CVC5-CLOUD at the SMT Competition 2024

Amalee Wilson[1], Andres Nötzli[1], Andrew Reynolds[2], Cesare Tinelli[2], and Clark Barrett[1]

[1]Stanford University
[2]The University of Iowa

***Abstract*—This paper is a description of the CVC5-CLOUD SMT solver as entered into the cloud tracks at the SMT Competition 2023. CVC5-CLOUD wraps the SMT solver CVC5 and adds the infrastructure to run it in distributed settings. It instruments CVC5 to split problems into independent sub-problems using multiple theory-agnostic approaches. Each set of sub-problems is run like a portfolio solver such that we are applying a portfolio of divide-and-conquer techniques. In addition to using partitioning, some cores run a traditional portfolio solver that uses scrambling. In every case, CVC5 is used to solve the sub-problems, and MPI to distribute the sub-problems across different cores and machines.**

Overview

CVC5-CLOUD consists of four main components: (a) multiple divide-and-conquer algorithms, (b) a *base solver* that attempts to solve queries, (c) a *splitter* that divides queries into hopefully easier ones using the divide-and-conquer algorithms, and (d) an infrastructure which schedules and executes these tasks. CVC5-CLOUD wraps CVC5 [3] and uses it for both the splitter and the base solver. More information about CVC5 can be found on its website [2]. For the traditional portfolio solving portion of the cores, the SMT-COMP scrambler [6] is used to create different scrambles with unique random seeds for each problem. Each of these problems is then solved using the base solver.

Divide-and-Conquer SMT Solving

Divide-and-conquer SMT solving recursively solves SMT formulas by splitting an original formula F into multiple independent sub-problems. The algorithm invokes a *splitter* which is given F and must split F into d *partitions* $C_1, \ldots C_d$ that partition the search space. More precisely, F and $(F \wedge C_1) \vee \cdots \vee (F \wedge C_d)$ must be equisatisfiable and, as a consequence, if there exists a satisfiable sub-problem $F \wedge C_i$, then F is satisfiable. Conversely, iff F is unsatisfiable, all sub-problems $F \wedge C_i$ are unsatisfiable. After the splitter completes, the system schedules the solving of sub-problems $F \wedge C_1$ through $F \wedge C_d$ to the pool of workers. Our approach uses multiple divide-and-conquer partitioning strategies as a partitioning portfolio.

Solving Queries

To solve the sub-problems, CVC5-CLOUD invokes CVC5 as a base solver. Currently, CVC5-CLOUD uses CVC5 with a single set of options that depend on the logic of the problem. CVC5-CLOUD currently does not retrieve any additional information from the base solver other than the satisfiability of a given sub-problem. Currently no information is shared among the sub-problems or in the traditional portfolio, but we plan to explore this in the future.

Scheduling and Executing Tasks

The submission of CVC5-CLOUD for SMT-COMP 2023 is based on the example code provided.[1] CVC5-CLOUD uses the Message Passing Interface (MPI) [5] through MPI for Python [4] to partition the problems and coordinate their solutions. Several dedicated leader cores are used to manage each partitioning strategy in the partitioning portfolio and the scrambling portfolio. The leader cores are responsible for invoking the splitter, scheduling sub-problems, scheduling the traditional portfolio, and processing answers. The leader immediately terminates and reports the answer if any of these situations occur: any sub-problem is satisfiable; all subproblems for a given partitioning strategy are unsatisfiable; any of the traditional portfolio problem instances are satisfiable or unsatisfiable. If all of the traditional portfolio problem instances are unknown, then the leader returns unknown.

Under some circumstances, the splitter can contribute to solving a problem, because our approach uses the regular solving infrastructure of CVC5. If the splitter finds that a problem is satisfiable, the leader terminates solving immediately and reports the answer. If the splitter does not produce any partitions and reports that the problem is unsatisfiable, the leader skips that problem and moves on the splitting the next sub-problem. This can happen if CVC5 only makes few decisions before concluding that the problem is unsatisfiable. If no partitions are created and no solution is found, then the given partitioning strategy is skipped.

Conclusion

We believe the approach of combining a traditional portfolio solver with a partitioning portfolio solver to be promising. The divide-and-conquer architecture permits both theory-agnostic splitters (which can be easily applied to new theories) and theory-specific splitters (which can be optimized to the dynamics of a specific theory, or even specific family of benchmarks). We hope to explore more partitioning strategies as well as information sharing in the future. Our work on CVC5-CLOUD is ongoing.

Acknowledgments

Like CVC5, CVC5-CLOUD is supported in part by the organizations listed on CVC5's website [1].

[1]https://github.com/aws-samples/aws-batch-comp-infrastructure-sample and https://github.com/aws-samples/aws-satcomp-solver-sample

REFERENCES

[1] cvc5 acknowledgments. https://cvc5.github.io/acknowledgements.html, 2022.

[2] CVC5 website. https://cvc5.github.io/, 2022.

[3] Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength SMT solver. In *TACAS (1)*, volume 13243 of *Lecture Notes in Computer Science*, pages 415–442. Springer, 2022.

[4] Lisandro Dalcín and Yao-Lung L. Fang. mpi4py: Status update after 12 years of development. *Comput. Sci. Eng.*, 23(4):47–54, 2021.

[5] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard Version 4.0*, June 2021. URL https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf.

[6] Tjark Weber, Aina Niemetz, Jochen Hoenicke, Antti Hyvärinen, Haniel Barbosa, and Matthias Schlaipfer. SMT-COMP benchmark scrambler. https://github.com/SMT-COMP/scrambler, 2023.