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

4.º Ano da LMCC (7007N2) + LESI (5307P6)
Ano Lectivo de 2006/07

Exame (2.ª chamada da época normal) — 25 de Janeiro 2007
14h00
Sala 2306

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

PROVA SEM CONSULTA (2 horas)

Questão 1 Considere o seguinte texto, extraído da *Wikipedia* (tópico: *Priority Queue*):

A priority queue is an abstract data type supporting the following three operations:

- *add an element to the queue with an associated priority*
- *remove the element from the queue that has the highest priority, and return it*
- *(optionally) peek at the element with highest priority without removing it*

Suponha que alguém propõe o seguinte modelo para esta estrutura de dados, baseado em sequências finitas:

```
types

PQueue = seq of Entry;
Entry :: element: token
         priority: nat;
```

1. É intuitivo ver que este modelo só funciona se a fila estiver ordenada por prioridades. Acrescente essa propriedade ao tipo `PQueue` sob a forma de um invariante não recursivo.
2. Apresente especificações (não recursivas) para as três operações `add`, `remove` e `peek` referidas no texto da Wikipedia supra citado.

Questão 2 O cálculo relacional dito *point-free* é particularmente útil para raciocinar sobre funções, que como sabe são casos particulares de relações — exactamente aquelas que são simples (imagem coreflexiva) e inteiras (núcleo reflexivo). Por exemplo, um método construtivo para mostrar que uma função f é um isomorfismo (ie. função injectiva e sobrejectiva) é calcular o seu converso f°: se este for uma função, tanto f como f° são isomorfismos.

1. Demonstre o método acima proposto, isto é mostre que a equivalência

$$R \text{ é função } \wedge\ R^\circ \text{ é função } \quad \equiv \quad R \text{ é isomorfismo} \tag{1}$$

se verifica.

2. Aplica-se de seguida o método proposto acima ao isomorfismo $[id+i_1\ , i_2 \cdot i_2]$ (que testemunha a associatividade de $A+B$), calculando-se o seu converso $[i_1 \cdot i_1\ , i_2 + id]$:

$$
\begin{array}{ll}
& [id + i_1\ , i_2 \cdot i_2]^\circ \\
= & \quad \{ \ \ldots\ldots\ldots \ \} \\
& ((id + i_1) \cdot i_1^\circ \cup i_2 \cdot i_2 \cdot i_2^\circ)^\circ \\
= & \quad \{ \ \ldots\ldots\ldots \ \} \\
& i_1 \cdot (id + i_1^\circ) \cup i_2 \cdot i_2^\circ \cdot i_2^\circ \\
= & \quad \{ \ \ldots\ldots\ldots \ \} \\
& i_1 \cdot [i_1\ , i_2 \cdot i_1^\circ] \cup i_2 \cdot i_2^\circ \cdot i_2^\circ \\
= & \quad \{ \ \ldots\ldots\ldots \ \} \\
& i_1 \cdot (i_1 \cdot i_1^\circ \cup i_2 \cdot i_1^\circ \cdot i_2^\circ) \cup i_2 \cdot i_2^\circ \cdot i_2^\circ \\
= & \quad \{ \ \ldots\ldots\ldots \ \} \\
& i_1 \cdot i_1 \cdot i_1^\circ \cup (i_1 \cdot i_2 \cdot i_1^\circ \cup i_2 \cdot i_2^\circ) \cdot i_2^\circ \\
= & \quad \{ \ \ldots\ldots\ldots \ \} \\
& [i_1 \cdot i_1\ , [i_1 \cdot i_2\ , i_2]] \\
= & \quad \{ \ \ldots\ldots\ldots \ \} \\
& [i_1 \cdot i_1\ , i_2 + id]
\end{array}
$$

Apresente justificações para todos os passos do cálculo.

Questão 3 Sejam $\underline{a}$ e $\underline{b}$ duas funções constantes. Qual o tipo e o significado da relação $\underline{a} \cdot \underline{b}^\circ$? Responda introduzindo variáveis e simplificando.

Questão 4 Mostre, por cálculo-PF, que a relação $\underline{a} \cdot \underline{b}^\circ$ da Questão 3 é simples (para todo o a, b) e codifique-a em notação VDM-SL.

Questão 5 Como sabe, a união de dois *mappings* em VDM-SL não dá garantidamente um *mapping* como resultado, cf. o seguinte extrato do respectivo manual:

Operator	Name	Semantics description
`m1 munion m2`	Merge	yields a map combined by `m1` and `m2` such that the resulting map maps the elements of dom `m1` as does `m1`, and the elements of dom `m2` as does `m2`. The two maps must be compatible.

Um pouco adiante no mesmo manual refere-se que dois *mappings* `m1` e `m2` são compatíveis se e só se a condição

```
forall d in set dom m1 inter dom m2 & m1(d) = m2(d)
```

se verifica.

Apresente justificações para o cálculo que se segue e que mostra que, de facto, a compatibilidade é *equivalente* à simplicidade da união de dois *mappings* M e N:

$M \cup N$ é simples

$\equiv$ { ..... }

$(M \cup N) \cdot (M \cup N)^\circ \subseteq id$

$\equiv$ { ..... }

$(M \cup N) \cdot (M^\circ \cup N^\circ) \subseteq id$

$\equiv$ { ..... }

$M \cdot M^\circ \cup M \cdot N^\circ \cup N \cdot M^\circ \cup N \cdot N^\circ \subseteq id$

$\equiv$ { ..... }

$M \cdot M^\circ \subseteq id \ \wedge \ M \cdot N^\circ \subseteq id \ \wedge \ N \cdot M^\circ \subseteq id \ \wedge \ N \cdot N^\circ \subseteq id$

$\equiv$ { ..... }

M é simples $\wedge$ $M \cdot N^\circ \subseteq id$ $\wedge$ $(M \cdot N^\circ)^\circ \subseteq id$ $\wedge$ N é simples

$\equiv$ { ..... }

M é simples $\wedge$ $M \cdot N^\circ \subseteq id$ $\wedge$ N é simples

$\equiv$ { ..... }

M é simples $\wedge$ $M \cdot \delta\, N \subseteq N$ $\wedge$ N é simples

$\equiv$ { ..... }

M é simples $\wedge$ $\delta\, M \cdot \delta\, N \subseteq M^\circ \cdot N$ $\wedge$ N é simples

$\equiv$ { ..... }

M é simples $\wedge$ M e N são compatíveis $\wedge$ N é simples

---

Questão 6 Um par **pre**/**post** diz-se, em VDM-SL, uma especificação *realizável*, ou *satisfiável*, se a condição

$$\langle \forall\, a \ :: \ pre(a) \Rightarrow \langle \exists\, b \ :: \ post(b, a) \rangle \rangle \tag{2}$$

se verificar.

Mostre que a especificação que segue da operação de *divisão* de números inteiros (a chamada *divisão inteira*)

```
idiv (a,b: int) q: int
post q * b <= a and (q+1) * b > a ;
```

não é realizável, e acrescente-lhe uma pré-condição por forma a que o passe a ser (**NB:** não tem de provar a propriedade formalmente.)
