Métodos Formais em Engenharia de Software

1.º Ano de Mestrado de Informática da Universidade do Minho
Ano Lectivo de 2007/08

Prova de avaliação — Teórica — 21 de Fevereiro 2008
09h30
Sala 1.08

NB*: Esta prova consta de **8** alíneas todas com as mesma cotação. Responda a grupos diferentes em folhas diferentes.*

PROVA COM CONSULTA (2 horas)

GRUPO I

Questão 1 A ideia de que (a) a função identidade preserva qualquer invariante e de que (b) a ausência de invariante no tipo de saída de uma função trivializa a respectiva obrigação de prova é captada pelas equivalências

$$\Phi \xleftarrow{id} \Phi \quad \equiv \quad \text{TRUE} \quad \equiv \quad id \xleftarrow{f} \Phi \tag{1}$$

Prove-as recorrendo ao cálculo relacional *pointfree*.

Questão 2 O produto $R \times S$ de relações (funções) é útil para especificar (programar) computações independentes que "correm paralela e separadamente". A soma $R + S$ especifica computações alternativas.

1. Sendo p e q dois invariantes, e designando por Φ_p (resp. Φ_q) a coreflexiva associada a p (resp. q), introduza variáveis na expressão $\Phi_p \times \Phi_q$ e simplifique. Faça o mesmo para $\Phi_p + \Phi_q$.
2. Dada a independência dos dois factores de um produto relacional, não custa aceitar que a preservação de invariantes admita a lei de separação

$$\Phi' \times \Psi' \xleftarrow{f \times g} \Phi \times \Psi \quad \equiv \quad \Phi' \xleftarrow{f} \Phi \ \wedge \ \Psi' \xleftarrow{g} \Psi \tag{2}$$

Demonstre (2) sabendo que a lei **functor**-$\times$

$$(g \cdot h) \times (i \cdot j) \quad = \quad (g \times i) \cdot (h \times j) \tag{3}$$

generaliza a quaisquer relações e que

$$R \times S \subseteq U \times V \quad \equiv \quad R \subseteq U \ \wedge \ S \subseteq V \tag{4}$$

se verifica também.

Questão 3 Uma outra lei de separação de obrigações de prova é (para funções) a que se segue:

$$\Psi \times \Upsilon \xleftarrow{\langle f,g\rangle} \Phi \quad \equiv \quad \Psi \xleftarrow{f} \Phi \;\wedge\; \Upsilon \xleftarrow{g} \Phi \tag{5}$$

Sabendo que a lei de **absorção-**$\times$

$$(f \times g) \cdot \langle k, h \rangle = \langle f \cdot k, g \cdot h \rangle \tag{6}$$

generaliza a quaisquer quatro relações, apresente justificações detalhadas para os passos do seguinte cálculo de (5):

$$
\begin{array}{ll}
 & \Psi \times \Upsilon \xleftarrow{\langle f,g\rangle} \Phi \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \langle f,g\rangle \cdot \Phi \subseteq (\Psi \times \Upsilon) \cdot \langle f,g\rangle \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \langle f,g\rangle \cdot \Phi \subseteq \langle \Psi \cdot f, \Upsilon \cdot g\rangle \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \pi_1 \cdot \langle f,g\rangle \cdot \Phi \subseteq \Psi \cdot f \;\wedge\; \pi_2 \cdot \langle f,g\rangle \cdot \Phi \subseteq \Upsilon \cdot g \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & f \cdot \Phi \subseteq \Psi \cdot f \;\wedge\; g \cdot \Phi \subseteq \Upsilon \cdot g \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \Psi \xleftarrow{f} \Phi \;\wedge\; \Upsilon \xleftarrow{g} \Phi
\end{array}
$$

GRUPO II

Questão 4 Atente no anexo desta prova em que é dado um fragmento da especificação funcional da operação que apaga um ficheiro ou directoria de um sistema de ficheiros de acordo com os requisitos que constam do *Intel ® Flash File System Core Reference Guide* (versão 1, Out. 2004).

Repare que o sistema de ficheiros é modelado por um par de 'mappings', um (de tipo `OpenFileDescriptorTable`) que regista os ficheiros que estão abertos para operações de leitura ou escrita e outro (de tipo `Tar`) que regista o conteúdo de cada ficheiro dado o seu *'path'*.

A função `dirName` (omitida no fragmento apresentado) indica, de acordo com os requisitos, a directoria a que pertence um dado ficheiro ou directoria.

1. Analise <u>cuidadosamente</u> os invariantes envolvidos nos tipos `System` e `Tar` por forma a completar as pre-condições que se encontram (intencionalmente) omitidas.
2. Especifique a função auxiliar `dirName : Path -> Path` sabendo que os requisitos impoem que a directoria da <Root> seja ela própria.

GRUPO III

Questão 5 O teste orientado ao modelo (MBT = 'model based testing') foi assunto de várias aulas desta UCE. Responda às seguintes questões de forma personalizada (isto é, não se limite a *copiar e colar* frases do material pedagógico que tem para consulta):

1. Quais são os principais obstáculos a ultrapassar quando se pretende implementar um processo de ”Model-Based Testing”? Justifique.
2. Que técnicas poderiam ajudar a ultrapassar esses obstáculos? Justifique.

Anexo — Fragmento de Modelo de um Sistema de Ficheiros

Modelo de dados do sistema de ficheiros:

```
-------------------------------------------------------------------------
System :: table : OpenFileDescriptorTable
          tar   : Tar
        inv sys ==
            forall openfiledescriptor in set rng sys.table &
                   openfiledescriptor.path in set dom sys.tar;
-------------------------------------------------------------------------
OpenFileDescriptorTable =
        map FileHandler to OpenFileDescriptor;
FileHandler = nat;
OpenFileDescriptor :: path     : Path;
-------------------------------------------------------------------------
Tar = map Path to File
        inv tar ==
         forall path in set dom tar &
                dirName(path) in set dom tar and
                tar(dirName(path)).attributes.fileType = <Directory>;
File :: attributes : Attributes
        contents   : [FileContents]
        inv file ==
        (file.attributes.fileType = <Directory> and file.contents = nil) or
        (file.attributes.fileType = <RegularFile> and file.contents <> nil);
Attributes :: fileType : FileType;
FileType = <RegularFile> | <Directory>;
-------------------------------------------------------------------------
Path = <Root> | seq1 of FileName;
FileName = seq1 of char;
FileContents = seq of char;
-------------------------------------------------------------------------
```

NB: todos os tipos de dados não declarados devem ser considerados `token`.

Operação de remoção de ficheiro ou directoria:

```
FS_DeleteFileDir : System * Path -> System * FFS_Status
FS_DeleteFileDir(sys, full_path) ==
        if full_path <> <Root> and
           full_path in set dom sys.tar and
           pre_FS_DeleteFileDir_System(sys, full_path)
        then mk_(FS_DeleteFileDir_System(sys, full_path), <FFS_StatusSuccess>)
        else mk_(sys, FS_DeleteFileDir_Exception(sys, full_path));
-------------------------------------------------------------------------
FS_DeleteFileDir_System : System * Path -> System
FS_DeleteFileDir_System(sys, full_path) ==
        mu(sys, tar |-> FS_DeleteFileDir_Tar(sys.tar, full_path))
pre     ..............................................................
        ......................................;
-------------------------------------------------------------------------
FS_DeleteFileDir_Tar : Tar * Path -> Tar
FS_DeleteFileDir_Tar(tar, full_path) ==
```

```
        {full_path} <-: tar
pre     ...................................................;
--------------------------------------------------------------------------------
FS_DeleteFileDir_Exception : System * Path -> FFS_Status
FS_DeleteFileDir_Exception(sys, full_path) ==
        if full_path = <Root>
        then <FS_ErrorInvalidPath>
        elseif full_path not in set dom sys.tar
        then <FS_ErrorPathNotFound>
        elseif exists buffer in set rng sys.table & buffer.path = full_path
        then <FS_ErrorFileStillOpen>
        elseif sys.tar(full_path).attributes.fileType = <Directory> or
                exists path in set dom sys.tar & full_path = dirName(path)
        then <FS_ErrorDirectoryNotEmpty>
        else <FFS_StatusSuccess>;
--------------------------------------------------------------------------------
```