Hermes: Final Report

July 9, 2020

1 Query Dispatcher

Part of the approach that underlies the HERMES project is "choosing the right tool for the job". And indeed, various sub-problems that are described in a HERMES module are dispatched to various reasoning engines such as SMT-solvers, model checkers, etc. This approach can be leverged also in a lower leverl of abstraction. When a (sub-)problem that is decided by HERMES to be solved by an SMT-solver, it is still required to determine *which* SMT-solver will be used.

The approach implemented by the query dispatcher makes use of the fact that several solvers can be run in parallel. Thus, given an input SMT-LIB file, we can dispatch it to several solvers in parallel, return the result obtain by the first solver to get an answer, and terminate the run of the rest of the solvers.

In order to not overuse the system's resources, and given the large number of SMT-solvers, not all possible solvers are executed, but a subset of them is being chosen. To choose this subset efficiently, we need to expect which subset of solvers is best for the particular input problem. The parameter of the input problems that we chose to focus on is the problem's *logic*. Problems in SMT-LIB format are divided into logic, that determine the various types and operators that are allowed to be used. Prominent such logics include, e.g., QF_BV, the quantifier-free logic of bit-vectors, QF_NRA, the quantifier-free logic of non-linear real arithmetic, UFNIA, the quantified logic of non-linear integer arithmetic combined with uninterpreted functions, etc.

Given the input problem's logic (which is typically specified in the first line of the input SMT-LIB file), the way that we project which solvers should be employed is data-driven. For each logic, we choose the solvers that were in the first four places in the yearly SMT competition. The current version of the dispatcher relies on the results from the 2019 competition. In case less than four solvers are supported for the given logic, all of them are chosen.

The dispatcher currently supports the following pool of solvers:

1. CVC4
2. Z3
3. Yices

4. VERIT
5. mathsat
6. SPASS_SATT

In addition to the above functionallity, the dispatcher allows the user to manually choose which solvers will be used, by specifying a list of their names in the command line.

The following snippet is a trace from running the dispatcher. In each line, there is a timeout of 3 seconds which is enforced on the command that is being executed (using `timeout 3s`). In the first line, CVC4 is specified by the user as the chosen solver, and returns a result. Next, Yices is chosen to run on the same file, and does not return a result. In the next two commands, a different benchmark is used with the opposite results: Yices returns an answer while CVC4 does not. Next, the `-s` option of the tool allows the user to specify more than one solver, and so when both CVC4 and Yices are specified, a result is obtained. Finally, the `normal` keyword of the tool invokes the selection algorithm described above, which succesfully results in a solution.

```
$ timeout 3s python3 dispatcher.py term-mzEEAc.smt2 -s cvc4
sat
$ timeout 3s python3 dispatcher.py term-mzEEAc.smt2 -s yices
$ timeout 3s python3 dispatcher.py term-n2YjEt.smt2 -s cvc4
$ timeout 3s python3 dispatcher.py term-n2YjEt.smt2 -s yices
sat
$ timeout 3s python3 dispatcher.py term-n2YjEt.smt2 -s cvc4 yices
sat
$ timeout 3s python3 dispatcher.py term-n2YjEt.smt2 -s normal
sat
```