

## SMT solving - Questão para avaliação

- A resposta ao exercício deverá ser feita num notebook Colab.
- A primeira secção do notebook deverá conter o **Número**, **Nome** e **Curso** do aluno.
- A resposta deverá ser entregue até **7 de Novembro**, via Blackboard, onde deverá submeter o link para o notebook Colab, assim como uma cópia PDF do notebook.
- Não se esqueça de dar permissões de visualização para todos os utilizadores que tenham acesso ao link.

Responda apenas a uma das seguintes secções (**Futoshiki Puzzle** ou **Verificação de Software**).

### 1 Futoshiki Puzzle

Futoshiki é um puzzle lógico japonês jogado num tabuleiro  $N \times N$ , onde são assinaladas restrições de desigualdade entre algumas posições contíguas do tabuleiro.

O objetivo é colocar os números  $1..N$  de forma a que cada número não apareça repetido em cada linha nem em cada coluna do tabuleiro, e que as relações de desigualdade assinaladas sejam respeitadas. Alguns números podem estar fixos no tabuleiro inicial. Pode ver mais informações sobre o puzzle em

- <http://en.wikipedia.org/wiki/Futoshiki>
- <http://www.brainbashers.com/futoshiki.asp>

Desenvolva um programa em Python para resolver este jogo como auxílio de um SMT solver.

- **Input:** a configuração do tabuleiro inicial deverá ser fornecida num ficheiro de texto, em formato que entendam adequado para o descrever. Como alternativa, pode receber o texto do tabuleiro diretamente numa string.
- **Output:** a solução do puzzle deverá ser impressa no ecrã.

### 2 Verificação de Software

#### 2.1 Codificação lógica de um programa

Considere o seguinte programa C sobre inteiros.

```
z = 0;
x = x + y;
if (y >= 0) {
    y = x - y;
    x = x - y;
}
```

```

else {
    z = x - y;
    x = y;
    y = 0;
}
z = x + y + z;

```

1. Faça a codificação lógica deste programa. Recorde que será útil escrever primeiro uma versão *single-assignment* do programa.
2. Tendo por base a codificação lógica que fez do programa, utilize o API do Z3 para Python para se pronunciar quanto à veracidade das seguintes afirmações. Justifique a sua resposta. No caso da afirmação ser falsa, apresente o contra-exemplo indicado pelo solver.
  - (a) “Se o valor inicial de  $y$  for positivo, o programa faz a troca dos valores de  $x$  e  $y$  entre si.”
  - (b) “O valor final de  $y$  nunca é negativo.”
  - (c) “O valor final de  $z$  corresponde à soma dos valores de entrada de  $x$  e  $y$ .”

## 2.2 Verificação dedutiva de SW

Considere o seguinte programa anotado (com pré-condição, pós-condição e invariante de ciclo) que calcula o máximo de um array de inteiros.

```

PRE:  $n \geq 1 \wedge i = 1 \wedge m = A[0]$ 
while (i < n)
    INV:  $i \leq n \wedge \forall j. 0 \leq j < i \rightarrow m \geq A[j]$ 
    if (A[i] > m)
        m = A[i];
    i = i+1;
POS:  $\forall j. 0 \leq j < n \rightarrow m \geq A[j]$ 

```

Para verificar que o programa satisfaz a especificação são geradas as seguintes condições de verificação.

- **Inicialização:**  $PRE \rightarrow INV$
- **Preservação:**  
 $(i < n \wedge INV) \rightarrow (A[i] > m \rightarrow INV[(i+1)/i][A[i]/m]) \wedge (A[i] \leq m \rightarrow INV[(i+1)/i])$
- **Utilidade:**  $(INV \wedge i \geq n) \rightarrow POS$

Utilize o API do Z3 para Python para verificar a validade das condições de verificação geradas.

Notas: Utilize a teoria de arrays do Z3, descrita nos “Advanced Topics” manual. Pode utilizar a função `substitute` do z3 que permite substituir sub-expressões. `substitute(a, (b,c))` tem o efeito de substituir em `a` todas as ocorrências de `b` por `c`.