Satisfiability Checking - WS 2019/2020

Programming exercise

gereon.kremer@cs.rwth-aachen.de
https://ths.rwth-aachen.de/teaching/

Deadline: January 6th

Task

Implement a SAT solver for propositional logic following the DPLL architecture.

You have to implement a trail, boolean constraint propagation, decisions, backtracking as presented for DPLL in the lecture.

You may optionally implement the two-watched-literals scheme, CDCL-style conflict analysis and clause learning, proper variable heuristics.

Your solver needs to correctly solve at least 90% of the benchmark files within 10 seconds.

Technical requirements

Please submit **a zip archive** that contains directly (not in a subfolder) at least two scripts:

- `build.sh` that compiles your program, if necessary.
- `solve.sh` that runs your program on a given input.

The results (`SAT` or `UNSAT`) need to be indicated as follows:

- `SAT`: output „`sat`“, a satisfying assignment and return with exit code `10`.
- `UNSAT`: output „`unsat`“ and return with exit code `20`.

For implementation, you may choose C++, Python or Java. Please do not use any external libraries except for the respective standard libraries. The solver is limited to use **at most 4GB of memory** which is enforced using `ulimit -S -v 4194304`. It may be necessary to adapt your script `solver.sh` accordingly, in particular if using Java. You may want to consult `https://stackoverflow.com/a/44532547` on this issue.

C++ Use `g++` for compilation, the compiler we use will be `g++ 8`.

Java Use `javac` for compilation and `java` for execution. The Java version is `11.0.4`.

Python Use `python3` for execution. The version is `python 3.7`.

The provided zip file contains

- a C++ example file `example.cpp`,
- a Java example file `example.java`,
- a Python example file `example.py`,
- a DIMACS parser for each language in the respective files,
- a build script `build.sh` that builds all examples and
- a run script `solve.sh` that runs all examples on a given file.