Riss 4.27

Norbert Manthey
Knowledge Representation and Reasoning Group
TU Dresden, Germany

***Abstract*—The solver RISS combines the improved Minisat-style solving engine of GLUCOSE 2.2 with a state-of-the-art preprocessor COPROCESSOR and adds further modifications to the search process.**

I. INTRODUCTION

The CDCL solver RISS in version 4.27 was first build on the MINISAT search engine [1], and next incorporated the improvements that have been proposed for GLUCOSE 2.2 [2], [3]. Afterwards, more search algorithm extensions have been added. RISS is equipped with the preprocessor COPROCESSOR [4], that implements most of the recently published formula simplification techniques, ready to be used as inprocessing as well by taking care of learned clauses. The aim of the solver is to provide a huge portfolio of options on the one hand, to be able to adopt to new SAT applications, with an efficient implementation on the other hand.

II. MAIN TECHNIQUES

RISS uses the modifications of the CDCL algorithm that are proposed in [5], namely *local look-ahead*, *all-unit-UIP learning* and *on-the-fly probing*. An additional learned clause minimization technique, based on *hidden literal elimination* is added. Another addition to the solver is the ability to perform *restricted extended resolution* (RER) during search as described by Audemard et al. [6]. However, the used implementation differs to reduce the overhead of the procedure: introduced variables are not deleted, and new learned clauses are only rewritten with the most recent extension of a literal. During unit propagation, *lazy hyper binary resolution* [7] can be used, and *on-the-fly clause improvement* [8] is available during conflict analysis. During learning, usual first-UIP clauses can be learned [9], but alternatively the first *bi-asserting* clause can be learned instead [10]. For large learned clauses with a high LBD value, a so called *decision clause* is learned, which simply negates the current decision literals [11], because this clause is assumed to be removed from the formula again soon. The activity decay for variables that are used in clause learning steps can be adopted dynamically, similarly to the method used in GLUCOSE 2.3 [12]. When learned clauses are removed again, then the set of the heuristically worst clauses can be kept. Finally, the algorithm proposed by Wieringa [13] to strengthen learned clauses in parallel to the actual CDCL search has been sequentialized, so that search can be interleaved with such an clause strengthening technique.

For the initialization of the activities of variables, as well as the polarities for polarity caching [14], several opportunities are offered, among them the *Jeroslow-Wang* heuristic [15]. For scheduling restarts, the scheme presented for GLUCOSE 2.2 is used by default, but also the *geometric series* of MINISAT can be used [1], or the LUBY series [16]. *Partial restart* are also available [17].

The built-in preprocessor COPROCESSOR incorporates the following formula simplification techniques: Unit Propagation, Subsumption, Strengthening (also called self-subsuming resolution) – where for small clauses all subsuming resolvents can be produced, (Bounded) Variable Elimination (BVE) [18] combined with Blocked Clause Elimination (BCE) [19], (Bounded) Variable Addition (BVA) [20], Probing [21], Clause Vivification [22], Covered Clause Elimination [23], Hidden Tautology Elimination [24], Equivalent Literal Substitution [25], Unhiding (Unhide) [26], Adding Binary Resolvents [27], a 2SAT algorithm [28], and a walksat implementation [29]. The preprocessor furthermore supports parallel subsumption, strengthening and variable elimination [30].

Since the last version of RISS [31], the following simplification techniques have been added: The implementation of Unhide now supports finding equivalent literals and can remove redundant binary clauses. *Structural hashing* is performed during equivalent literal elimination [32], and the *Tarjan* algorithm to find strongly connected components in the binary implication graph is now implemented iteratively. The *Fourier-Motzkin method* (FM) for reasoning on cardinality constraints, similar to the procedure by Biere [11], is used as a preprocessing step, where the cardinality constraints can be extracted syntactically for degrees less than 3, and for higher degrees a semantic method based on unit propagation is used [33]. Furthermore, an algorithm is embedded, which is able to derive new clauses from a special set of cardinality constraints: when at-most-one constraints for rows, and at-least-one constraints for columns encode a possibly over constraint matching problem, then the at-least-one constraints for the rows can are deduced and added in form of clauses. Another deduction systems that is stronger than resolution is available in COPROCESSOR: by retrieving XOR gates from the CNF, with the help of the Gaussian elimination equivalent literals and unit clauses can be found. The elimination of *resolution asymmetric tautologies* is implemented in a first naive version, which performs the same steps as the theoretical description: building the resolvent and testing this resolvent for being an asymmetric tautology by running full unit propagation [34]. For an extension $x \leftrightarrow (a \wedge b)$, BVA also replaces the disjunction $(\overline{a} \vee \overline{b})$ in the formula with $\overline{x}$, and adds the full extension. Furthermore, BVA can be search for

other gate types than AND-gates: XOR-gates and IF-THEN-ELSE-gates can be used additionally. Finally, the counter technique of *covered literal addition* [23], which is used to eliminate covered clauses, is added to COPROCESSOR: during the computation of *BCE*, *covered literal elimination* (CLE) is performed [35], which removes literals from a clause, if these literals can be added by *covered literal addition* again [23].

RISS is able to output proofs for unsatisfiable formulas in the DRUP format [36], [37], also when look-ahead or the all-units-learning modifications are enabled. Furthermore, most of the techniques inside COPROCESSOR are able to produce proofs. If a technique introduces fresh variables or cannot be simulated by unit propagation (easily), DRAT proofs are printed instead [38], as required for example by RER, BVA or CLE. Inside the solver the generated proof can be verified during its construction. Techniques that do not support producing proofs yet are Gaussian elimination and FM. Furthermore, for some probing based simplification techniques proofs are not produced. For more details on the generation of proofs and simplification techniques see [35].

III. MAIN PARAMETERS

RISS offers all the parameters that are available in GLUCOSE 2.2. Furthermore, all the techniques that are mentioned above can be enabled or disabled, and the number of execution steps per technique can be limited, as well as variants can be produced. The total number of parameters for the solver is 486, where 190 of these parameters are Boolean, and the remaining parameters have either floating point or integer domains.

For the SAT Competition 2014 the formula size limits for the formula simplification techniques have been set, so that these techniques do not consume too much run time. Next, a set of well performing techniques was determined by a local-search like selection procedure. Based with this configuration, the parameters for the search algorithm have been tuned. Finally, the search configuration has been combined with the techniques of the formula simplification.

IV. SPECIAL ALGORITHMS, DATA STRUCTURES, AND OTHER FEATURES

In GLUCOSE 2.2, binary clauses are handled specially during propagation: both literals of the clause can be retrieved from the watch list, so that the actual clause is not touched. Therefore, GLUCOSE 2.2 introduces an extra watch list for binary clauses. RISS keeps all clauses in a single watch list, but applies that same idea by modifying the elements in the watch list so that they know whether the watched clause is binary or not. This modification reduces the memory consumption of the solver. Another reduction of the memory consumption is achieved by merging multiple bit arrays into a single array and using bit operations. Especially when new variables are introduced by BVA or RER, the memory fragmentation is lower with this modification.

V. IMPLEMENTATION DETAILS

RISS and COPROCESSOR are implemented in C++. All simplification techniques inside COPROCESSOR are implemented in separate classes to increase the structure of the code.

VI. SAT COMPETITION 2013 SPECIFICS

RISS is submitted as a 64-bit binary to the SAT and SAT+UNSAT tracks for the categories Application and Crafted. The compilation uses the flag "-O3".

The submitted configuration of RISS 4.27 uses the following techniques, where FM and variable renaming is disabled for the certified unsatisfiable tracks: BVE, FM, five iterations of UNHIDE, CLE and variable renaming to compact the representation of the formula during search.

Furthermore, the version DRAT uses the same configuration as RISS3G [31] for all application tracks and crafted tracks, because the used techniques BVE, BVA, Unhide, and Local Look-Ahead, support the DRAT proof format.

VII. AVAILABILITY

RISS, as well as the formula simplifier COPROCESSOR are available for research. The collection additionally contains the parallel search space splitting SAT solver PCASSO, the portfolio SAT solver PRISS that can produce DRUP proofs [39], and a CNF feature extraction. The framework can be downloaded from http://tools.computational-logic.org.

ACKNOWLEDGMENT

The author would like to thank the developers of GLUCOSE 2.2 and MINISAT 2.2. Furthermore, many thanks go to Marijn Heule for discussions on the proof formats DRUP and DRAT, as well as to Armin Biere for discussions on formula simplification techniques. The computational resources to develop, evaluate and configure the SAT solver have been provided by the BWGrid [40] project and the ZIH of TU Dresden.

REFERENCES

[1] N. Eén and N. Sörensson, "An extensible SAT-solver," in *SAT 2003*, ser. LNCS, E. Giunchiglia and A. Tacchella, Eds., vol. 2919. Heidelberg: Springer, 2004, pp. 502–518.

[2] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern SAT solvers," in *IJCAI 2009*, C. Boutilier, Ed. Pasadena: Morgan Kaufmann Publishers Inc., 2009, pp. 399–404.

[3] ——, "Refining restarts strategies for sat and unsat," in *CP'12*, 2012, pp. 118–126.

[4] N. Manthey, "Coprocessor 2.0 a flexible CNF simplifier," in *SAT 2012*, ser. LNCS, A. Cimatti and R. Sebastiani, Eds., vol. 7317. Heidelberg: Springer, 2012, pp. 436–441.

[5] ——, "Enhancing CDCL solvers with local look-ahead, all-unit-uip learning and on-the-fly probing," in *LaSh 2014*, 2014, submitted.

[6] G. Audemard, G. Katsirelos, and L. Simon, "A restriction of extended resolution for clause learning sat solvers," M. Fox and D. Poole, Eds. AAAI Press, 2010.

[7] A. Biere, "PrecoSAT system description," http://fmv.jku.at/precosat/preicosat-sc09.pdf, 2009.

[8] H. Han and F. Somenzi, "On-the-fly clause improvement," in *SAT*, 2009, pp. 209–222.

[9] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in *DAC 2001*. New York: ACM, 2001, pp. 530–535.

[10] K. Pipatsrisawat and A. Darwiche, "A new clause learning scheme for efficient unsatisfiability proofs," in *AI 2008*. AAAI Press, 2008, pp. 1481–1484.

[11] A. Biere, "Lingeling, Plingeling and Treengeling entering the SAT competition 2013," ser. Department of Computer Science Series of Publications B, A. Balint, A. Belov, M. J. Heule, and M. Järvisalo, Eds., vol. B-2013-1. University of Helsinki, Helsinki, Finland, 2013, pp. 51–52.

[12] G. Audemard and L. Simon, "Glucose 2.3 in the SAT 2013 competition," ser. Department of Computer Science Series of Publications B, A. Balint, A. Belov, M. J. Heule, and M. Järvisalo, Eds., vol. B-2013-1. University of Helsinki, Helsinki, Finland, 2013, pp. 42–43.

[13] S. Wieringa and K. Heljanko, "Concurrent clause strengthening," in *SAT*, ser. LNCS, vol. 7962, 2013, pp. 116–132.

[14] K. Pipatsrisawat and A. Darwiche, "A lightweight component caching scheme for satisfiability solvers," in *SAT*, ser. LNCS, vol. 4501, 2007, pp. 294–299.

[15] R. G. Jeroslow and J. Wang, "Solving propositional satisfiability problems," *Annals of Mathematics and Artificial Intelligence*, vol. 1, pp. 167–187, 1990.

[16] M. Luby, A. Sinclair, and D. Zuckerman, "Optimal speedup of las vegas algorithms," *Inf. Process. Lett.*, vol. 47, pp. 173–180, September 1993.

[17] P. van der Tak, A. Ramos, and M. Heule, "Reusing the assignment trail in cdcl solvers," *JSAT*, vol. 7, no. 4, pp. 133–138, 2011.

[18] N. Eén and A. Biere, "Effective preprocessing in SAT through variable and clause elimination," in *SAT 2005*, ser. LNCS, F. Bacchus and T. Walsh, Eds., vol. 3569. Heidelberg: Springer, 2005, pp. 61–75.

[19] M. Järvisalo, A. Biere, and M. Heule, "Blocked clause elimination," in *TACAS 2010*, ser. LNCS, J. Esparza and R. Majumdar, Eds., vol. 6015. Heidelberg: Springer, 2010, pp. 129–144.

[20] N. Manthey, M. J. H. Heule, and A. Biere, "Automated reencoding of Boolean formulas," in *HVC 2012*, 2012.

[21] I. Lynce and J. P. Marques-Silva, "Probing-based preprocessing techniques for propositional satisfiability," in *ICTAI 2003*. Sacramento, California, USA: IEEE Computer Society, 2003, pp. 105–110.

[22] C. Piette, Y. Hamadi, and L. Saïs, "Vivifying propositional clausal formulae," in *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2008, pp. 525–529.

[23] M. Heule, M. Järvisalo, and A. Biere, "Covered clause elimination," ser. LNCS, C. G. Fermüller and A. Voronkov, Eds., vol. 6397. Heidelberg: Springer, 2010, pp. 41–46.

[24] ——, "Clause elimination procedures for cnf formulas," ser. LNCS, C. G. Fermüller and A. Voronkov, Eds., vol. 6397. Heidelberg: Springer, 2010, pp. 357–371.

[25] A. V. Gelder, "Toward leaner binary-clause reasoning in a satisfiability solver," *Ann. Math. Artif. Intell.*, vol. 43, no. 1, pp. 239–253, 2005.

[26] M. Heule, M. Järvisalo, and A. Biere, "Efficient CNF simplification based on binary implication graphs," in *SAT*, ser. LNCS, vol. 6695, 2011, pp. 201–215.

[27] W. Wei and B. Selman, "Accelerating random walks," in *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, ser. CP '02. London, UK, UK: Springer-Verlag, 2002, pp. 216–232.

[28] A. del Val, "On 2-sat and renamable horn," in *AAAI/IAAI*, H. A. Kautz and B. W. Porter, Eds., 2000, pp. 279–284.

[29] B. Selman, H. A. Kautz, and B. Cohen, "Noise strategies for improving local search," in *AAAI*, B. Hayes-Roth and R. E. Korf, Eds., 1994, pp. 337–343.

[30] K. Gebhardt and N. Manthey, "Parallel variable elimination on CNF formulas," in *KI 2013: Advances in Artificial Intelligence,*, ser. LNCS, I. J. Timm and M. Thimm, Eds., vol. 8077, 2013, pp. 61–73.

[31] N. Manthey, "The SAT solver RISS3G at SC 2013," ser. Department of Computer Science Series of Publications B, A. Balint, A. Belov, M. J. Heule, and M. Järvisalo, Eds., vol. B-2013-1. University of Helsinki, Helsinki, Finland, 2013, pp. 72–73.

[32] A. Kuehlmann and F. Krohm, "Equivalence checking using cuts and heaps," in *DAC*, 1997, pp. 263–268.

[33] A. Biere, D. L. Berre, E. Lonca, and N. Manthey, "Detecting cardinality constraints in CNF," in *SAT*, 2014, accepted.

[34] M. Järvisalo, M. J. H. Heule, and A. Biere, "Inprocessing rules," in *IJCAR 2012*, ser. LNCS, B. Gramlich, D. Miller, and U. Sattler, Eds., vol. 7364. Heidelberg: Springer, 2012, pp. 355–370.

[35] N. Manthey and T. Philipp, "Formula simplifications as DRAT derivations," in *German Conference on A.I. (KI 2014)*, 2014, submitted.

[36] E. Goldberg and Y. Novikov, "Verification of proofs of unsatisfiability for cnf formulas," in *Proceedings of the conference on Design, Automation and Test in Europe - Volume 1*, ser. DATE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 10 886–.

[37] M. Heule, W. A. H. Jr., and N. Wetzler, "Trimming while checking clausal proofs," in *FMCAD*, 2013.

[38] N. Wetzler, M. Heule, and W. A. H. Jr., "Drat-trim: Efficient checking and trimming using expressive clausal proof," in *SAT*, 2014, accepted.

[39] M. J. H. Heule, N. Manthey, and T. Philipp, "Validating unsatisfiability results from clause sharing parallel sat solvers," 2014, submitted.

[40] bwGRiD (http://www.bw grid.de/), "Member of the german d-grid initiative, funded by the ministry of education and research (bundesministerium für bildung und forschung) and the ministry for science, research and arts baden-wuerttemberg (ministerium für wissenschaft, forschung und kunst baden-württemberg)," Universities of Baden-Württemberg, Tech. Rep., 2007-2010.

[41] A. Balint, A. Belov, M. J. Heule, and M. Järvisalo, Eds., *Proceedings of SAT Challenge 2013*, ser. Department of Computer Science Series of Publications B. University of Helsinki, Helsinki, Finland, 2013, vol. B-2013-1.

[42] C. G. Fermüller and A. Voronkov, Eds., *LPAR 2010*, ser. LNCS, vol. 6397. Heidelberg: Springer, 2010.