KI 2019 Class - SAT Solving Competition

September 12, 2019

Deadlines

- *Fri, 13. of Sep, evening*: Send us a *tempory version* of your solver. We will try to get it running and give you feedback.
- *Sun, 15. of Sep, 15:00*: You MAY send us an *updated version* of your solver until 15:00, if you do so, we will use that one on monday. You can also submit up to three DIMACS CNF problems that will be employed during the final competition.
- *Mon, 16. of Sep, morning*: We will run a pre-tournament on some test problems so you know what to expect from your competitors.
- *Tue, 17. of Sep, before 11:45*: Send us the *final version* of your solver.
- *Tue, 17. of Sep, after lunch*: Final tournament.

Any solvers can be submitted to Marco by Mail, any questions will be answered by Marco and Tobias via Mail.

Marco Träger `traeger@inf.fu-berlin.de`
Tobias Gleißner `tobias.gleissner@fu-berlin.de`

Solver Requirements

Misc

The prover should be able to be executed on a unix-like operating system. Please submit as an archive containing only source code and a file `INSTALL.md` that explains the requirements of the solver, its installation procedure and invocation on the command line in detail. Binaries will not be accepted.

Input Format

The input format of the problems is DIMACS cnf. Each line of a problem may contain

- a comment starting with character `c`,
- exactly once a description of the problem starting with character `p` followed by the number of atoms and the number of clauses, each separated by non-linebreaking whitespaces, and
- a clause consisting of literals for which each literal can be a positive number if it is an atom or a negative number if it is a negated atom, the end for each clause is indicated by `0` and all literals and `0` are separated by non-linebreaking whitespaces.

A problem will be passed to a solver through a filename parameter of the solver.

Output Format

The output should conform to the TPTP SZS Ontology [1]. In summary, this means if a solver found a problem to be satisfiable it should report

```
% SZS status Satisfiable
```

If the problem is identified as unsatisfiable it should report

```
% SZS status Unsatisfiable
```

If the solver was not able to find a solution it should report

```
% SZS status GaveUp
```

The status has to be reported exactly once in a separate line of the standard output stream.

Test your implementation

The competition will be conducted employing the python package *tptp* which is available at `https://github.com/leoprover/tptp/releases/tag/0.0.1`. This package ships with a test competition containing some satisfiable and unsatisfiable CNF problems in DIMACS format. Download and extract this release of the *tptp* python package. The package does not have to be installed. Python ≥ 3.5 is required.
The test competition consists of a directory `competition-test/problems` containing test problems and a configuration file `competition-test/definition.py`. The configuration file contains a python tuple `SOLVERS` whose elements are python dictionaries of which each represents a solver configuration. A solver configuration possesses the mandatory keys `'type'`, `'name'`, and `'command'`. `'type'` should always be set to `'local'`, `'name'` is a string representing the solver name, and `'command'` has to be set to the command that can invoke the solver from the command line. `'command'` should contain the substring `%s` which will be replaced by an absolute filename on each invocation. An example solver configuration may look like this:

```
SOLVERS = (
    {
        'type': 'local',
        'name': 'my-solver',
        'command': 'my-solver-binary-or-shell-script %s',
    },
)
```

Here, if an input problem `my-problem.cnf` is passed to the solver, the command `my-solver-binary-or-shell-script my-problem.cnf` will be executed on the command line. After the timeout defined in the competition configuration has come to pass or the solver is finished, the solver's SZS status is parsed and printed on the command line with its name and the expected status of the problem.
Adjust the configuration to contain your solver. Finally, you will be able to do the following:
Run the test competition with all configured solvers by invoking

```
python3 -m tptp competition competition-test/definition.py
```

from the repository root directory.

Run the test competition with more output. Good for error tracking.

```
python3 -m tptp competition competition-test/definition.py --verbose
```

[1] See `http://www.cs.miami.edu/home/geoff/Papers/Conference/2008_Sut08_KEAPPA-38-49.pdf`.