Especificação e Desenvolvimento Formal de 'Software'

Mestrado em Informática + Curso de Especialização em Informática
Ano Lectivo de 2001/02

Exame (Época normal) — 23 de Fevereiro de 2002
10h00
Sala do Mestrado

NB: *Esta prova consta de **5** questões de **4 valores** cada uma.*

PROVA COM CONSULTA (2 horas)

Questão 1 A possibilidade de uma função dar resultado indefinido em VDM-SL conduz a uma lógica de funções parciais estendida com o valor lógico *indefinido* ($\star$). Neste contexto, complete a tabela de verdade do predicado $A \Rightarrow \neg B$ que se segue:

A	B	$A \Rightarrow \neg B$
$true$	$true$	
$true$	$false$	
$true$	$\star$	
$false$	$true$	
$false$	$false$	
$false$	$\star$	
$\star$	$true$	
$\star$	$false$	
$\star$	$\star$	

Questão 2 Recorde os seguintes modelos abstractos para "mapas", abordados nas aulas desta disciplina:

Modelo (a)

```
Map = set of Path;
Path :: From: Location
        To: Location;
Location = token;
```

Modelo (b)

```
UMap = set of UPath;
UPath :: From: Location
         Info: PathInfo
         To: Location;
PathInfo :: name: token
            distance: real
            speed: real;
Location = token;
```

As seguintes duas funções `f` e `g` foram definidas no contexto do **Modelo (a)**:

```
f : Map * Map -> Map
f(m1,m2) == { mk_Path(x.From,y.To) | x in set m1, y in set m2 & x.To = y.From };

g : Map -> Map
g(m) == let c = f(m,m) in if m = c then m else m union g(c);
```

Explique por palavras suas o significado destas funções e adapte-as ao **Modelo (b)**.
NB: tome a liberdade de re-especificar tipos `token`, se necessário.

Questão 3 O registo de chamadas não atendidas em telemóveis é vulgarmente organizado de forma a que os últimos números que chamaram estejam imediatamente acessíveis. Essa estrutura corresponde à de um "stack" cuja operação de "push" se redefine da forma seguinte:

```
npush : int * seq of int -> seq of int
-- push int or move it to the front
npush(n, s) == [n] ^ [ s(i) | i in set inds s & n <> s(i) ] ;
```

Contudo, um construtor de "software" para telemóveis rejeitou este modelo dizendo que apenas queria que ficassem registadas as últimas 10 chamadas. Perante este novo requisito, foram apresentadas as seguintes 3 propostas de revisão de `npush`:

1.ª proposta:

```
npush(n, s) == [n] ^ [ s(i) | i in set inds s & n <> s(i) and len s <= 10 ] ;
```

2.ª proposta:

```
npush(n, s) == [n | n in set {1,...,10}] ^ [ s(i) | i in set inds s & n <> s(i) ] ;
```

3.ª proposta:

```
npush(n, s) == [n] ^ [ s(i) | i in set inds s & n <> s(i) ]
pre len s <= 10;
```

Mostre (*e.g.* através de contra-exemplos) que todas estas três propostas são inadequadas, explicando brevemente por que é que "não funcionam". Termine formulando a sua própria proposta de revisão de `npush`.

Questão 4 Uma das formas de se calcular n^2, o quadrado de um número natural n, é somar os n primeiros ímpares — *cf.* $1^2 = 1$, $2^2 = 1 + 3$, $3^2 = 1 + 3 + 5$, — e, no caso geral, $n^2 = (2n - 1) + (n - 1)^2$. Em VDM-SL alguém escreveu:

```
sq : nat -> nat
sq(n) == setsum(nthodds(n));
```

Complete as seguintes definições em VDM-SL das funções `setsum` (= "somar conjunto de naturais") e `nfodds` (= "os n primeiros ímpares") a que `sq` (= "quadrado") recorre:

```
setsum : set of nat -> nat
setsum(s) ==
        cases s:
          {}     -> 0,
          others -> let e in set s
                    in ....... setsum(.........)
        end;

nthodds : nat -> set of nat
nthodds(n) == ..............................;
```

Questão 5 A função

```
invert : WWW -> map Word to set of Ref
invert(www) == transpose({ r |-> getWords(www(r)) | r in set dom www })
```

— que invoca como auxiliares as funções

```
transpose(m) ==
        { w |-> { r | r in set dom m & w in set m(r) } | w in set dunion rng m };

getWords(url) ==
         dunion { let u = url(i)
                  in cases u:
                        mk_HyperLink(-,t) -> elems t,
                        others            -> elems u
                     end
                  | i in set inds url };
```

— é proposta como solução para a última alínea da questão dada em anexo (página 3). Identifique os tipos de entrada e saída de `getWords` e `transpose`, e calcule o resultado de `invert(www)` para

```
www = { "r1" |-> [ ["b","c"], mk_HyperLink("r2",["a"]) ],
        "r2" |-> [ mk_HyperLink("r2",["a"]) ]
      };
```

Anexo — A Naive Model of the WWW

Consider the following VDM-SL model specifying, at abstract level, the structure of an information system based on the 'World Wide Web' over the INTERNET, where `Ref` (page address) is a datatype of which no further details are required:

```
WWW  = map Ref to URL;           -- (URL=Universal Resource Location)
URL  = seq of Unit;
Unit = PlainText | HyperLink;
PlainText = seq of Word;
Word = seq of char;
HyperLink :: link: Ref
             txt: PlainText;     -- "underlined text"
Ref = seq of char;
```

1. Add an invariant to `WWW` ensuring that no URL mentions a non-existing URL (NB: although this cannot be ensured for the WWW as a whole, it makes sense for the particular fragment which embodies our information system.)
2. Assuming

   ```
   wc : seq of char -> nat
   -- counts the number of words in a string
   ```

   specify

   ```
   nrofwd : URL -> nat
   -- counts the number of words in a URL
   ```
3. Specify

   ```
   refsTo : Ref * WWW -> set of Ref
   -- retrieves the addresses of all URLs which point at Ref
   ```
4. Search engines on the WWW are based on text inversion, that is, on a function

   ```
   invert : WWW -> map Word to set of Ref
   ```

   which computes, for every word, the URLs which mention it. Specify `invert`. (NB: this inversion operation is far more elaborate in practice!)