SMT Solving - TPC2

Ana Luísa Lira Tomé Carneiro - PG46983 - Mestrado em Engenharia Informática (MEI)

Futoshiki Puzzle

Para o programa é necessário a instalação e importe da API z3 do Python

In [ ]:
```
!pip install z3-solver
```

```
Collecting z3-solver
  Downloading z3_solver-4.8.12.0-py2.py3-none-manylinux1_x86_64.whl (33.0 MB)
     |████████████████████████████████| 33.0 MB 18 kB/s
Installing collected packages: z3-solver
Successfully installed z3-solver-4.8.12.0
```

In [ ]:
```
from z3 import *
```

Ficheiro de Input

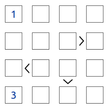
Para a realização deste exercício é necessário a criação de um ficheiro que contém informação pertinente sobre o puzzle futoshiki que se pertende solucionar. Este ficheiro de input será lido pelo o programa e contém a seguinte informação:

- valores fixos do puzzle;
- tamanho da tabuleiro de jogo;
- restrições de desigualdade.

O ficheiro de input *futoshikiPuzzle.txt* a ser lido pelo sistema que representa um puzzle futoshiki apresenta o seguinte formato:

```
values
1 0 0 0
0 0 0 0
0 0 0 0
3 0 0 0
restrict
1,2 1,3
2,1 2,0
2,2 3,2
```

Na secção *values* podemos ver um tabuleiro de jogo 4x4 onde as células com valor 0 estão por preencher e as células (0,0) e (0,3) tem números que não podem ser alterados. Na secção *restrict* também podemos ver as várias restrições de desigualdade, por exemplo a célula (1,2) terá um número maior que o da célula (1,3). Na figura abaixo podemos ver o puzzle representativo do ficheiro acima.



Criação do Ficheiro de input

Para termos acesso ao ficheiro apresentado acima foi necessário criá-lo através da função *createFile*. A função podia ser removida se o ficheiro já tivesse sido criado na diretoria do programa, contudo para o programa funcionar na plataforma colab foi necessário a criação desta função.

In [ ]:
```
def createFile():

    f = open("futoshikiPuzzle.txt", "w")
    f.write("values\n1 0 0 0\n0 0 0 0\n0 0 0 0\n3 0 0 0\n")
    f.write("restrict\n1,2 1,3\n2,1 2,0\n2,2 3,2")
    f.close()

print("Creating File...")
createFile()
print("DONE!")
```

```
Creating File...
DONE!
```

Leitura do Ficheiro de Input

Abaixo encontra-se o código de *parsing* do ficheiro para posteriormente ser utilizado pelo programa para solucionar o puzzle em questão. Na função *parseFile* à medida que se lê o ficheiro de input vamos armazenar a sua informação em duas matrizes: **data** e **rules**. A primeira matriz contém a informação sobre o tabuleiro de jogo (*values*) e a segunda matriz contém a informação sobre as restrições de desigualdade (*restrict*). No final, a função devolve o tamanho do tabuleiro.

In [ ]:
```
def parseFile(data, rules):

    f = open("futoshikiPuzzle.txt","r")
    tag = 0
    n = 0

    for content in f.readlines():
        content = content.rstrip("\n")

        if content == "values":
            tag = 1
        elif content == "restrict":
            tag = 2
        elif tag == 1:

            content = content.rstrip("\n")
            values = list(content.split(" "))
            n = len(values)

            for i in range(0, len(values)):
                values[i] = int(values[i])

            data.append(values)

        elif tag == 2:

            content = content.rstrip("\n")
            res = list(content.split(" "))
            line = []

            for i in range(0, len(res)):

                restrict = tuple(map(int, res[i].split(",")))
                line.append(restrict)

            rules.append(line)
    return n


#DATA - Parse File
print("Parsing File...")
data = []
rules = []
n = parseFile(data, rules)
print("DONE!")
```

```
Parsing File...
DONE!
```

Restrições do SMT-Solver

Para se conseguir solucionar o puzzle é necessário cumprir com as seguintes regras (N - tamanho do tabuleiro):

- Em cada célula só pode haver um número entre 1 e N
- Em cada linha só pode haver um único número entre 1 e N
- Em cada coluna só pode haver um único número entre 1 e N
- Existem células cujos os números não podem ser alterados
- Restrições de desigualdade presentes no ficheiro de input

Abaixo encontra-se o código com as diversas restrições adicionadas ao SMT-Solver. Numa primeira fase temos de criar todas as variáveis que identificam as células do tabuleiro. De seguida definimos as restrições apresentadas anteriormente. A lista cells contém as restrições para que cada célula tenha valores entre 1 e N. A lista *rows* e *columns*, com a ajuda da função *Distinct*, restringe os valores das células para que todos as linhas e colunas, respetivamente, tenham um só número de 1 a N e a lista *numbers* obriga a que certas células tenham valores fixos definidos no ficheiro de input. Finalmente, adicionamos as restrições de desigualdade às células presentes na matriz *rules*.

In [ ]:
```
#SMT-Solver - RESTRICTIONS
print("Reading Restrictions...")
s = Solver()

puzzle = [[ Int("va_%s_%s" % (i, j)) for j in range(n)] for i in range(n)]

# -- Numbers between 1 and N
cells  = [ And(1 <= puzzle[i][j], puzzle[i][j] <= n)
                for i in range(n) for j in range(n) ]

# -- Each row contains a digit at most once
rows   = [ Distinct(puzzle[i]) for i in range(n) ]

# -- Each column contains a digit at most once
columns  = [ Distinct([ puzzle[i][j] for i in range(n) ]) for j in range(n) ]

# -- Fixed Numbers
numbers = [ If(data[i][j] == 0,
                    True,
                    puzzle[i][j] == data[i][j])
                for i in range(n) for j in range(n) ]

s.add(cells + rows + columns + numbers)

# -- Restrictions - Cell must be larger than other
for rule in rules:

    major = rule[0]
    minor = rule[1]

    maxi = major[0]
    maxj = major[1]

    mini = minor[0]
    minj = minor[1]

    s.add(puzzle[maxi][maxj] > puzzle[mini][minj])

print("DONE!")
```

```
Reading Restrictions...
DONE!
```

Solução

Por fim, resolvemos o puzzle com as restrições adicionadas. A matriz *solution* coloca as variáveis da solução que se encontra no modelo *model* pela ordem correta. De seguida fazemos o print da matriz *solution* para o ecrã. Caso o tabuleiro não tenha solução então o programa devolve o print "Something went wrong!"

In [ ]:
```
# RESOLVE PROBLEM
if s.check() == sat:

    model = s.model()

    solution = [ [ model.evaluate(puzzle[i][j]) for j in range(n) ]
                                                for i in range(n) ]

    print("\nSOLUTION",end=' ')
    for i in range(n):
        print("\n",end=' ')
        for j in range(n):
            print("%s " % solution[i][j], end=' ')
else:
    print("Something went wrong!")

print("\n")
```

```
SOLUTION
 1  4  2  3
 4  1  3  2
 2  3  4  1
 3  2  1  4
```

Tal como mostra o programa, obtemos a seguinte solução para o tabuleiro.