DPS-Kissat

Hidetomo Nabeshima Tsubasa Fukiage Yuto Obitsu Xiao-Nan Lu
University of Yamanashi
Yamanashi, JAPAN
{nabesima,xnlu}@yamanashi.ac.jp

Katsumi Inoue
National Institute of Informatics
Tokyo, JAPAN
inoue@nii.ac.jp

***Abstract*—DPS is a new framework for easily constructing efficient deterministic parallel SAT solvers, providing the delayed clause exchange technique introduced in ManyGlucose. DPS-Kissat applies our framework to Kissat, a state-of-the-art sequential SAT solver.**

I. Introduction

DPS is a framework for easily implementing deterministic portfolio parallel SAT solvers for shared memory multi-core environment, that guarantee reproducible behavior. Reproducibility means that the runtime and a found model if satisfiable are the same across runs. The framework provides the delayed clause exchange proposed in ManyGlucose [1] and supports the realization of efficient deterministic parallel solvers. We applied DPS to Kissat [2], one of the state-of-the-art sequential SAT solvers, to construct a portfolio parallel SAT solver DPS-Kissat.

II. Delayed Clause Exchange

The delayed clause exchange introduced in ManyGlucose allows a certain delay in the timing of clause exchanges, thereby absorbing fluctuations in the exchange interval and significantly reducing the waiting time, even in many-core environments. However, implementing delayed clause exchange requires expert knowledge of concurrent programming, so introducing it into existing sequential SAT solvers is a time-consuming task. We have extracted the delayed clause exchange method from ManyGlucose and developed a framework DPS with a generic interface to facilitate its integration into existing sequential solvers.

III. Portfolio Strategy

The diversity strategy of DPS-Kissat is simple and consists of the following two elements:

1) random variable selection until the first conflict occurs except for the first thread. The random seeds use different values for each thread.
2) disabled elimination in half of threads.

The former strategy was introduced in ManySAT 2.0 [3], the first definitive parallel SAT solver. Clause exchange is performed asynchronously in non-deterministic parallel SAT solvers and is one of the reasons for the diversity of search in these solvers. However, this is not expected in deterministic solvers, and random decision is required to ensure diversity. The latter strategy was introduced because there were some instances where a lot of time was spent on in-processing.

IV. Implementation

DPS-Kissat is a parallelization of Kissat-SC2021 [4], which required about 400 lines of modification to Kissat and about 400 lines for the wrapper class to incorporate Kissat into DPS. DPS-Kissat, submitted to the SAT competition 2022, launches 32 threads, each of which executes Kissat in parallel. The results are guaranteed to be reproducible.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Numbers JP20H05963, JP20K11934. In this research work we used the supercomputer of ACCMS, Kyoto University.

References

[1] H. Nabeshima and K. Inoue, “Reproducible efficient parallel SAT solving,” in *Proceedings of the 23rd International Conference on Theory and Applications of Satisfiability Testing (SAT 2020), LNCS 12178*, 2020, pp. 123–138.

[2] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger, “CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT competition 2020,” http://hdl.handle.net/10138/318450, 2020, SAT Competition 2020 Solver Description.

[3] Y. Hamadi, S. Jabbour, C. Piette, and L. Sais, “Deterministic parallel DPLL,” *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 7, no. 4, pp. 127–132, 2011.

[4] A. Biere, M. Fleury, and M. Heisinger, “CaDiCaL, Kissat, Paracooba entering the SAT competition 2021,” http://hdl.handle.net/10138/333647, 2021, SAT Competition 2021 Solver Description.