

```

1  class F2CSP_COUNTER:
2      def __init__(self, inF = "", outF = "", nDom = 0, nVar = 0, nRes = 0, nAccept =
0, nReject = 0) :
3          self.inFile = inF
4          self.outFile = outF
5          self.nDom = nDom
6          self.nVar = nVar
7          self.nRes = nRes
8          self.nAccept = nAccept
9          self.nReject = nReject
10
11
12
13  def readFile(self) :
14      self.inFile = input("Enter input file name:")
15      self.outFile = input("Enter output file name:")
16
17      file = open(self.inFile,"r")
18
19
20      markDom = 0
21      markVar = 0
22      markRes = 0
23      markAccept = 0
24      markReject = 0
25
26      for line in file :
27          line = line.split()
28          for l in line :
29              if(markDom == 1) :
30                  markDom -= 1
31                  self.nDom = int(l)
32
33              if(markVar == 1) :
34                  markVar -= 1
35                  self.nVar = int(l)
36
37              if(markRes == 1) :
38                  markRes -= 1
39                  self.nRes = int(l)
40
41              if(markAccept == 1) :
42                  markAccept -= 1
43                  self.nAccept += int(l)
44
45              if(markReject == 1) :
46                  markReject -= 1
47                  self.nReject += int(l)
48
49              if("Domains" in l) :
50                  markDom += 1
51
52              if("Variables" in l) :
53                  markVar += 1
54
55              if("Constraints:" in l) :
56                  markRes += 1
57
58              if("Reject" in l) :
59                  markReject += 1
60
61              if("Accept" in l) :
62                  markAccept += 1
63
64      file.close()
65
66  def writeStatistics(self) :
67      f = open(self.outFile,"w+")
68
69      f.write("Numero de Dominios: "+str(self.nDom))
70      f.write("\n")
71      f.write("Numero de Variaveis: "+str(self.nVar))

```

```
72         f.write("\n")
73         f.write("Numero de Restricoes: "+str(self.nRes))
74         f.write("\n")
75         f.write("Numero de Tuplos Aceites: "+str(self.nAccept))
76         f.write("\n")
77         f.write("Numero de Tuplos Rejeitados: "+str(self.nReject))
78
79
80     test = F2CSP_COUNTER()
81     test.readFile()
82     test.writeStatistics()
83     print("DONE")
```

```

1  import re
2  import sys
3  import math
4
5  class InputSudokuToF2CSP:
6      def __init__(self, inF = "", outF = "", matrix = [[]], n = 0) :
7          self.inFile = inF
8          self.outFile = outF
9
10     def readfile(self) :
11         self.inFile = input("Enter input file name:")
12         self.outFile = input("Enter output file name:")
13
14         file = open(self.inFile,"r")
15         self.n = int(file.readline())
16
17         self.matrix = [[0 for x in range(self.n)] for y in range(self.n)]
18
19
20         numbersOnSlot = int(file.readline())
21
22         while(numbersOnSlot > 0) :
23             tuple = re.split("\s", file.readline())
24             self.matrix[int(tuple[1])-1][int(tuple[0])-1] = int(tuple[2])
25             numbersOnSlot -= 1
26
27         file.close
28
29     def writeFile(self) :
30         f = open(self.outFile,"w+")
31
32         f.write("Title: Sudoku"+str(self.n)+"x"+str(self.n)+"\n") #Titulo
33         f.write("\n")
34         f.write("Domains:\n1\nD1 "+"1.." +str(self.n)+"\n") #Dominios
35         f.write("\n")
36         f.write("Variables:\n"+str(self.n*self.n)+"\n") #Variaveis
37         for i in range(1,self.n + 1) :
38             for j in range(1,self.n + 1) :
39                 f.write("V"+str(i)+"-"+str(j)+" D1\n")
40
41         f.write("\n")
42
43         numberConstraints = 1
44         stringTotal = ""
45
46         for row in range(1,self.n + 1) :
47             for col in range(1,self.n + 1) :
48                 constraintsROW = ""
49                 constraintsCOLUMN = ""
50                 constraintsSQUARE = ""
51
52                 #Restricoes da Linha
53                 resColl = col+1
54                 for resCol in range(resColl,self.n + 1) :
55                     constraintsROW = constraintsROW +
56                     "C"+str(numberConstraints)+":\nVars:\n2\n"+"V"+str(row)+"-"+str(co
57                     l)+"\n"+"V"+str(row)+"-"+str(resCol)+"\nReject:\n" + str(self.n)
58                     +"\n"
59
60                     for c in range(1,self.n+1) : #1..9 Rejects
61                         constraintsROW = constraintsROW + str(c) + " " + str(c) + "\n"
62
63                     numberConstraints += 1
64
65                     constraintsROW = constraintsROW + "\n"
66
67                 stringTotal += constraintsROW
68
69                 #Restricoes da Coluna
70                 resRows = row+1
71                 for resRow in range(resRows,self.n + 1) :
72                     constraintsCOLUMN = constraintsCOLUMN +

```

```

        "C"+str(numberConstraints)+":\nVars:\n2\n"+"V"+str(row)+"-"+str(col)
        +"\n"+"V"+str(resRow)+"-"+str(col)+"\nReject:\n" + str(self.n)
        +"\n"

70
71         for r in range(1,self.n+1) :
72             constraintsCOLUMN = constraintsCOLUMN + str(r) + " " +
                str(r) + "\n"

73
74             numberConstraints += 1
75
76             constraintsCOLUMN = constraintsCOLUMN + "\n"
77
78             stringTotal += constraintsCOLUMN
79
80             #Restricoes do Quadrado
81             frontRows = row
82
83             while((frontRows % math.sqrt(self.n)) != 0 ) :
84                 frontRows += 1
85                 frontCols = col
86                 backCols = col
87
88                 while(backCols % math.sqrt(self.n) != 1) : #numero de Colunas
89                     entre 0 e o numero
90                     backCols -= 1
91                     constraintsSQUARE = constraintsSQUARE +
92                     "C"+str(numberConstraints)+":\nVars:\n2\n"+"V"+str(row)+"-"+str
93                     r(col)+"\n"+"V"+str(frontRows)+"-"+str(backCols)+"\nReject:\n"
94                     + str(self.n) +"\n"
95
96                     for b in range(1,self.n+1) :
97                         constraintsSQUARE = constraintsSQUARE + str(b) + " " +
98                         str(b) + "\n"
99
100                     numberConstraints += 1
101                     constraintsSQUARE = constraintsSQUARE + "\n"
102
103                     while(frontCols % math.sqrt(self.n) != 0) : #numero de Colunas
104                         entre o numero e n
105                         frontCols += 1
106                         constraintsSQUARE = constraintsSQUARE +
107                         "C"+str(numberConstraints)+":\nVars:\n2\n"+"V"+str(row)+"-"+str
108                         r(col)+"\n"+"V"+str(frontRows)+"-"+str(frontCols)+"\nReject:\n"
109                         + str(self.n) +"\n"
110
111                         for fr in range(1,self.n+1) :
112                             constraintsSQUARE = constraintsSQUARE + str(fr) + " " +
113                             str(fr) + "\n"
114
115                             numberConstraints += 1
116                             constraintsSQUARE = constraintsSQUARE + "\n"
117
118                     stringTotal += constraintsSQUARE
119
120             acceptSQUARE = ""
121             for row in range(1,self.n + 1) :
122                 for col in range(1,self.n + 1) :
123                     if self.matrix[row-1][col-1] != 0 :
124                         acceptSQUARE = acceptSQUARE +
125                         "C"+str(numberConstraints)+":\nVars:\n1\nV"+str(row)+"-"+str(col)+
126                         "\nAccept:\n1\n"+str(self.matrix[row-1][col-1])+"\n\n"
127                         numberConstraints += 1
128
129             stringTotal += acceptSQUARE
130
131             f.write("Constraints:\n"+str(numberConstraints-1)+"\n\n")
132             f.write(stringTotal)
133             f.write("Goal:\nSatisfy")
134
135             f.close()

```

```
126 test = InputSudokuToF2CSP()
127 test.readFile()
128 test.writeFile()
129 print("DONE")
130
```

Relatório Engenharia Do Conhecimento

Fase 1

Grupo 21:

João Marques nº49038

João David nº49448

Luís Moreira nº49531

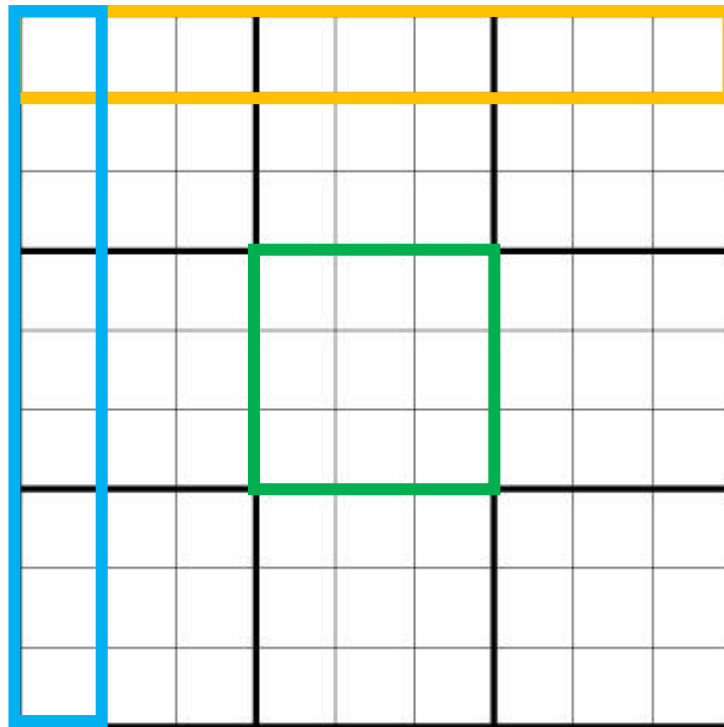
CONTEÚDO

Puzzle Escolhido: Sudoku	3
Representação CSP do sudoku	4
Programa que cria o Ficheiro com a representação F2CSP do Sudoku	5
Representação do input.....	6
Exemplo de Ficheiro de input.....	6
Algoritmo Para as restrições	7
Estatísticas do ficheiro F2CSP	9
Programa que devolve o ficheiro com as estatísticas do ficheiro F2CSP.....	9

PUZZLE ESCOLHIDO: SUDOKU

Como sabemos, o puzzle Sudoku tem três principais restrições, em que um número não pode ser repetido em:

1. Linha (Amarelo)
2. Coluna (Azul)
3. Quadrado (Verde)



De forma a ser resolvido, todas as casas têm de estar preenchidas de 1 até ao valor N que representa o Sudoku. O tabuleiro representado em cima é um Sudoku 9x9, sendo os valores dos quadrados de 1 a 9, e pela mesma lógica se fosse um Sudoku 4x4 o valor de n era igual a 4 ($n=4$), sendo os valores representados de 1 a 4, e por aí em diante para 16x16, 25x25, etc. Existe ainda mais uma regra, um Sudoku só pode ser uma matriz quadrada, logo será sempre da forma $N \times N$ e nunca $N \times M$.

REPRESENTAÇÃO CSP DO SUDOKU

A representação tomada foi a F2CSP, sendo esta do género:

1. **Título:** que indica qual é o tipo de Sudoku (se é 4x4, 9x9, etc)
2. **Domínios das variáveis:** quais os valores que elas podem tomar
3. **Variáveis:** todas as variáveis do problema
4. **Restrições:** o número de restrições do problema
 - a. Restrições em detalhe: representação de todas as restrições entre as respetivas variáveis
 - b. Existe no fim as Restrições de aceitação, quando uma variável já tem à partida um valor fixo, estes valores são as pistas para a resolução do Sudoku
5. Acaba com o objectivo, que é satisfazer as restrições todas

As variáveis existem em relação às casas do Sudoku, sendo um par de Linha-Coluna, por exemplo a primeira posição será V1-1, a segunda posição da primeira linha na segunda coluna será V1-2, etc.

Exemplo:

V1-1	V1-2	V1-3	V1-4	V1-5	V1-6	V1-7	V1-8	V1-9
V2-1								
V3-1								
V4-1								
V5-1								
V6-1								
V7-1								
V8-1								
V9-1								V9-9

As restrições em detalhe são dadas na forma:

1. Número da Restrição
2. Variáveis envolvidas nesta restrição
3. Tipo de restrição
 - a. Reject: Valores impossíveis para cada par de variável que esteja na mesma linha, coluna e quadrado
 - b. Accept: No caso de ter sido dado uma pista, é atribuída a uma variável um valor

Existe mais uma particularidade no formato F2CSP que é o facto de a seguir a cada campo, Domínio, Variáveis, Restrições, existe o número das mesmas.

Domains:

1 (número de domínios)

...

... (representação de todos os domínios)

Variables:

16 (número de variáveis)

... (representação de todas as variáveis e respectivos domínios)

...

Constraints:


59 (número de restrições do puzzle)

Vars:

2 (número de variáveis)

Reject:

4 (número de tuplos rejeitados)



Informação das variáveis
afectas e tuplos rejeitados
em cada restrição

PROGRAMA QUE CRIA O FICHEIRO COM A REPRESENTAÇÃO F2CSP
DO SUDOKU

SudokuToF2CSP.py

Ao correr, deve-se escrever na consola o ficheiro de input com a representação correta (próxima página), premir **enter**, e escrever o nome do ficheiro de output que se quer.

REPRESENTAÇÃO DO INPUT

A representação do input é bastante simples, e é dada por linhas, e esta é a seguinte:

1º Linha: Número que representa o tamanho da matriz do Sudoku, ou seja o N Sudoku (N×N)

2º Linha: O número (**x**) de pistas que vão estar no tabuleiro do Sudoku

Restantes x linhas: As pistas com a posição e valor que vai estar nessa posição, e que vai ser associada a uma variável. Exemplo: 1 1 5 → 1 1 é posição, logo é a primeira casa do Sudoku e o 5 é o valor que vai estar representado nessa casa

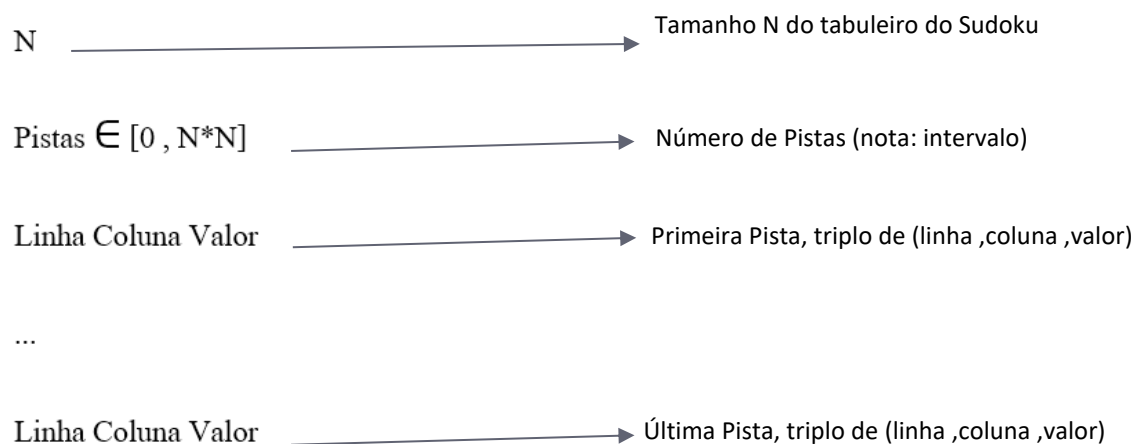
Notas:

O número de pistas não deve ser maior que (N×N). Caso não haja pistas é **obrigatório** referir zero pistas.

O valor da pista não pode ser maior que N, nem menor que 1. ($1 \leq \text{VALOR} \leq N$)

A posição da pista, X Y, não pode superar N, nem ser menor que 1. ($1 \leq X \leq N$) e ($1 \leq Y \leq N$)

EXEMPLO DE FICHEIRO DE INPUT



ALGORITMO PARA AS RESTRIÇÕES

Como foi dito no capítulo PUZZLE ESCOLHIDO: SUDOKU, as restrições são de linha, coluna e quadrado. Desta forma fizemos a leitura das restrições não ser mais excessiva do que ela já é, limitando o número de comparações que temos de fazer ao mínimo possível.

Vocabulário:

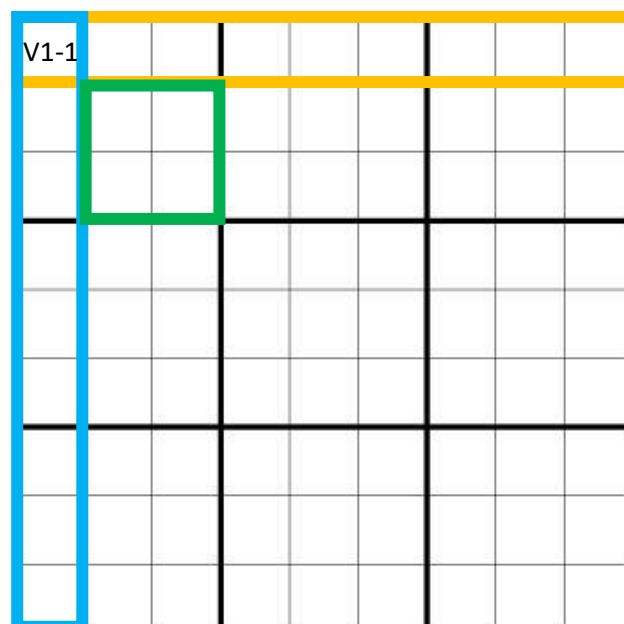
\neq : diferente

Sabemos da particularidade da lógica que se $Z \neq Y$, então $Y \neq Z$, desta forma sempre que fazemos uma restrição que implica $V1-1$ e $V1-2$, por exemplo, temos que

$V1-1 \neq V1-2$ logo não é necessário fazer $V1-2 \neq V1-1$. Isto leva-nos a nunca andar para trás no puzzle, limitando o número de comparações.

Exemplo: Para a variável $V1-1$ temos as seguintes restrições

- Todas as variáveis da linha são diferentes de $V1-1$
 - $V1-1 \neq V1-Y$, com $Y \in [1,N]$
- Todas as variáveis da coluna são diferentes de $V1-1$
 - $V1-1 \neq VX-1$, com $X \in [1,N]$
- Todas as variáveis do quadrado exceto as já verificadas nas linhas e colunas são diferentes de $V1-1$
 - $V1-1 \neq VX-Y$, com $X \in [2,\sqrt{N}]$ e $Y \in [2,\sqrt{N}]$



A verificação das linhas e colunas é tarefa fácil, um ciclo de 1 até N+1 resolve todas essas restrições, porém a verificação do subquadrado requer algumas verificações e a nossa solução passou por usar os restos de divisões que as linguagens de programação oferecem.

Para a restrição dos subquadrados temos para a verificação das linhas:

```
frontRows = row

while((frontRows % math.sqrt(self.n)) != 0 )
```

Que indica-nos quando estamos na última linha do quadrado e temos de parar as restrições, e basta apenas esta verificação porque pelo nosso método nunca é preciso voltar para trás no puzzle.

Para os lados temos as seguintes verificações:

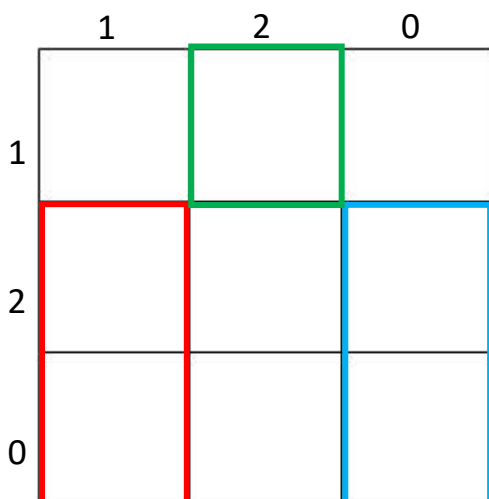
```
frontCols = col
backCols = col

while(backCols % math.sqrt(self.n) != 1)

.....
.....
    MAIS CÓDIGO
.....
.....

while(frontCols % math.sqrt(self.n) != 0)
```

Isto indica-nos o limite nos lados dos quadrados, pois o resto da divisão na última posição é 0 enquanto na primeira é 1.



Representado estão os restos das divisões por \sqrt{N} , que é o que permite o algoritmo não fazer mais comparações que as necessárias, isto é, ele nunca passa uma única vez por alguma casa já com restrições, logo não é necessário o uso de ifs no código e sim apenas os ciclos. Representado na figura a cores, a **Vermelho** as posições que faltam fazer restrições, com linha(s) à frente, e coluna(s) atrás da variável a **verde**, e a **azul** as posições que faltam fazer restrições, com linha(s) e coluna(s) à frente da variável **verde**. É um exemplo do algoritmo numa posição diferente da primeira, aplicando-se igual para as outras posições.

ESTATÍSTICAS DO FICHEIRO F2CSP

A leitura é feita pelos valores já explicados no capítulo REPRESENTAÇÃO CSP DO SUDOKU.

Vai-se lendo o ficheiro F2CSP dado pelo o outro programa, e sempre que achamos palavras chaves como “Domains”, “Variables”, “Constraints”, “Reject” e “Accept” somamos os valores na linha seguinte a um contador e ignoramos o número de linhas seguintes igual ao número lido.

Dominios: dado pelo número seguinte à palavra chave “Domains”

Variaveis: dado pelo número seguinte à palavra chave “Variables”

Restricoes: dado pelo número seguinte à palavra chave “Constraints”

Tuplos Aceites: dado pela soma de todos os números seguintes à palavra chave “Accept” em cada restrição que a contenha

Tuplos Rejeitados: dado pela soma de todos os números seguintes à palavra chave “Reject” em cada restrição que a contenha

PROGRAMA QUE DEVOLVE O FICHEIRO COM AS ESTATISTICAS DO FICHEIRO F2CSP

F2CSP_Stats.py

Ao correr, deve-se escrever na consola o ficheiro de input, este sendo um ficheiro de output do programa SudokuCSP.py, premir **enter**, e escrever o nome do ficheiro de output que se quer.

```
1 Title: 2x2F2CSP
2
3 Domains:
4 1
5 D1 1..2
6
7 Variables:
8 4
9 V11 D1
10 V12 D1
11 V21 D1
12 V22 D1
13
14 Constraints:
15 4
16
17 C1:
18 Vars:
19 2
20 V11
21 V12
22 Reject:
23 2
24 1 1
25 2 2
26
27 C2:
28 Vars:
29 2
30 V11
31 V21
32 Reject:
33 2
34 1 1
35 2 2
36
37 C3:
38 Vars:
39 2
40 V12
41 V22
42 Reject:
43 2
44 1 1
45 2 2
46
47 C4:
48 Vars:
49 2
50 V21
51 V22
52 Reject:
53 2
54 1 1
55 2 2
56
57 Goal:
58 Satisfy
```

```

1  #----- Domain Class-----
2  class Domain():
3      def __init__(self, name, start, end):
4          self.name = name
5          self.values = range(start,end+1)
6          self.vars = []
7
8
9
10     def addVariable(self,variable):
11         self.vars.append(variable)
12
13     def getValues(self):
14         return self.values
15
16     def getVars(self):
17         return vars
18
19     def strSelectVariables(self):
20         print("vars"+str(self.vars))
21         res = ""
22         for v in self.vars:
23             res += "?" + v + " "
24         return res
25
26     def __str__(self):
27         res = ":" + self.name + " rdf:type :Domain ;\n" + "\t:values "
28         for val in self.values[:-1]:
29             res += str(val) + ", "
30         res += str(self.values[-1]) + " ;\n\t:variables"
31         for var in self.vars[:-1]:
32             res += " : " + str(var) + ", "
33         res += " : " + self.vars[-1] + ".\n\n"
34         return res
35
36
37  #----- Constraint Class-----
38  class Constraint:
39      def __init__(self):
40          self.typeCons = ""
41          self.vars = []
42          self.values = []
43          self.first = ""
44          self.second = ""
45          self.third = ""
46
47
48     def addVar(self, var):
49         self.vars.append(var)
50
51     def addValue(self, value):
52         self.values.append(value)
53
54     def setTypeCons(self,typeCons):
55         self.typeCons = typeCons
56         if self.typeCons == "Reject:\n":
57             self.first = " != "
58             self.second = " || "
59             self.third = " && "
60         elif self.typeCons == "Accept:\n":
61             self.first = " = "
62             self.second = " && "
63             self.third = " || "
64
65     def __str__(self):
66         print(self.vars)
67         print(self.values)
68         res = "\t\t( "
69         for i in range(len(self.values)):
70             res += "("
71             for j in range(len(self.vars)):
72                 res += "?" + str(self.vars[j]) + self.first + str(self.values[i][j])
73                 if j == len(self.vars) - 1:

```



```

73         res += ")"
74     else:
75         res += self.second
76
77         if i != len(self.values) - 1:
78             res += self.third
79     res += " )\n"
80     return res
81
82
83
84
85 #----- Script itself -----
86 class MainRun:
87     def __init__(self):
88         self.inFileName = ""
89         self.outFileName = ""
90         self.domains = {}
91         self.fileOutRDF = None
92         self.fileOutSPAQRL = None
93
94     def writeDomains(self):
95         for d in self.domains.keys():
96             self.fileOutRDF.write(str(self.domains[d]))
97
98     def parseConstraints(self, file, nConst):
99         nConstParsed = 0
100         for line in file:
101             constraint = Constraint()
102             if("Vars:" in line):
103                 if nConstParsed < nConst:
104                     nVars = int(file.readline())
105                     for x in range(nVars):
106                         var = file.readline().rstrip('\n')
107                         constraint.addVar(var)
108                     typeCons = file.readline()
109                     constraint.setTypeCons(typeCons)
110                     nValues = int(file.readline())
111                     for x in range(nValues):
112                         lineValue = file.readline().rstrip('\n')
113                         constraint.addValue(lineValue.split())
114                     if nConstParsed + 1 < nConst:
115                         self.fileOutSPAQRL.write(str(constraint) + "\t\t&& \n")
116                     else:
117                         self.fileOutSPAQRL.write(str(constraint))
118                     nConstParsed += 1
119
120     def writeSelect(self):
121         self.fileOutSPAQRL.write("SELECT ")
122         for _ , value in self.domains.items():
123             self.fileOutSPAQRL.write(value.strSelectVariables())
124         self.fileOutSPAQRL.write("\n")
125
126     def writeWhere(self):
127         self.fileOutSPAQRL.write("WHERE {\n")
128         for key , value in self.domains.items():
129             listVar = value.vars
130             for v in listVar:
131                 self.fileOutSPAQRL.write("\t:" + key + " :values ?" + v + " .\n")
132         self.fileOutSPAQRL.write("\tFILTER {\n")
133
134     def run(self):
135         self.inFileName = input("Enter F2CSP file name:")
136         self.outFileName = input("Enter output file name:")
137         self.fileOutRDF = open(self.outFileName + ".ttl", "w+")
138         self.fileOutRDF.write("@prefix : <http://www.w3.org> .\n")
139         self.fileOutRDF.write("@prefix rdf:
140         <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n\n")
141         self.fileOutSPAQRL = open(self.outFileName + ".rq", "w+")
142         self.fileOutSPAQRL.write("PREFIX : <http://www.w3.org>\n")
143         self.fileOutSPAQRL.write("PREFIX rdf:
144         <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n")

```

```

143
144     fileIn = open(self.inFileName, "r")
145     for line in fileIn:
146         if("Domains:" in line):
147             nDomains = int(fileIn.readline())
148             for _ in range(nDomains):
149                 currD = fileIn.readline()
150                 d = currD.split()
151                 self.domains[d[0]] = Domain(d[0], int(d[1][0]),int(d[1][-1]))
152         if("Variables:" in line):
153             nVars = int(fileIn.readline())
154             for _ in range(nVars):
155                 currV = fileIn.readline()
156                 v = currV.split()
157                 self.domains[v[1]].addVariable(v[0])
158             self.writeDomains()
159         if("Constraints:" in line):
160             self.writeSelect()
161             self.writeWhere()
162             self.parseConstraints(fileIn,int(fileIn.readline()))
163             self.fileOutSPAQRL.write("\t)\n")
164             self.fileOutSPAQRL.write("}")
165     fileIn.close()
166     self.fileOutRDF.close()
167     self.fileOutSPAQRL.close()
168     print("SCRIPT END")
169
170
171
172 scriptRun = MainRun()
173 scriptRun.run()

```

Relatório de Engenharia do Conhecimento

Fase 2

Engenharia Informática

Grupo 21

João David n49448

João Marques n49038

Luís Moreira n49531

Índice

Script para gerar query SPARQL	3
Implementação do script F2CSPtoSPARQL	3
Escrita da query	4
Execução da query	7
Análise do desempenho	8
Entrega e anexos	10

Script para gerar query SPARQL

O ficheiro F2CSPtoSPARQL.py corresponde ao script responsável por gerar o ficheiro .ttl que contem o conhecimento dos domínios e respectivas variáveis do problema F2CSP e também por gerar o ficheiro .rq que contém a query SPARQL que resolve o problema com base nas constraints do ficheiro F2CSP.

Ao correr o script, é pedido ao utilizador o nome do ficheiro F2CSP, e o nome do ficheiro de saída (como exemplo, considere o nome do ficheiro de saída “out”), após o script interpretar o ficheiro F2CSP, vão ser criados dois ficheiros, um denominado out.ttl e outro out.rq, para obter a solução do problema basta utilizar estes dois ficheiros com o SPARQL.

O script espera que o ficheiro F2CSP fornecido esteja corretamente escrito, caso contrário terá um comportamento imprevisível e os ficheiros de saída não deveram ser considerados.

Implementação do script F2CSPtoSPARQL

Na implementação do script foram criadas três classes (todas no mesmo ficheiro, F2CSPtoSPARQL.py), a class Domain, Constraint e MainRun.

Ao correr a script a class MainRun vai ser executada e inicia a interpretação do ficheiro F2CSP, ao encontrar a label “Domains:” sabe que estão definidos em baixo os domínios, por cada domínio encontrado, é criado um objecto Domain e adiciona-o a um dicionário (key: “D1”, value: Domain), quando posteriormente encontra a label “Variables:” procede à análise das variáveis que estão definidas na forma “V11 D1”, para adicionar esta variável ao domínio definido anteriormente, acede-se ao dicionário com a chave do domínio, neste caso “D1”, e adiciona-se “V11” à lista de variáveis pertencentes ao respectivo domínio.

Ao encontrar a label “Constraints:”, o MainRun sabe que a partir daí vai encontrar todas as constraints do F2CSP, portanto, para cada constraint encontrada vai criar um objecto “Constraint”, e enquanto analisa a mesma constraint, vai adicionando informação ao objecto, nomeadamente as variáveis envolvidas, o tipo de constraint (Reject ou Accept) e os valores associados a cada variável que serão rejeitados ou aceites com base no tipo de constraint, por fim, escreve no ficheiro out.rq a condição dessa constraint na zona do FILTER, quando isto acontece, há que ter em conta que operadores lógicos utilizar, visto que o tipo de constraint influencia a forma como a condição é escrita, o seguinte tópico do relatório irá explicar como é feita a escrita da query.

Escrita da query

Para simplificar, tome como exemplo o ficheiro de input F2CSP denominado “2x2F2CSP.txt”, que trata de resolver as restrições de um tabuleiro 2x2, em que na mesma linha e na mesma coluna não há valores repetidos, valores esses que estão contidos no único domínio “D1 1..2”.

V11	V12
V21	V22

Assim que é iniciado a escrita do ficheiro out.rq, são escritos os PREFIX utilizados, e de seguida escrito o SELECT para todas as variáveis de todos os domínios envolvidos no ficheiro F2CSP.

```
PREFIX : <http://www.w3.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?V11 ?V12 ?V21 ?V22
```

Depois segue-se a escrita do WHERE, onde se escreve para cada domínio, as suas variáveis em cada linha seguindo o triplo utilizado no ficheiro out.ttl

No ficheiro out.ttl temos:

```
:D1 rdf:type :Domain ;
    :values 1, 2 ;
    :variables :V11, :V12, :V21, :V22.
```

No ficheiro out.rq temos:

```
WHERE {
    :D1 :values ?V11.
    :D1 :values ?V12.
    :D1 :values ?V21.
    :D1 :values ?V22.
```

De seguida é realizada a escrita do FILTER, onde estão as restrições de todas as constraints lidas do ficheiro F2CSP, como dito anteriormente, sempre que é lido uma constraint, o seu tipo (Accept ou Reject) influencia a forma como é escrito a condição.

Tome como exemplo a escrita das seguintes constraints, em que a única diferença é no seu tipo:

```
C1:
Vars:
2
V11
V12
Reject:
2
1 1
2 2
```

```
C2:
Vars:
2
V11
V12
Accept:
2
1 1
2 2
```

No caso do Reject, a condição será:

```
( (?V11 != 1 || ?V12 != 1) && (?V11 != 2 || ?V12 != 2) )
```

Enquanto que no Accept será:

```
( (?V11 = 1 && ?V12 = 1) || (?V11 = 2 && ?V12 = 2) )
```

A única diferença entre estas duas condições são os operadores utilizados (coloridos com amarelo, castanho e verde), foi com base neste padrão que foi definido o método que define o tipo de constraint na class Constraint.

```
class Constraint:
    def setTypeCons(self,typeCons):
        self.typeCons = typeCons
        if self.typeCons == "Reject:\n":
            self.first = " != "
            self.second = " || "
            self.third = " && "
        elif self.typeCons == "Accept:\n":
            self.first = " = "
            self.second = " && "
            self.third = " || "
```

As condições são escritas para cada constraint presente no ficheiro F2CSP e adicionado ao filter da query sparql, cada condição escrita é sempre separada por &&.

Considere o ficheiro “.rq” do jogo 2x2 mencionado anteriormente, em que em cada linha e coluna não há números repetidos.

```
PREFIX : <http://www.w3.org>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?V11 ?V12 ?V21 ?V22
WHERE {
  :D1 :values ?V11.
  :D1 :values ?V12.
  :D1 :values ?V21.
  :D1 :values ?V22.
  FILTER (
    ( (?V11 != 1 || ?V12 != 1) && (?V11 != 2 || ?V12 != 2) )
    &&
    ( (?V11 != 1 || ?V21 != 1) && (?V11 != 2 || ?V21 != 2) )
    &&
    ( (?V12 != 1 || ?V22 != 1) && (?V12 != 2 || ?V22 != 2) )
    &&
    ( (?V21 != 1 || ?V22 != 1) && (?V21 != 2 || ?V22 != 2) )
  )
}
```


Execução da query

Para obtermos os resultados ao problema F2CSP é necessário utilizar os ficheiros de saída “.ttl” e “.rq” com o SPARQL.

Recorrendo à consola do Linux e ao programa sparql executa-se o seguinte comando:

```
sparql -data file_name.ttl -query file_name.rq
```

O resultado da query referente ao jogo enunciado no início deste relatório é a seguinte. Em que cada linha corresponde a uma possível solução.

V11	V12	V21	V22
1	2	2	1
2	1	1	2

As duas soluções para o problema do 2x2F2CSP.txt. Como previsto, não há números repetidos em cada linha e cada coluna.

1	2
2	1

2	1
1	2

Análise do desempenho

Para realizar o cálculo de tempo de execução da query, foi utilizado o programa time do Linux, o comando utilizado foi o seguinte, o tempo obtido corresponde ao wall clock time, o tempo desde que se clicou no enter para submeter o comando e o momento em que a execução do comando terminou.

```
time sparql -data file_name.ttl -query file_name.rq
```

A máquina utilizada para realizar as query foi o servidor do DI da FCUL, gcc.alunos.di.fc.ul.pt, utilizando a VPN da fcul para estabelecer conexão.

Sudoku 4x4

Considere os três puzzles de sudoku 4x4, os ficheiros F2CSP correspondentes estão anexados a este relatório.

NoClues (288 soluções possíveis)

4	1	3	2
3	2	4	1

A (4 soluções possíveis)

4	1	3	2
3	2	4	1
1	3	2	4
2	4	1	

B (1 solução possível)

Foram corridos 5 series de testes para cada puzzle, o tempo foi calculado em segundos:

tempo
(sec)
1,949
1,968
1,886
1,863
1,932
Média
1,9196

tempo
(sec)
1,351
1,347
1,341
1,350
1,345
Média
1,3468

tempo
(sec)
1,341
1,338
1,331
1,310
1,360
Média
1,3360

Analisando os resultados, é possível concluir que a diferença no tempo de execução da query de um puzzle sudoku 4x4 sem pistas, para um outro sudoku 4x4 em que apenas falta 1 valor para terminar o puzzle é de apenas 0,5836 segundos, não é muito significativo. Porém este caso já não se verifica para um sudoku de 9x9, como mostram os seguintes resultados

Sudoku 9x9

Para o sudoku 9x9 foi feita uma serie de testes de forma análoga ao sudoku 4x4.

2	8	9			5	4		
7		4					5	3
		3	7					2
6	9				4		3	
		5		8		6		
	1		5				9	8
8					1	5		
9	7					3		4
		1	2			8	6	7

C (35 pistas, 1 solução)

2	8	9	3	1	5	4	7	6
7	6	4					5	3
		3	7					2
6	9				4		3	
		5		8		6		
	1		5				9	8
8					1	5		
9	7					3		4
		1	2			8	6	7

D (40 pistas, 1 solução possível)

2	8	9	3	1	5	4	7	6
7	6	4	8	9	2	1	5	3
1	5	3	7	4	6	9	8	2
6	9	8	1	2	4	7	3	5
3	2	5	9	8	7	6	4	1
4	1	7	5	6	3	2	9	8
8	3	6	4	7	1	5	2	9
9	7	2	6	5	8	3	1	4
5	4	1	2	3	9	8	6	

E (80 pistas, 1 solução possível)

Foram corridos 5 series de testes para cada puzzle, o tempo foi calculado é o seguinte:

tempo	tempo (sec)	tempo (sec)
5m 15,317 s	21,59	2,662
5m 27,419 s	21,141	2,596
5m 43,606 s	21,184	2,557
5m 31,698 s	21,159	2,597
5m 26,322 s	20,979	2,601
Média	Média	Média
5m 28,872s	21,2106	2,6026

Ao contrário do que se verificou com o sudoku 4x4, o sudoku 9x9 apresenta resultados muito diferentes em relação o número de pistas fornecidas. Aquando da realização do sudoku 9x9 sem pistas, a query ficou a calcular os valores por mais de 4 horas, e não obteve qualquer tipo de resultados.

Ao fornecermos 35 pistas, a query levou em média 5 minutos e 28 segundos a obter a solução do sudoku. Ao aumentar o número de pistas em 5, ficando agora com 40, o tempo que levou foi de apenas 21 segundos, houve uma diferença de considerável no tempo de execução da query. Por último, quando se tentou obter a solução do sudoku, ficando apenas uma pista por revelar, a query demorou em média 2,6 segundos. Ou seja, mesmo faltando apenas um valor do sudoku 9x9, a query do sudoku 4x4 sem pistas consegue ser ainda mais rápida. Isto deve-se ao facto de quanto maior o número de restrições e variáveis, mais tempo leva a query a obter a solução do problema F2CSP.

Entrega e anexos

Os ficheiros entregues são os seguintes:

- F2CSPtoSPARQL.py
 - Script em Python 3 responsável por converter F2CSP em query SPARQL
- 2x2F2CSP.txt
 - Ficheiro F2CSP referente ao jogo enunciado anteriormente
- SudokuToF2CSP.py
 - Script em Python 3 responsável por gerar sudokus em F2CSP, foi esta script a utilizada para gerar os ficheiros F2CSP dos sudokus usados na análise de desempenho
- Sudoku4x4NoClueF2CSP.txt, Sudoku4x4_F2CSP_A.txt e Sudoku4x4_F2CSP_B.txt
 - Ficheiros F2CSP usados para gerar as query da análise de desempenho do sudoku 4x4
- Sudoku9x9_F2CSP_C.txt, Sudoku9x9_F2CSP_D.txt e Sudoku9x9_F2CSP_E.txt
 - Ficheiros F2CSP usados para gerar as query da análise de desempenho do sudoku 9x9

```
1 PREFIX : <http://www.w3.org>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 SELECT ?V11 ?V12 ?V21 ?V22
4 WHERE {
5     :D1 :values ?V11.
6     :D1 :values ?V12.
7     :D1 :values ?V21.
8     :D1 :values ?V22.
9     FILTER (
10         ( (?V11 != 1 || ?V12 != 1) && (?V11 != 2 || ?V12 != 2) )
11         &&
12         ( (?V11 != 1 || ?V21 != 1) && (?V11 != 2 || ?V21 != 2) )
13         &&
14         ( (?V12 != 1 || ?V22 != 1) && (?V12 != 2 || ?V22 != 2) )
15         &&
16         ( (?V21 != 1 || ?V22 != 1) && (?V21 != 2 || ?V22 != 2) )
17     )
18 }
```

```
1 @prefix : <http://www.w3.org> .
2 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3
4 :D1 rdf:type :Domain ;
5     :values 1, 2 ;
6     :variables :V11, :V12, :V21, :V22.
7
8
```

```

1  #----- Domain Class-----
2  class Domain():
3      def __init__(self, name, start, end):
4          self.name = name
5          self.values = range(start,end+1)
6          self.vars = []
7
8      def addVariable(self,variable):
9          self.vars.append(variable.lower())
10
11     def getValues(self):
12         return self.values
13
14     def getVars(self):
15         return vars
16
17     def __str__(self):
18         res = ":" + self.name + " rdf:type :Domain ;\n" + "\t:values "
19         for val in self.values[:-1]:
20             res += str(val) + ", "
21         res += str(self.values[-1]) + " ;\n\t:variables"
22         for var in self.vars[:-1]:
23             res += " : " + str(var) + ", "
24         res += " : " + self.vars[-1] + ".\n\n"
25         return res
26
27
28
29  #----- Constraint Class-----
30  class Constraint:
31      def __init__(self,varDom):
32          self.typeCons = ""
33          self.vars = []
34          self.values = []
35          self.varDom = varDom
36
37      def addVar(self, var):
38          self.vars.append(var)
39
40      def addValue(self, value):
41          self.values.append(value)
42
43      def setTypeCons(self,typeCons):
44          self.typeCons = typeCons
45          if self.typeCons == "Reject:\n":
46              self.first = " != "
47              self.second = " || "
48              self.third = " && "
49          elif self.typeCons == "Accept:\n":
50              self.first = " = "
51              self.second = " && "
52              self.third = " || "
53
54      def __str__(self):
55          print(self.vars)
56          print(self.values)
57          res = ""
58          for i in range(len(self.values)):
59              if self.typeCons == "Reject:\n":
60                  res += " ObjectComplementOf("
61              if len(self.vars) > 1:
62                  res += " ObjectIntersectionOf("
63              for j in range(len(self.vars)):
64                  res += " ObjectHasValue("
65                  res += ":" + self.vars[j].lower()
66                  res += " "
67                  res += ":" + self.varDom[self.vars[j]].lower() + "val" +
68                      str(self.values[i][j]).lower() + ")"
69              if len(self.vars) > 1:
69                  res += ")"
70              if self.typeCons == "Reject:\n":
71                  res += ")"

```

```

72
73     return res
74
75
76
77
78 #----- Script itself -----
79 class MainRun:
80     def __init__(self):
81         self.inFileName = ""
82         self.outFileName = ""
83         self.domains = {}
84         self.varDom = {}
85         self.fileOutOWL = None
86
87     def parseConstraints(self, file, nConst):
88         if nConst > 1:
89             self.fileOutOWL.write(" ObjectIntersectionOf(")
90         nConstParsed = 0
91         for line in file:
92             constraint = Constraint(self.varDom)
93             if("Vars:" in line):
94                 if nConstParsed < nConst:
95                     nVars = int(file.readline())
96                     for x in range(nVars):
97                         var = file.readline().rstrip('\n')
98                         constraint.addVar(var)
99                     typeCons = file.readline()
100                     constraint.setTypeCons(typeCons)
101                     nValues = int(file.readline())
102                     for x in range(nValues):
103                         lineValue = file.readline().rstrip('\n')
104                         constraint.addValue(lineValue.split())
105                     self.fileOutOWL.write(str(constraint))
106                     nConstParsed += 1
107         if nConst > 1:
108             self.fileOutOWL.write(")")
109
110
111     def writeHashTagSeparator(self, title):
112         self.fileOutOWL.write("#####\n")
113         self.fileOutOWL.write("# " + title + "\n")
114         self.fileOutOWL.write("#####\n\n")
115
116
117     def writeObjectProperties(self):
118         self.writeHashTagSeparator("Object Properties")
119         for _, d in self.domains.items():
120             for v in d.vars:
121                 self.fileOutOWL.write("# Object Property: " + v + " (: " + v +
122                                     ")\n\n")
123                 self.fileOutOWL.write("FunctionalObjectProperty(: " + v + ")\n")
124                 self.fileOutOWL.write("ObjectPropertyDomain(: " + v + " :Var)\n")
125                 self.fileOutOWL.write("ObjectPropertyRange(: " + v + " : " + d.name +
126                                     ")\n\n")
127
128     def writeClasses(self, fileIn):
129         self.writeHashTagSeparator("Classes")
130         for _, d in self.domains.items():
131             self.fileOutOWL.write("# Class: " + d.name + " (: " + d.name + ")\n\n")
132             self.fileOutOWL.write("EquivalentClasses(: " + d.name + " ObjectOneOf(")
133             for val in d.values:
134                 self.fileOutOWL.write(": " + d.name.lower() + "val" + str(val) + " ")
135             self.fileOutOWL.write("))\n\n")
136             self.fileOutOWL.write("# Class: :Fml (:Fml)\n\n")
137             self.writeFml(fileIn)
138             self.fileOutOWL.write("# Class: :Var (:Var)\n\n")
139             for _, d in self.domains.items():
140                 for v in d.vars:
141                     self.fileOutOWL.write("EquivalentClasses(:Var
142                                         ObjectExactCardinality(1 : " + v + " : " + d.name + ")\n")
143                     self.fileOutOWL.write("\n\n")

```



```

141 def writeNamesIndividuals(self):
142     self.writeHashTagSeparator("Named Individuals")
143     for _ , d in self.domains.items():
144         for val in d.values:
145             self.fileOutOWL.write("# Individual: ":" + d.name.lower() + "val" +
146                 str(val) + " (::" + d.name.lower() + "val" + str(val) + ")\n\n")
147             self.fileOutOWL.write("ClassAssertion(:" + d.name + " ::" +
148                 d.name.lower() + "val" + str(val) + ")\n\n")
149 self.fileOutOWL.write("# Individual: :fml (:fml)\n\n")
150 self.fileOutOWL.write("ClassAssertion(:Fml :fml)\n\n")
151 self.fileOutOWL.write("SameIndividual(:fml :map)\n\n")
152 self.fileOutOWL.write("# Individual: :map (:map)\n\n")
153 self.fileOutOWL.write("ClassAssertion(:Var :map)\n\n")
154 self.fileOutOWL.write("\n\n")
155 for _ , d in self.domains.items():
156     self.fileOutOWL.write("DifferentIndividuals(")
157     for val in d.values:
158         self.fileOutOWL.write("::" + d.name.lower() + "val" + str(val) + " ")
159     self.fileOutOWL.write(")\n\n")
160
161 def writeFml(self,fileIn):
162     self.fileOutOWL.write("EquivalentClasses(:Fml")
163     n = int(fileIn.readline())
164     self.parseConstraints(fileIn,n)
165     self.fileOutOWL.write(")\n\n")
166
167 def run(self):
168     self.inFileName = input("Enter F2CSP file name:")
169     self.outFileName = input("Enter output file name:")
170     self.fileOutOWL = open(self.outFileName + ".owl","w+")
171
172     self.fileOutOWL.write("Prefix(:=<http://www.semanticweb.org/grupo21/ontologies/2019/4/grupo21#>)\n")
173     self.fileOutOWL.write("Prefix(owl:=<http://www.w3.org/2002/07/owl#>)\n")
174
175     self.fileOutOWL.write("Prefix(rdf:=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)\n")
176     self.fileOutOWL.write("Prefix(xml:=<http://www.w3.org/XML/1998/namespace>)\n")
177     self.fileOutOWL.write("Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)\n")
178
179     self.fileOutOWL.write("Prefix(rdfs:=<http://www.w3.org/2000/01/rdf-schema#>)\n")
180
181     self.fileOutOWL.write("\n\nOntology(<http://www.semanticweb.org/grupo21/ontologies/2019/4/" + self.outFileName + ">\n\n")
182
183     fileIn = open(self.inFileName, "r")
184     for line in fileIn:
185         if("Domains:" in line):
186             nDomains = int(fileIn.readline())
187             for _ in range(nDomains):
188                 currD = fileIn.readline()
189                 d = currD.split()
190                 self.domains[d[0]] = Domain(d[0], int(d[1][0]),int(d[1][-1]))
191             for d in self.domains.keys():
192                 self.fileOutOWL.write("Declaration(Class(:" + d + "))\n")
193             self.fileOutOWL.write("Declaration(Class(:Fml))\n")
194             self.fileOutOWL.write("Declaration(Class(:Var))\n")
195
196         if("Variables:" in line):
197             nVars = int(fileIn.readline())
198             for _ in range(nVars):
199                 currV = fileIn.readline()
200                 v = currV.split()
201                 self.domains[v[1]].addVariable(v[0])
202                 self.varDom[v[0]] = v[1]
203             for _ , d in self.domains.items():
204                 for v in d.vars:
205                     self.fileOutOWL.write("Declaration(ObjectProperty(:" + v +
206                         ")\n\n")

```

```
202         for _, d in self.domains.items():
203             for val in d.values:
204                 self.fileOutOWL.write("Declaration(NamedIndividual(:" +
205                                         d.name.lower() + "val" + str(val) + "))\n")
206                 self.fileOutOWL.write("Declaration(NamedIndividual(:fml))\n")
207                 self.fileOutOWL.write("Declaration(NamedIndividual(:map))\n")
208                 self.writeObjectProperties()
209                 self.fileOutOWL.write("\n")
210
211             if("Constraints:" in line):
212                 self.writeClasses(fileIn)
213                 self.writeNamesIndividuals()
214                 self.fileOutOWL.write(" ")
215                 fileIn.close()
216                 self.fileOutOWL.close()
217             print("SCRIPT END")
218
219
220 scriptRun = MainRun()
221 scriptRun.run()
```

Relatório de Engenharia do Conhecimento

Fase 3

Engenharia Informática

Grupo 21

João David n49448

João Marques n49038

Luís Moreira n49531

Índice

Script para gerar ficheiro com ontologia em OWL 2 DL.....	3
Implementação do script F2CSPtoOWL	3
Interpretação da solução desenvolvida	4
Entrega e anexos	8

Script para gerar ficheiro com ontologia em OWL 2 DL

O ficheiro F2CSPtoOWL.py corresponde ao script responsável por interpretar o ficheiro F2CSP e gerar o ficheiro “.owl”.

O ficheiro de entrada F2CSP convêm estar na mesma directoria do script, visto que ao utilizar determinadas formas de caminhos para o ficheiro poderá causar algum erro, desta forma, ao ter o ficheiro de entrada junto do script, basta escrever o seu nome, o mesmo acontece quando definir o nome do ficheiro de saída.

Ao correr o script, é pedido ao utilizador o nome do ficheiro F2CSP, e o nome do ficheiro de saída (como exemplo, considere o nome do ficheiro de saída “out”), após o script interpretar o ficheiro F2CSP, será criado o ficheiro out.owl que pode posteriormente ser aberto utilizando o Protege.

O script espera que o ficheiro F2CSP fornecido esteja corretamente escrito, caso contrário terá um comportamento imprevisível e os ficheiros de saída não deveram ser considerados.

Implementação do script F2CSPtoOWL

Na implementação do script foram criadas três classes (todas no mesmo ficheiro, F2CSPtoOWL.py), a class Domain, Constraint e MainRun.

Ao correr a script a class MainRun vai ser executada e inicia a interpretação do ficheiro F2CSP, ao encontrar a label “Domains:” sabe que estão definidos em baixo os domínios, por cada domínio encontrado, é criado um objecto Domain e adiciona-o a um dicionário (key: “D1”, value: Domain), quando posteriormente encontra a label “Variables:” procede à análise das variáveis que estão definidas na forma “V11 D1”, para adicionar esta variável ao domínio definido anteriormente, acede-se ao dicionário com a chave do domínio, neste caso “D1”, e adiciona-se “V11” à lista de variáveis pertencentes ao respectivo domínio.

Ao encontrar a label “Constraints:”, o MainRun sabe que a partir daí vai encontrar todas as constraints do F2CSP, portanto, para cada constraint encontrada vai criar um objecto “Constraint”, e enquanto analisa a mesma constraint, vai adicionando informação ao objecto, nomeadamente as variáveis envolvidas, o tipo de constraint (Reject ou Accept) e os valores associados a cada variável que serão rejeitados ou aceites com base no tipo de constraint.

Interpretação da solução desenvolvida

Para simplificar, tome como exemplo o ficheiro de input F2CSP denominado “2x2F2CSP.txt”, que trata de resolver as restrições de um tabuleiro 2x2, em que na mesma linha e na mesma coluna não há valores repetidos, valores esses que estão contidos no único domínio “D1 1..2”.

V11	V12
V21	V22

A interface do Protege será utilizada no auxílio da explicação da solução desenvolvida.

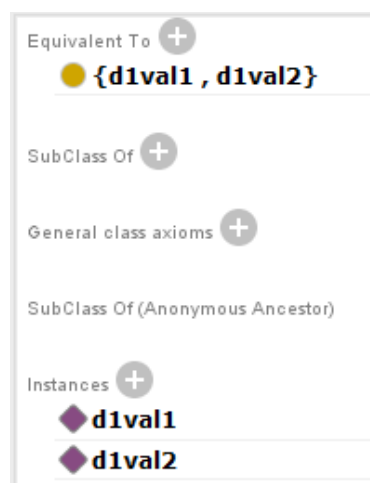
Neste caso, serão criadas três Classes, duas classes obrigatórias independentemente do problema (Fml e Var), e uma classe Dx por cada domínio definido no F2CSP.



Classe Dx

Neste caso existe apenas a classe D1. Tendo em conta o número de valores do domínio definido no ficheiro F2CSP, o mesmo número de instâncias desta classe será criado, utilizando a sintaxe “dXvalY”, em que X corresponde ao número do domínio, e Y ao valor respectivo a esse domínio. Por exemplo, o ficheiro F2CSP do problema enunciado anteriormente tem um único domínio “D1 1..2”. Logo será criada apenas uma classe referente aos domínios do problema, a classe D1, e essa classe terá como instância d1val1 e d1val2, visto que o domínio vai de 1 a 2.

Por fim é ainda definido em todas as classes domínio criadas a propriedade “Equivalent to” onde serão escritas todas as instâncias desse mesmo domínio. Esta propriedade permite-nos fechar o mundo, é uma forma de dizer que esta classe apenas tem estas instâncias.



Classe Fml

É nesta classe que serão escritas as restrições do problema, mais precisamente na propriedade “Equivalent To”, será criada uma conjunção com todas as constraints definidas no F2CSP, quando uma constraint é do tipo Reject, será aplicado o prefixo not de forma a negar a atribuição desses mesmos valores em conjunto. É também criada uma única instância desta classe com o nome fml onde serão inferidas as object properties assim que se ligar o Reasoner.

Por exemplo, no ficheiro “2x2F2CSP.txt” temos quatro constraints:

```
C1:
Vars:
2
V11
V12
Reject:
2
1 1
2 2
```

```
C2:
Vars:
2
V11
V21
Reject:
2
1 1
2 2
```

```
C3:
Vars:
2
V12
V22
Reject:
2
1 1
2 2
```

```
C4:
Vars:
2
V21
V22
Reject:
2
1 1
2 2
```

O desenvolvimento das restrições é o seguinte:

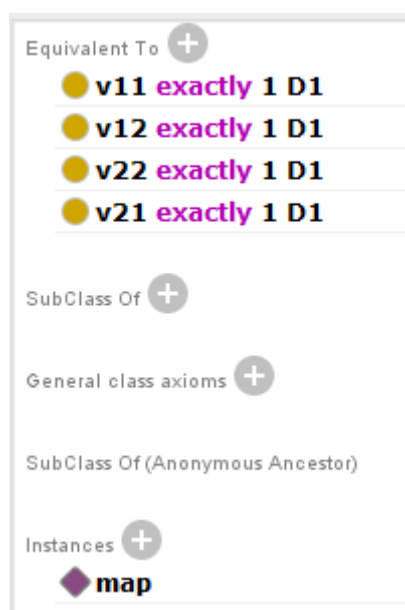
- Satisfy(C1) **and** Satisfy(C2) **and** Satisfy(C3) **and** Satisfy(C4)
 - ((not ((v11 value d1val1) and (v12 value d1val1)))
 - **and**
 - ((not ((v11 value d1val1) and (v21 value d1val1)))
 - **and**
 - ((not ((v12 value d1val1) and (v22 value d1val1)))
 - **and**
 - ((not ((v21 value d1val1) and (v22 value d1val1)))

Estas condições não são suficientes para obter uma solução, visto que para o protege conseguir devolver uma solução válida, o problema necessita de ter apenas uma solução possível, desta forma é necessário neste caso criar uma constraint do tipo Accept: de forma a que só haja uma solução possível. Por exemplo, ao adicionar a seguinte constraint às anteriores permitiria obter uma solução válida do problema. Se o problema tiver mais que uma solução, o Protege lançará um erro de “inconsistente ontologies”.

```
C5:  
Vars:  
2  
V11  
V12  
Reject:  
1  
1 2
```

Classe Var

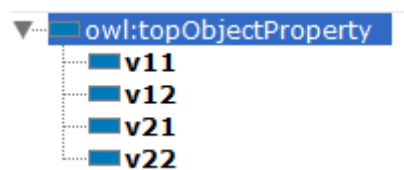
Nesta classe será definido que para cada object property terá apenas um domínio correspondente ao seu domínio definido no ficheiro F2CSP. Isto obriga a que exista apenas um valor do domínio em causa atribuído à variável do problema. É também definida apenas uma instância com nome “map” correspondente ao mapa do problema.



Object Properties

Serão criados “x” Object Properties, em que “x” corresponde ao número total de variáveis definidas no ficheiro F2CSP. Cada object property vai ter como nome a variável que foi definida no ficheiro F2CSP e terá como domain a classe Var e como range a classe domínio a que pertence a variável no ficheiro F2CSP.

No caso do problema referido em cima, existira 4 object properties, cujos nomes serão v11, v12, v21 e v22, cujas definições são como consta em baixo. É ainda definida a característica “Functional” de forma a que haja apenas seja atribuído para cada variável, se o problema tiver mais que uma solução, o reasoner lançará um erro de “inconsistente ontologies”.



<input checked="" type="checkbox"/> Functional	Equivalent To +
<input type="checkbox"/> Inverse functional	SubProperty Of +
<input type="checkbox"/> Transitive	Inverse Of +
<input type="checkbox"/> Symmetric	Domains (intersection) +
<input type="checkbox"/> Asymmetric	Var
<input type="checkbox"/> Reflexive	Ranges (intersection) +
<input type="checkbox"/> Irreflexive	D1

São as object properties que efectivamente permitem saber os valores os resultados que se deve atribuir às variáveis do problema de forma a obter a solução. Após o reasoner correr, estas object properties serão inferidas na instância fml.

Entrega e anexos

Os ficheiros entregues são os seguintes:

- F2CSPtoOWL.py
 - Script em Python 3 responsável por converter F2CSP em OWL
- 2x2F2CSP.txt
 - Ficheiro F2CSP referente ao jogo enunciado anteriormente
- SudokuToF2CSP.py
 - Script em Python 3 responsável por gerar sudokus em F2CSP, foi esta script a utilizada para gerar os ficheiros F2CSP usados no desenvolvimento do script
- Ficheiros F2CSP
 - 2x2F2CSP.txt
 - Referente ao jogo enunciado na explicação do desenvolvimento
 - Sudoku4x4_F2CSP_B.txt
 - Sudoku 4x4 com 15 pistas, 1 solução possível
 - Sudoku9x9_F2CSP_C.txt
 - Sudoku 9x9 com 35 pistas, 1 solução possível
 - Sudoku9x9_F2CSP_D.txt
 - Sudoku 9x9 com 40 pistas, 1 solução possível
 - Sudoku9x9_F2CSP_E.txt
 - Sudoku 9x9 com 80 pistas, 1 solução possível

```

1 Prefix(=<http://www.semanticweb.org/grupo21/ontologies/2019/4/grupo21#>)
2 Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
3 Prefix(rdf:=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
4 Prefix(xml:=<http://www.w3.org/XML/1998/namespace>)
5 Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
6 Prefix(rdfs:=<http://www.w3.org/2000/01/rdf-schema#>)
7
8
9 Ontology(<http://www.semanticweb.org/grupo21/ontologies/2019/4/2x2>
10
11 Declaration(Class(:D1))
12 Declaration(Class(:Fml))
13 Declaration(Class(:Var))
14 Declaration(ObjectProperty(:v11))
15 Declaration(ObjectProperty(:v12))
16 Declaration(ObjectProperty(:v21))
17 Declaration(ObjectProperty(:v22))
18 Declaration(NamedIndividual(:d1val1))
19 Declaration(NamedIndividual(:d1val2))
20 Declaration(NamedIndividual(:fml))
21 Declaration(NamedIndividual(:map))
22 #####
23 # Object Properties
24 #####
25
26 # Object Property: :v11 (:v11)
27
28 FunctionalObjectProperty(:v11)
29 ObjectPropertyDomain(:v11 :Var)
30 ObjectPropertyRange(:v11 :D1)
31
32 # Object Property: :v12 (:v12)
33
34 FunctionalObjectProperty(:v12)
35 ObjectPropertyDomain(:v12 :Var)
36 ObjectPropertyRange(:v12 :D1)
37
38 # Object Property: :v21 (:v21)
39
40 FunctionalObjectProperty(:v21)
41 ObjectPropertyDomain(:v21 :Var)
42 ObjectPropertyRange(:v21 :D1)
43
44 # Object Property: :v22 (:v22)
45
46 FunctionalObjectProperty(:v22)
47 ObjectPropertyDomain(:v22 :Var)
48 ObjectPropertyRange(:v22 :D1)
49
50
51 #####
52 # Classes
53 #####
54
55 # Class: :D1 (:D1)
56
57 EquivalentClasses(:D1 ObjectOneOf(:d1val1 :d1val2 ))
58
59 # Class: :Fml (:Fml)
60
61 EquivalentClasses(:Fml ObjectIntersectionOf( ObjectComplementOf(
ObjectIntersectionOf( ObjectHasValue(:v11 :d1val1) ObjectHasValue(:v12 :d1val1)))
ObjectComplementOf( ObjectIntersectionOf( ObjectHasValue(:v11 :d1val2)
ObjectHasValue(:v12 :d1val2))) ObjectComplementOf( ObjectIntersectionOf(
ObjectHasValue(:v11 :d1val1) ObjectHasValue(:v21 :d1val1))) ObjectComplementOf(
ObjectIntersectionOf( ObjectHasValue(:v11 :d1val2) ObjectHasValue(:v21 :d1val2)))
ObjectComplementOf( ObjectIntersectionOf( ObjectHasValue(:v12 :d1val1)
ObjectHasValue(:v22 :d1val1))) ObjectComplementOf( ObjectIntersectionOf(
ObjectHasValue(:v12 :d1val2) ObjectHasValue(:v22 :d1val2))) ObjectComplementOf(
ObjectIntersectionOf( ObjectHasValue(:v21 :d1val1) ObjectHasValue(:v22 :d1val1)))
ObjectComplementOf( ObjectIntersectionOf( ObjectHasValue(:v21 :d1val2)
ObjectHasValue(:v22 :d1val2))) ObjectIntersectionOf( ObjectHasValue(:v11 :d1val1)
ObjectHasValue(:v12 :d1val2))))))

```

```
62
63 # Class: :Var (:Var)
64
65 EquivalentClasses(:Var ObjectExactCardinality(1 :v11 :D1))
66 EquivalentClasses(:Var ObjectExactCardinality(1 :v12 :D1))
67 EquivalentClasses(:Var ObjectExactCardinality(1 :v21 :D1))
68 EquivalentClasses(:Var ObjectExactCardinality(1 :v22 :D1))
69
70
71 #####
72 #   Named Individuals
73 #####
74
75 # Individual: :dlval1 (:dlval1)
76
77 ClassAssertion(:D1 :dlval1)
78
79 # Individual: :dlval2 (:dlval2)
80
81 ClassAssertion(:D1 :dlval2)
82
83 # Individual: :fml (:fml)
84
85 ClassAssertion(:Fml :fml)
86 SameIndividual(:fml :map)
87
88 # Individual: :map (:map)
89
90 ClassAssertion(:Var :map)
91
92
93 DifferentIndividuals(:dlval1 :dlval2 )
94 )
```

```

1  Londres é uma cidade.
2
3  John McKenzie é uma pessoa.
4
5  AI Log é uma empresa. Que tem 40 funcionário com sede em Londres. Em que John
  McKenzie é o CEO.
6
7  Para além do CEO tem os seguintes funcionários (que são pessoas):
8
9  José, Ana e Lopes.
10 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
11 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
12 @prefix foaf: <http://xmlns.com/foaf/0.1/> .
13 @prefix comp: <http://companies.world.org/> .
14 @prefix : <http://ec-tp-l.pt/#> .
15
16
17 :ld
18   a :cidade ;
19   rdfs:label "Londres" .
20
21 :cidade
22   a rdfs:Class ;
23   rdfs:label "Cidade" .
24
25 :jm
26   a foaf:Person ;
27   foaf:name "John McKenzie" .
28
29 :ailog
30   a :empresa ;
31   comp:num_funcionarios "40" ;
32   rdfs:label "AI Log" ;
33   comp:sede :ld ;
34   comp:ceo :jm .
35
36
37 :ailog
38   comp:tem_funcionario :f1 , :f2 , :f3 .
39
40 :f1
41   a foaf:Person ;
42   foaf:name "José" .
43 :f2
44   a foaf:Person ;
45   foaf:name "Ana" .
46 :f3
47   a foaf:Person ;
48   foaf:name "Lopes" .
49
50 @prefix : <http://www.algumas.equipas.org/#> .
51 @prefix sp: <http://www.sports.org/#> .
52 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
53 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
54
55 :astros rdf:type sp:Equipa ;
56         sp:tem_jogador :joao , :manel , :paulo , :pedro , :ze ;
57         sp:nome_equipa "Astros FC"^^xsd:string .
58
59 :fcl rdf:type sp:Equipa ;
60      sp:tem_jogador :carlos , :jacinto , :lemos , :rodolfo , :silva ;
61      sp:nome_equipa "FC Lisboa"^^xsd:string .
62
63 :carlos rdf:type :Pessoa ;
64         sp:tem_irmao :rodolfo ;
65         sp:altura "1.95"^^xsd:float .
66
67 :jacinto rdf:type sp:Pessoa ;
68         sp:tem_irmao :manel ;
69         sp:altura "1.80"^^xsd:float .
70 :lemos rdf:type sp:Pessoa ;
71        sp:altura "2.05"^^xsd:float .
72 :rodolfo rdf:type sp:Pessoa .

```

```

73 :silva rdf:type sp:Pessoa .
74 :joao rdf:type sp:Pessoa ;
75     sp:altura "1.85"^^xsd:float .
76 :manel rdf:type sp:Pessoa ;
77     sp:altura "1.92"^^xsd:float .
78 :paulo rdf:type sp:Pessoa ;
79     sp:altura "1.87"^^xsd:float .
80 :pedro rdf:type sp:Pessoa .
81 :ze rdf:type sp:Pessoa ;
82     sp:altura "1.94"^^xsd:float .
83
84 Escreva uma query SPARQL que permita obter os jogadores que tenham um irmão a jogar
na mesma equipa.
85 @prefix : <http://www.algumas.equipas.org/#> .
86 @prefix sp: <http://www.sports.org/#> .
87 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
88 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
89 SELECT DISTINCT ?jogador
90 WHERE {
91     ?equipa rdf:type sp:Equipa .
92     ?equipa2 rdf:type sp:Equipa .
93     ?equipa sp:tem_jogador ?jogador .
94     ?jogador sp:tem_irmao ?irmao .
95     ?equipa2 sp:tem_jogador ?irmao .
96 }
97 Considerando, para além das que viu no guião, as seguintes propriedades:
98 - dbo:birthYear, que permite obter o ano de nascimento de uma pessoa;
99 - dbo:deathYear, que permite obter o ano com que uma pessoa morreu.
100
101 Escreva uma query SPARQL, sobre a dbpedia, que permita obter os nomes das pessoas
nascidas no século XX
102 e que morreram com 100 ou mais anos. O resultado deverá ser uma tabela com quatro
colunas (nome, ano de
103 nascimento, ano da morte, idade da morte) ordenada pela idade da morte
(decrescente). Deverão ser apresentados
104 apenas os primeiros 20 resultados.
105 PREFIX dbo: <http://dbpedia.org/ontology/>
106 PREFIX dbr: <http://dbpedia.org/resource/>
107 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
108 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
109 SELECT DISTINCT ?name (YEAR( ?birth) as ?yb) (YEAR( ?death) as ?yd)
110 WHERE {
111     ?person dbo:birthPlace dbr:London .
112     ?person dbo:birthDate ?birth .
113
114     ?person dbo:deathDate ?death .
115
116     ?person rdfs:label ?name .
117     FILTER (lang(?name) = "en") .
118     FILTER ((?birth > "1900-01-01"^^xsd:date) && (YEAR(?death) - YEAR(?birth)) >= 100 ) .
119 }
120 ORDER BY (?birth)

```