Algaroba at the SMT Competition 2024

Amar Shah
University of California, Berkeley
amarshah@cmu.edu

Federico Mora
University of California, Berkeley
fmora@cs.berkeley.edu

Sanjit A. Seshia
University of California, Berkeley
sseshia@berkeley.edu

1 Introduction

This paper is a description of the SMT solver Algaroba as entered into the 2024 SMT competition. Algaroba is an SMT solver for the quantifier-free theory of Algebraic Datatypes. We compete in the Single-Query track in the **QF_DT** and **QF_UFDT** logics. For a more detailed description of Algaroba, see [3] and [4].

2 Description

Algaroba is an eager solver that reduces the theory of Algebraic Datatypes to the theory of Equality & Uninterpreted Functions by blasting the axioms of Algebraic Datatypes. This is not simple because the theory of Algebraic Datatypes requires all datatypes to be well-founded, i.e. no ADT term can reference itself. On its own, this cannot be blasted.

Algaroba works using a small model property. We only need to blast the well-foundedness axioms up to some depth k, where k is the number of variables in the flattened query. We prove that this gives a sound and complete decision procedure [4].

3 Solver Architecture

Algaroba is written in approximately 2900 lines of OCaml. We use the Smtlib-utils [1] to parse and print SMT-Lib queries. Algaroba can support any solver for **UF**. For the SMTCOMP submission, we use the Z3 [2] OCaml API to solve the reduced **UF** query. It includes a number of simple optimizations, like hash-consing, incremental solving, and theory-specific query simplifications.

Algorithm 1 Algaroba(ψ)

$\psi_1 \leftarrow \mathit{NNF}(\psi)$
$\psi_2 \leftarrow \mathit{Flatten}(\psi_1)$
$k \leftarrow$ Number of variables in ψ_2
$\tilde{\psi} \leftarrow$ Transform ψ by blasting the axioms of **DT**
$\phi_1, ..., \phi_m \leftarrow$ Well-foundedness axioms up to a depth k
return $\mathit{UF\text{-}SMT\text{-}Solver}(\tilde{\psi} \wedge \phi_1 \wedge ... \wedge \phi_m)$

Algorithm 1 shows Algaroba's architecture. We initially apply simplifications to our query ψ to flatten it and put it in NNF. Then we transform ψ to $\tilde{\psi}$ to blast all axioms of **DT** (except well-foundedness). Finally, we introduce our well-foundedness axioms up to a depth k. We describe two modifications to this algorithm below.

3.1 Pause

We do not have to add all of $\phi_1 ... \phi_m$ to solve. We can "pause" in the middle. We do this by instantiating our axioms up to a depth $\frac{k}{2}$ and checking for **unsat**. Then, we restrict our model's size and check for **sat**. Only if we do not get **unsat** or **sat** at either of these steps, do we continue up to depth k.

3.2 Finite Universe Instantiation

If an ADT has a finite universe, i.e. it is only composed of records and enums and not inductive types, we treat this as a special case. We create constants for every term in the universe and instantiate the axioms of **DT** over the entire, finite universe.

References

[1] Simon Cruanes. *Smtlib-utils*. Version 0.5. URL: https://github.com/c-cube/smtlib-utils.

[2] Leonardo de Moura and Nikolaj Bjørner. "Z3: An Efficient SMT Solver". In: *Tools and Algorithms for the Construction and Analysis of Systems*. 2008. ISBN: 978-3-540-78800-3.

[3] Amar Shah, Federico Mora, and Sanjit A. Seshia. *Algaroba*. Version 1.0. URL: https://github.com/uclid-org/algaroba.

[4] Amar Shah, Federico Mora, and Sanjit A. Seshia. "An Eager Satisfiability Modulo Theories Solver for Algebraic Datatypes". In: *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI*. 2024. URL: https://doi.org/10.1609/aaai.v38i8.28649.