EmergenTheta: Variations on Symbolic Transition Systems (Competition Contribution)

Milán Mondok, Levente Bajczi(✉), Dániel Szekeres, and Vince Molnár

Department of Artificial Intelligence and Systems Engineering
Budapest University of Technology and Economics, Budapest, Hungary
{mondok,bajczi,szekeres,molnarv}@mit.bme.hu

Abstract. EmergenTheta is our sandbox for experimental analyses. After its successful debut in SV-COMP'24, we kept some well-performing but still under-tested configurations, and complemented them with a new saturation algorithm over decision diagrams, and two ways of extending their verification power: wrapping them in a lightweight, counterexample-guided abstraction refinement (CEGAR) loop based on implicit predicate abstraction; and backwards traversal of the state space. All such analyses now rely on a common interface to the underlying symbolic transition system, integrating seamlessly into the existing Theta framework. Using this combination of proven analyses and novel extensions, EmergenTheta outperformed our expectations in SV-COMP'25.

Funding. This research was partially funded by the EKÖP-24-3 New National Excellence Program under project numbers EKÖP-24-3-BME-78, EKÖP-24-3-BME-213, and EKÖP-24-3-BME-159, and the Doctoral Excellence Fellowship Programme under project numbers 400434/2023 and 400443/2023; funded by the NRDI Fund of Hungary.

1 Verification Approach

EmergenTheta parses the C programs and transforms them into the extended control flow automaton (XCFA) formalism through a series of preprocessing steps. XCFA are then transformed into the low-level symbolic transition system (STS) formalism, which captures the model behavior using two SMT formulas, one characterizing the initial states and the other characterizing the transition relation.

Bounded model checking (BMC) [8] can prove the faultiness of a model by constructing path constraints that characterize all possible counterexamples of a given length k and checking the satisfiability of such constraints using an SMT solver. The bound k is incremented until either the model is proven unsafe, or

L. Bajczi—Jury member representing EmergenTheta at SV-COMP 2025.

A. Gurfinkel and M. Heule (Eds.): TACAS 2025, LNCS 15698, pp. 217–222, 2025.
https://doi.org/10.1007/978-3-031-90660-2_15

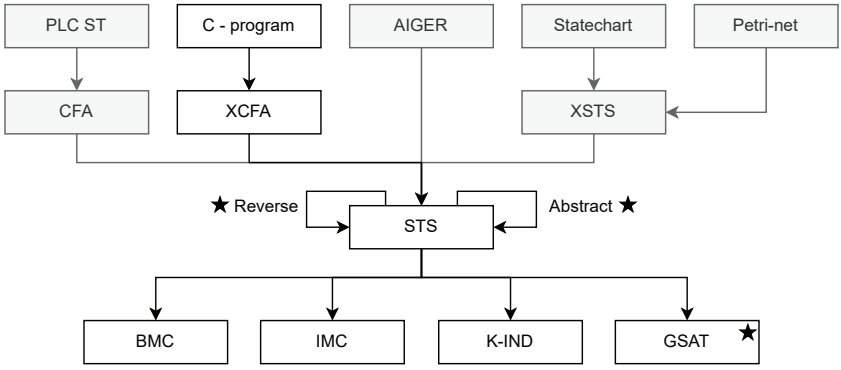

Fig. 1: Overview of the verification approach of EmergenTheta. 2025 additions are denoted with the ★ symbol. Parts irrelevant to SV-COMP are grayed out.

until the available resources allow. *K-induction* (K-IND) [17] and *interpolation-based model checking* (IMC) [13] extend BMC with additional checks in an attempt to prove that the model is safe. K-induction does so by trying to prove the k-inductivity of the property with k being the current BMC length, while IMC derives Craig interpolants to compute an overapproximation of the set of reachable states.

Decision-diagram-based algorithms are a novel addition to the set of available algorithms this year. *Substitution diagrams* [15] allow us to handle SMT formulas as multi-valued decision diagrams (MDDs) in a top-down approach supported by efficient caching and lazy evaluation for the presence of edges. Using substitution diagrams, we can represent the states and transitions of the system with MDDs constructed from the SMT formulas of the STS representation. The *saturation* algorithm [9] can compute the set of reachable states on decision diagrams in a bottom-up manner. It decomposes the exploration into smaller explorations on submodels exploiting the locality of the system's events. The *generalized saturation* (GSAT) [14] algorithm employs a weaker notion of locality to further enhance the saturation effect.

The modular architecture of EmergenTheta also allows us to wrap the aforementioned analyses into a CEGAR loop based on *implicit predicate abstraction* [18] and to traverse the state space in a *reversed* manner.

We run the aforementioned analyses in a sequential portfolio. We start with the GSAT algorithm (because it can find a safety proof fairly quickly, or report the task not solvable with the configuration), then move on to BMC and K-Induction. Afterwards, we intended to start the *abstracted* and *reversed* IMC, but ended up using K-Induction due to a bug (see Sect. 3).

Configuration	unreach-call safe	unreach-call unsafe	termination safe	termination unsafe	no-overflow safe	no-overflow unsafe
Preprocessing	1	0	51	0	232	0
GSAT	152	33	0	0	0	0
BMC	209	134	11	38	34	143
K-Induction	380	34	4	1	6	4
Abstract K-Induction	37	0	112	0	63	1
Reversed K-Induction	2	0	0	0	0	0
Sum	781	201	178	39	335	148
Wrong	2	0	20	0	15	0

Fig. 2: Successful verification results per configuration

2 Software Architecture

EmergenTheta is a JVM-based tool, written primarily in Kotlin (and legacy parts of the code are written in Java).

EmergenTheta is part of the Theta framework [12], and can therefore use its integrated frontend- and solver-infrastructure. Parsing C files is done using a custom ANTLR grammar [2]. Satisfiability Modulo Theories (SMT) solvers can be invoked three different ways:

- Z3 [16] is natively integrated, its Java API is a dependency of Theta
- Other solvers are integrated via SMT-LIBv2 [11], such as MathSAT5 [10] or CVC5 [5]
- JavaSMT [1] has been integrated since last year's SV-COMP

In SV-COMP'25, we use Z3, MathSAT5, and CVC5.

3 Discussion of Strengths and Weaknesses of the Approach

EmergenTheta took part in SV-COMP'25 with 3 checkable properties: the conventional unreach-call (as previously [3]); and we debuted with termination and no-overlow this year using a pre-processing step using TransVer [7]. This broadened verification reach is a definitive strength of our tool, but cannot be attributed to its architecture, as the pre-processing step is transparent to EmergenTheta. In the future we plan to integrate it into our portfolio more, as experiences with reachability need not translate to other properties when it comes to verification efficacy (based on results in Figure 2 [6]).

It is also worth mentioning that a large number of tasks (232 for no-overflow, 51 for termination, but only 1 for unreach-call) were solved by sanity checks *before* a verifier algorithm was started. In these instances safety was evident, as in the control flow graph, we do not have any paths from the initial location to the error locations.

Unfortunately, the portfolio implementation contained an oversight, making IMC-based analyses become K-Induction-based analyses by mistake. This is reflected in our results: we wanted to run IMC with abstraction and with reversal, but ended up running K-Induction in these instances, and these represent our least efficient configurations (see Figure 2). We will move to a more robust portfolio-engine in the future so that oversights like this can be more easily avoided. However, both abstraction and reversal solved tasks that the non-reversed and not-abstracted configurations did not, therefore, we are confident in their advantages overall.

The new algorithm, GSAT, produced really promising results this year. It proved safety in 152 tasks, and found a bug in 33 instances. However, it could only output correctness witnesses this year, and no violation witnesses, so the 33 successful verification tasks are correct, but unconfirmable.

EmergenTheta produced a few wrong results. In the unreach-call category it produced 2, in termination 20, and in no-overflow 15 false positive results. It produced no false negative (false safe) results. The cause of the wrong results are partly due to a mishandling of floating point NaN values, and due to the preprocessing step mapping the new properties to reachability.

4 Tool Setup and Configuration

EmergenTheta is widely configurable, and choosing a successful configuration for a verification task at hand can be complicated. If using the competition archive [4] for software verification, we recommend using the following input options, starting our EMERGENT portfolio:

```
theta-start.sh <input> --portfolio EMERGENT --svcomp
```

To minimize the output verbosity, the option `--loglevel RESULT` can be added to the arguments. We also used these options at SV-COMP 2025.

5 Software Project and Data-Availability Statement

EmergenTheta is integrated into the Theta verification framework maintained by the Critical Systems Research Group[1] of the Budapest University of Technology and Economics. The project is available open-source on GitHub under an Apache 2.0 license[2], and the binary release of the competition version (`6.8.6`) is published on Zenodo [4].

References

1. Baier, D., Beyer, D., Friedberger, K.: JavaSMT 3: Interacting with SMT Solvers in Java. In: Silva, A., Leino, K.R.M. (eds.) Computer Aided Verification. pp. 195–208. Springer International Publishing, Cham (2021)

[1] https://ftsrg.mit.bme.hu/en/
[2] https://github.com/ftsrg/theta/releases/tag/svcomp25

2. Bajczi, L., Ádám, Z., Molnár, V.: C for Yourself: Comparison of Front-End Techniques for Formal Verification. In: 10th FormaliSEICSE 2022. pp. 1–11. ACM (2022). https://doi.org/10.1145/3524482.3527646
3. Bajczi, L., Szekeres, D., Mondok, M., Ádám, Z., Somorjai, M., Telbisz, C., Dobos-Kovács, M., Molnár, V.: EmergenTheta: Verification Beyond Abstraction Refinement (Competition Contribution). In: Finkbeiner, B., Kovács, L. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 371–375. Springer Nature Switzerland, Cham (2024)
4. Bajczi, L., Telbisz, C., Somorjai, M., Ádám, Z., Dobos-Kovács, M., Szekeres, D., Mondok, M., Molnár, V.: EmergenTheta - SV-COMP'25 Verifier Archive (Nov 2024). https://doi.org/10.5281/zenodo.14194484
5. Barbosa, H., Barrett, C., Brain, M., Kremer, G., Lachnitt, H., Mann, M., Mohamed, A., Mohamed, M., Niemetz, A., Nötzli, A., Ozdemir, A., Preiner, M., Reynolds, A., Sheng, Y., Tinelli, C., Zohar, Y.: cvc5: A Versatile and Industrial-Strength SMT Solver. In: Fisman, D., Rosu, G. (eds.) TACAS 2022. pp. 415–442. Springer International Publishing, Cham (2022)
6. Beyer, D., Strejček, J.: Improvements in software verification and witness validation: SV-COMP 2025. In: Proc. TACAS. LNCS, Springer (2025)
7. Beyer, D., Jankola, M., Lingsch-Rosenfeld, M., Xia, T., Zheng, X.: A modular program-transformation framework for reducing specifications to reachability (2025), https://arxiv.org/abs/2501.16310
8. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic Model Checking without BDDs. In: TACAS (1999). https://doi.org/10.1007/3-540-49059-0_14
9. Ciardo, G., Lüttgen, G., Siminiceanu, R.: Saturation: An efficient iteration strategy for symbolic state—space generation. In: Margaria, T., Yi, W. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 328–342. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
10. Cimatti, A., Griggio, A., Schaafsma, B., Sebastiani, R.: The MathSAT5 SMT Solver. In: TACAS 2013, LNCS, vol. 7795, pp. 93–107. Springer (2013). https://doi.org/10.1007/978-3-642-36742-7_7
11. Dobos-Kovács, M., Vörös, A.: Evaluation of SMT solvers in abstraction-based software model checking. In: Proceedings of the 11th Latin-American Symposium on Dependable Computing. p. 109–116. LADC '22, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3569902.3570187
12. Hajdu, Á., Micskei, Z.: Efficient Strategies for CEGAR-based Model Checking. Journal of Automated Reasoning **64**(6), 1051–1091 (2020). https://doi.org/10.1007/s10817-019-09535-x
13. McMillan, K.L.: Interpolation and SAT-Based Model Checking. In: Hunt, W.A., Somenzi, F. (eds.) Computer Aided Verification (2003). https://doi.org/10.1007/978-3-540-45069-6_1
14. Molnár, V., Majzik, I.: Saturation Enhanced with Conditional Locality: Application to Petri Nets. In: Donatelli, S., Haar, S. (eds.) Application and Theory of Petri Nets and Concurrency. pp. 342–361. Springer International Publishing, Cham (2019)
15. Mondok, M., Molnár, V.: Efficient Manipulation of Logical Formulas as Decision Diagrams. In: Renczes, B. (ed.) Proceedings of the 31st PhD Mini-Symposium. pp. 61–65. Budapest University of Technology and Economics, Department of Measurement and Information Systems (2024). https://doi.org/10.3311/MINISY2024-012
16. de Moura, L., Bjørner, N.: Z3: An Efficient SMT Solver. In: TACAS 2008, LNCS, vol. 4963, pp. 337–340. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_24

17. Sheeran, M., Singh, S., Stålmarck, G.: Checking Safety Properties Using Induction and a SAT-Solver. In: Formal Methods in Computer-Aided Design (2000). https://doi.org/10.1007/3-540-40922-X_8
18. Tonetta, S.: Abstract Model Checking without Computing the Abstraction. In: Cavalcanti, A., Dams, D.R. (eds.) FM 2009: Formal Methods. pp. 89–105. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)