Número: PG47520

Nome: Melânia Rafaela Sousa Pereira

Curso: Mestrado em Engenharia Informática

1

Variáveis:

1. B - Usar bigode
2. C - Ser casado
3. R - Ser de Ribeirão
4. A - Usar camisola amarela
5. D - Assistir aos jogos ao Domingo

Regras:

- Todos os sócios que usam bigode são casados.

  $$B \rightarrow C \equiv \neg B \vee C$$

- Cada sócio do clube que não é de Ribeirão tem que usar camisola amarela.

  $$\neg R \rightarrow A \equiv R \vee A$$

- Os sócios casados não podem assistir aos jogos ao Domingo.

  $$C \rightarrow \neg D \equiv \neg C \vee \neg D$$

- Um sócio vai aos jogos de Domingo se e só se é de Ribeirão.

  $$D \leftrightarrow R \equiv D \rightarrow R \wedge R \rightarrow D \equiv (\neg D \vee R) \wedge (\neg R \vee D)$$

- Cada sócio usa bigode ou não usa camisola amarela.

  $$B \vee \neg A$$

- Todos os sócios de Ribeirão usam bigode.

  $$R \rightarrow B \equiv \neg R \vee B$$

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

2

```
# alínea 2
# Considera-se a numeração de variáveis feita na alínea 1

from pysat.solvers import Minisat22

s = Minisat22()                 # cria o solver s

s.add_clause([-1, 2])           # acrescenta uma cláusula
s.add_clause([3, 4])
s.add_clause([-2, -5])
s.add_clause([-5, 3])
s.add_clause([-3, 5])
s.add_clause([1, -4])
s.add_clause([-3, 1])

if s.solve():                   # testa a satisfatibilidade
    print("SAT")
    print(s.get_model())        # imprime o modelo
else:
    print("UNSAT")

s.delete()                      # apaga o solver s
```

```
SAT
[1, 2, -3, 4, -5]
```

> O resultado anterior foi SAT, logo o conjunto de regras é consistente.

3.a

```
# alínea 3.a) A afirmação “Quem usa bigode não pode ir ao jogo ao Domingo.” é co
#
# testar a afirmação: usar bigode implica não assistir aos jogos ao Domingo
# 1 -> -5
# -1 V -5
# -(-1 V -5) UNSAT?
# verificar se (1 ∧ 5) UNSAT?

from pysat.solvers import Minisat22

s = Minisat22()                 # cria o solver s

s.add_clause([-1, 2])           # acrescenta uma cláusula
s.add_clause([3, 4])
s.add_clause([-2, -5])
s.add_clause([-5, 3])
s.add_clause([-3, 5])
s.add_clause([1, -4])
s.add_clause([-3, 1])
s.add_clause([1])
s.add_clause([5])

if s.solve():                   # testa a satisfatibilidade
    print("SAT")
    print(s.get_model())        # imprime o modelo
else:
    print("UNSAT")

s.delete()                      # apaga o solver s
```

```
UNSAT
```

> O resultado desta verificação foi UNSAT, por isso a fórmula testada é válida, logo, a afirmação é correta, ou seja, quem usa bigode não pode ir aos jogos ao Domingo.

3.b

```
# alínea 3.b) Pode um membro de camisola amarela ser casado?
#
# testar a afirmação: usar camisola amarela e ser casado
# 4 ∧ 2
# −(4 ∧ 2) UNSAT?
# verificar se (−4 V −2) UNSAT?

from pysat.solvers import Minisat22

s = Minisat22()                 # cria o solver s

s.add_clause([-1, 2])           # acrescenta uma cláusula
s.add_clause([3, 4])
s.add_clause([-2, -5])
s.add_clause([-5, 3])
s.add_clause([-3, 5])
s.add_clause([1, -4])
s.add_clause([-3, 1])
s.add_clause([-4, -2])

if s.solve():                   # testa a satisfatibilidade
    print("SAT")
    print(s.get_model())        # imprime o modelo
else:
    print("UNSAT")

s.delete()                      # apaga o solver s
```

```
UNSAT
```

> O resultado desta verificação foi UNSAT, por isso a fórmula testada é válida, logo, a afirmação é correta, ou seja, um membro pode usar camisola amarela e ser casado.

3.c

```
# alínea 3.c) A afirmação “Afinal o clube não pode ter sócios Ribeironenses.” é
#
# testar a afirmação: não ser de Ribeirão
# -3
# -(-3) UNSAT?
# verificar se 3 UNSAT?
from pysat.solvers import Minisat22

s = Minisat22()                # cria o solver s

s.add_clause([-1, 2])          # acrescenta uma cláusula
s.add_clause([3, 4])
s.add_clause([-2, -5])
s.add_clause([-5, 3])
s.add_clause([-3, 5])
s.add_clause([1, -4])
s.add_clause([-3, 1])
s.add_clause([3])

if s.solve():                  # testa a satisfatibilidade
    print("SAT")
    print(s.get_model())       # imprime o modelo
else:
    print("UNSAT")

s.delete()                     # apaga o solver s
```

```
UNSAT
```

> O resultado desta verificação foi UNSAT, por isso a fórmula testada é válida, logo, a afirmação é correta, ou seja, o clube não pode ter sócios Ribeironenses.

3.d

```
# alínea 3.d) Os sócios casados têm todos bigode?
#
# testar a afirmação: ser casado implica ter bigode
# 2 -> 1
# -2 V 1
# -(-2 V 1) UNSAT?
# verificar se 2 ∧ -1 UNSAT?

from pysat.solvers import Minisat22

s = Minisat22()                 # cria o solver s

s.add_clause([-1, 2])           # acrescenta uma cláusula
s.add_clause([3, 4])
s.add_clause([-2, -5])
s.add_clause([-5, 3])
s.add_clause([-3, 5])
s.add_clause([1, -4])
s.add_clause([-3, 1])
s.add_clause([2])
s.add_clause([-1])

if s.solve():                   # testa a satisfatibilidade
    print("SAT")
    print(s.get_model())        # imprime o modelo
else:
    print("UNSAT")

s.delete()                      # apaga o solver s
```

```
UNSAT
```

> O resultado desta verificação foi UNSAT, por isso a fórmula testada é válida, logo, a afirmação é correta, ou seja, os sócios casados têm todos bigode.

3.e

```
# alínea 3.e) A afirmação “Ao domingo nunca há sócios a assistir aos jogos.” é c
#
# testar a afirmação: não há sócios a assistir aos jogos ao Domingo
# -5
# -(-5) UNSAT?
# verificar se 5 UNSAT?

from pysat.solvers import Minisat22

s = Minisat22()                 # cria o solver s

s.add_clause([-1, 2])           # acrescenta uma cláusula
s.add_clause([3, 4])
s.add_clause([-2, -5])
s.add_clause([-5, 3])
s.add_clause([-3, 5])
s.add_clause([1, -4])
s.add_clause([-3, 1])
s.add_clause([5])

if s.solve():                   # testa a satisfatibilidade
    print("SAT")
    print(s.get_model())        # imprime o modelo
else:
    print("UNSAT")

s.delete()                      # apaga o solver s
```

```
UNSAT
```

> O resultado desta verificação foi UNSAT, por isso a fórmula testada é válida, logo, a afirmação é correta, ou seja, ao domingo nunca há sócios a assistir aos jogos.