SAT Solver for Obligatory 2

The solver checks the satisfiability of a propositional formula in clause form and provides a satisfying interpretation if applicable.

Usage

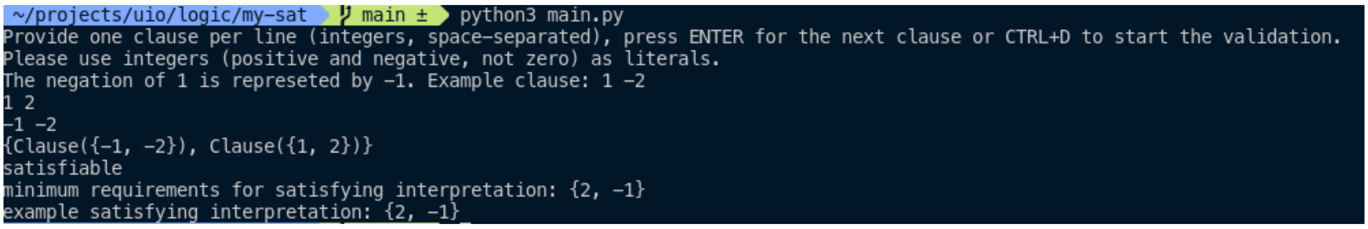
The formula may be provided in two ways: interactively or via an input file

With interactive input

Run `main.py` and follow the prompts on screen. You may provide an arbitrary number of clauses – one per line – with an arbitrary number of literals each. Propositional variables are represented by integers. A negative integer represents a negated literal, e.g., `-1` is the negation of `1`. Zero (`0`) is therefore **NOT** valid.

To end the clause definition and start the validation, press `Ctrl+D` to close the standard input (stdin).

Example run:



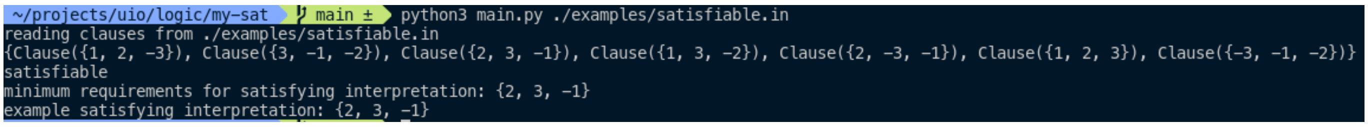
```
~/projects/uio/logic/my-sat  main ±  python3 main.py
Provide one clause per line (integers, space-separated), press ENTER for the next clause or CTRL+D to start the validation.
Please use integers (positive and negative, not zero) as literals.
The negation of 1 is represeted by -1. Example clause: 1 -2
1 2
-1 -2
{Clause({-1, -2}), Clause({1, 2})}
satisfiable
minimum requirements for satisfying interpretation: {2, -1}
example satisfying interpretation: {2, -1}
```

The last line of the output the first satisfying interpretation found – in this case to set the variable `1=False` (represented by `-1`) and `2=True` (represented by `(+)2`).

With file input

Alternatively, the user may provide their formula in a separate file. The file path must be provided as the first (optional) command line argument when executing the script. The file may only contain a single formula. Valid examples can be found in the folder `examples`.

Example run:



```
~/projects/uio/logic/my-sat  main ±  python3 main.py ./examples/satisfiable.in
reading clauses from ./examples/satisfiable.in
{Clause({1, 2, -3}), Clause({3, -1, -2}), Clause({2, 3, -1}), Clause({1, 3, -2}), Clause({2, -3, -1}), Clause({1, 2, 3}), Clause({-3, -1, -2})}
satisfiable
minimum requirements for satisfying interpretation: {2, 3, -1}
example satisfying interpretation: {2, 3, -1}
```

Performance

The case with all 2**n clauses one can build from n propositional variables takes about 0.12 seconds for n==3 but does not terminate in reasonable time for n==4 on my machine.