PySAT: a Satisfiability Solver Python Package

Xudong Liu, Ph.D.
Associate Professor
School of Computing
University of North Florida

What is PySAT?

1. PySAT is a SAT solver Python package that can take a Boolean formula in the conjunctive normal form (CNF) format and decides if it is satisfiable.
    - In case satisfiable, it computes one, or even all, satisfying truth assignments.
2. To install, run `pip install python-sat[pblib,aiger]`
    - Use right version of pip, e.g., if using python3.11, use pip3.11.
    - See `https://pysathq.github.io/` for details.
3. For Project 3, you also may check if a formula is satisfiable by a truth assignment (aka, object) by writing your own programs without using PySAT.

Quick Example

$$\varphi = (a_1 \vee \neg a_3) \wedge (a_2 \vee a_3 \vee \neg a_1)$$

```python
"""
PySAT example
Download and install page: https://pysathq.github.io/
"""

# the standard way to import PySAT:
from pysat.formula import CNF
from pysat.solvers import Solver

# create a CNF formula "(a1 ∨ ¬a3) ∧ (a2 ∨ a3 ∨ ¬a1)":
cnf = CNF(from_clauses=[[1, -3], [2, 3, -1]])

# create a SAT solver for this formula:
with Solver(bootstrap_with=cnf) as solver:
    # call the solver for this formula:
    print('formula is', f'{"s" if solver.solve() else "uns"}atisfiable')

    # the formula is satisfiable and so has a model:
    print('and the model is:', solver.get_model())

    # enumerate all models
    print('here are all the models for this formula:')
    for m in solver.enum_models():
        print(m)
```

Quick Example

$$\varphi = (a_1 \lor \neg a_3) \land (a_2 \lor a_3 \lor \neg a_1)$$

```
n01237497-office [PySATTutorial]$ python3.11 example.py
formula is satisfiable
and the model is: [-1, -2, -3]
here are all the models for this formula:
[-1, -2, -3]
[1, -2, 3]
[1, 2, 3]
[1, 2, -3]
[-1, 2, -3]
n01237497-office [PySATTutorial]$
```

Dinner Meals

1. Appetizer: {soup (s or 1), salad ($\overline{s}$ or -1)}
2. Entree: {fish (f or 2), beef ($\overline{f}$ or -2)}
3. Drink: {wine (w or 3), beer ($\overline{w}$ or -3)}
4. Dissert: {cake (c or 4), ice cream ($\overline{c}$ or -4)}

Hard Constraint

$$\neg(s \wedge \overline{c})$$

Note we have:

1. $\neg(p \wedge q) \equiv \neg p \vee \neg q$ (De Morgan's laws)
2. $\neg(p \vee q) \equiv \neg p \wedge \neg q$ (De Morgan's laws)

$$\neg(s \wedge \overline{c}) \equiv \neg s \vee c$$

```
# create a CNF formula "-x1 v x4":
cnf = CNF(from_clauses=[[-1, 4], [-1, 1], [-2, 2], [3, -3], [-4, 4]])
```

```
n01237497-office [PySATTutorial]$ python3.11 example.py
formula is satisfiable
and the model is: [-1, -2, -3, -4]
here are all the models for this formula:
[-1, -2, -3, -4]
[-1, -2, -3, 4]
[-1, 2, -3, -4]
[-1, 2, 3, -4]
[-1, 2, 3, 4]
[-1, 2, -3, 4]
[1, -2, -3, 4]
[1, -2, 3, 4]
[-1, -2, 3, 4]
[-1, -2, 3, -4]
[1, 2, 3, 4]
[1, 2, -3, 4]
```

Example: Possibilistic Logic Theory

$$T = \{(\overline{f} \rightarrow (\overline{s} \wedge \overline{c}), 0.9), (f \rightarrow (s \wedge c), 0.8), (w \vee c, 0.6)\}.$$

Note we have:

1. $p \rightarrow q \equiv \neg p \vee q$
2. $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ (Distributive laws)

$$\varphi = \overline{f} \rightarrow (\overline{s} \wedge \overline{c}) \equiv f \vee (\overline{s} \wedge \overline{c}) \equiv (f \vee \overline{s}) \wedge (f \vee \overline{c})$$

Example: Possibilistic Logic Theory

$$(f \vee \overline{s}) \wedge (f \vee \overline{c})$$

How to see if dinner $\langle s, \overline{f}, \overline{w}, c \rangle$ satisfies property φ?

```
# see if "1 ^ -2 ^ -3 ^ 4" satisfies CNF formula "(x2 v -x1) ^ (x2 v -x4)":
cnf = CNF(from_clauses=[[2, -1], [2, -4], [1], [-2], [-3], [4]])
```

```
n01237497-office [PySATTutorial]$ python3.11 example.py
formula is unsatisfiable
and the model is: None
here are all the models for this formula:
n01237497-office [PySATTutorial]$
```