Métodos Formais em Engenharia de Software

Trabalho Prático 1 - SAT Solving

Nome: Francisco Reis Izquierdo

Número: PG50384

Curso: Mestrado em Engenharia Informática

> **Nota:** Para a realização do trabalho prático, foi necessário o auxílio de um *solver*, nomeadamente o **PySAT**, por forma a obter resposta ás questões do enunciado, sendo que a primeira etapa passou pela instalação da biblioteca necessária ao supramencionado *solver*.

```
!pip install python-sat[pblib,aiger]
```

Exercício 1

Após a leitura do enunciado, conseguimos identificar o conjunto de variáveis proposicionais que nos irão ajudar a modelar o problema. Assim, temos o seguinte conjunto de variáveis proposicionais.

- CPU1 = 1
- CPU2 = 2
- RAM1 = 3
- RAM2 = 4
- MB1 = 5
- MB2 = 6
- PG1 = 7
- PG2 = 8
- PG3 = 9
- MON1 = 10
- MON2 = 11
- MON3 = 12

Obtemos assim um conjunto de 12 variáveis proposicionais. Com isto, o passo seguinte foi transformar o conjunto de restrições e regras que descrevem o âmbito do problema sobre a forma de fórmulas proposicionais, convertendo-as ao formato **CNF**.

Restrições

- *"Cada computador tem que ter obrigatoriamente uma única motherboard..."*

> $(MB_1 \vee MB_2) \wedge (\neg MB_1 \vee \neg MB_2)$

- *"...um único CPU..."*

> $(CPU1 \vee CPU2) \wedge (\neg CPU1 \vee \neg CPU2)$

- *"...uma única placa gráfica..."*

> $(PG1 \vee PG2 \vee PG3) \wedge (\neg PG1 \vee \neg PG2 \vee \neg PG3) \wedge (\neg PG1 \vee \neg PG2 \vee PG3) \wedge (\neg PG1 \vee PG2$

- *"...e uma única memória RAM."*

> $(RAM1 \vee RAM2) \wedge (\neg RAM1 \vee \neg RAM2)$

- *"O computador poderá ter ou não ter monitores."*

> Esta restrição em nada implica.

Regras

- *"A motherboard MB1 quando combinada com a placa gráfica PG1, obriga à utilização da RAM1."*

> $(MB_1 \wedge PG_1) \rightarrow RAM_1 \equiv$
>
> {Conversão para formato CNF}
>
> $\equiv \neg(MB1 \wedge PG1) \vee RAM1 \equiv$
>
> $\equiv \neg MB1 \vee \neg PG1 \vee RAM1$

- *"A placa gráfica PG1 precisa do CPU1, excepto quando combinada com uma memória RAM2."*

> $(PG_1 \wedge \neg RAM_2) \rightarrow CPU_1 \equiv$
>
> {Conversão para formato CNF}
>
> $\equiv \neg(PG_1 \wedge \neg RAM_2) \vee CPU_1 \equiv$
>
> $\equiv \neg PG_1 \vee RAM_2 \vee CPU_1 \equiv$

- *"O CPU2 só pode ser instalado na motherboard MB2."*

> $CPU_2 \rightarrow MB_2 \equiv$
>
> {Conversão para formato CNF}
>
> $\equiv \neg CPU_2 \vee MB_2$

- *"O monitor MON1 para poder funcionar precisa da placa gráfica PG1 e da memória RAM2."*

> $MON_1 \rightarrow (PG_1 \wedge RAM_2) \equiv$
>
> {Conversão para formato CNF}
>
> $\equiv \neg MON_1 \vee (PG_1 \wedge RAM_2) \equiv$
>
> $\equiv (\neg MON_1 \vee PG_1) \wedge (\neg MON_1 \vee RAM_2)$

- "*O monitor MON2 precisa da memória RAM2 para poder trabalhar com a placa gráfica PG3.*"

> $(MON_2 \wedge PG_3) \rightarrow RAM_2 \equiv$
>
> {Conversão para formato CNF}
>
> $\equiv \neg(MON_2 \wedge PG_3) \vee RAM_2 \equiv$
>
> $\equiv \neg MON_2 \vee \neg PG_3 \wedge RAM_2$

Exercício 2

Dado o obtido no exercício anterior, uma vez que agora queremos provar se o modelo é **consistente** teremos então de, através do *solver* verificar se este é satisfazível, isto é, se existe, pleo menos uma solução. Com isto, é verificado que para cada variável proposicional do modelo, existe um valor tal que as várias fórmulas proposicionais do mesmo são todas verdadeiras.

Assim, o primeiro passo passa por transformar as várias fórmulas proposicionais no formato **DIMACS**.

```
p cnf 11 17
5 6 0
-5 -6 0
1 2 0
-1 -2 0
7 8 9 0
-7 -8 -9 0
-7 -8 9 0
-7 8 -9 0
7 -8 -9 0
3 4 0
-3 -4 0
-5 -7 3 0
-7 4 1 0
-2 6 0
-10 7 0
-10 4 0
-11 -9 4 0
```

No qual, em cabeçalho é feita a referência ao uso de 11 variáveis proposicionais (das 12 previamente definidas) ao longo de 17 cláusulas proposicionais.

> **Nota:** É de realçar o facto de que para este problema, apesar de haver 12 variáveis proposicionais, apenas 11 têm relevância para o desenvolvimento no que concerne ao *solver*, dado que o conjunto de restrições e regras que definem o problema, não envolvem umas das variáveis proposicionais, nomeadamente **MON3**.

Após isto, iremos usar o *solver* já mencionado, o **PySAT**.

```
from pysat.solvers import Minisat22

s = Minisat22()

s.add_clause([5, 6])
s.add_clause([-5, -6])
s.add_clause([1, 2])
s.add_clause([-1, -2])
s.add_clause([7, 8, 9])
s.add_clause([-7, -8, -9])
s.add_clause([-7, -8, 9])
s.add_clause([-7, 8, -9])
s.add_clause([7, -8, -9])
s.add_clause([3, 4])
s.add_clause([-3, -4])
s.add_clause([-5, -7, 3])
s.add_clause([-7, 4, 1])
s.add_clause([-2, 6])
s.add_clause([-10, 7])
s.add_clause([-10, 4])
s.add_clause([-11, -9, 4])

if s.solve():
    print("SAT")
    print(s.get_model())
else:
    print("UNSAT")

s.delete()
```

```
SAT
[1, -2, 3, -4, 5, -6, 7, -8, -9, -10, -11]
```

Após a análise do resultado obtido, confirma-se a existência de, pelo menos, uma solução, indo de encontro ao anteriormente referido. Uma vez que existe, pelo menos, uma solução o problema é satisfazível, sendo consequentemente consistente.

Exercício 3

Para este tipo de exercício iremos, primeiramente, formalizar a questão sobre a forma de lógica proposicional.

- **Alínea a)** *O monitor MON1 só poderá ser usado com uma motherboard MB1?*

> $MON_1 \rightarrow MB_1$

Para resolver esta questão, queremos provar se o contrário é impossível, isto é, se não há nenhuma solução que nos garanta que o monitor **MON1** não pode ser usado com uma *motherboard* **MB1**. Ou seja, queremos aplicar a seguinte fórmula proposicional:

$\Gamma \models F$ iff $\Gamma, \neg F$ UNSAT

A fórmula supramencionada, diz-nos que uma dada fórmula F é verdadeira se e só se, a negação da mesma for **insatisfazível**. Com isto, teremos de converter a fórmula proposicional que descreve a questão sobre o formato **CNF** sendo posteriormente negada e aplicada ao *solver*.

> $MON_1 \rightarrow MB_1$ ≡
>
> {Conversão para formato CNF}
>
> $\equiv \neg MON_1 \vee MB_1 \equiv$
>
> {Negação da fórmula}
>
> $MON_1 \wedge \neg MB_1$

```
from pysat.solvers import Minisat22

s = Minisat22()

s.add_clause([5, 6])
s.add_clause([-5, -6])
s.add_clause([1, 2])
s.add_clause([-1, -2])
s.add_clause([7, 8, 9])
s.add_clause([-7, -8, -9])
s.add_clause([-7, -8, 9])
s.add_clause([-7, 8, -9])
s.add_clause([7, -8, -9])
s.add_clause([3, 4])
s.add_clause([-3, -4])
s.add_clause([-5, -7, 3])
s.add_clause([-7, 4, 1])
s.add_clause([-2, 6])
s.add_clause([-10, 7])
s.add_clause([-10, 4])
s.add_clause([-11, -9, 4])
```

```
s.add_clause([10])
s.add_clause([-5])

if s.solve():
    print("SAT")
    print(s.get_model())
else:
    print("UNSAT")

s.delete()
```

```
SAT
[1, -2, -3, 4, -5, 6, 7, -8, -9, 10, -11]
```

Uma vez que existe uma solução, sendo por isso satisfazível, o monitor **MON1** não tem de ser combinado exclusivamente com uma *motherboard* MB1.

- **Alínea b):** "*Um cliente pode personalizar o seu computador da seguinte forma: uma motherboard MB1, o CPU1, a placa gráfica PG2 e a memória RAM1?*"

> $MB_1 \wedge CPU_1 \wedge PG_2 \wedge RAM_1$

Para esta questão queremos ver se existe alguma solução que contenha a combinação das variáveis proposicionais acima referidas. Com isto, fornecemos ao *solver* a fórmula proposicional que define a questão.

```
from pysat.solvers import Minisat22

s = Minisat22()

s.add_clause([5, 6])
s.add_clause([-5, -6])
s.add_clause([1, 2])
s.add_clause([-1, -2])
s.add_clause([7, 8, 9])
s.add_clause([-7, -8, -9])
s.add_clause([-7, -8, 9])
s.add_clause([-7, 8, -9])
s.add_clause([7, -8, -9])
s.add_clause([3, 4])
s.add_clause([-3, -4])
s.add_clause([-5, -7, 3])
s.add_clause([-7, 4, 1])
s.add_clause([-2, 6])
s.add_clause([-10, 7])
s.add_clause([-10, 4])
s.add_clause([-11, -9, 4])
s.add_clause([5])
s.add_clause([1])
s.add_clause([8])
s.add_clause([3])
```

```
if s.solve():
    print("SAT")
    print(s.get_model())
else:
    print("UNSAT")

s.delete()
```

```
SAT
[1, -2, 3, -4, 5, -6, -7, 8, -9, -10, -11]
```

Uma vez que existe solução, sendo por isso satisfazível, um cliente pode de facto personalizar o seu computador com os seguintes componentes: uma motherboard **MB1**, o **CPU1**, a placa gráfica **PG2** e a memória **RAM1**.

- **Alínea c):** "*É possível combinar a motherboard MB2, a placa gráfica PG3 e a RAM1 num mesmo computador?*"

> $MB_2 \land PG_3 \land RAM_1$

Para esta questão queremos ver se existe alguma solução que contenha a combinação das variáveis proposicionais acima referidas. Com isto, fornecemos ao *solver* a fórmula proposicional que define a questão.

```
from pysat.solvers import Minisat22

s = Minisat22()

s.add_clause([5, 6])
s.add_clause([-5, -6])
s.add_clause([1, 2])
s.add_clause([-1, -2])
s.add_clause([7, 8, 9])
s.add_clause([-7, -8, -9])
s.add_clause([-7, -8, 9])
s.add_clause([-7, 8, -9])
s.add_clause([7, -8, -9])
s.add_clause([3, 4])
s.add_clause([-3, -4])
s.add_clause([-5, -7, 3])
s.add_clause([-7, 4, 1])
s.add_clause([-2, 6])
s.add_clause([-10, 7])
s.add_clause([-10, 4])
s.add_clause([-11, -9, 4])
s.add_clause([6])
s.add_clause([9])
```

```
s.add_clause([3])



if s.solve():
    print("SAT")
    print(s.get_model())
else:
    print("UNSAT")

s.delete()
```

```
SAT
[1, -2, 3, -4, -5, 6, -7, -8, 9, -10, -11]
```

Uma vez que existe solução, sendo por isso satisfazível, é possível combinar de facto os seguintes componentes: motherboard **MB2**, a placa gráfica **PG3** e a **RAM1**.

- **Alínea d):** "*Para combinarmos a placa gráfica PG2 e a RAM1 temos que usar o CPU2?*"

> $PG_2 \land RAM_1 \to CPU_2$

Para resolver esta questão, queremos provar se o contrário é impossível, isto é, se não há nenhuma solução que nos garanta que a placa gráfica **PG2** e a memória **RAM1** possam ser combinadas sem ter que ser usado o **CPU2**. À semelhança do que fora feito na **alínea a)**, iremos ir de encontro à fórmula proposicional supramencionada, transformando a fórmula proposicional que descreve a questão no formato **CNF**, sendo esta posteriormente negada e aplicada ao **solver**.

> $PG_2 \land RAM_1 \to CPU_2 \equiv$
>
> {Conversão para formato CNF}
>
> $\equiv \neg(PG_2 \land RAM_1) \lor CPU_2 \equiv$
>
> $\equiv \neg PG_2 \lor RAM_1 \lor CPU_2$
>
> {Negação da fórmula}
>
> $PG_2 \land \neg RAM_1 \land \neg CPU_2$

```
from pysat.solvers import Minisat22

s = Minisat22()

s.add_clause([5, 6])
s.add_clause([-5, -6])
s.add_clause([1, 2])
s.add_clause([-1, -2])
s.add_clause([7, 8, 9])
s.add_clause([-7, -8, -9])
```

```
s.add_clause([-7, -8, 9])
s.add_clause([-7, 8, -9])
s.add_clause([7, -8, -9])
s.add_clause([3, 4])
s.add_clause([-3, -4])
s.add_clause([-5, -7, 3])
s.add_clause([-7, 4, 1])
s.add_clause([-2, 6])
s.add_clause([-10, 7])
s.add_clause([-10, 4])
s.add_clause([-11, -9, 4])
s.add_clause([8])
s.add_clause([-3])
s.add_clause([-2])

if s.solve():
    print("SAT")
    print(s.get_model())
else:
    print("UNSAT")

s.delete()
```

```
SAT
[1, -2, -3, 4, 5, -6, -7, 8, -9, -10, -11]
```

Uma vez que existe uma solução, sendo por isso satisfazível, a placa gráfica **PG2** e a memória **RAM1** podem ser combinadas sem ter que ser obrigatoriamente usado o **CPU2**.