Especificação e Desenvolvimento Formal de 'Software'

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

Exame (Época de recurso) — 14 de Setembro de 2002
9h00
Sala do Mestrado

NB: *Esta prova consta de **7** alíneas todas com a mesma cotação.*

PROVA COM CONSULTA (2 horas)

Questão 1 Um dos benefícios da especificação formal de software é a fácil identificação de padrões de projecto ("design patterns") a partir de modelos formais, conduzindo a soluções genéricas "costumizáveis" em diferentes domínios semânticos. Por exemplo, o problema do sistema de vigilância de uma fábrica química (`alarm.vdm`), que foi analisado com detalhe ao longo da disciplina e é dado em anexo, é generalizável ao de "sistema crítico" baseado em recursos humanos especializados.

Pretende-se nesta questão realizar o exercício de transporte do modelo `alarm.vdm` para o domínio semântico do *serviço de urgência* de uma unidade hospitalar.

Neste contexto, faça um paralelo entre os tipos de dados e as funções existentes no modelo e as entidades correspondentes no domínio hospitalar. Justifique as suas opções e identifique eventuais necessidades de extensão do modelo, tanto a nível de tipos de dados como de operações.

Questão 2 Atente na seguinte especificação do modelo de dados de um sistema de gestão de contas bancárias (muito!) simplificado:

```
BAMS = map AccId to Account;
Account :: H: set of AccHolder
           B: Amount;

AccId     = seq of char;
AccHolder = seq of char;
Amount = int;
```

1. Para cada uma das descrições informais que se seguem escreva a expressão/definição formal em VDM-SL, onde `bams` é um valor de `BAMS`:
   (a) O conjunto dos titulares das contas que em `bams` têm saldo inferior a 10 (euro).
   (b) A função finita (`map`) que associa a cada número de conta presente em `bams` o respectivo saldo.
2. Qual das seguintes duas alternativas para a especificação da operação de abertura de uma conta *nova*,

   ```
   OpenAccount : BAMS * AccId * set of AccHolder * Amount -> BAMS
   OpenAccount(bams,n,h,m) == bams munion { n |-> mk_Account(h,m) }
   ```

   ou

   ```
   OpenAccount : BAMS * AccId * set of AccHolder * Amount -> BAMS
   OpenAccount(bams,n,h,m) == { n |-> mk_Account(h,m) } ++ bams ;
   ```

   escolheria? Justifique a sua resposta sem omitir a comparação do comportamento esperado de cada uma das alternativas apresentadas para a situação em que se tenta abrir uma conta que já existe.
3. A seguinte especificação da operação de crédito numa conta,

   ```
   Credit: BAMS * AccId * Amount -> BAMS
   Credit(bams,n,m) ==
        selUp({n}, credit(m), bams);
   -- pre n in set dom bams;
   ```

   baseia-se numa função auxiliar

   ```
   credit : Amount -> Account -> Account
   credit(m)(x) == mk_Account(x.H, x.B + m)
   ```

que é usada pelo operador de "actualização selectiva" de um conjunto de contas,

```
selUp: set of AccId * (Account -> Account) * map AccId to Account -> map AccId to Account
selUp(s,f,x) == x ++ ffmap(f)(s <: x);
```

que por sua vez se baseia no operador genérico

```
ffmap: (Account -> Account) -> map AccId to Account -> map AccId to Account
ffmap(f)(x) == { k |-> f(x(k)) | k in set dom x };
```

Note que a pre-condição de `Credit` foi posta em comentário. Qual o comportamento de `Credit` para a situação em que tal predicado se não verifica? Justifique.

4. Adapte a definição de `Credit` por forma a passar a especificar a operação de débito de uma quantia, não sendo permitidos quaisquer débitos que conduzam a saldos negativos.
5. Teste a validade da seguinte igualdade

```
selUp({},f,x) = selUp(s,f,{|->})
```

calculando o resultado de ambos os seus membros para `s` e `x` arbitrários.

Questão 3 As companhias de um vasto contingente militar em missões de alto risco podem estar em dois estados possíveis: operacional ou repouso. Logo que uma companhia prefaz determinado tempo de actividade entra, para recuperar do desgaste, em repouso. Sempre que as operações militares assim o exijam, a companhia há mais tempo em repouso regressa de imediato à actividade operacional.

Indique que tipo de modelação (`set of`..., `seq of`..., `map` ... `to` ..., ou outra) escolheria para completar as reticências no respectivo modelo formal

```
Contingente :: operacional: ....
               repouso: ....
```

e sobre ela especifique a operação de *regresso* de uma companhia à actividade operacional. Inclua eventuais tipos elementares (`token`) de que necessite e acrescente um invariante ao seu modelo de dados, se necessário.

Anexo

```
--
-- Chemical plant model
-- Source: adapted from "Modelling Systems" by JNO
-- File: -rw-------    1 jno      jno           899 Jan 12 14:16 alarm.vdm
--

types

Plant :: sch    : Schedule
         alarms : set of Alarm ;

Schedule = map Period to set of Expert
inv sch == forall exs in set rng sch &
                  exs <> {} and
                  forall ex1,ex2 in set exs &
                         ex1.expertId = ex2.expertId => ex1 = ex2 ;

Alarm :: alarmtext : seq of char
         quali     : Qualification ;

Qualification = <Elec> | <Mech> | <Bio> | <Chem> ;

Expert :: expertId : ExpertId
          quali    : set of Qualification
inv ex == ex.quali <> {};

ExpertId = token ;

Period = token ;
```

```
functions

NumberOfExperts: Period * Plant -> nat
NumberOfExperts(per,pl) == card pl.sch(per)
pre per in set dom pl.sch ;

ExpertIsOnDuty: Expert * Plant -> set of Period
ExpertIsOnDuty(ex,pl) ==
{per | per in set dom pl.sch &
       ex in set pl.sch(per)} ;

ExpertToPage(al: Alarm, per: Period, pl: Plant) r: Expert
pre per in set dom pl.sch and
    al in set pl.alarms
post r in set pl.sch(per) and
     al.quali in set r.quali ;

-- end of alarm.vdm
```