Métodos Formais de Programação II

4.º Ano da LMCC (7008N2)
Ano Lectivo de 1998/99

Exame (época normal, 1.ª chamada) — 16 de Junho de 1999
09h30
Salas 2301 a 2306

NB: *Esta prova consta de* ***5*** *questões de* ***4 valores*** *cada uma.*
Sempre que os seus cálculos utilizarem resultados, leis, teoremas etc. *disponíveis no material pedagógico desta disciplina não os copie para a sua folha de exame — limite-se a citar a sua referência.*

PROVA COM CONSULTA $(3\ horas)$

Questão 1 Recorde das aulas teóricas desta disciplina a seguinte definição e diagrama referentes à especificação do invariante que garante que uma lista não vazia está ordenada:

$$
\begin{array}{l}
ordered : \ A^+ \longrightarrow 2 \\
ordered <a> = V \\
ordered\,(cons(a,l)) = a > (Max\,l) \wedge (ordered\,l)
\end{array}
$$

$$
\begin{tikzcd}
A^+ \arrow[d, "ordered"'] & A + A \times A^+ \arrow[l, "{[singl, cons]}"'] \arrow[d, "id + id \times \langle Max, ordered \rangle"] \\
2 & A + A \times (A \times 2) \arrow[l, "{[\underline{V}, \alpha]}"]
\end{tikzcd}
$$

onde *singl* $a \stackrel{\rm def}{=} <a>$, $\alpha(a,(m,b)) \stackrel{\rm def}{=} a > m \wedge b$ e $Max = (\![[id, max]]\!)$.

Adapte este esquema à especificação do invariante que garante que uma *árvore binária de procura* está *equilibrada*, isto é, que em qualquer dos seus nós o *peso* das duas sub-árvores (esquerda e direita) difere quando muito de uma unidade.

RESOLUÇÃO: *Árvores binárias de procura* são listas bi-lineares, isto é, soluções da equação $X \cong 1 + A \times (X \times X)$. Logo, em lugar de $\mathsf{F}X = A + A \times X$ temos $\mathsf{F}X = 1 + A \times (X \times X)$, isto é, $\mathsf{F}f = id + id \times (f \times f)$. Vamos designar por $\mathsf{Bt}\,A$ e $[\underline{empty}, node]$ o correspondente a A^+ e a $[singl, cons]$, respectivamente, e por $balanced$ o invariante a definir. Para concluir a adaptação, basta notar que em lugar de Max se tem $weight$ (peso) definido por

$$
\begin{array}{l}
weight : \ \mathsf{Bt}\,A \longrightarrow I\!N_0 \\
weight\,empty = 0 \\
weight\,(node(a,(x,y))) = 1 + max(weight\,x, weight\,y)
\end{array}
$$

isto é, $weight = (\![[\underline{0}, succ \cdot max \cdot \pi_2]]\!)$.
Tem-se, então:

$$
\begin{tikzcd}
\mathsf{Bt}\,A \arrow[d, "balanced"'] & 1 + A \times (\mathsf{Bt}\,A \times \mathsf{Bt}\,A) \arrow[l, "{[\underline{empty}, node]}"'] \arrow[d, "id + id \times (\langle weight, balanced \rangle \times \langle weight, balanced \rangle)"] \\
2 & 1 + A \times ((I\!N_0 \times 2) \times (I\!N_0 \times 2)) \arrow[l, "{[\underline{V}, \alpha]}"]
\end{tikzcd}
$$

onde

$$\alpha(a,((w,b),(w',b'))) \stackrel{\rm def}{=} |w - w'| \leq 1 \wedge b \wedge b'$$

isto é:

$$
\begin{array}{lll}
balanced : \ \mathsf{Bt}\,A \longrightarrow 2 & & \\
balanced\,empty = V & & \\
balanced\,(node(a,(x,y))) = & let & w = weight\,x \\
 & & w' = weight\,y \\
 & in & |w - w'| \leq 1 \wedge (balanced\,x) \wedge (balanced\,y)
\end{array}
$$

□

Questão 2 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:

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

Sejam f e g duas funções de abstracção (sobrejectivas). Indique, justificando formalmente, quais das expressões funcionais seguintes designam também funções de abstracção (sobrejectivas):

$$[f, g] \tag{1}$$
$$f \times g \tag{2}$$
$$\langle f, g\rangle \tag{3}$$
$$f + g \tag{4}$$

RESOLUÇÃO:

(1) — seja r a inversa à direita de f, isto é, $f \cdot r = id_A$. Tem-se então

$$
\begin{array}{cl}
& f \cdot r = id_A \\
\equiv & \quad \{ \text{ por cancelamento-}+ \textit{ cf. } (1.37) \text{ dos apontamentos em inglês} \} \\
& ([f, g] \cdot i_1) \cdot r = id_A \\
\equiv & \quad \{ \text{ associatividade da composição, ou seja, } (1.8) \text{ dos apontamentos em inglês} \} \\
& [f, g] \cdot (i_1 \cdot r) = id_A \\
\equiv & \quad \{ \text{ definindo } r' = i_1 \cdot r \} \\
& [f, g] \cdot r' = id_A
\end{array}
$$

Logo $[f, g]$ é função de abstracção (sobrejectiva) pois foi possível encontrar uma sua inversa à direita $r' = i_1 \cdot r$ [1].

(2) — a resposta é de novo positiva como se verifica de imediato:

$$
\begin{array}{cl}
& f \cdot r = id_A \wedge g \cdot s = id_B \\
\equiv & \quad \{ \text{ produto entre iguais} \} \\
& (f \cdot r) \times (g \cdot s) = id_A \times id_B \\
\equiv & \quad \{ \text{ produto é bifunctor, } \textit{cf. } (1.27,1.28) \text{ dos apont. em inglês} \} \\
& (f \times g) \cdot (r \times s) = id_{A \times B}
\end{array}
$$

Logo, $r \times s$ é inversa à direita de $f \times g$.

(3) — a resposta é negativa neste caso. Vejamos a que corresponde a existência de uma inversa à direita r' de $\langle f, g\rangle$:

$$
\begin{array}{cl}
& \langle f, g\rangle \cdot r' = id_{A \times B} \\
\equiv & \quad \{ \text{ fusão-}\times \text{ e reflexão-}\times \textit{ cf. } \text{resp. } (1.23,1.29) \text{ dos apont. em inglês} \} \\
& \langle f \cdot r', g \cdot r'\rangle = \langle \pi_1, \pi_2\rangle \\
\equiv & \quad \{ \text{ unicidade de `splits'} \} \\
& f \cdot r' = \pi_1 \wedge g \cdot r' = \pi_2
\end{array}
$$

É fácil de ver que nem todo o r' satisfaz estas duas cláusulas. Por exemplo, para $f = g = id$ ter-se-ia que ter $r' = \pi_1 = \pi_2$, o que é impossível.

(4) — em tudo igual a (2), basta substituir $\times$ (produto) por $+$ (coproduto).

□

---

[1] Idêntico raciocínio poderia ter sido baseado na inversa de g.

Questão 3 Considere o seguinte modelo formal (muito!!) simplificado para uma base de dados de apuramento eleitoral:

$$
\begin{aligned}
Bd &= Freg \rightharpoonup Inscritos \times Apuramento && (5)\\
Inscritos &= I\!N && (6)\\
Apuramento &= Partido \rightharpoonup I\!N && (7)
\end{aligned}
$$

em que $Inscritos$ e $Partido$ são representados por 'strings'. Mostra-se a seguir um fragmento (idealizado) dessa base de dados, numa noite de eleições, tal como apareceria animado em CAMILA:

```
[ "S.Victor-Braga"->  < 1150, [ "CDS/PP"-> 100, "PS"-> 400, "PSD"-> 350, "CDU"-> 150 ]>,
  "Ucha-Barcelos"->   < 1030, [ "CDS/PP"-> 190, "PS"-> 590, "PSD"-> 290, "CDU"-> 110 ]>,
  "Paranhos-Porto"->  < 1270, [ "CDS/PP"-> 120, "PS"-> 450, "PSD"-> 370, "CDU"-> 110 ]>,
  "Vila Verde"->      < 1230, []>,
  "V.N.Foz-Coa"->     < 1010, [ "CDS/PP"-> 190, "PS"-> 330, "PSD"-> 390, "CDU"-> 70 ]>,
  "Anais-P.Lima"->    < 1450, []>
]
```

(Note-se que os dados de duas freguesias ainda não chegaram à base de dados.)

Solicitado a especificar a operação que realiza o apuramento total nacional, um técnico escreveu a seguinte especificação funcional,

$$Bd \xrightarrow{f_1} (a) \xrightarrow{f_2} (b) \xrightarrow{f_3} (c) \xrightarrow{f_4} (Partidos \rightharpoonup I\!N)$$

tendo desafiado os colegas a identificarem as funções f_1, f_2, f_3 e f_4, mostrando-lhes apenas o resultado da sua animação, tal como se segue:

Formato (a), após f_1:

```
[ "S.Victor-Braga"->  [ "CDS/PP"-> 100, "PS"-> 400, "PSD"-> 350, "CDU"-> 150 ],
  "Anais-P.Lima"->    [],
  "Vila Verde"->      [],
  "Paranhos-Porto"->  [ "CDS/PP"-> 120, "PS"-> 450, "PSD"-> 370, "CDU"-> 110 ],
  "V.N.Foz-Coa"->     [ "CDS/PP"-> 190, "PS"-> 330, "PSD"-> 390, "CDU"-> 70 ],
  "Ucha-Barcelos"->[ "CDS/PP"-> 190, "PS"-> 590, "PSD"-> 290, "CDU"-> 110 ]
]
```

Formato (b), após f_2:

```
[ < "CDS/PP","S.Victor-Braga"> -> 100,
  < "PS","S.Victor-Braga"> -> 400,
  < "PSD","S.Victor-Braga"> -> 350,
  < "CDU","S.Victor-Braga"> -> 150,
  < "CDS/PP","Paranhos-Porto"> -> 120,
  < "PS","Paranhos-Porto"> -> 450,
  < "PSD","Paranhos-Porto"> -> 370,
  < "CDU","Paranhos-Porto"> -> 110,
  < "CDS/PP","V.N.Foz-Coa"> -> 190,
  < "PS","V.N.Foz-Coa"> -> 330,
  < "PSD","V.N.Foz-Coa"> -> 390,
  < "CDU","V.N.Foz-Coa"> -> 70,
  < "CDS/PP","Ucha-Barcelos"> -> 190,
  < "PS","Ucha-Barcelos"> -> 590,
  < "PSD","Ucha-Barcelos"> -> 290,
  < "CDU","Ucha-Barcelos"> -> 11
]
```

Formato (c), após f_3:

```
[ "CDS/PP"->[ "S.Victor-Braga"-> 100, "Paranhos-Porto"-> 120, "V.N.Foz-Coa"-> 190, "Ucha-Barcelos"-> 190 ],
  "PS"->[ "S.Victor-Braga"-> 400, "Paranhos-Porto"-> 450, "V.N.Foz-Coa"-> 330, "Ucha-Barcelos"-> 590 ],
  "PSD"->[ "S.Victor-Braga"-> 350, "Paranhos-Porto"-> 370, "V.N.Foz-Coa"-> 390, "Ucha-Barcelos"-> 290 ],
  "CDU"->[ "S.Victor-Braga"-> 150, "Paranhos-Porto"-> 110, "V.N.Foz-Coa"-> 70, "Ucha-Barcelos"-> 110 ]
]
```

Formato final ($Partidos \rightharpoonup I\!N$), após f_4:

```
[ "CDS/PP"-> 600, "CDU"-> 440, "PS"-> 1770, "PSD"-> 1400 ];
```

1. Defina as funções f_1, f_2, f_3 e f_4, justificando a sua resposta.
2. Especifique o invariante (relevante!) sobre Bd que garante que em nenhuma freguesia o total dos votos apurados excede o número de eleitores inscritos.

RESOLUÇÃO:

1. Abreviando $Freg$ por F e $Partido$ por P, parte-se de $Bd = F \rightharpoonup I\!N \times (P \rightharpoonup I\!N)$ por substituição de $Inscritos$ e $Apuramento$. Tem-se, então:

   - $f_1 = F \rightharpoonup \pi_2$ por

   $$\frac{I\!N \times (P \rightharpoonup I\!N) \xrightarrow{\pi_2} (P \rightharpoonup I\!N)}{F \rightharpoonup I\!N \times (P \rightharpoonup I\!N) \xrightarrow{F \rightharpoonup \pi_2} F \rightharpoonup (P \rightharpoonup I\!N)}$$

   Logo $(a) = F \rightharpoonup (P \rightharpoonup I\!N)$.

   - $f_2 = swap \rightharpoonup I\!N \cdot unpcurry$ por $F \rightharpoonup (P \rightharpoonup I\!N) \xrightarrow{unpcurry} F \times P \rightharpoonup I\!N$ e $F \times P \xrightarrow{swap} P \times F$. Logo $(b) = P \times F \rightharpoonup I\!N$.
   - $f_3 = pcurry$ pois $P \times F \rightharpoonup I\!N \xrightarrow{pcurry} P \rightharpoonup (F \rightharpoonup I\!N)$. Logo $(c) = P \rightharpoonup (F \rightharpoonup I\!N)$.
   - $f_4 = P \rightharpoonup rsum$ onde $(F \rightharpoonup I\!N) \xrightarrow{rsum} I\!N$ calcula o total de elementos de um multi-conjunto:

   $$rsum = [\![[\underline{0}, add \cdot (\pi_2 \times id)], \alpha]\!]$$

   onde α designa o "destructor" standard de funções finitas estudado nas aulas.

2. O invariante pedido é dado por

   $$Freg \rightharpoonup (\geq \cdot (id \times rsum))$$

$\square$

Questão 4 Responda às seguintes duas alíneas:

1. Complete o diagrama que se segue por forma a nele se especificar o predicado fdp (='functional dependence'):

$$\begin{tikzcd} 2^{A\times B} \arrow[r, "\cdots"] \arrow[rrrr, bend right, "fdp"'] & A \rightharpoonup 2^B \arrow[r, "\cdots"] & A \rightharpoonup I\!N \arrow[r, "\cdots"] & 2^{I\!N} \arrow[r, "\cdots"] & 2 \end{tikzcd}$$

2. Uma tabela relacional pode ser vista como um conjunto de tuplos, sendo um tuplo uma atribuição de valores a atributos:

$$\begin{array}{rcl} Table & = & 2^{Tuple} \\ Tuple & = & AtrId \rightharpoonup Value \end{array}$$

   Baseie em fdp a especificação da função

   $$checkKey : 2^{AtrId} \longrightarrow (Table \longrightarrow 2)$$

   que testa se um dado conjunto de atributos $K \subseteq AtrId$ é chave numa determinada tabela $t \in Table$.

RESOLUÇÃO:

1. Propõe-se:

$$
\begin{tikzcd}
2^{A\times B} \arrow[r, "collect"] \arrow[rrrr, bend right, "fdp"'] & A \rightharpoonup 2^{B} \arrow[r, "A\rightharpoonup card"] & A \rightharpoonup \mathbb{N}_0 \arrow[r, "rng"] & 2^{\mathbb{N}_0} \arrow[r, "=_{\{1\}}"] & 2
\end{tikzcd}
$$

2. Vamos designar por $|_K$ a função $\lambda\sigma.(\sigma \mid K)$, isto é, $|_K = \overline{flip\,(|)}\;K$ onde $flip$ é a função, definida em `setcat.cam` que troca a ordem dos argumentos de uma função. De forma análoga se usa a designação $\backslash_K$. Tem-se então:

$$checkKey\;K \stackrel{\mathrm{def}}{=} fdp \cdot 2^{\langle |_K, \backslash_K \rangle}$$

□

Questão 5 Uma cadeia de lojas ligadas ao pequeno comércio foi informatizada há alguns anos, tendo-se perdido a documentação sobre o seu sistema de informação. No contexto do problema "Y2K", foi contratada uma empresa especialista em *especificação reversa* que verificou, por inspecção, que para cada loja se regista o 'stock' de artigos armazenados ou expostos para venda (cada artigo tem um código próprio) e se produzem diariamente mapas de vendas. Cada mapa de venda regista as facturas emitidas nas vendas de cada dia útil. Cada factura indica os artigos vendidos e em que quantidade. As facturas são numeradas e emitidas sequencialmente, sendo possível anular facturas. Apurou-se ainda que a operação do sistema se baseia em 4 tabelas relacionais T_1 a T_4, sobre as quais se sabe o seguinte:

- T_1 — Indica o conjunto das lojas que estão correntemente com 'stock' vazio.
- T_2 — É da forma $\#L \times \#A \rightharpoonup \mathbb{N}$, onde $\#L$ (identificador de loja) e $\#A$ (identificador de artigo) são tipos elementares atómicos.
- T_3 — É da forma $\#L \times \#F \rightharpoonup Data$, onde $\#F \cong \mathbb{N}$ (número de factura) e $Data$ é um tipo descritor de datas.
- T_4 — É da forma $\#L \times \#F \times \#A \rightharpoonup \mathbb{N}$ e indica quais os artigos que foram vendidos em cada factura, e em que quantidades.

Sabendo que este 'software' foi calculado a partir de uma especificação formal, conjecture o modelo de dados de que se partiu sabendo que se tratou de uma única função finita com domínio $\#L$. Apoie a sua resolução com a síntese da função de abstracção global.

RESOLUÇÃO: T_1 e T_2 têm a ver com 'stock' e T_3 e T_4 têm a ver com facturas. Logo parece sensato começar por criar dois pares à custa do iso

$$\langle\langle \pi_1, \pi_2\rangle, \langle \pi_3, \pi_4\rangle\rangle$$

Quanto a T_1 e T_2, uma loja ou tem 'stock' vazio ou não-vazio, o que sugere o recurso à abstracção $\stackrel{+}{\bowtie}$ — *cf.* (8.75) e $cojoin$ em `setcat.cam`. Mas é preciso preparar T_1 e T_2 para esse efeito. Sendo T_1 do tipo $2^{\#L}$, é preciso convertê-la em $\#L \rightharpoonup 1$ via $set2fm$ (*cf.* lei (8.64) e `setcat.cam`). Quanto a T_2, basta aplicar-lhe $pcurry$. Em suma, vamos usar sobre o par (T_1, T_2) a função

$$\stackrel{+}{\bowtie} \cdot (set2fm \times pcurry)$$

cujo resultado é do tipo $\#L \rightharpoonup (1 + (\#A \rightharpoonup \mathbb{N}))$. Este modelo ainda tem redundância pois permite representar o 'stock' vazio de duas formas distintas que se unificam facilmente à custa da aplicação de $[\underline{(\ \ )}, id]$. Em suma, a função de abstracção "do lado T_1 e T_2" é

$$(\#L \rightharpoonup [\underline{(\ \ )}, id]) \cdot \stackrel{+}{\bowtie} \cdot (set2fm \times pcurry)$$

cujo resultado é do tipo $\#L \rightharpoonup (\#A \rightharpoonup \mathbb{N})$.

Quanto a T_3 e T_4, podemos "pôr $\#L \times \#F$ em evidência" à custa de $\bowtie_n$ (*cf.* lei (8.74) e $njoin$ em `setcat.cam`) uma vez que toda a factura tem que ter uma data (embora possa não ter artigos vendidos, situação correspondente à sua anulação). Sobre o par (T_3, T_4) aplicamos então

$$\bowtie_n \cdot (id \times (nestl \rightharpoonup \mathbb{N}))$$

onde $(A \times B) \times C \xleftarrow{nestl} A \times B \times C$ é iso (*cf.* `setcat.cam`). O resultado virá do tipo $\#L \times \#F \rightharpoonup Data \times (\#A \rightharpoonup \mathbb{N})$.

Finalmente, combinamos estes dois resultados via $\bowtie_n$ de novo pois toda a loja com facturas tem de ser uma loja conhecida, com ou sem 'stock', sendo o resultado final do tipo

$$\#L \rightharpoonup (\#A \rightharpoonup \mathbb{N}) \times (\#F \rightharpoonup Data \times (\#A \rightharpoonup \mathbb{N}))$$

que é a especificação formal pedida. A função de abstracção global é, assim:

$$\bowtie_n \cdot (((\#L \rightharpoonup [\underline{(\ \ )}, id]) \cdot \stackrel{+}{\bowtie} \cdot (set2fm \times pcurry)) \times \bowtie_n \cdot (id \times nestl \rightharpoonup \mathbb{N})) \cdot \langle\langle \pi_1, \pi_2\rangle, \langle \pi_3, \pi_4\rangle\rangle$$

□
