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

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 analise 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 dai 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 influência 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) influência a forma como é escrito a condição.

Tome como exemplo a escrita da 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) )
    )
}
```

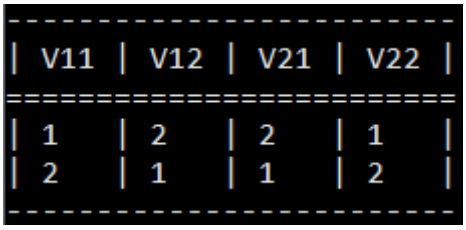
Execução da query

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)	tempo (sec)	tempo (sec)
1,949	1,351	1,341
1,968	1,347	1,338
1,886	1,341	1,331
1,863	1,350	1,310
1,932	1,345	1,360
Média	**Média**	**Média**
1,9196	**1,3468**	**1,3360**

Analisando os resultamos, 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