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

4.º Ano da LMCC (7007N2) + LESI (5307P2)
Ano Lectivo de 1996/97

Exame (2.ª chamada) — 26 de Fevereiro 1997
9h30
Salas 2321, 2322

NB: *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 (*e.g. *lei (2.102), exercício 2.15,* etc.*).*

PROVA COM CONSULTA (3 *horas*)

Questão 1 Demonstre ou refute a seguinte igualdade, onde letras maiúsculas designam espécies de dados e minúsculas designam funções,

$$2^{(f \circ \pi_2)} \circ discollect \quad = \quad 2^{\pi_2} \circ discollect \circ (A \hookrightarrow 2^f)$$

e $discollect$ é uma função a que se refere o Exercício 2.59 abordado nas aulas práticas.

Questão 2 Recorde do Exercício 2.59 a função

$$\begin{aligned} listByIndex &: \quad A^\star \times (A \hookrightarrow B) \longrightarrow (A \times B)^\star \\ listByIndex(l, \sigma) &\stackrel{\rm def}{=} \quad <\langle a, \sigma(a)\rangle \mid a \leftarrow l \wedge a \in dom(\sigma)> \end{aligned}$$

1. Demonstre ou refute a seguinte igualdade sobre essa função, que se supõe universalmente quantificada nas variáveis livres:

$$listByIndex(l, (\quad)) \quad = \quad listByIndex(<>, \sigma)$$

(**Sugestão**: evite recorrer a métodos de prova indutiva.)

2. Suponha que

$$Students \quad \cong \quad Nr \hookrightarrow Name \times Course$$

é a estrutura de informação que regista, para cada número de aluno desta disciplina, o seu nome e o curso a que pertence (suponha $Nr \cong \mathbb{N}$, $Name \cong Str$ e $Course \cong Str$). Suponha ainda que

$$Marks \quad \cong \quad Nr \hookrightarrow Mark$$

é a estrutura de informação que regista, para cada número de aluno que entregou o exame da 1.ª chamada, a respectiva classificação (onde $Mark \cong \mathbb{N}$, por exemplo a nota 10.23 é representada pelo natural 1023).

Especifique formalmente a função

$$\begin{aligned} listStudents &: \quad Students \times Marks \longrightarrow (Nr \times Name \times Course \times Mark)^\star \\ listStudents(\sigma, \mu) &\stackrel{\rm def}{=} \quad \ldots \end{aligned}$$

que deverá gerar a pauta do exame da 1.ª chamada, ordenada por ordem crescente de números de aluno, onde cada linha da pauta indica, para cada aluno, o seu número, nome, curso e nota. (**Sugestão**: recorra a $listByIndex$ e a outras funções auxiliares que conhece para resolver esta alínea.)

3. Repita a alínea anterior por forma a $listStudents$ gerar a pauta ordenada alfabeticamente por nomes de alunos. (**Sugestão**: a mesma da alínea anterior.)

Questão 3 Recorde o Exercício 2.25 do texto de apoio desta disciplina, onde se definiu a função

$$\begin{array}{rcl} invl & : & X^\star \longrightarrow X^\star \\ invl(l) & \stackrel{\text{def}}{=} & \left\{ \begin{array}{lcl} l = \langle\rangle & \Rightarrow & \langle\rangle \\ l \neq \langle\rangle & \Rightarrow & invl(tail(l)) \frown \langle head(l) \rangle \end{array} \right. \end{array}$$

para inversão de sequências. Verifique (*e.g.* por indução) se a especificação de $invl$ satisfaz a seguinte propriedade:

$$\forall s \in X^\star : elems(invl(s)) = elems(s)$$

Questão 4 CAMILA é uma linguagem (interpretada) destinada à prototipagem rápida de especificações formais previamente formuladas em SETS. Em CAMILA é possível definir módulos (ficheiros com a extensão `.cam`) que incluem, através de uma cláusula `#include` semelhante à do C, outros módulos CAMILA. Por exemplo, um módulo `mset.cam` (para manipulação de multiconjuntos) pode ter que invocar, para poder correr, o módulo `ffun.cam` (para manipulação de funções finitas).

A 'shell' de interpretação de módulos CAMILA é um interpretador de LISP evoluído, que se designa por `xmetoo`. A interpretação de um qualquer módulo `x.cam` pressupõe a sua compilação prévia para `xmetoo`, originando o ficheiro `x.met`. Sempre que tal compilação encontra uma cláusula da forma `#include y.cam`, processa-se `y.cam`, gerando-se `y.met`, e assim sucessivamente.

É fácil de ver que a interdependência repetitiva entre módulos CAMILA torna a interpretação desnecessariamente lenta (um mesmo módulo pode ser compilado/carregado mais do que uma vez) e a sua interdependência cíclica (que, em teoria, é normal) pode ter consequências perfeitamente indesejáveis.

Pretendendo-se uma pequena aplicação que racionalize este processo de inclusão, suponha que dispõe já de uma pequena aplicação que inspecciona o sistema de ficheiros e dele infere uma tabela cuja estrutura formal é a seguinte:

$$\begin{array}{rcl} CamilaFiles & \cong & 2^{Name \times Date \times (Met + Cam)} \\ Met & \cong & 1 \\ Cam & \cong & 2^{Name} \\ Name & \cong & Str \\ Date & \cong & \mathbb{N} \end{array}$$

Esta tabela indica os nomes dos ficheiros `.met` ou `.cam` que se encontraram, a data da sua última alteração e, no caso de um ficheiro `.cam`, os nomes dos ficheiros `.cam` que aquele inclui.

1. Especifique formalmente a função,

$$\begin{array}{rcl} traInclude & : & CamilaFiles \times 2^{Name} \longrightarrow 2^{Name} \\ traInclude(\sigma, s) & \stackrel{\text{def}}{=} & \ldots \end{array}$$

que deverá calcular os nomes de todos os ficheiros `.cam` que devem ser (transitivamente) carregados quando se pretende carregar aqueles cujos nomes estão em S. (**Importante**: $traInclude$ deve conseguir tratar inclusão cíclica entre ficheiros.)

2. Sempre que a data de um ficheiro `.met` é posterior a data do correspondente `.cam`, torna-se desnecessária a compilação prévia deste. Assumindo disponível a função $traInclude$, especifique uma função capaz de separar os módulos que têm de ser compilados e carregados daqueles para os quais basta carregar o correspodente código `.met`, por este estar actualizado.

3. Suponha que em lugar da estrutura de informação $CamilaFiles$ que acima se caracterizou, alguém propõe a seguinte estrutura formal para registar a mesma informação:

$$CamilaFiles' \cong 2^{Date \times Name} \times 2^{Name \times Date \times 2^{Name}}$$

Será $CamilaFiles'$ isomorfa a $CamilaFiles$? Justifique formalmente.

Questão 5 Discuta formalmente a validade da seguinte afirmação:

Existe pelo menos uma relação $R \subseteq P \times P$ tal que o seu fecho simétrico, dado pela equação $x = R \cup x^{-1}$, coincide com o seu fecho transitivo, dado pela equação $y = R \cup R \circ y$.

Questão 6 Considere a seguinte gramática de contexto livre, bem conhecida, para expressões aritméticas simples:

$$\begin{array}{rcl} G & = & \langle NT, T, \langle E \rangle, P \rangle \\ NT & = & \{\langle E \rangle\} \\ T & = & \{\, \texttt{+}\,,\, \texttt{*}\,,\, \texttt{(}\,,\, \texttt{)}\,,\, \texttt{id}\,,\, \texttt{num} \,\} \\ P & = & \left\{\ \langle E \rangle \quad ::= \quad \langle E \rangle \ \texttt{+}\ \langle E \rangle \mid \langle E \rangle \ \texttt{*}\ \langle E \rangle \mid \ \texttt{(}\ \langle E \rangle\ \texttt{)}\ \mid \texttt{id} \mid \texttt{num} \quad \right\} \end{array}$$

Recorde do exame da 1.ª chamada o método aí proposto para representação de uma gramática independente de contexto $G = \langle N, T, S, P \rangle$, expressa em BNF, num sistema de equações em SETS: (1) começa-se por eliminar todas as ocorrências de símbolos terminais nas produções em P; (2) a cada símbolo não-terminal $\langle A \rangle$ em N corresponde a variável A em SETS; (3) usa-se a seguinte tabela de conversão de BNF em SETS:

BNF		SETS
$\alpha \mid \beta$	$\mapsto$	$\alpha + \beta$
$\alpha\ \beta$	$\mapsto$	$\alpha \times \beta$
α^*	$\mapsto$	α^*
α^+	$\mapsto$	$\alpha \times \alpha^*$
ϵ	$\mapsto$	1

Recorrendo a este método e ao cálculo de isomorfismos de SETS, mostre que a gramática acima é convertível numa equação polinomial do 2.º grau em SETS. Finalmente, enuncie o método de indução polinomial que está associado à equação que obteve.
