SolveSurvo
Z3

José Pinheiro pg23208@alunos.uminho.pt
Tiago Guimarães pg22832@alunos.uminho.pt

28 de Abril de 2013

1 Survo

O Survo é um puzzle onde o objectivo é preencher uma tabela m * n com inteiros entre 1...M*N, de tal maneira que cada um desses números aparece apenas uma vez e a soma da linha e coluna é igual ao inteiro do fundo da tabela e do lado direito da tabela. Podem ser dados inteiros na tabela inicial para garantir a unicidade da solução e para fazer a tarefa mais fácil.

2 Z3

Como SMT-solver estamos a usar o z3 da Microsoft. Escolhemos este smt-solver porque nós foi recomendado nas aulas de vfs. Como linguagem de interface com o z3 escolhemos o python (z3Py) devido a ser uma linguagem simples, eficaz e devido a mesma suportar listas por compreensão.

3 solveSurvo

O solveSurvo é invocado com $ solveSurvo.py <ficheiro_input>, se conseguir resolver o puzzle devolve um ficheiro output com a solução do puzzle. Necessita que o z3 esteja instalado e conectado com o python.

3.1 Ficheiros input

Para representar um tabuleiro de survo, escolhemos uma sintaxe muito simples. A primeira linha é os valores que o as colunas tem de somar, e a segunda linha é os valores que as linhas tem de somar. Para representar os pontos dados, temos a posição da matriz e o valor correspondente.

Exemplo :

```
27 16 10 25
30 18 30
1 2 6
2 1 8
3 3 3
```

3.2 Ficheiros output

o SolveSurvo da como output um ficheiro com a representação da matriz obtida e os valores que as linhas e as colunas tem de somar.

Exemplo :

```
[7, 6, 5, 12]
[8, 3, 2, 5]
```

```
[12, 7, 3, 8]

Linhas   : [30, 18, 30]
Colunas  : [27, 16, 10, 25]
```

3.3 Codificação do Survo

Para codificar o puzzle, codificamos 4 propriedades do puzzle:

- **(A)** Cada célula tem valores entre 1..m*n;
- **(B)** Cada número é único no tabuleiro;
- **(C)** O somatório das linhas tem de ser igual ao valor na coluna final;
- **(D)** O somatório das colunas tem de ser igual ao valor na linha final.

Sendo que X é a matriz que representa o tabuleiro, e cx_size*cy_size é tamanho da mesma então a sua codificação é:

(A)

```
[ And(1 <= X[j][i], X[j][i] <= cx_size*cy_size) for i in range(cx_size)
for j in range(cy_size) ]
```

(B)

```
[ Distinct(X[j][i]) for i in range(cx_size) for j in range(cy_size) ]
```

(C)

```
[ cy[i] == sum(X[i]) for i in range(cy_size) ]
```

(D)

```
[ cx[j] == sum([ X[i][j] for i in range(cy_size) ])
             for j in range(cx_size) ]
```