Amaya v2.0

System description for SMT-COMP 2024

VOJTĚCH HAVLENA, Brno University of Technology, Czech Republic
MICHAL HEČKO, Brno University of Technology, Czech Republic
LUKÁŠ HOLÍK, Brno University of Technology, Czech Republic
ONDŘEJ LENGÁL, Brno University of Technology, Czech Republic

This is a brief overview of Amaya's submission to SMT-COMP 2024. Amaya is a standalone solver for quantified linear integer arithmetic (LIA) based on finite automata. It implements the standard automata-based decision procedure with several crucial optimizations that prune vast fractions of the underlying state space. Contrary to standard approaches, Amaya constructs a compact representation of all models of the LIA formula.

1 OVERVIEW

Amaya[1] is a standalone solver for quantified *linear integer arithmetic* (LIA)[2] based on finite automata (FAs). The foundation of the decision procedure implemented in Amaya is the work of Büchi [3] (in the context of showing automata-based decidability of the *weak monadic second-order theory of one successor*—WS1S—into which LIA can be encoded), which was later tailored for LIA (including the linear congruence operator) in the works [1, 2, 4]. The basic procedure works by constructing an FA $\mathcal{A}_{\varphi_{atom}}$ for each atomic formula φ_{atom}, which is of the form $a_1x_1 + \ldots a_nx_n \sim c$ where $\sim \in \{=, \leq, \equiv_m\}$ (with $\equiv_m$ being the linear congruence modulo $m \in \mathbb{N} \setminus \{0\}$) for integers $a_1, \ldots, a_n, c$ and a natural number m. The FA $\mathcal{A}_{\varphi_{atom}}$ represents all solutions of the formula φ_{atom}, encoded as tuples of binary numbers in the two's complement encoding. FAs for complex formulae are then obtained inductively by performing standard automata operations: *intersection* for conjunction, *union* for disjunction, *complementation* for negation. Existential quantification is performed by the operation of *projection* on the alphabet of the corresponding FA (followed by saturation of the FAs accepting states).

Amaya implements the procedure with several crucial optimizations described in [5]. In particular, it fights the issue of constructing FAs with too many states (which emerge, e.g., when translating modulo congruence predicates) by directly creating FAs for non-atomic formulae using a derivative based construction with implicit pruning and subsumption. Moreover, the procedure also uses certain techniques of quantifier instantiation (based on formula monotonicity, finite range of models, and modulo linearization) as well as standard simplification techniques for on-the-fly rewriting of the formula for which the FA is being constructed.

[OL: preprocessing?]

2 IMPLEMENTATION

The high-level part of the decision procedure of Amaya is implemented in Python. Since the size of the underlying FA alphabet grows exponentially with the number of variables in the formula, needing to deal with alphabets having thousands up to millions of symbols, causing an explicit representation of the transition relation infeasible. To deal with this issue, Amaya encodes the transition relation using *multi-terminal binary decision diagrams* (MTBDDs) implemented within the Sylvan library [6], which often yields a much more compact representation. As Sylvan is a C library, Amaya includes a C++ backend that connects to the Python frontend via a `ctypes` interface. [OL: what else is in C++?]

[1] https://github.com/MichalHe/amaya
[2] also sometimes referred to as *Presburger arithmetic*

[OL:]

3 AMAYA IN SMT-COMP 2024

We are submitting version v2.0 of Amaya to compete in the single query track of the division Arith, in logics LIA and NIA (nonlinear integer arithmetic; we participate in NIA because it contains many formulae that are linear in nature, but include the $\equiv_m$ predicate, which is not considered linear by SMT-LIB).

REFERENCES

[1] Bernard Boigelot, Sébastien Jodogne, and Pierre Wolper. 2005. An effective decision procedure for linear arithmetic over the integers and reals. *ACM Trans. Comput. Log.* 6, 3 (2005), 614–633. https://doi.org/10.1145/1071596.1071601

[2] Alexandre Boudet and Hubert Comon. 1996. Diophantine Equations, Presburger Arithmetic and Finite Automata. In *Trees in Algebra and Programming - CAAP'96, 21st International Colloquium, Linköping, Sweden, April, 22-24, 1996, Proceedings (Lecture Notes in Computer Science, Vol. 1059)*, Hélène Kirchner (Ed.). Springer, 30–43. https://doi.org/10.1007/3-540-61064-2_27

[3] Julius Richard Büchi. 1960. Weak Second-Order Arithmetic and Finite Automata. *Zeitscrift fur mathematische Logic und Grundlagen der Mathematik* 6 (1960), 66–92.

[4] Antoine Durand-Gasselin and Peter Habermehl. 2010. On the Use of Non-deterministic Automata for Presburger Arithmetic. In *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings (Lecture Notes in Computer Science, Vol. 6269)*, Paul Gastin and François Laroussinie (Eds.). Springer, 373–387. https://doi.org/10.1007/978-3-642-15375-4_26

[5] Peter Habermehl, Vojtěch Havlena, Michal Hečko, Lukáš Holík, and Ondřej Lengál. 2024. Algebraic Reasoning Meets Automata in Solving Linear Integer Arithmetic. In *To appear in CAV'24 (Lecture Notes in Computer Science)*. Springer. An extended version is available at https://arxiv.org/abs/2403.18995..

[6] Tom van Dijk and Jaco van de Pol. 2015. Sylvan: Multi-Core Decision Diagrams. In *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings (Lecture Notes in Computer Science, Vol. 9035)*, Christel Baier and Cesare Tinelli (Eds.). Springer, 677–691. https://doi.org/10.1007/978-3-662-46681-0_60