SEQFROST at the SAT Race 2022

Muhammad Osama and Anton Wijs
Department of Mathematics and Computer Science
Eindhoven University of Technology, Eindhoven, The Netherlands
{o.m.m.muhammad, a.j.wijs}@tue.nl

I. INTRODUCTION

This paper presents a brief description of our solver SEQFROST which stands for *Sequential Formal ReasOning about SaTisfiability* in 3 different configurations. SEQFROST is a new solver mostly rewritten from scratch based on our last year submission [1] with efficient data structures and many code optimizations. This year, we observed a large amount of time is spent on function calls in Boolean Constraint Propagation (BCP) and data sorting especially in Multiple-Decision Making (MDM) and inprocessing. Thus, we resorted to inlined code and pointer prefetching in BCP and replaced the standard *sort* procedure with faster modern alternatives, e.g., *pdqsort*. Further, we augment the Multi-Arm Bandit (MAB) rewarding scheme as implemented in the last year winner KISSAT-MAB to MDM strategy [2], and implement functional dependency extraction to enhance the effectiveness of variable elimination. Finally, we extend our elimination method *eager redundancy elimination* (ERE) [1], [3] with clause strengthening to remove redundant literals.

II. DECISION MAKING

The decision-making step in SEQFROST switches periodically from the standard single-decision procedure as originally introduced in CDCL search to our MDM procedure previously presented in [2]. Both single and multiple decisions are chosen according to VSIDS, VMTF, and CHB [4] branching heuristics. This year, we add the latter to our solver decision heuristics to alleviate the quality of the picked decisions in MDM. SEQFROST decides whether to use VSIDS or CHB based on MAB restarts [5]. The decision phases of multiple decisions are still improved via local search but only once at the initial MDM call.

III. VARIABLE ELIMINATION

In gate-equivalence reasoning, we substitute eliminated variables with deduced logical equivalent expressions. Combining gate equivalence reasoning with the resolution rule tends to result in smaller formulas compared to only applying the resolution rule [1], [3], [6]–[9]. Let G_ℓ be the gate clauses having ℓ as the gate output and H_ℓ the non-gate clauses, i.e., clauses not contributing to the gate itself. For regular gates (e.g. AND), substitution can be performed by resolving non-gate with gate clauses as follows: $R_x = \{\{G_x \otimes H_{\neg x}\}, \{G_{\neg x} \otimes H_x\}\}$, omitting the tautological and the redundant parts $\{G_x \otimes G_{\neg x}\}$ and $\{H_x \otimes H_{\neg x}\}$, respectively [6].

In this submission, we focus on finding definitions for irregular gates by checking the unsatisfiability of the *co-factors* formula $\{\mathcal{S}_x|_{\neg x} \cup \mathcal{S}_{\neg x}|_x\}$, that is, the formula obtained by removing all occurrences of x from $\mathcal{S}_x$ and $\neg x$ from $\mathcal{S}_{\neg x}$. In [10], a BDD-based approach is used to solve the co-factors. In this work, we replace the BDD structure with a function table (bit-vector) encoding the clausal core of the co-factors. The clausal core is mapped back to the original gate clauses G_x and $G_{\neg x}$ by adding back x and $\neg x$, respectively. Then, the set of resolvents $R_x = \mathcal{S}_x \otimes \mathcal{S}_{\neg x}$ is reduced to $\{\{G_x \otimes G_{\neg x}\}, \{G_x \otimes H_{\neg x}\}, \{G_{\neg x} \otimes H_x\}\}$, dropping the redundant part $\{H_x \otimes H_{\neg x}\}$. In contrast to gate substitution, the resolvents $\{G_x \otimes G_{\neg x}\}$ are not necessarily tautological.

IV. EAGER REDUNDANCY ELIMINATION

ERE was designed originally to target and remove redundant equivalences after a resolution step. It repeats the following until a fixpoint has been reached: for a given formula $\mathcal{S}$ and clauses $C_1 \in \mathcal{S}, C_2 \in \mathcal{S}$ with $x \in C_1$ and $\bar{x} \in C_2$ for some variable x, if there exists a clause $C \in \mathcal{S}$ for which $C \equiv C_1 \otimes_x C_2$, then let $\mathcal{S} := \mathcal{S} \setminus \{C\}$ iff (C is $learnt$ $\vee$ (C_1 is $original \wedge C_2$ is $original$)). The clause C in this case is called a *redundancy* and can be removed without altering the original satisfiability. In addition to the redundancies removal, we observed that if the resolvent $C_1 \otimes_x C_2$ is not equivalent to any clause, it can still subsume many others in $\mathcal{S}$. However, to preserve correctness, subsumed clauses are only strengthened via the generated resolvents. Suppose that $C = (C' \cup C'')$. Extended-ERE (i.e. as we call it in this submission) may strengthen C by removing the redundant literals C' (resp. C'') if $C'' = C_1 \otimes_x C_2$ (resp. $C' = C_1 \otimes_x C_2$).

V. CODE OPTIMIZATIONS

As mention earlier in the introduction section, all pointers of vector-type variables are prefetched to save the time spent in calling the overloaded indexing operator `[]`. Additionally, all functions repeatedly called in unit propagation and conflict analysis are replaced with macros as inlining is not always guaranteed by the compiler. Lastly, the bytes generated by DRAT proof are now stored in a 1-MB buffer. Once, the buffer is full, the data is written to the output file via a single call to `fwrite` (i.e. writes data in burst mode). Compared to previous submissions and other solvers, `putc_unlock` was being called to write on disk byte by byte which, of course, adds unnessary overhead to the proof generation.

VI. SUBMISSIONS

The solver instance SEQFROST comprises all configurations described in the previous sections, in which MDM with local search, CHB decision heuristic, and all simplifications are enabled with Extended ERE to strengthen original clauses only (e.g. the option `redundancyextend=1` is set). The second configuration SEQFROST-ERE-ALL extends ERE with both original and learnt clause strengthening (e.g. `redundancyextend=2`). The third configuration SEQFROST-NO-EXTEND disables Extended ERE (e.g. `redundancyextend=0`). The initial settings of the SEQFROST have been tuned and tested on the DAS-5 cluster [11] and the Dutch national supercomputer SNELLIUS[1].

REFERENCES

[1] M. Osama and A. Wijs, "ParaFROST at the SAT Race 2021," in *Proc. of SC (2021)*, ser. Report Series B, vol. B-2021-1. University of Helsinki, 2021, pp. 32–34. [Online]. Available: http://hdl.handle.net/10138/333647

[2] ——, "Multiple Decision Making in Conflict-Driven Clause Learning," in *Proc. of ICTAI (Nov. 2020), Baltimore, USA*. IEEE, 2020, pp. 161–169.

[3] M. Osama, A. Wijs, and A. Biere, "SAT Solving with GPU Accelerated Inprocessing," in *Proc. of TACAS (Mar. 2021), Luxembourg*, ser. LNCS, vol. 12651. Springer, 2021, pp. 133–151.

[4] J. H. Liang, V. Ganesh, P. Poupart, and K. Czarnecki, "Exponential recency weighted average branching heuristic for sat solvers," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, p. 3434–3440.

[5] M. S. Cherif, D. Habet, and C. Terrioux, "Combining VSIDS and CHB Using Restarts in SAT," in *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), L. D. Michel, Ed., vol. 210. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, pp. 20:1–20:19.

[6] M. Järvisalo, M. Heule, and A. Biere, "Inprocessing Rules," in *Proc. of IJCAR (Jun. 2012), Manchester, UK*, ser. LNCS, vol. 7364. Springer, 2012, pp. 355–370.

[7] M. Osama and A. Wijs, "GPU Acceleration of Bounded Model Checking with ParaFROST," in *Proc. of CAV (Jul. 2021), USA*, ser. LNCS, vol. 12760. Springer, 2021, pp. 447–460.

[8] ——, "Parallel SAT Simplification on GPU Architectures," in *Proc. of TACAS (Apr. 2019), Prague, Czech Republic*, ser. LNCS, vol. 11427. Springer, 2019, pp. 21–40.

[9] ——, "SIGmA: GPU Accelerated Simplification of SAT Formulas," in *Proc. of IFM (Dec. 2019), Bergen, Norway*, ser. LNCS, vol. 11918. Springer, 2019, pp. 514–522.

[10] A. Biere, "Lingeling, Plingeling and Treengeling Entering the Sat Competition 2013," in *Proc. of SC (2013)*, ser. Report Series B, vol. B-2013-1. University of Helsinki, 2013, pp. 51–52. [Online]. Available: http://hdl.handle.net/10138/40026

[11] H. E. Bal, D. H. J. Epema, C. de Laat, R. van Nieuwpoort, J. W. Romein, F. J. Seinstra, C. Snoek, and H. A. G. Wijshoff, "A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term," *Computer*, vol. 49, no. 5, pp. 54–63, 2016.

[1]This work was carried out on the Dutch national e-infrastructure with the support of SURF cooperative.