Planning and Automated Reasoning – Automated Reasoning
II term, Academic Year 2022-23
Project: Implementation of the congruence closure algorithm

Assigned May 8, to be submitted before the final exam (tentatively June 21 at 3:00pm)

Assignment

In this project you are to implement the congruence closure algorithm with DAG for the satisfiability of a set of equalities and disequalities in the quantifier-free fragment of the theory of equality. The algorithm was explained in class and is described in Sect. 9.3 of the Bradley-Manna textbook. Variants to be considered:

- Forbidden list: when calling `MERGE` s t if s is in the forbidden list of t or vice versa, return `unsat` (see page 388 bottom in reference [2b]).
- Non-arbitrary choice of the representative of the new class in the `UNION` function: pick the one with the largest `ccpar` set (see page 761 top in reference [2c] and page 423 top in reference [2b]).

You can choose your favorite programming language, provided it is a general-purpose portable programming language that can be compiled on a Linux PC. The program will have an interface (stdin/stdout or a simple GUI) that allows the user to submit an input formula (typically contained in a file) and get the answer.

In order to test your program, consider sets of equalities and disequalities from the following sources:

1. Examples and exercises from the books in the book list of the course, and from books and papers in the references below.
2. Sets of equalities and disequalities obtained from more general formulas (also from books or papers) as follows, e.g.:
   - If the formula contains predicate symbols other than equality, eliminate them by implementing the transformation explained in class;
   - If the formula contains defined (constant or function) symbols from other theories (e.g., arithmetical symbols), replace them with free symbols;
   - If the formula is not a conjunction of literals, transform it into DNF, so that your program can be applied to each disjunct;

- If the formula contains quantifiers, drop them and treat the variables as free variables.

Since not all these transformations preserve equisatisfiability, one may end up solving a different problem. The suggested transformations are merely intended to get more inputs for the algorithm.

3. The benchmarks in the `QF-UF` class of the SMT-LIB repository. QF stands for quantifier free and UF stands for undefined function symbols. Since all logics/theories in the SMT-LIB repository feature equality, `QF-UF` stands for the quantifier-free fragment of the theory of equality. The benchmarks of SMT-LIB are currently hosted on GitLab and those for `QF-UF` are at `https://clc-gitlab.cs.uiowa.edu:2443/SMT-LIB-benchmarks/QF_UF/-/tree/master/` These banchmarks are written in the SMT-LIB language that is far more complex than the logic used in class, books, and papers. Thus, if you wish to give these inputs to your program, you need a parser that handles the subset of SMT-LIB used in the `QF-UF` benchmarks.

Furthermore, you are to write a report (max 6 pages 11pt) presenting

- Your implementation of the algorithm, emphasizing major choices (e.g., data structures, the above mentioned variants, heuristics if any) or any other information that you deem significant;
- A summary of the results of the experiments in the form of one or more tables or plots, reporting data such as the answer (SAT/UNSAT), the run time, the source of the problems, or other data that you deem relevant;
- Some analysis of the experiments, such as comments about performance, impact of features, or any other remarks that you deem interesting.

Hand-in

1. A compressed archive (e.g., .tgz[1] or .zip) to be sent to the instructor by e-mail. The archive should contain source code, executable, `README` file, input files used in the experiments and the corresponding output files. The input files should include a comment stating the source of the problem. The `README` file should contain instructions to execute the program. The archive should be named FirstNameLastNameStudentId (i.e., NomeCognomeNumeroMatricola).
2. A double-sided print-out of the report that you can put in the instructor's mailbox.

[1] E.g., `tar czvf - FolderName > FolderName.tgz` compresses and `tar xzpvf FolderName.tgz` decompresses.

References

1. Textbooks:

   (a) Aaron R. Bradley, Zohar Manna. *The Calculus of Computation. Decision Procedures with Applications to Verification.* Springer, 2007, ISBN 978-3-642-09347-0.

   (b) Daniel Kroening, Ofer Strichman: *Decision Procedures. An Algorithmic Point of View*, Springer, 2008, ISBN: 978-3-540-74104-6.

2. Papers:

   (a) Leo Bachmair, Ashish Tiwari and Laurent Vigneron. Abstract congruence closure. Journal of Automated Reasoning 31(2):129–168, 2003.

   (b) David L. Detlef, Greg Nelson and James B. Saxe. Simplify: a theorem prover for program checking. Journal of the ACM 52(3):365–473, 2005.

   (c) Peter J. Downey, Ravi Sethi, and Robert Endre Tarjan. Variations on the common subexpression problem. Journal of the ACM 27(4):758–771, 1980.

   (d) Dexter Kozen. Complexity of finitely presented algebras. Technical Report TR-76-294, Department of Computer Science, Cornell University, 1976.

   (e) Greg Nelson and Derek C. Oppen. Fast decision procedures based on congruence closure. Journal of the ACM 27(2):356–364, 1980.

   (f) Robert Nieuwenhuis and Albert Oliveras. Fast congruence closure and extensions. Information and Computation 205:557–580, 2007.

   (g) Robert E. Shostak. An algorithm for reasoning about equality. Communications of the ACM 21(7):583–585, 1978.