DSL Project Report : Final Milestone

Olivier Zhang

January 6, 2020

1 Introduction

This document is the report of the final milestone of the DSL course's project.

All experiments of this milestone have been performed on an Intel Core i5-8350U CPU 1.70GHz ×8, and a RAM of 16Go.

The solvers that are evaluated are Minisat[1], Cryptominisat[2] and Sat4J[3]. This last solver is called in 3 different ways : using java code, calling directly the jar file and using java code through a Maven project.

To launch our own evaluation on a desired benchmark, open the test file `org.xtext.example.msat.tests/src/ozhang/MSatParsingMacherTest.xtend` and in the test named `benchmark`, replace the variable `benchmark_path` value by the actual benchmark location. All .cnf files within the specified location will then be used for evaluation.

Results of evaluations are stored in a generated `CSV` file which name is specified is `Solver.xtend`, at the beginning.

To make experiments not too long, a timeout of 5 minutes is set for the evaluation. Any solving that is exceeding this duration is annotated as `TIMED OUT` in the `CSV` file.

For the experiments described in this report, the benchmark used is composed of all the 70 `CNF` files within the following directory : `https://github.com/diverse-project/samplingfm/tree/master/Benchmarks`.

This benchmark has been processed 2 times, and results are stored in 2 generated files, `results1.csv` and `results2.csv`. The file `resultsFused.cvs` is a handcrafted file where results of both executions are compared.

2 Experiments

After the experiments, we have the mean execution time of our 5 solvers, which values are displayed in Table 1. This table also show the standard deviation of each solver, to have an insight of the dispersion of the execution time.

Solver	Execution time (ms)	Standard Deviation
Sat4j-java	6705,283	11609,744
Sat4j-jar	0,651	0,666
Sat4j-maven	0,762	0,791
Minisat	0,591	0,604
Cryptominisat	0,478	0,568

Table 1: Mean execution time of solvers

As we can see, the Sat4j-java solver seems to be by far the worse one, with a mean execution time of 6.705 seconds. It also seems that the best solver is Cryptominisat, with an average execution time of 0.4 milliseconds.

If we look at the standard deviation, it looks like the Sat4j-java solver is quite unstable concerning the execution time. When looking to the data, it is clear that some files are difficult to solve for this solver, whereas they are solved with a few seconds for other solvers.

[1]`http://minisat.se/`
[2]`https://github.com/msoos/cryptominisat`
[3]`https://www.sat4j.org/`

When looking at the results, some bugs seem to appear, because some solvers succeed to solve a file almost instantly in the first run, but are timed out in the second one. That happens one time for Minisat for the file `xpose.sk_6_134.cnf`, and one time for Sat4j-jar and Sat4j-maven, on the same file `diagStencil.sk_35_36.cnf`.

Concerning the consistency of the predictions (SAT or UNSAT), the benchmark always generate the same prediction for each file : solvers always provide the same result for a given file, except bugs noticed earlier.

We can notice some particular files :

- `diagStencilClean.sk_41_36.cnf`, where all 3 solvers based on Sat4j are timed out, but other solvers, Minisat and Cryptominisat, solve this file in a few milliseconds. It would be interesting to run Sat4j based solvers on this file with a greater timeout, to see if they actually succeed to solve it.
- `listReverse.sk_11_43.cnf` is taking too much time for all solvers. It would be also interesting to run solvers on this file with a higher timeout, to see if it is actually solvable.

3 Conclusion

Based on the experiments, it is clear that the Sat4j-java solver is the worse one, this the highest mean execution time. It also seems that the Cryptominisat solver is the best one. Some bugs are also exposed by the benchmark, on some particular files.

To improve the evaluation performed in this experiment, more iterations through the benchmark should be done. It would also be interesting to extend the benchmark to other `CNF` files and also random generated formulas. A higher timeout should also be set, 5 minutes is actually to short.