SAT solving - Questões para Avaliação

Luís Ribeiro (A85954)
Mestrado em Engenharia Informática
Universidade do Minho

1 Puzzle

A “Associação Recreativa do Paraíso” tem o seguinte conjunto de regras:

- Os sócios loiros não podem ir ao Sábado.
- Quem não for adulto têm que usar chapéu.
- Cada sócio usa o anel ou não usa chapéu.
- Um sócio vai ao Sábado se e só se é adulto.
- Todos os sócios adultos têm que usar anel.
- Quem usa anel tem que ser loiro.

1.1 Alínea 1

Por forma a codificar este puzzle como problema SAT, defina um conjunto adequado de variáveis proposicionais, exprima as regras acima como fórmulas proposicionais, e converta essas fórmulas para CNF.

Variáveis proposicionais e correspondente associação:

ser sócio	$\rightarrow$	1
ser loiro	$\rightarrow$	2
ir ao sábado	$\rightarrow$	3
ser adulto	$\rightarrow$	4
usar chapéu	$\rightarrow$	5
usar anel	$\rightarrow$	6

Restrições definidas em Lógica Proposicional

— Os sócios loiros não podem ir ao Sábado:

$$1 \wedge 2 \Longrightarrow \neg 3 \equiv \neg(1 \wedge 2) \vee \neg 3 \equiv \neg 1 \vee \neg 2 \vee \neg 3$$

— Quem não for adulto têm que usar chapéu:

$$\neg 4 \Longrightarrow 5 \equiv \neg(\neg 4) \vee 5 \equiv 4 \vee 5$$

— Cada sócio usa o anel ou não usa chapéu:

$$1 \Longrightarrow 6 \vee \neg 5 \equiv \neg 1 \vee 6 \vee \neg 5$$

— Um sócio vai ao Sábado se e só se é adulto. (Primeira implicação, porque "se e só se"é uma equivalência)

$$(1 \wedge 3) \Longrightarrow 4 \equiv \neg 1 \vee \neg 3 \vee 4$$

— Um sócio vai ao Sábado se e só se é adulto. (Segunda implicação, porque "se e só se"é uma equivalência)

$$4 \Longrightarrow (1 \wedge 3) \equiv \neg 4 \vee (1 \wedge 3) \equiv (\neg 4 \vee 1) \wedge (\neg 4 \vee 3)$$

— Todos os sócios adultos têm que usar anel.

$$(1 \wedge 4) \Longrightarrow 6 \equiv \neg(1 \wedge 4) \vee 6 \equiv \neg 1 \vee \neg 4 \vee 6$$

— Quem usa anel tem que ser loiro.

$$6 \Longrightarrow 2 \equiv \neg 6 \vee 2$$

Conversão para CNF

```
p cnf 6 8
-1 -2 -3 0
4 5 0
-1 6 -5 0
-1 -3 4 0
-4 1 0
-4 3 0
-1 -4 6 0
-6 2 0
```

Estes dados da conversão para CNF encontram-se no ficheiro `puzzle-e1.cnf`.

1.2 Alínea 2

Comprove, utilizando um SAT solver, que o problema é consistente.

Para provar que este problema é consistente, temos que verificar se é satisfazível dado as condições acima referidas. Para isso usei `minisat`, um *SAT solver*, dando-lhe como *input* o ficheiro `puzzle-e1.cnf`.

O resultado devolvido foi **SAT** e com isto podemos concluir que o problema é consistente. Este encontra-se no ficheiro `puzzle-e2`.

```
SAT
-1 2 -3 -4 5 6 0
```

O resultado permite-nos concluir que é possível existir uma pessoa loira que não é sócia nem adulta, usa chapéu e anel e não vai aos sábados.

1.3 Alínea 3

Use agora o SAT solver para o ajudar a responder às seguintes questões.

1.3.1 Alínea (a)

A afirmação "Quem usa anel não pode ir ao Sábado." é correcta?

Para provar que isto é verdade, temos que demonstrar que esta afirmação é uma consequência lógica das condições acima. Para isso temos que negar esta afirmação e verificar se o problema é NÃO satisfazível.

Afirmação:

$$6 \implies \neg 3 \equiv \neg 6 \vee \neg 3$$

Negação da afirmação:

$$\neg(\neg 6 \vee \neg 3) \equiv \neg(\neg 6) \wedge \neg(\neg 3) \equiv 6 \wedge 3$$

Restrições adicionadas:

```
3 0
6 0
```

Resultado: O problema é satisfazível (SAT), logo a afirmação é falsa, por não ser uma consequência lógica às condições anteriores.

O ficheiro correspondente a esta alínea tem nome de `puzzle-e3a.cnf`.

1.3.2 Alínea (b)

Pode um sócio de chapéu ser loiro?

Para provar que isto é verdade, temos que demonstrar que o problema continua a ser satisfazível, adicionando estas restrições às condições já feitas, isto é, se encontra solução. Note que não queremos provar uma consequência lógica, portanto não há a necessidade de negar a afirmação.

Afirmação:

$$1 \wedge 2 \wedge 5$$

Restrições adicionadas:

```
1 0
2 0
5 0
```

Resultado: O problema continua a ser satisfazível (SAT), então uma solução em que existe um sócio de chapéu e loiro é possível dentro destas condições.

O ficheiro correspondente a esta alínea tem nome de `puzzle-e3b.cnf`.

1.3.3 Alínea (c)

A afirmação "Afinal a associação não pode ter sócios adultos." é correcta?

Para provar que isto é verdade, temos que demonstrar que esta afirmação é uma consequência lógica das condições acima. Para isso temos que negar esta afirmação e verificar se o problema é NÃO satisfazível.

Afirmação:

$$\neg(1 \wedge 4) \equiv \neg 1 \vee \neg 4$$

Negação da afirmação:

$$\neg(\neg 1 \vee \neg 4) \equiv \neg(\neg 1) \wedge \neg(\neg 4) \equiv 1 \wedge 4$$

Restrições adicionadas:

```
1 0
4 0
```

Resultado: O problema é NÃO satisfazível (UNSAT), logo a afirmação é verdadeira, por ser uma consequência lógica às condições anteriores.

O ficheiro correspondente a esta alínea tem nome de `puzzle-e3c.cnf`.

2 Sudoku

Os puzzles Sudoku são problemas de colocação de números inteiros entre 1 e N^2 numa matriz quadrada de dimensão N^2, por forma a que cada coluna e cada linha contenha todos os números, sem repetições. Além disso, cada matriz contém N^2 sub-matrizes quadradas disjuntas, de dimensão N, que deverão também elas conter os números entre 1 e N^2.
Cada problema é dado por uma matriz parcialmente preenchida, cabendo ao jogador completá-la. Exemplo de um problema para N=2:

4		1	
	2		
		3	
	4		1

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

O problema pode ser codificado em lógica proposicional criando uma variável proposicional para cada triplo (l, c, n), onde l é uma linha, c é uma coluna, e n é um número. $x_{l,c,n} = 1$ se na linha l, coluna c, estiver o número n, caso contrário será 0.

2.1 Alínea 1

Modele o problema do Sudoku (para N=2) como um problema SAT, escrevendo as restrições correspondentes às regras do puzzle. Acrescente depois as restrições correspondentes à definição de um tabuleiro concreto (por exemplo, o da figura).

O primeiro passo no desenvolvimento das fórmulas proposicionais que irão representar o problema, foi a atribuição de uma variável que irá representar o trio (l,c,n). Assim, com este modelo, teremos **64** $(= (N^2)^3)$ variáveis proposicionais associadas a cada posição $x_{l,c,n}$.
O cálculo da variável associada a cada posição $x_{l,c,n}$ é feita de forma consistente e que pode ser aplicado a um $N! = 2$.

```
(l-1)*(nmr_colunas*nmr_ns) + (c-1)*(nmr_linhas) + n
```

Sendo o $nmr\_colunas$ a **quantidade de colunas**, o nmr_ns a **quantidade de números** possíveis em cada posição e o nmr_linhas a **quantidade de linhas**. Estas variáveis dependem todas do N, e tem todas os mesmo valor, $nmr\_colunas = nmr\_ns = nmr\_linhas = N^2$. Neste caso $N = 2$, então cada uma das variáveis terá o valor de 4.

Para uma melhor compreensão, apresento alguns exemplos da atribuição de $x_{l,c,n}$.

$$
\begin{array}{lcl}
x_{1,1,1} & \rightarrow & 1 \\
x_{1,1,2} & \rightarrow & 2 \\
... & & \\
x_{1,2,1} & \rightarrow & 5 \\
x_{1,2,2} & \rightarrow & 6 \\
... & & \\
x_{2,1,1} & \rightarrow & 17 \\
x_{2,1,2} & \rightarrow & 18 \\
... & & \\
x_{4,4,4} & \rightarrow & 64
\end{array}
$$

Restrições definidas em Lógica Proposicional

— Pelo menos um número n dentro de cada **bloco** (l, c).

$$(1 \lor 2 \lor 3 \lor 4) \land (5 \lor 6 \lor 7 \lor 8) ... \land (61 \lor 62 \lor 63 \lor 64)$$

— Apenas um número n dentro de cada **bloco** (l, c). **Restringir através de implicações ($\Longrightarrow$).**

`Exemplificando:` O bloco $(1, 1)$ é definido pelas posições $\{1, 2, 3, 4\}$, por poder ter um número variado entre 1 a 4. Se dentro desse bloco estiver o número $n = 1$, então o restante conjunto $\{2, 3, 4\}$ tem que obrigatoriamente ser falso. Temos assim,

$$1 \Longrightarrow (\neg 2 \land \neg 3 \land \neg 4) \equiv (\neg 1 \lor \neg 2) \land (\neg 1 \lor \neg 3) \land (\neg 1 \lor \neg 4)$$

Agora temos que repetir este processo para $n = 2$, $n = 3$ e $n = 4$. Simplificando as implicações e eliminando as restrições repetidas, para o bloco $(1, 1)$ teremos as seguintes restrições:

$$(\neg 1 \lor \neg 2) \land (\neg 1 \lor \neg 3) \land (\neg 1 \lor \neg 4) \land (\neg 2 \lor \neg 3) \land (\neg 2 \lor \neg 4) \land (\neg 3 \lor \neg 4)$$

Para o cálculo das restantes restrições deste domínio, teremos que aplicar o mesmo raciocínio para cada bloco (l, c).

— Não pode haver números repetidos dentro de cada **sub-Matriz** (bloco com 4 posições). **Restringir através de implicações ($\Longrightarrow$).**

`Exemplificando:` A primeira sub-Matriz é definida pelas posições $\{(1, 1), (1, 2), (2, 1), (2, 2)\}$. Se dentro dessa sub-Matriz o número $n = 1$ já estiver numa das posições definidas, então não poderá estar noutra qualquer. Isto é, se $x_{1,1,1} = 1$ então $x_{1,2,1} = 0 \land x_{2,1,1} = 0 \land x_{2,2,1} = 0$. Temos assim,

$$1 \Longrightarrow (\neg 5 \land \neg 17 \land \neg 21) \equiv (\neg 1 \lor \neg 5) \land (\neg 1 \lor \neg 17) \land (\neg 1 \lor \neg 21)$$

Agora temos que repetir este processo para as posições $(1,2)$, $(2,1)$ e $(2,2)$. Simplificando as implicações e eliminando as restrições repetidas, para sub-Matriz $\{(1,1),(1,2),(2,1),(2,2)\}$, com a **atribuição do** $n = 1$, teremos as seguintes restrições:

$$(\neg 1 \vee \neg 5) \wedge (\neg 1 \vee \neg 17) \wedge (\neg 1 \vee \neg 21) \wedge (\neg 5 \vee \neg 17) \wedge (\neg 5 \vee \neg 21) \wedge (\neg 17 \vee \neg 21)$$

Para o cálculo das restantes restrições deste domínio, temos que primeiro atribuir todos os possíveis valores de n, a cada sub-Matriz definida.

— Não pode haver números repetidos na mesma **linha** ou na mesma **coluna**. **Restringir através de implicações** ($\Longrightarrow$)**.**

Como o processo de restringir a repetição de números na mesma linha é análogo às colunas, decidi explicar dentro do mesmo ponto, com respetivos exemplos.

De notar também que o raciocínio aplicado aqui é o mesmo da última restrição apresentada, porque uma linha ou uma coluna pode ser vista como uma sub-Matriz linear ou colunar, respetivamente.

`Exemplificando:` A **primeira linha** pode ser vista como uma sub-Matriz definida pelas posições $\{(1,1),(1,2),(1,3),(1,4)\}$. Se dentro dessa sub-Matriz o número $n = 1$ já estiver numa das posições definidas, então não poderá estar noutra qualquer. Isto é, se $x_{1,1,1} = 1$ então $x_{1,2,1} = 0 \wedge x_{1,3,1} = 0 \wedge x_{1,4,1} = 0$. Temos assim,

$$1 \Longrightarrow (\neg 5 \wedge \neg 9 \wedge \neg 13) \equiv (\neg 1 \vee \neg 5) \wedge (\neg 1 \vee \neg 9) \wedge (\neg 1 \vee \neg 13)$$

Agora temos que repetir este processo para as posições $(1,2)$, $(1,3)$ e $(1,4)$. Simplificando as implicações e eliminando as restrições repetidas, para a linha $\{(1,1),(1,2),(1,3),(1,4)\}$, com a **atribuição do** $n = 1$, teremos as seguintes restrições:

$$(\neg 1 \vee \neg 5) \wedge (\neg 1 \vee \neg 9) \wedge (\neg 1 \vee \neg 13) \wedge (\neg 5 \vee \neg 9) \wedge (\neg 5 \vee \neg 13) \wedge (\neg 9 \vee \neg 13)$$

Para o cálculo das restantes restrições deste domínio, temos que primeiro atribuir todos os possíveis valores de n, a cada linha definida.

`Exemplificando:` A **primeira coluna** pode ser vista como uma sub-Matriz definida pelas posições $\{(1,1),(2,1),(3,1),(4,1)\}$. Se dentro dessa sub-Matriz o número $n = 1$ já estiver numa das posições definidas, então não poderá estar noutra qualquer. Isto é, se $x_{1,1,1} = 1$ então $x_{2,1,1} = 0 \wedge x_{3,1,1} = 0 \wedge x_{4,1,1} = 0$. Temos assim,

$$1 \Longrightarrow (\neg 17 \wedge \neg 33 \wedge \neg 49) \equiv (\neg 1 \vee \neg 17) \wedge (\neg 1 \vee \neg 33) \wedge (\neg 1 \vee \neg 49)$$

Agora temos que repetir este processo para as posições $(2,1)$, $(3,1)$ e $(4,1)$. Simplificando as implicações e eliminando as restrições repetidas, para a linha $\{(1,1),(2,1),(3,1),(4,1)\}$, com a **atribuição do** $n = 1$, teremos as seguintes restrições:

$$(\neg 1 \vee \neg 17) \wedge (\neg 1 \vee \neg 33) \wedge (\neg 1 \vee \neg 49) \wedge (\neg 17 \vee \neg 33) \wedge (\neg 17 \vee \neg 49) \wedge (\neg 33 \vee \neg 49)$$

Para o cálculo das restantes restrições deste domínio, temos que primeiro atribuir todos os possíveis valores de n, a cada linha definida.

Todas estas restrições encontram-se definidas e exemplificadas dentro do ficheiro `sudoku-e1.cnf`.

2.2 Alínea 2

Converta as restrições geradas para CNF.

A conversão para CNF está explícita no ficheiro `sudoku-e2.cnf`.

Para facilitar o processo de conversão de Lógica Proposicional para CNF, usei as ferramentas do `VIM` para a transformação direta e repetitiva. Em baixo, apresento as técnicas usadas, mas também se pode encontrar no ficheiro `sudoku-e2-vim.txt`.

```
:%s//\n/ (Substituição dos and's proposicionais por \n,
           para haver uma separação por linhas)
:%s/(// :%s/)// (Tirar os parênteses)
:%s/¬/-/  (Substituição dos not's proposicionais pelo caracter \-)
:%s///   (Remoção dos or's proposicionais)
:%s/c//   (Remoção dos c's que significam comentários)
:%s/$/ 0/ (Adicionar ao fim de cada linha um '0')
:%s/ \+/ / (Normalização dos espaços)
:%sort u  (Eliminação de condições repetidas)
```

Usei `minisat`, um *SAT solver*, dando-lhe como *input* o ficheiro `sudoku-e2.cnf`.

O resultado devolvido foi **SAT** e com isto podemos concluir que o problema é consistente.

```
SAT
-1 -2 -3 4 -5 -6 7 -8 9 -10 -11 -12 -13 14 -15 -16
17 -18 -19 -20 -21 22 -23 -24 -25 -26 -27 28 -29 -30 31 -32
-33 34 -35 -36 37 -38 -39 -40 -41 -42 43 -44 -45 -46 -47 48
-49 -50 51 -52 -53 -54 -55 56 -57 58 -59 -60 61 -62 -63 -64 0
```

Fazendo o processo inverso e fazendo a respetiva correspondência, temos a seguinte solução sudoku:

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

2.3 Alínea 3

Resolva o problema usando um SAT solver. Sugestão: Implemente um pequeno programa (por exemplo, em C ou em Phyton) para gerar o ficheiro DIMACS CNF para enviar ao SAT solver. Note que pode criar uma matriz ou um dicionário, X, de forma a fazer o mapeamento entre cada variável proposicional X[l][c][n] e o valor inteiro que lhe corresponde no formato DIMACS.

Como consigo associar a cada posição $x_{l,c,n}$ de forma consistente, onde o N pode variar, não tenho a necessidade de criar uma matriz ou dicionário associado a cada posição.

```
(l-1)*(nmr_colunas*nmr_ns) + (c-1)*(nmr_linhas) + n
```

Sendo o *nmr_colunas* a **quantidade de colunas**, o *nmr_ns* a **quantidade de números** possíveis em cada posição e o *nmr_linhas* a **quantidade de linhas**. Estas variáveis dependem todas do N, e tem todas os mesmo valor, $nmr\_colunas = nmr\_ns = nmr\_linhas = N^2$.

O programa criado tem nome de `sudoku-program` e recebe como *input* um ficheiro, onde começa com a atribuição do valor N, seguido por posições que nós queremos já preenchidas no nosso Sudoku.
Para testar diferentes valores de N, criei 2 ficheiros de *input* com nomes `sudoku-n2` e `sudoku-n3`, para valores $N = 2$ e $N = 3$, respetivamente.
Em baixo apresento o ficheiro `sudoku-n2`. Este exemplo é igual ao exemplo dado no enunciado.

```
N=2
(1,1,4)
(1,3,1)
(2,2,2)
(3,3,3)
(4,2,4)
(4,4,1)
```

A primeira linha do ficheiro de *input* deve ser neste formato, para a leitura do N e o cálculo do número de colunas, linhas e números ($= N^2$).

Na implementação do programa criei uma **Matriz de Condições** (Lista de Listas), onde cada lista da Matriz é uma condição separados por $\vee$. As listas são separadas entre elas por $\wedge$. Desta forma, consigo facilmente converter para o formato CNF.
O cálculo destas Condições/Restrições foi diferente das alíneas anteriores. Aqui, invés de fazer implicações de negações, digo que cada número n deverá estar dentro de cada sub-Matriz, de cada linha l e de cada coluna c, negando assim a possibilidade de haver repetidos nestas. Isto permitiu-me calcular facilmente as condições através de funções/métodos auxiliares, e reduziu-me drasticamente o número de condições CNF.

Todas isto encontra-se dentro do ficheiro executável `sudoku-program`.
\$ ./sudoku-program *ficheiro_input*