[PG50770] Teresa Costa Pires Gil Fortes, Mestrado em Engenharia Informática

Métodos Formais em Engenharia de Software (2022/2023)

Questão 1

Variáveis Proposicionais:

1. CPU1: O computador tem o CPU1.
2. CPU2: O computador tem o CPU2.
3. RAM1: O computador tem a RAM1.
4. RAM2: O computador tem a RAM2.
5. MB1: O computador tem a MB1.
6. MB2: O computador tem a MB2.
7. PG1: O computador tem a PG1.
8. PG2: O computador tem a PG2.
9. PG3: O computador tem a PG3.
10. MON1: O computador tem o MON1.
11. MON2: O computador tem o MON2.
12. MON3: O computador tem o MON3.

Fórmulas Proposicionais:

1. (MB1 ∧ PG1) -> RAM1: A motherboard MB1 quando combinada com a placa gráfica PG1, obrig
2. (PG1 ∧ ¬RAM2) -> CPU1: A placa gráfica PG1 precisa do CPU1, exceto quando combinada co
3. CPU2 -> MB2: O CPU2 só pode ser instalado na motherboard MB2.
4. MON1 -> (PG1 ∧ RAM2): O monitor MON1 para poder funcionar precisa da placa gráfica PG1
5. (MON2 ∧ PG3) -> RAM2: O monitor MON2 precisa da memória RAM2 para poder trabalhar com
6. (CPU1 ∧ ¬CP2) ∨ (¬CPU1 ∧ CP2): O computador tem um único CPU.
7. (RAM1 ∧ ¬RAM2) ∨ (¬RAM1 ∧ RAM2): O computador tem uma única memória RAM.
8. (MB1 ∧ ¬MB2) ∨ (¬MB1 ∧ MB2): O computador tem uma única Motherboard.
9. (PG1 ∧ ¬PG2 ∧ ¬PG3) ∨ (¬PG1 ∧ PG2 ∧ ¬PG3) ∨ (¬PG1 ∧ ¬PG2 ∧ PG3): O computador tem uma
10. MON1 ∨ MON2 ∨ MON3 ∨ (¬MON1 ∧ ¬MON2 ∧ ¬MON3): O computador poderá ter ou não ter moni

Conversão para CNF:

1. ¬(MB1 ∧ PG1) ∨ RAM1 = (¬MB1 ∨ ¬PG1 ∨ RAM1)
2. PG1 ∧ ¬RAM2 -> CPU1 = (¬PG1 ∨ RAM2 ∨ CP1)
3. (¬CPU2 ∨ MB2)
4. ¬MON1 ∨ (PG1 ∧ RAM2) = (¬MON1 ∨ PG1) ∧ (¬MON1 ∨ RAM2)

5. ¬(MON2 ∧ PG3) ∨ RAM2 = (¬MON2 ∨ ¬PG3 ∨ RAM2)
6. (CP1 ∧ ¬CP2) ∨ (¬CP1 ∧ CP2) = (¬CP1 ∨ ¬CP2) ∧ (CP1 ∨ CP2)
7. (RAM1 ∧ ¬RAM2) ∨ (¬RAM1 ∧ RAM2) = (¬RAM1 ∨ ¬RAM2) ∧ (RAM1 ∨ RAM2)
8. (MB1 ∧ ¬MB2) ∨ (¬MB1 ∧ MB2) = (¬MB1 ∨ ¬MB2) ∧ (MB1 ∨ MB2)
9. (PG1 ∧ ¬PG2 ∧ ¬PG3) ∨ (¬PG1 ∧ PG2 ∧ ¬PG3) ∨ (¬PG1 ∧ ¬PG2 ∧ PG3) = (¬PG1∨¬PG2) ∧ (¬PG1
10. True

Questão 2

O conjunto de fórmulas é consistente, isto é, satisfazível (**SAT**), pois o solver encontra uma solução para o problema.

Executando o código abaixo, podemos confirmar a solução do problema.

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

```
    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
    Requirement already satisfied: python-sat[aiger,pblib] in /usr/local/lib/pyth
    Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
    Requirement already satisfied: pypblib>=0.0.3 in /usr/local/lib/python3.7/dis
    Requirement already satisfied: py-aiger-cnf>=2.0.0 in /usr/local/lib/python3.
    Requirement already satisfied: bidict<0.22.0,>=0.21.0 in /usr/local/lib/pytho
    Requirement already satisfied: funcy<2.0,>=1.12 in /usr/local/lib/python3.7/d
    Requirement already satisfied: py-aiger<7.0.0,>=6.0.0 in /usr/local/lib/pytho
    Requirement already satisfied: attrs<22,>=21 in /usr/local/lib/python3.7/dist
    Requirement already satisfied: sortedcontainers<3.0.0,>=2.3.0 in /usr/local/l
    Requirement already satisfied: toposort<2.0,>=1.5 in /usr/local/lib/python3.7
    Requirement already satisfied: parsimonious<0.9.0,>=0.8.1 in /usr/local/lib/p
    Requirement already satisfied: pyrsistent<0.19.0,>=0.18.0 in /usr/local/lib/p
```

```
from pysat.solvers import Minisat22

s = Minisat22()

componentes = ['CPU1', 'CPU2', 'RAM1', 'RAM2', 'MB1', 'MB2', 'PG1', 'PG2', 'PG3',
#                1      2       3       4      5      6      7      8      9

x = {}
c = 1
for d in componentes:
    x[d] = c
    c += 1

# fórmula 1
s.add_clause([-x['MB1'], -x['PG1'], x['RAM1']])

# fórmula 2
s.add_clause([-x['PG1'], x['RAM2'], x['CPU1']])

# fórmula 3
```

```
s.add_clause([-x['CPU2'], x['MB2']])

# fórmula 4
s.add_clause([-x['MON1'], x['PG1']])
s.add_clause([-x['MON1'], x['RAM2']])

# fórmula 5
s.add_clause([-x['MON2'], -x['PG3'], x['RAM2']])

# fórmula 6
s.add_clause([-x['CPU1'], -x['CPU2']])
s.add_clause([x['CPU1'], x['CPU2']])

# fórmula 7
s.add_clause([-x['RAM1'], -x['RAM2']])
s.add_clause([x['RAM1'], x['RAM2']])

# fórmula 8
s.add_clause([-x['MB1'], -x['MB2']])
s.add_clause([x['MB1'], x['MB2']])

# fórmula 9
s.add_clause([-x['PG1'], -x['PG2']])
s.add_clause([-x['PG1'], -x['PG3']])
s.add_clause([x['PG1'], x['PG2'], x['PG3']])
s.add_clause([-x['PG2'], -x['PG3']])

# fórmula 10 - tautologia dos monitores
s.add_clause([x['MON3'], -x['MON3']]) #(MON3 V ¬MON3) para que seja considerado no

#Fórmulas das regras de personalização
#fórmula p1
s.add_clause([-x['MB1'], -x['PG1'], x['RAM1']])

#fórmula p2
s.add_clause([-x['PG1'], x['RAM2'], x['CPU1']])

#fórmula p3
s.add_clause([-x['CPU2'], x['MB2']])

#fórmula p4
s.add_clause([-x['MON1'], x['PG1']])
s.add_clause([-x['MON1'], x['RAM2']])

#fórmula p5
s.add_clause([-x['MON2'], -x['PG3'], x['RAM2']])


if s.solve():
    print("SAT")
    m = s.get_model()
    print(m)
    print("\n")
    print("O computador pode ser formado pelos componentes: ")
    for w in componentes:
```

```
        if m[x[w]-1] > 0:
          print("%s " %w)

else:
    print("O computador não pode ser produzido.")
s.delete()
```

```
SAT
[1, -2, 3, -4, 5, -6, 7, -8, -9, -10, -11, -12]


O computador pode ser formado pelos componentes:
CPU1
RAM1
MB1
PG1
```

Questão 3

> **(a).** O monitor MON1 só poderá ser usado com uma motherboard MB1?

> $MON1 \rightarrow MB1 = MON1 \wedge \neg MB1$

Não. Como podemos verificar através da solução dada pelo solver, há uma solução que torna o sistema satisfazível em que o monitor MON1 também pode ser usado com a motherboard MB2.

```
from pysat.solvers import Minisat22

s = Minisat22()

componentes = ['CPU1', 'CPU2', 'RAM1', 'RAM2', 'MB1', 'MB2', 'PG1', 'PG2', 'PG3',
#                1      2       3       4      5      6      7      8      9

x = {}
i = 1
for c in componentes:
    x[c] = i
    i += 1

# fórmula 1
s.add_clause([-x['MB1'], -x['PG1'], x['RAM1']])

# fórmula 2
s.add_clause([-x['PG1'], x['RAM2'], x['CPU1']])

# fórmula 3
s.add_clause([-x['CPU2'], x['MB2']])

# fórmula 4
s.add_clause([-x['MON1'], x['PG1']])
s.add_clause([-x['MON1'], x['RAM2']])
```

```
# fórmula 5
s.add_clause([-x['MON2'], -x['PG3'], x['RAM2']])

# fórmula 6
s.add_clause([-x['CPU1'], -x['CPU2']])
s.add_clause([x['CPU1'], x['CPU2']])

# fórmula 7
s.add_clause([-x['RAM1'], -x['RAM2']])
s.add_clause([x['RAM1'], x['RAM2']])

# fórmula 8
s.add_clause([-x['MB1'], -x['MB2']])
s.add_clause([x['MB1'], x['MB2']])

# fórmula 9
s.add_clause([-x['PG1'], -x['PG2']])
s.add_clause([-x['PG1'], -x['PG3']])
s.add_clause([x['PG1'], x['PG2'], x['PG3']])
s.add_clause([-x['PG2'], -x['PG3']])

# fórmula 10 - tautologia dos monitores
s.add_clause([x['MON3'], -x['MON3']]) #(MON3 V ¬MON3) para que seja considerado no


#Fórmulas das regras de personalização
#fórmula p1
s.add_clause([-x['MB1'], -x['PG1'], x['RAM1']])

#fórmula p2
s.add_clause([-x['PG1'], x['RAM2'], x['CPU1']])

#fórmula p3
s.add_clause([-x['CPU2'], x['MB2']])

#fórmula p4
s.add_clause([-x['MON1'], x['PG1']])
s.add_clause([-x['MON1'], x['RAM2']])

#fórmula p5
s.add_clause([-x['MON2'], -x['PG3'], x['RAM2']])

#restrições adicionais da alínea 3.a)
s.add_clause([x['MON1']])
s.add_clause([-x['MB1']])

if s.solve():
    print("SAT")
    m = s.get_model()
    print(m)
    print("\n")
    print("O computador pode ser formado pelos componentes: ")
    for w in componentes:
      if m[x[w]-1] > 0:
```

```
        print("%s" %w)

else:
    print("O computador não pode ser produzido.")
s.delete()
```

```
SAT
[1, -2, -3, 4, -5, 6, 7, -8, -9, 10, -11, -12]


O computador pode ser formado pelos componentes:
CPU1
RAM2
MB2
PG1
MON1
```

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

> $MB1 \wedge CP1 \wedge PG2 \wedge RAM1$

Sim. Tal como verificamos através da solução dada pelo solver, há uma solução que torna o sistema satisfazível em que um computador é formado por esses componentes.

```
from pysat.solvers import Minisat22

s = Minisat22()

componentes = ['CPU1', 'CPU2', 'RAM1', 'RAM2', 'MB1', 'MB2', 'PG1', 'PG2', 'PG3',
#                1      2       3       4      5      6      7      8      9

x = {}
i = 1
for c in componentes:
    x[c] = i
    i += 1

# fórmula 1
s.add_clause([-x['MB1'], -x['PG1'], x['RAM1']])

# fórmula 2
s.add_clause([-x['PG1'], x['RAM2'], x['CPU1']])

# fórmula 3
s.add_clause([-x['CPU2'], x['MB2']])

# fórmula 4
s.add_clause([-x['MON1'], x['PG1']])
s.add_clause([-x['MON1'], x['RAM2']])

# fórmula 5
s.add_clause([-x['MON2'], -x['PG3'], x['RAM2']])
```

```
# fórmula 6
s.add_clause([-x['CPU1'], -x['CPU2']])
s.add_clause([x['CPU1'], x['CPU2']])

# fórmula 7
s.add_clause([-x['RAM1'], -x['RAM2']])
s.add_clause([x['RAM1'], x['RAM2']])

# fórmula 8
s.add_clause([-x['MB1'], -x['MB2']])
s.add_clause([x['MB1'], x['MB2']])

# fórmula 9
s.add_clause([-x['PG1'], -x['PG2']])
s.add_clause([-x['PG1'], -x['PG3']])
s.add_clause([x['PG1'], x['PG2'], x['PG3']])
s.add_clause([-x['PG2'], -x['PG3']])

# fórmula 10 - tautologia dos monitores
s.add_clause([x['MON3'], -x['MON3']]) #(MON3 V ¬MON3) para que seja considerado no


#Fórmulas das regras de personalização
#fórmula p1
s.add_clause([-x['MB1'], -x['PG1'], x['RAM1']])

#fórmula p2
s.add_clause([-x['PG1'], x['RAM2'], x['CPU1']])

#fórmula p3
s.add_clause([-x['CPU2'], x['MB2']])

#fórmula p4
s.add_clause([-x['MON1'], x['PG1']])
s.add_clause([-x['MON1'], x['RAM2']])

#fórmula p5
s.add_clause([-x['MON2'], -x['PG3'], x['RAM2']])

# restrições adicionais da alínea 3.b)
s.add_clause([x['MB1']])
s.add_clause([x['CPU1']])
s.add_clause([x['PG2']])
s.add_clause([x['RAM1']])

if s.solve():
    print("SAT")
    m = s.get_model()
    print(m)
    print("\n")
    print("O computador pode ser formado pelos componentes: ")
    for w in componentes:
      if m[x[w]-1] > 0:
        print("%s " %w)
```

```
else:
    print("O computador não pode ser produzido.")
s.delete()
```

```
SAT
[1, -2, 3, -4, 5, -6, -7, 8, -9, -10, -11, -12]


O computador pode ser formado pelos componentes:
CPU1
RAM1
MB1
PG2
```

> **(c).** É possivel combinar a motherboard MB2, a placa gráfica PG3 e a RAM1 num mesmo computador?

> $MB2 \wedge PG3 \wedge RAM1$

Sim. Tal como verificamos através da solução dada pelo solver, há uma solução que torna o sistema satisfazível em que um computador é formado por esses componentes.

```
from pysat.solvers import Minisat22

s = Minisat22()

componentes = ['CPU1', 'CPU2', 'RAM1', 'RAM2', 'MB1', 'MB2', 'PG1', 'PG2', 'PG3',
#                 1      2       3        4      5      6      7      8      9

x = {}
i = 1
for c in componentes:
    x[c] = i
    i += 1

# fórmula 1
s.add_clause([-x['MB1'], -x['PG1'], x['RAM1']])

# fórmula 2
s.add_clause([-x['PG1'], x['RAM2'], x['CPU1']])

# fórmula 3
s.add_clause([-x['CPU2'], x['MB2']])

# fórmula 4
s.add_clause([-x['MON1'], x['PG1']])
s.add_clause([-x['MON1'], x['RAM2']])

# fórmula 5
s.add_clause([-x['MON2'], -x['PG3'], x['RAM2']])

# fórmula 6
s.add_clause([-x['CPU1'], -x['CPU2']])
```

```
    s.add_clause([-x['CPU1'], -x['CPU2']])
    s.add_clause([x['CPU1'], x['CPU2']])

    # fórmula 7
    s.add_clause([-x['RAM1'], -x['RAM2']])
    s.add_clause([x['RAM1'], x['RAM2']])

    # fórmula 8
    s.add_clause([-x['MB1'], -x['MB2']])
    s.add_clause([x['MB1'], x['MB2']])

    # fórmula 9
    s.add_clause([-x['PG1'], -x['PG2']])
    s.add_clause([-x['PG1'], -x['PG3']])
    s.add_clause([x['PG1'], x['PG2'], x['PG3']])
    s.add_clause([-x['PG2'], -x['PG3']])

    # fórmula 10 - tautologia dos monitores
    s.add_clause([x['MON3'], -x['MON3']]) #(MON3 V ¬MON3) para que seja considerado no


    #Fórmulas das regras de personalização
    #fórmula p1
    s.add_clause([-x['MB1'], -x['PG1'], x['RAM1']])

    #fórmula p2
    s.add_clause([-x['PG1'], x['RAM2'], x['CPU1']])

    #fórmula p3
    s.add_clause([-x['CPU2'], x['MB2']])

    #fórmula p4
    s.add_clause([-x['MON1'], x['PG1']])
    s.add_clause([-x['MON1'], x['RAM2']])

    #fórmula p5
    s.add_clause([-x['MON2'], -x['PG3'], x['RAM2']])

    # restrições adicionais da alínea 3.c)
    s.add_clause([x['MB2']])
    s.add_clause([x['PG3']])
    s.add_clause([x['RAM1']])

    if s.solve():
        print("SAT")
        m = s.get_model()
        print(m)
        print("\n")
        print("O computador pode ser formado pelos componentes: ")
        for w in componentes:
          if m[x[w]-1] > 0:
            print("%s " %w)

    else:
        print("O computador não pode ser produzido.")
    s.delete()
```

```
s.delete()

SAT
[1, -2, 3, -4, -5, 6, -7, -8, 9, -10, -11, -12]


O computador pode ser formado pelos componentes:
CPU1
RAM1
MB2
PG3
```

> **(d).** Para combinarmos a placa gráfica PG2 e a RAM1 temos que usar o CPU2?

> $PG2 \wedge RAM1 \rightarrow CPU2 = \neg(PG2 \wedge RAM1) \vee CPU2 = (\neg PG2 \vee \neg RAM1 \vee C$

Não. Tal como verificamos através da solução dada pelo solver, há uma solução que torna o sistema satisfazível em que um computador é formado pela PG2 e pela RAM1, mas não é formado pelo CPU2 (é formado pelo CPU1).

```
from pysat.solvers import Minisat22

s = Minisat22()

componentes = ['CPU1', 'CPU2', 'RAM1', 'RAM2', 'MB1', 'MB2', 'PG1', 'PG2', 'PG3',
#                1      2       3       4      5      6      7      8      9

x = {}
i = 1
for c in componentes:
    x[c] = i
    i += 1

# fórmula 1
s.add_clause([-x['MB1'], -x['PG1'], x['RAM1']])

# fórmula 2
s.add_clause([-x['PG1'], x['RAM2'], x['CPU1']])

# fórmula 3
s.add_clause([-x['CPU2'], x['MB2']])

# fórmula 4
s.add_clause([-x['MON1'], x['PG1']])
s.add_clause([-x['MON1'], x['RAM2']])

# fórmula 5
s.add_clause([-x['MON2'], -x['PG3'], x['RAM2']])

# fórmula 6
s.add_clause([-x['CPU1'], -x['CPU2']])
s.add_clause([x['CPU1'], x['CPU2']])
```

```
# fórmula 7
s.add_clause([-x['RAM1'], -x['RAM2']])
s.add_clause([x['RAM1'], x['RAM2']])

# fórmula 8
s.add_clause([-x['MB1'], -x['MB2']])
s.add_clause([x['MB1'], x['MB2']])

# fórmula 9
s.add_clause([-x['PG1'], -x['PG2']])
s.add_clause([-x['PG1'], -x['PG3']])
s.add_clause([x['PG1'], x['PG2'], x['PG3']])
s.add_clause([-x['PG2'], -x['PG3']])

# fórmula 10 - tautologia dos monitores
s.add_clause([x['MON3'], -x['MON3']]) #(MON3 V ¬MON3) para que seja considerado no


#Fórmulas das regras de personalização
#fórmula p1
s.add_clause([-x['MB1'], -x['PG1'], x['RAM1']])

#fórmula p2
s.add_clause([-x['PG1'], x['RAM2'], x['CPU1']])

#fórmula p3
s.add_clause([-x['CPU2'], x['MB2']])

#fórmula p4
s.add_clause([-x['MON1'], x['PG1']])
s.add_clause([-x['MON1'], x['RAM2']])

#fórmula p5
s.add_clause([-x['MON2'], -x['PG3'], x['RAM2']])

# restrições adicionais da alínea 3.d)
s.add_clause([x['PG2']])
s.add_clause([x['RAM1']])
s.add_clause([-x['CPU2']])

if s.solve():
    print("SAT")
    m = s.get_model()
    print(m)
    print("\n")
    print("O computador pode ser formado pelos componentes: ")
    for w in componentes:
      if m[x[w]-1] > 0:
        print("%s " %w)

else:
    print("O computador não pode ser produzido.")
s.delete()
```

```
 SAT
```

```
[1, -2, 3, -4, 5, -6, -7, 8, -9, -10, -11, -12]


O computador pode ser formado pelos componentes:
CPU1
RAM1
MB1
PG2
```


✓ 0 s concluído à(s) 18:21