Métodos Formais em Engenharia de Software
Teste, 14 de Janeiro de 2022

Número: _____ Nome: _____

1. SAT *(6 pontos)*

Uma loja de electrónica permite aos seus clientes personalizar o seu computador, escolhendo entre dois modelos de CPU, dois modelos de placa gráfica, dois modelos de memória RAM, dois modelos de *motherboards* e dois modelos de monitor. Cada computador tem que ter obrigatoriamente uma única *motherboard*, um único CPU, uma única placa gráfica e uma única memória RAM. O computador poderá ter ou não ter monitor. A personalização do computador deverá obedecer às seguintes regras:

- A *motherboard* MB1 quando combinada com a placa gráfica PG1, obriga à utilização da RAM1.
- A placa gráfica PG1 precisa do CPU1, excepto quando combinada com uma memória RAM2.
- O CPU2 só pode ser instalado na *motherboard* MB2.
- O monitor MON1 para poder funcionar precisa da placa gráfica PG1 e na memória RAM2.

a) Por forma a codificar este problema como um problema SAT, defina um conjunto adequado de variáveis proposicionais, e escreva um conjunto de fórmulas proposicionais adequado à sua modelação.
b) Indique o que deveria ser o conteúdo de um ficheiro com a codificação em formato DIMACS CNF, para comprovar que este conjunto de restrições é consistente, e qual a resposta que o *SAT solver* deveria dar.
c) Indique, justificando a sua resposta, como faria para com o *SAT solver* confirmar que:
   1. O monitor MON1 só poderá ser usado com uma *motherboard* MB1.
   2. Um cliente pode personalizar o seu computador da seguinte forma:
      uma *motherboard* MB2 , o CPU1, a placa gráfica PG2 e a memória RAM1.

Métodos Formais em Engenharia de Software
Teste, 14 de Janeiro de 2022

Número: _____ Nome: _____

2. Modelação estrutural com Alloy *(4 pontos)*

Especifique as seguintes propriedades sobre este modelo de um sistema de informação de uma universidade.

```
sig UC {
      precedencias : set UC,
      inscritos : set Aluno
}
sig Curso {
      plano : some UC,
      inscritos : set Aluno
}
sig Aluno {}
```

a) Todas as UCs têm que pertencer ao plano de algum curso.
b) Os alunos inscritos num curso devem estar inscritos nalguma das suas UCs.
c) Se uma UC tem precedências então estas devem fazer parte do plano de todos os cursos aos quais ela pertence.
d) Um aluno inscrito numa UC não pode estar inscrito nalguma das suas precedentes (directas ou indirectas).

Métodos Formais em Engenharia de Software
Teste, 14 de Janeiro de 2022

Número: _____ Nome: _____

3. Modelação comportamental com Alloy *(4 pontos)*

Considere o seguinte modelo, onde o **fact** caminho garante que a **sig** Actual faz um caminho válido no grafo (ou seja, só se desloca para nós adjacentes).

```
sig Node { adj : set Node }
var one sig Actual in Node {}
pred stutter { Actual' = Actual }
pred avanca  { some n : Actual.adj | Actual' = n }
fact caminho { always (stutter or avanca) }
run hamiltoniano { ... }
```

a) Complete a definição do **run** hamiltoniano por forma a garantir que o caminho percorrido pela **sig** Actual seja *Hamiltoniano*, ou seja, visita todos os nós mas não mais do que uma vez (note que o caminho não tem que ser um ciclo).
b) Modifique a definição dos eventos stutter e avanca por forma a que qualquer comando **run** gere sempre caminhos Hamiltonianos. Pode declarar novas assinaturas ou relações se desejar.

Métodos Formais em Engenharia de Software
Teste, 14 de Janeiro de 2022

Número: _____ Nome: _____

4. Frama-C *(6 pontos)*

Recorde a definição em C do algoritmo de Warshall, abordado no exercício para casa:

```
#define MAXVERTICES 10
typedef int Graph[MAXVERTICES][MAXVERTICES];

void WarshallTC (Graph A, Graph R, int n) {
    int i, j, k;

    for (i=0 ; i<n ; i++)
        for (j=0 ; j<n ; j++)
            R[i][j] = A[i][j];

    for (k=0 ; k<n; k++)
        for (i=0 ; i<n ; i++)
            for (j=0 ; j<n ; j++)
                if (R[i][k] && R[k][j])
                    R[i][j] = 1;
}
```

a) Apresente a anotação completa desta função (incluindo pré-condições) com todos os elementos necessários para provar:

- a sua *segurança de execução* (operações de memória e aritméticas), terminação, e *frame condition* adequada;

- a sua correcção face ao seguinte contrato:

```
@ requires \forall integer a, b; 0 <= a < n && 0 <= b < n ==>
@           A[a][b]==0 || A[a][b]==1 ;
@ ensures  \forall integer a, b; 0 <= a < n && 0 <= b < n ==>
@           R[a][b]==0 || R[a][b]==1 ;
```

Métodos Formais em Engenharia de Software
Teste, 14 de Janeiro de 2022

Número: _____ Nome: _____

b) Considere agora o comportamento funcional do algoritmo. Pretende-se provar que ele satisfaz a seguinte pós-condição:

```
@ ensures \forall integer a, b ; 0 <= a < n && 0 <= b < n ==>
@           A[a][b] <==> reachable{Here}((Graph) R, a, b, n) ;
```

Para isso, comece por completar a seguinte axiomatização, escrevendo axiomas ou uma definição para o predicado `reachable`, e outros elementos que entenda necessários:

```
/*@ axiomatic GraphTClosure {
  @ predicate reachable{L}(Graph g, integer a, integer b, integer V) ;
  @
  @ ...
  @ }
  @*/
```

Anote depois o programa com invariantes de ciclo adequados para provar a pós-condição.

Dica: Utilize o seguinte invariante para o ciclo `for (k=0 ; k<n; k++)`.

> *No início de cada iteração, `R[a][b] == 1` sse existir em `A` um caminho de `a` para `b` que passe apenas pelos vértices do conjunto `{0, …, k-1}`.*