Z3-Parti-Z3++ at SMT-COMP 2025

Mengyu Zhao[1,2] and Shaowei Cai[1,2]

[1] Key Laboratory of System Software (Chinese Academy of Sciences) and
State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences, Beijing, China
{zhaomy, caisw}@ios.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

1 Introduction

Z3-Parti-Z3++ is a derived SMT solver from Z3 [3] and participates in the Parallel Track of the Arithmetic theories, namely QF_RDL, QF_IDL, QF_LRA, QF_LIA, QF_NRA, and QF_NIA logics. For SMT-COMP 2024, you can find the solver, experimental scripts, and Docker files we have prepared at GitHub-Z3-Parti-Z3++-at-SMT-COMP-2025.

Z3-Parti-Z3++ comprises the following three primary components: The master is implemented by Python for task management and scheduling in distributed solving. The partitioner is derived from Z3 (v4.12.1). As for base solvers, we apply our method to Z3++ [1], which participates in SMT-COMP 2023. Z3++ is also developed by our team and derived from Z3, and further information about Z3++ can be found at GitHub-Z3++. Regrettably, there is no Cloud Track this year; we hope it will return next year.

Building on our CAV 2024 work on variable-level partitioning [5], `Z3-Parti-Z3++` has evolved from a parallel solver into a fully *distributed* engine that scales smoothly from a handful of cores to hundreds of nodes. The new implementation adopts a two-tier *leader–coordinator–worker* hierarchy that cleanly separates distributed scheduling from parallel solving.

Detailed information on the original parallel framework appears in our paper "Distributed SMT Solving Based on Dynamic Variable-level Partitioning" [5]. We have provided our solver, evaluation scripts, and related experimental results in GitHub-AriParti.

In our future work, a comprehensive description of the additional mechanisms introduced in the distributed version will be presented.

2 Features

Dynamic parallel framework. We propose a dynamic parallel framework based on arithmetic variable-level partitioning. This framework ensures full utilization of computing resources, preventing idle core resources from lacking executable tasks. The dynamic parallel framework provides flexibility for parallel trees to grow. Thus, it can easily collaborate with other partitioning strategies

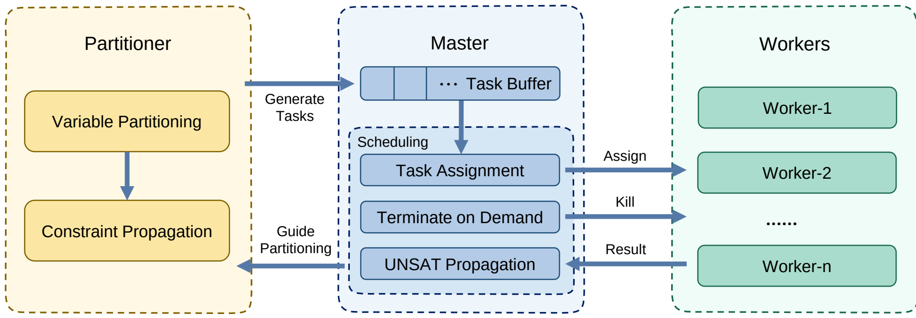

Fig. 1. Our dynamic parallel framework.

— any sub-problem yielded previously by pre-partitioning strategies can be partitioned further.

Variable-level partitioning. This is the first attempt to perform variable-level partitioning for arithmetic theories. Each time it picks a variable and partitions the problem by dividing the feasible domain of the variable, leading to sub-problems, which can be further simplified via constraint propagation. Our proposed variable-level partitioning permits robust, comprehensive partitioning. Regardless of the Boolean structure of any given instance, our partitioning algorithm can keep partitioning to the last moment of the solving process.

Algorithm 1: Arithmetic Variable-level Partitioning

$\phi \leftarrow$ choose a leaf node from the *partition tree*
$x \leftarrow$ choose a partitioning variable for the node ϕ
$\{\phi_l, \phi_r\} \leftarrow$ perform interval partitioning on variable x within ϕ
$\{\mathcal{R}_l, \hat{\phi}_l\} \leftarrow$ perform the *BICP* on ϕ_l
if $\mathcal{R}_l \neq$ UNSAT **then**
    Add node $\hat{\phi}_l$ into the *partition tree*
    Send $\hat{\phi}_l$ to the *task buffer*
$\{\mathcal{R}_r, \hat{\phi}_r\} \leftarrow$ perform the *BICP* on ϕ_r
if $\mathcal{R}_r \neq$ UNSAT **then**
    Add node $\hat{\phi}_r$ into the *partition tree*
    Send $\hat{\phi}_r$ to the *task buffer*

Improved constraint propagation. The effectiveness of our partition strategy is closely related to the underlying constraint propagation techniques to simplify the sub-problems. We propose an improved version of Interval Constraint Propagation (ICP) [2,4], named Boolean and Interval Constraint Propagation (BICP), and integrate it within our variable-level partitioning strategy. The

BICP conducts arithmetic feasible interval reasoning and successfully integrates Boolean propagation, allowing stronger propagation.

References

1. Cai, S., Li, B., Zhan, B., Zhang, X., Zhao, M.: Z3++ at SMT-COMP 2023. 18th International Satisfiability Modulo Theories Competition (SMT-COMP 2023) (2023)
2. Kulisch, U.W.: Complete interval arithmetic and its implementation on the computer. In: Cuyt, A., Krämer, W., Luther, W., Markstein, P. (eds.) Numerical Validation in Current Hardware Architectures. pp. 7–26. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
3. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
4. Schupp, S., Ábrahám, E., Rossmanith, P., Loup, D.I.U.: Interval constraint propagation in SMT compliant decision procedures. Master's thesis, RWTH Aachen (2013)
5. Zhao, M., Cai, S., Qian, Y.: Distributed smt solving based on dynamic variable-level partitioning. In: CAV 2024, Montreal, Canada, July 24-27, 2024, Proceedings. Lecture Notes in Computer Science, Springer (2024), (to appear)