

Tree-based Modular SMT Solving

Georg Schädler <georg.schadler@student.tugraz.at> and
Georg Hofferek <georg.hofferek@iaik.tugraz.at>

Graz University of Technology, Austria

Abstract

Modern Satisfiability Modulo Theories (SMT) solvers are highly efficient and can generate a resolution proof in case of unsatisfiability. Some applications, such as synthesis of Boolean controllers¹, compute multiple coordinated interpolants from a single refutation proof. In order to do so, the proof is required to have two properties: It must be colorable and local-first. The latter means that resolution over a literal that occurs just in one partition, has to have both premises derived from this partition. Off-the-shelf SMT solvers do not necessarily produce proofs that have these properties. In particular, proofs are usually not local-first. Hofferek et al.¹ suggest to perform proof transformations to obtain a colorable and local-first proof, but these transformations can be computationally expensive. Our goal is to introduce a new method to directly compute a local-first, colorable resolution proof for an unsatisfiable SMT formula. This proof can then be directly used for n -interpolation.¹

Our approach uses a tree-based structure of SMT solvers, where every node in the tree is associated with a formula (possibly empty initially), and a (possibly empty) set of literals. The semantics of the modular SMT problem is the conjunction of the formulas of all nodes. The set of literals associated with a node is computed recursively as follows. Every literal, which appears in more than one descendant of a parent node, is assigned to the parent node. During solving, every node makes decisions about its associated literals only. We start at the root node and communicate the partial assignments to the child nodes until we reach a leaf. Every node has its own solver instance and tries to compute a satisfying assignment w.r.t. the given partial assignments. A blocking clause is added to the parent node if no satisfying assignment can be found. If all child nodes find a satisfying assignment and the conjunction of them is theory-consistent, we either decide more literals, or return to the parent node if we already have a full assignment for all the literals of the current node. In case the conjunction is inconsistent within the theory, we respectively add a blocking clause for the current assignment to the children. In order to obtain these clauses while keeping the modular structure intact, we perform interpolation over the assignments of the children and use the interpolants, which are guaranteed to contain only “global” literals (which can safely be added to the children without breaking the modularity), to learn a blocking clause for every child node. The algorithm terminates if either solely full assignments are communicated to the root node and there is no literal left to decide, or when enough clauses have been learned at the root node to show inconsistency. In the latter case we are able to extract a resolution proof with the same modular structure as the original problem.

We are currently working on implementing this approach in the hope that we can use it to generate colorable, local-first proofs for synthesis problems much faster than with post-processing proof transformations.