} \mu\mathsf{G} \arrow[r, bend left] \arrow[r, phantom, "\leq"] & (K \rightharpoonup \mathsf{G}\,K) \times K \arrow[l, bend left, "F"] \end{tikzcd} \qquad (3)$$

representa estruturas indutivas ($\mu\mathsf{G}$) sob a forma de pares *(heap, apontador)*. Seja $K \xleftarrow{f} K$ uma função de transformação de apontadores, usada como parâmtero na seguinte operação de re-alocação de células de um *heap* (vulg. *compressão*):

$$\begin{aligned} compress &: (K \longrightarrow K) \longrightarrow (K \rightharpoonup \mathsf{G}\,K) \longrightarrow (K \rightharpoonup \mathsf{G}\,K) \\ compress\ f\ H &\stackrel{\rm def}{=} (\mathsf{G}\,f) \cdot H \cdot f^{\circ} \end{aligned} \qquad (4)$$

A compressão de um *heap* só estará "correcta" se não destruir o que é representado, isto é, se

$$F(compress\ f\ H, f\ k) = F(H, k) \qquad (5)$$

se verificar, onde F é a abstracção em (3).

Complete o cálculo seguinte de uma condição que é suficiente para (5) estar garantida. Que condição é essa?

$$\begin{array}{ll}
 & F(compress\ f H, f\ k) = F(H, k) \\
\equiv & \{\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & (\!|compress\ f H|\!)(f\ k) = (\!|H|\!) k \\
\equiv & \{\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & (\!|(\mathsf{G}\,f) \cdot H \cdot f^{\circ}|\!) \cdot f = (\!|H|\!) \\
\equiv & \{\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & [\![\ in, (\mathsf{G}\,f) \cdot H \cdot f^{\circ}\ ]\!] \cdot f = [\![\ in, H\ ]\!] \\
\Leftarrow & \{\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & (\mathsf{G}\,f) \cdot H \cdot f^{\circ} \cdot f = (\mathsf{G}\,f) \cdot H \\
\Leftarrow & \{\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & H \cdot f^{\circ} \cdot f = H \\
\equiv & \{\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & H \cdot f^{\circ} \cdot f \subseteq H \\
\Leftarrow & \{\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
 & f^{\circ} \cdot f \subseteq id
\end{array}$$

Questão 6 Em especificações VDM é muito vulgar a utilização de compreensões de sequências do tipo da que é usada na função seguinte:

```
mapAndFilter(p)(f)(l) == [ f(l(i)) | i in set inds l & p(l(i))  ];
```

Todas essas compreensões podem ser, se se entender conveniente, convertidas para ciclos-while, desde que previamentes sejam convertidas para catamorfismos como, por exemplo

```
mapAndFilter'(p)(f)(l) ==
         cases l:
            [] -> [],
            others -> if p(hd l) then [f(hd l)] ^ mapAndFilter'(p)(f)(tl l)
                                  else       [] ^ mapAndFilter'(p)(f)(tl l)
         end;
```

Converta o corpo de `mapAndFilter'` num ciclo-while usando as leis que estudou para esse efeito.

Anexo–Algumas leis de cálculo que podem ser úteis

Transposição:

$$f = \Lambda R \quad \equiv \quad R = \in \cdot f \tag{6}$$

Relações simples:

$$R \cdot R^\circ \cdot R = R \quad \Leftarrow \quad R \text{ é simples} \tag{7}$$

"Splits":

$$(\mathit{ker}\, R) \cap (\mathit{ker}\, S) \quad = \quad \mathit{ker}\, \langle R, S \rangle \tag{8}$$

"Eithers":

$$(\mathit{ker}\, R) + (\mathit{ker}\, S) \quad \subseteq \quad \mathit{ker}\, [R\, , S] \tag{9}$$

Refinamento algorítmico:

$$S \vdash R \quad \equiv \quad (\delta\, S \subseteq \delta\, R) \,\wedge\, (R \cdot \delta\, S \subseteq S) \tag{10}$$

Refinamento de dados:

$$A \rightharpoonup 1 \quad \cong \quad \mathcal{P} A \tag{11}$$

$$A^\star \quad \leq \quad I\!N \rightharpoonup A \tag{12}$$

$$A \rightharpoonup (D \times (B \rightharpoonup C)) \quad \leq \quad (A \rightharpoonup D) \times ((A \times B) \rightharpoonup C) \tag{13}$$

Hilomorfismos:

$$[\![R, S]\!] \quad = \quad R \cdot \mathsf{F}\, [\![R, S]\!] \cdot S \tag{14}$$

$$[\![R, S]\!] \subseteq T \quad \Leftarrow \quad R \cdot \mathsf{F} T \cdot S \subseteq T \tag{15}$$

$$V \cdot [\![S, H]\!] \subseteq [\![T, H]\!] \quad \Leftarrow \quad V \cdot S \subseteq T \cdot (\mathsf{G}\, V) \tag{16}$$

$$[\![R, S]\!]^\circ = [\![S^\circ, R^\circ]\!] \tag{17}$$

$$V \cdot [\![S, R]\!] = [\![T, R]\!] \quad \Leftarrow \quad V \cdot S = T \cdot (\mathsf{F}\, V) \tag{18}$$

$$[\![S, R]\!] \cdot V = [\![S, U]\!] \quad \Leftarrow \quad R \cdot V = \mathsf{F}\, V \cdot U \tag{19}$$

Factorização iterativa: para θ associativa, tem-se

$$\begin{array}{lcl} \langle \mu\, f \;::\; p \rightarrow b\, ,\; \theta \cdot \langle d, f \cdot e \rangle \rangle & = & p \rightarrow b\, ,\; \theta \cdot (id \times b) \cdot w \cdot \langle d, e \rangle \\ & & \mathit{onde} \\ & & w = \underline{\mathit{while}}\; (\neg \cdot p \cdot \pi_2)\; \underline{\mathit{do}}\; \langle \theta \cdot (id \times d), e \cdot \pi_2 \rangle \end{array} \tag{20}$$

Sendo (θ, u) um monoide, tem-se

$$\langle \mu\, f \;::\; p \rightarrow \underline{u}\, ,\; \theta \cdot \langle d, f \cdot e \rangle \rangle \quad = \quad \pi_1 \cdot w \cdot \langle \underline{u}, id \rangle \tag{21}$$

onde w é o mesmo que em (20).