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

4.º Ano da LMCC (7008N2) + LESI (5308P3)
Ano Lectivo de 2001/02

Exame (1.ª chamada) — 12 de Junho de 2002
09h30
Salas 2207, 2208

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

PROVA SEM CONSULTA (2 horas)

Questão 1 Na teoria de refinamento estudada nesta disciplina a comparação entre dois tipos de dados A e B é feita com base na existência de uma *função de abstracção* f sobrejectiva, isto é, invertível à direita:

$$A \leq B \quad sse \quad \exists\ A \xleftarrow{f} B\ ,\ A \xrightarrow{r} B\ : f \cdot r = id_A$$

1. Sendo f e g duas funções de abstracção (sobrejectivas), mostre que $\langle f, g \rangle$ não é necessariamente uma função de abstracção sobrejectiva.
2. Mostre que toda a função involutiva estabelece um isomorfismo de dados, dando exemplos que conheça.
   ***NB:** uma função involutiva é tal que a sua aplicação sucessiva se cancela, por exemplo $f\ x = -x$, pois verifica-se $-(-x) = x$, a popular regra "menos por menos dá mais".*

Questão 2 Uma tabela de "hashing"

```
HTable = map Index to set of Data;
```

pode ser vista como a implementação de uma tabela relacional

```
Table = set of Data;
```

sob a seguinte função de representação

```
rep: Table -> HTable
rep(t) == collect[Index,Data]({ mk_(hash(d),d) | d in set t});
```

que assume uma função de "hashing"

```
hash: Data -> Index
```

e uma função que conhece:

```
collect[@A,@B]: set of (@A*@B) -> map @A to set of @B
collect(r) ==  { a |-> { q.#2 | q in set r & a = q.#1 }
                   | a in set { p.#1 | p in set r } };
```

1. Supondo `Index` = `Data` = `nat` e a função de hashing `hash(d) == d mod 11` , calcule a representação em tabela de "hashing" do conjunto `{ 1 , 10, 11, 23, 35, 456 }` . Com base neste exemplo, conjecture o invariante concreto (de representação) garantido pela função `rep`.
2. `collect` é uma função de representação ou de abstracção? Justifique a sua resposta com base nas propriedades desta função.
3. Especifique em notação VDM-SL a função de abstracção que é inversa à esquerda de `rep`.

Questão 3 Considere a seguinte definição (incompleta) de uma função em VDM-SL:

```
seq2m[@A]: ..........................
seq2m(l) == { x |-> (card { i | i in set (inds l) & l(i)=x })
               | x in set elems(l) };
```

Complete as reticências da assinatura da função e mostre que `seq2m` não pode ser considerada uma função de representação. (Pode justificar através da apresentação de contra-exemplos).

Questão 4 O registo de chamadas não atendidas em telemóveis é vulgarmente organizado de forma a que os últimos números que chamaram estejam imediatamente acessíveis, até um máximo de 10 números:

```
store : int * seq of int -> seq of int
store(n,s) ==
    take[int](10,
              [n] ^ filter[int](lambda x:int & not(x=n))(s));
```

onde `take` é a função

```
take[@A] : nat * seq of @A -> seq of @A
take(n,l) == if n=0 or l=[] then []
             else [hd l] ^ take[@A](n-1,tl l);
```

e `filter` é definida como se segue:

```
filter[@A] : (@A -> bool) -> seq of @A -> seq of @A
filter(p)(s) == [ s(i) | i in set inds s & p(s(i)) ];
```

Seguem-se os três primeiros passos do processo de cálculo de uma implementação de `store`:

$$
\begin{aligned}
& \texttt{store}(n, s) \\
= \quad & \{ \text{definição de } \texttt{store} \} \\
& \texttt{take}(10, [n] \frown \texttt{filter}(\lambda x.x \neq n)(s)) \\
= \quad & \{ \text{propriedade de } \texttt{take} \text{ — qual?} \} \\
& [n] \frown \texttt{take}(9, \texttt{filter}(\lambda x.x \neq n)(s)) \\
= \quad & \{ \text{introdução de } \texttt{filteratmost} \} \\
& [n] \frown \texttt{filteratmost}(\lambda x.x \neq n)(9, s)
\end{aligned}
$$

Formule a propriedade de `take` que foi útil no segundo passo do cálculo e conjecture a função `filteratmost` que é introduzida no terceiro passo do cálculo, especificando-a em VDM-SL.
