Prof. Dr. Rüdiger Ehlers Francisco Palau, M.Sc.

Applied Computational Engines 2018 – Notes on SAT Solving

Getting started with SAT solving

Students who are working on obtaining an M.Sc in Computer Science should normally already have access to the computers in the *Praktikums Computer-Pool* in the MZH building. It is easy to get a SAT solver to run on those machines.

For example, the solver `picosat` can be downloaded and compiled on these machines using the following commands (tested on machine `x16`):

```
cd /tmp
wget http://fmv.jku.at/picosat/picosat-960.tar.gz
tar -xvzf picosat-960.tar.gz
cd picosat-960
./configure
make
```

This will create the executable `picosat`, which can then be copied elsewhere. For example, the following commands will copy it to the folder `applied_computational_engines` in your local home folder:

```
mkdir ~/applied_computational_engines
cp picosat ~/applied_computational_engines/
```

As an alternative to `picosat`, there is the solver `minisat`. It can be built in a similar way as `picosat`:

```
cd /tmp
wget http://minisat.se/downloads/minisat-2.2.0.tar.gz
tar -xvzf minisat-2.2.0.tar.gz
cd minisat
export MROOT=/tmp/minisat
cd simp
make
mkdir ~/applied_computational_engines
cp minisat ~/applied_computational_engines/
```

The homepages of these solvers can be found at:

- `http://fmv.jku.at/picosat`
- `http://minisat.se/`

Finally, the SAT-solving exercises can also be solved using an online-version of `MiniSAT`, which can be accessed at `http://www.msoos.org/2013/09/minisat-in-your-browser/`.

DIMACS input format for SAT solvers

Many SAT solvers, including `MiniSAT` and `picosat`, require their input to be in CNF (conjunctive normal form). A boolean formula in CNF shape is build from three elements:

- literal: A literal is a boolean variable or its negation (e.g. if x_1 is a boolean variable, $\neg x_1$ is its negation).
- clause: A clause is a set of one or more literals, all of them connected by disjunction (`OR` or $\vee$).
- boolean formula: A boolean formula is a set of one or more clauses, all of them connected by conjunction (`AND` or $\wedge$).

An example of a boolean formula in CNF form is:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (\neg x_3)$$

Usually, SAT solvers accept its input instances in DIMACS format, which is simple text format. In DIMACS CNF format, every line beginning with character "`c`" is a comment, and will not be interpreted by the solver.

The first non-comment line must be of the form:

```
p cnf NUMBER_OF_VARIABLES NUMBER_OF_CLAUSES
```

Each non-comment line afterwards is a clause. Clauses are space-separated lists of literals. A positive occurrence of a variable is represented by its number. A negative occurrence of a variables is represented by its inverse element (for example, x_1 is represented by `1`, $\neg x_3$ is represented by `-3`). Each clause line must end with a space and number `0`. As the number `0` thus has a special meaning, the first variable in a DIMACS SAT instance thus has number 1.

You do not need to express the conjunction (AND) in the DIMACS CNF format.

The previous example of CNF in its DIMACS format could be expressed as:

```
c This is a comment
p cnf 3 4
1 2 0
-1 -3 0
2 3 0
c This is another comment
-3 0
```

The first line is a comment, the second line expresses that the SAT problem is in CNF format and that it has 3 variables and 4 clauses. The other lines, but the one starting with "`c`", are the clauses. The first clause can, for example, be read as $(x_1 \vee x_2)$.

The output of a SAT solver includes if the problem is satisfiable or not. Some SAT solvers include an assignments to the boolean variables that satisfies the instance. The solver `picosat` prints assignments by default, while `MiniSAT` writes an assignment to an output file if its name has been supplied to it in addition to the input file's name.