Métodos Formais de Programação II

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

Exame (época especial) — 17 de Novembro de 1999
14h30
Sala 2312

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 Considere o modelo formal especificado, em notação CAMILA, no anexo da página 2 e seguinte. Trata-se de um sistema que especifica (de forma obviamente simplificada) o processo de alocação de alunos a cursos do ensino superior (ler `AES` como abreviatura de "Acesso ao Ensino Superior"). A acção de inicialização do sistema cria 4 licenciaturas (`Resources`) indicando o respectivo número de vagas. Cada candidato indica os cursos que pretende por ordem de preferência, e já vem identificado com um número (`CanId`) que indica, após a avaliação das suas notas, a sua qualificação numa dada escala numérica.

1. Após uma análise cuidadosa de todo o texto, explique, justificando (a) se o processo de alocação ($ALOCA$) pode correr em simultâneo com o processo de chegada das candidaturas (INS); (b) o que acontece a um aluno que indica nas suas preferências um único curso e esse curso não existe.
2. Descreva por palavras suas a semântica da função auxiliar `aloca`.
3. Calcule uma implementação relacional de `AES` sintetizando a respectiva função de abstracção.

Questão 2 Recorde a lei de refinamento entre tipos de dados indutivos T e U

$$T \xleftarrow[in_{\mathsf{F}}]{\cong} \mathsf{F}\,T \qquad U \xleftarrow[in_{\mathsf{G}}]{\cong} \mathsf{G}\,U$$

pela qual, se o padrão de recursividade do segundo implementar o padrão de recursividade do primeiro, isto é se

$$\mathsf{F}\,X \leq_f \mathsf{G}\,X$$

se verificar para todo o X, se tem então que U implementa T da forma seguinte:

$$T \leq_{(\![in_{\mathsf{F}} \cdot f]\!)} U \tag{1}$$

1. Aplique a lei (1) à situação em que $T = I\!N$ — *i.é* $in_{\mathsf{F}} = [\underline{0}, succ]$ —, em que $U = A^{\star}$ e em que $f = id + \pi_2$. Calcule a função de abstracção $l = (\![in_{\mathsf{F}} \cdot f]\!)$ ao nível da variável. Que função conhecida é l, afinal?
2. Sendo sabido que $U \cong T$ sempre que f é iso (bijectiva) mostre, partindo da alínea anterior, que
$$I\!N \cong 1^{\star}$$
3. Mostre que também se verifica
$$1 \cong 0^{\star}$$

Questão 3 Uma das alíneas da Questão 2 sugere uma representação estranha (mas válida!) para número naturais: o número n é representado pela lista $[\underbrace{NIL, \ldots, NIL}_{n \text{ vezes}}]$. Suponha agora que quer implementar sobre esse modelo as operações de soma e subtracção (parcial) de naturais.

1. Quais são as operações sobre listas que implementam estas operações? Indique-as, esboçando o respectivo diagrama de refinamento.
2. Qual o comportamento da operação que escolheu para modelar a subtracção sempre que está a subtrair a um número um outro que lhe é superior, por exemplo $2 - 3$? Justifique.

Questão 4 Considere o seguinte par de funções mutuamente recursivas, sobre $I\!N_0$, escritas em notação CAMILA:

```
f(n) = if n == 0 then 0 else g(n .- 1);

g(n) = if n == 0 then 1 else f(n .- 1) .+ g(n .- 1);
```

Pode observar-se que g não é mais que a função que calcula o n-ésimo número de Fibonacci. Pode obter-se uma versão mais eficiente (linear) de g sabendo que $g = \pi_2 \cdot w$, onde $w = \langle f, g \rangle$, e aplicando a w a Lei de Fokkinga:

$$\begin{array}{c} f \cdot in = h \cdot \mathsf{F}\,\langle f, g \rangle \\ \wedge \\ g \cdot in = k \cdot \mathsf{F}\,\langle f, g \rangle \end{array} \quad \Rightarrow \quad \langle f, g \rangle = (\!|\langle h, k \rangle|\!) \tag{2}$$

Calcule o catamorfismo w dessa forma.

Questão 5 Faça uma avaliação succinta dos métodos estudados nesta disciplina. Enumere vantagens e desvantagens dos métodos formais de programação. Indique áreas onde a sua aplicação é inevitável, outras onde é apenas aconselhável e outras ainda onde a sua utilização é dispensável.

Anexo

```
/* ------------------------- 1. Types ------------------------------------- */

TYPE

   AES :: P: Candidates
          R: Resources;

   CanId     = INT;
   Candidates = Can-seq
                inv(l) = ordenada(l)  ; /* NB: lista ordenada */
   Can :: Rank : CanId
          Prefer: Preferences;
   Preferences = ResId-seq ;
   ResId = SYM;
   Resources = ResId -> Pair;
   Pair :: First: INT
           Second: CanId-seq ;

ENDTYPE

/* ------------------------- 2. Auxiliary functions ----------------------- */

FUNC melhor(p:X,r:X):BOOL
RETURN (r <= p) && ~(r==p);

FUNC ins(e:CanId,l:Preferences,p:Candidates):Candidates
RETURN if (p==<>)
       then < Can(e,l) >
           else let (h=head(p),
                     t=tail(p))
```

```
            in if melhor(e,Rank(h))
               then cons(Can(e,l),p)
               else cons(h,ins(e,l,t));

FUNC aloca(l: ResId-seq, p: Candidates, r: Resources) : AES
RETURN if (l==<>)
           then AES(tail(p),r)
       else let (o=head(l))
                    in if cheia(o,r)
                          then aloca(tl(l),p,r)
               else let (n=First(r[o]),
                          el=Second(r[o]))
                    in AES(tail(p),
                           r + [ o -> Pair(n.-1,
                                      el ^ <Rank(head(p))>)]);

FUNC cheia(o: ResId, r: Resources):BOOL
RETURN if o in dom(r) then (0==First(r[o])) else true;

/* ------------------------- 3. Main actions ------------------------------ */

FUNC INIC():SYM
RETURN aes <- AES(<>,
                  [ 'lesi -> Pair( 8, <>),
                    'lei  -> Pair( 4, <>),
                    'lmcc -> Pair( 5, <>),
                    'lepm -> Pair( 5, <>)
                  ]);

FUNC INS(e:Candidate,l:Preferences):SYM
RETURN let (p= P(aes))
       in aes <- AES(ins(e,l,p),R(aes));

FUNC ALOCA():SYM
RETURN let (p=P(aes),
            r=R(aes))
       in aes <- if (p==<>)
                 then aes
                 else let (l=Prefer(head(p)))
                      in aloca(l,p,r);

/* ------------------------- 4. Start-up script ----------------------------- */

INIC();
INS(15, <'lesi,'lmcc,'lei>);
INS(14, <'lmcc,'lesi>);
INS(17, <'lesi>);
INS(16, <'lmcc,'lesi>);
INS(13, <'lei,'lepm>);
INS(18, <'lei,'lepm>);
INS(12, <'lei,'lmcc>);
INS(11, <'lei,'lmcc>);
INS(10, <'lei,'lmcc>);
```