Z3-Owl at SMT-COMP 2023

Xinkai Ma[1], Jiahui Sun[1], Siyuan Zhu[1], Peisen Yao[1], Rui Chang[1], Yongwang Zhao[1], Wensheng Tang[2], and Charles Zhang[2]

[1] Zhejiang University
{`maxinkai, jasonj, syuanz, pyaoaa, crix1021, zhaoyw`}`@zju.edu.cn`
[2] The Hong Kong University and Science and Technology
{`wtangae, charlesz`}`@cse.ust.hk`

Abstract. In this report, we present Z3-Owl, a derived Satisfiability Modulo Theories (SMT) solver for the theories of bit-vectors, floating-points, arrays, uninterpreted functions, linear real arithmetic, linear integer arithmetic, and their combinations. We discuss the selected features for its participation in SMT-COMP 2023.

1 Introduction

Z3-Owl is a derived SMT solver based on Z3 [1] (version 4.8.11) and PySAT [4] (version 0.1.8.dev1). It participates in the single query and parallel tracks in the following divisions:

– *Single Query Track*: QF_BV, QF_UFBV, QF_ABV, QF_AUFBV, QF_FP, QF_BVFP.
– *Parallel Track*: QF_BV.

For more information, readers can refer to the following Web site.

`https://z3-owl.github.io/`

2 Features

In this section, we present the features of Z3-Owl on the above divisions.

2.1 Sequential Solving

Z3-Owl solves bit-vector and floating points formulas following the eager approach to SMT solving. Consider an input formula φ, Z3-Owl works as follows:

– First, we use the tactic system of Z3 to customize a word-level pre-processing strategy. The strategy can either solve φ directly or generate a simplified formula φ'.
– Second, if the input formula is not solved by the first phase, we use the "bit-blast" tactic of Z3 that translates the simplified formula into a Boolean formula φ_{bool}.

Table 1. Used SAT engine and its version for sequential solving.

Theory	SAT engine	Version
QF_BV	CaDiCaL	??
QF_UFBV	MiniSAT?	??
QF_ABV	CadiCaL?	??
QF_AUFBV	Glucose	??
QF_FP	??	??
QF_BVFP	??	??

– Finally, we solve te Boolean formula φ_{bool} via an off-the-shelf SAT solver through PySAT, which wrappers a set of state-of-the-art SAT engines.

Compared to Z3, our solver uses new word-level pre-processing strategies and allows for more diversified SAT engines. Specifically, Table 1 lists the configurations we use for this competition.

2.2 Parallel Solving

We adopt the portfolio approach for parallel solving. In a bit-blasting-based solver, the effectiveness of both the word-level pre-processing and SAT solving engine significantly contributes to the solver's performance [2]. Thus, our approach utilizes a two-layered design.

– First, we employs Z3's tactic system to establish a suite of parallel pre-processing strategies that operate on an input formula, with the aim of either solving the formula directly or producing a simplified variant of it. The resulting simplified formulas are then transformed into distinct Boolean formulas via bit-blasting.
– Second, to solve the Boolean formulas, we use pySAT to concurrently call multiple SAT engines and choose the result returned by the first engine.

In summary, our solver aims to improve the overall performance by utilizing a combination of pre-processing strategies and multiple SAT engines, following the idea of swarm verification [3].

Acknowledgement

We are grateful to the developers of Z3, pySAT, CadiCaL, MinisSAT, and ✎ [yao: ...] for making their systems available for open use.

References

1. De Moura, L., Bjørner, N.: Z3: An efficient smt solver. In: Tools and Algorithms for the Construction and Analysis of Systems: 14th International Conference, TACAS

2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings 14. pp. 337–340. Springer (2008)

2. Dutertre, B.: An empirical evaluation of SAT solvers on bit-vector problems. In: Bobot, F., Weber, T. (eds.) Proceedings of the 18th International Workshop on Satisfiability Modulo Theories co-located with the 10th International Joint Conference on Automated Reasoning (IJCAR 2020), Online (initially located in Paris, France), July 5-6, 2020. CEUR Workshop Proceedings, vol. 2854, pp. 15–25. CEUR-WS.org (2020), `https://ceur-ws.org/Vol-2854/paper1.pdf`
3. Holzmann, G.J., Joshi, R., Groce, A.: Swarm verification techniques. IEEE Trans. Software Eng. **37**(6), 845–857 (2011). https://doi.org/10.1109/TSE.2010.110, `https://doi.org/10.1109/TSE.2010.110`
4. Ignatiev, A., Morgado, A., Marques-Silva, J.: Pysat: A python toolkit for prototyping with sat oracles. In: Theory and Applications of Satisfiability Testing–SAT 2018: 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9–12, 2018, Proceedings. pp. 428–437. Springer (2018)