KoAT: An Automatic Complexity Analysis Tool for (Probabilistic) Integer Programs

Nils Lommen
RWTH Aachen University, Germany

Éléanore Meyer
RWTH Aachen University, Germany

Jürgen Giesl
RWTH Aachen University, Germany

Abstract

KoAT is an automated tool for inferring runtime and size bounds – and proving termination – of (probabilistic) integer programs by analyzing program fragments in a modular way and combining their bounds. It uses multiphase linear ranking functions to capture execution "phases" of loops and a technique for triangular weakly non-linear loops (*twn*-loops), enabling the analysis of programs with non-linear arithmetic.

2012 ACM Subject Classification Software and its engineering → Software verification and validation; Theory of computation → Automated reasoning; Theory of computation → Logic and verification

Keywords and phrases Complexity Analysis, Termination Analysis

KoAT is a tool to automatically infer complexity bounds for (probabilistic) integer programs, based on an alternating modular inference of upper runtime and size bounds for program parts [2]. By inferring runtime bounds for individual subprograms repeatedly, in the end we obtain a bound on the runtime complexity of the whole program. Furthermore, KoAT can be used to automatically prove termination of programs.

To infer runtime bounds for subprograms, we use multiphase linear ranking functions (MΦRFs) [1, 3]. In contrast to classical ranking functions, MΦRFs can also represent bounds on multiple "phases" of program executions. Moreover, we embedded a complete technique for inferring runtime bounds of triangular weakly non-linear loops (*twn*-loops) in our complexity analysis tool [4, 7]. The update of a *twn*-loop is triangular, i.e., we can order the program variables such that the update of any x_i does not depend on the variables $x_1, \ldots, x_{i-1}$ with smaller indices. So the restriction to triangular updates prohibits "cyclic dependencies" of variables. Due to this, one can compute closed forms for the repeated updates of *twn*-loops, which makes them especially suitable for automatic program analysis. In particular, this approach also allows us to analyze the complexity of programs with non-linear arithmetic. In addition, we developed a novel procedure to infer size bounds via closed forms as well, which is also suitable for programs with non-linear arithmetic [5, 7]. Furthermore, recently we extended our framework such that (possibly recursive) function calls can be represented naturally and analyzed by our tool KoAT. KoAT is also used in the framework AProVE (KoAT + LoAT) [8] to automatically prove termination of C programs.

We adapted KoAT's framework for automated complexity analysis to *probabilistic* integer programs in [9]. To improve the power of automatic complexity analysis further, we integrated control-flow refinement via partial evaluation into KoAT's approach [3] and recently adapted this refinement for probabilistic programs as well [6].

KoAT's source code, a web interface, a binary, and a Docker image are available at

https://koat.verify.rwth-aachen.de.

References

1 Amir M. Ben-Amram and Samir Genaim. On Multiphase-Linear Ranking Functions. In *Proc. CAV '17*, LNCS 10427, pages 601–620, 2017. doi:10.1007/978-3-319-63390-9_32.

2 Marc Brockschmidt, Fabian Emmes, Stephan Falke, Carsten Fuhs, and Jürgen Giesl. Analyzing Runtime and Size Complexity of Integer Programs. *ACM Trans. Program. Lang. Syst.*, 38:1–50, 2016. doi:10.1145/2866575.

3 Jürgen Giesl, Nils Lommen, Marcel Hark, and Fabian Meyer. Improving Automatic Complexity Analysis of Integer Programs. In *The Logic of Software. A Tasting Menu of Formal Methods*, LNCS 13360, pages 193–228, 2022. doi:10.1007/978-3-031-08166-8_10.

4 Nils Lommen, Fabian Meyer, and Jürgen Giesl. Automatic Complexity Analysis of Integer Programs via Triangular Weakly Non-Linear Loops. In *Proc. IJCAR '22*, LNCS 13385, pages 734–754, 2022. doi:10.1007/978-3-031-10769-6_43.

5 Nils Lommen and Jürgen Giesl. Targeting Completeness: Using Closed Forms for Size Bounds of Integer Programs. In *Proc. FroCoS '23*, LNCS 14279, pages 3–22, 2023. doi:10.1007/978-3-031-43369-6_1.

6 Nils Lommen, Éléanore Meyer, and Jürgen Giesl. Control-Flow Refinement for Complexity Analysis of Probabilistic Programs in KoAT (Short Paper). In *Proc. IJCAR '24*, LNCS 14739, pages 233–243, 2024. doi:10.1007/978-3-031-63498-7_14.

7 Nils Lommen, Éléanore Meyer, and Jürgen Giesl. Targeting Completeness: Automated Complexity Analysis of Integer Programs. *Corr*, abs/2412.01832, 2024. doi:10.48550/arXiv.2412.01832.

8 Nils Lommen and Jürgen Giesl. AProVE (KoAT + LoAT). In *Proc. TACAS '25*, LNCS 15698, pages 205–211, 2025. doi:10.1007/978-3-031-90660-2_13.

9 Fabian Meyer, Marcel Hark, and Jürgen Giesl. Inferring Expected Runtimes of Probabilistic Integer Programs Using Expected Sizes. In *Proc. TACAS '21*, LNCS 12651, pages 250–269, 2021. doi:10.1007/978-3-030-72016-2_14.