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

4.º Ano da LMCC (7008N2) + LESI (5308P3)
Ano Lectivo de 2002/03

Exame (época normal, 1ª chamada) — 28 de Junho 2003
09h30
Salas 3303, 3304

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

PROVA SEM CONSULTA (2 horas)

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

$$\begin{tikzcd} A \arrow[rr, bend left, "R"] \arrow[rr, phantom, "\leq"] & & B \arrow[ll, bend left, "F"] \end{tikzcd} \qquad \text{tais que} \qquad R \subseteq F^\circ \wedge \mathit{img}\, F = id \wedge \mathit{ker}\, R = id$$

onde R se diz relação de *representação* e F se diz função de *abstracção*. Mostre que $\langle R, S\rangle$ é uma relação de representação sempre que R e S individualmente o são.
NB: baseie o seu raciocínio na propriedade

$$\langle R, S\rangle^\circ \cdot \langle X, Y\rangle \quad = \quad (R^\circ \cdot X) \cap (S^\circ \cdot Y) \tag{1}$$

Questão 2 Sejam dadas as funções $untot$ e tot definidas por

$$\begin{aligned} untot &\stackrel{\mathrm{def}}{=} (i_1^\circ \cdot) \\ tot\ \sigma &= [\sigma^\circ, (id - \mathit{dom}\,\sigma) \cdot (!^\circ)]^\circ \end{aligned}$$

Identifique os tipos A e B em $B \xleftarrow{tot} A$ (desenhe um diagrama explicativo) e complete a seguinte prova de que tot é inversa à direita de $untot$:

$$\begin{aligned}
& untot \cdot tot = id \\
\equiv \quad & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& untot(tot\ \sigma) = \sigma \\
\equiv \quad & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& i_1^\circ \cdot [\sigma^\circ, (id - \mathit{dom}\,\sigma) \cdot (!^\circ)]^\circ = \sigma \\
\equiv \quad & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& ([\sigma^\circ, (id - \mathit{dom}\,\sigma) \cdot (!^\circ)] \cdot i_1)^\circ = \sigma \\
\equiv \quad & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& (\sigma^\circ)^\circ = \sigma \\
\equiv \quad & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& \mathrm{T}
\end{aligned}$$

Questão 3 Qualquer sequência finita pode ser representada por uma função finita que indica, explicitamente, que elemento ocupa que posição na sequência. Por exemplo, a sequência $[a, b, a]$ é representada por $\{1 \mapsto a, 2 \mapsto b, 3 \mapsto a\}$. Especifique em VDM-SL a representação $seq2fm$ e a abstracção $fm2seq$ que justificam a inequação

$$\begin{tikzcd} \mathsf{seq\,of}\,A \arrow[rr, bend left, "seq2fm"] \arrow[rr, phantom, "\leq"] & & \mathsf{map}\ nat\ \mathsf{to}\,A \arrow[ll, bend left, "fm2seq"] \end{tikzcd} \tag{2}$$

Por que razão é que a lei acima não é um isomorfismo?

Questão 4 Na especificação formal de um sistema de gestão de conhecimento, escrita em VDM-SL, encontra-se o seguinte modelo para expressões sintáticas arbitrárias:

```
Exp    = Var | Term ;
Var    :: variable: Symbol ;
Term   :: operator: Symbol
          arguments: seq of Exp
          inv t == len t.arguments <= 20 ;
Symbol = seq of char
         inv s == len s <= 10 ;
```

Inspeccionando a sua implementação em SQL, verificamos que a este fragmento de VDM-SL corresponde o seguinte código:

```
CREATE TABLE EXPRESSIONS (
       FatherId  NUMERIC (10) NOT NULL,
       ArgNr     NUMERIC (20) NOT NULL,
       ChildId   NUMERIC (10) NOT NULL
       CONSTRAINT EXPRESSIONS_pk PRIMARY KEY (FatherId,ArgNr)
       );

CREATE TABLE OPERATORS (
       NodeId   NUMERIC (10) NOT NULL,
       Operator CHAR    (10) NOT NULL
       CONSTRAINT OPERATORS_pk PRIMARY KEY(NodeId)
       );

CREATE TABLE VARIABLES (
       NodeId   NUMERIC (10) NOT NULL,
       Variable CHAR    (10) NOT NULL
       CONSTRAINT VARIABLES_pk PRIMARY KEY(NodeId)
       );

ALTER TABLE EXPRESSIONS ADD CONSTRAINT EXPRESSIONS_fk1
      FOREIGN KEY (ChildId) REFERENCES EXPRESSIONS(NodeId);
```

Concorda com as decisões que foram tomadas nesta codificação? Se a sua resposta é afirmativa, exponha o respectivo processo de cálculo. Se é negativa, indique quais as deficiências encontradas e refaça o respectivo processo de cálculo. Em qualquer dos casos, indique as leis de refinamento de dados que foram aplicadas em cada passo do seu raciocínio.

Questão 5 Sendo a diferença de relações definida pela seguinte conecção de Galois,

$$\begin{array}{|c|c|} \hline \multicolumn{2}{|c|}{(f\ X) \subseteq Y \equiv X \subseteq (g\ Y)} \\ \hline f = g^{\flat} & g = f^{\sharp} \\ \hline (\_ - R) & (R \cup\ ) \\ \hline \end{array} \tag{3}$$

estabeleça a correspondente propriedade universal e prove, por indirecção a partir desta, que a seguinte propriedade se verifica:

$$(X - S) - R = X - (S \cup R) \tag{4}$$

Questão 6 Atente no seguinte diálogo entre dois colegas seus:

A: O que se pretende é encontrar uma solução funcional para a especificação seguinte: *a lista resultado deverá ser uma permutação com os mesmos elementos (números naturais) da lista de entrada*.

B: Isso corresponde a resolver em ordem a f a equação $S \vdash f$, onde

```
S(i: seq of nat) r: seq of nat
post seq2bag(r) = seq2bag(i) and elems(r) = elems(i) ;
```

A: Exacto. Mas para efeitos de cálculo é melhor escrevermos

$$(\mathit{ker}\ seq2bag) \cap (\mathit{ker}\ elems) \vdash f \tag{5}$$

B: Para mim isso é a mesma coisa que

$$(\mathit{ker}\ seq2bag) \vdash f \wedge (\mathit{ker}\ elems) \vdash f \tag{6}$$

A: Talvez tenhas razão. Mas olha que nos vai bastar resolver $(\mathit{ker}\ seq2bag) \vdash f \ldots$

B: Porquê?

A: Porque $elems = dom \cdot seq2bag$ e dom é inteira. Logo $\mathit{ker}\ seq2bag \subseteq \mathit{ker}\ elems$.

B: Bem visto. Já agora vamos avisar quem escreveu a pos-condição de S que pode remover a cláusula `elems(r) = elems(i)` — está nitidamante a mais!

1. Deduza a lei geral em que **A** se baseou para reduzir $(\mathit{ker}\ seq2bag) \cap (\mathit{ker}\ elems)$ a $\mathit{ker}\ seq2bag$.
2. Faltando ainda verificar $elems = dom \cdot seq2bag$, complete a seguinte prova desse facto:

$$
\begin{array}{cl}
 & elems = dom \cdot seq2bag \\
\equiv & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
 & (\!|ins|\!) = dom \cdot (\!|[bnil, bcons]|\!) \\
\Leftarrow & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
 & dom \cdot [bnil, bcons] = ins \cdot (id + id \times dom) \\
\equiv & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
 & [dom \cdot bnil, dom \cdot bcons] = [\underline{\emptyset}, puts \cdot (id \times dom)] \\
\equiv & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
 & \left\{ \begin{array}{l} dom \cdot \underline{\{\mapsto\}} = \underline{\emptyset} \\ dom \cdot \oplus \cdot (singb \times id) = \cup \cdot (sings \times dom) \end{array} \right. \\
\equiv & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
 & \left\{ \begin{array}{l} \underline{dom\ \{\mapsto\}} = \underline{\emptyset} \\ \cup \cdot (dom \times dom) \cdot (singb \times id) = \cup \cdot (sings \times dom) \end{array} \right. \\
\equiv & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
 & \left\{ \begin{array}{l} dom\ \{\mapsto\} = \emptyset \\ \cup \cdot (dom \cdot singb \times dom) = \cup \cdot (sings \times dom) \end{array} \right. \\
\equiv & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
 & \left\{ \begin{array}{l} \mathrm{T} \\ \cup \cdot (sings \times dom) = \cup \cdot (sings \times dom) \end{array} \right. \\
\equiv & \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
 & \left\{ \begin{array}{l} \mathrm{T} \\ \mathrm{T} \end{array} \right.
\end{array}
$$