Programming Z3

Nikolaj Bjørner
Microsoft Research

The talk provides an update on the Satisfiability Modulo Theories Solver Z3.

It covers algorithmic trends around model constructing search that have been explored in different parts of Z3 and points to new directions on adding strategies to prune explored cases further. Modern SAT solvers use model constructing search when they use CDCL that builds partial assignments to literals extended through decisions, propagations and corrected using clause learning. The model-based theory combination method takes a modular view of model extension as each theory solver is allowed to extend a partial interpretation to a complete one for a subformula. It relies on conflict resolution and backjumping to reconcile separate models. Model-based quantifier instantiation uses an interpretation of free functions under a quantifier when searching for instantiations. Similarly model-based quantifier elimination uses a current interpretation to choose instantiations of a quantified variable. Engines, such as the solver for non-linear real arithmetic and property directed reachability use candidate interpretations to direct conflict resolution. A different, but so far only selectively explored, way to prune choices is through *strategies* that that restrict the set of possible partial interpretations through Herbrand functions. Quantifier reasoning offers an important use for forward pruning using strategies.

We then give a programmer's introduction to Z3. With the Python scripting interface as starting point we describe ways to interact with Z3 and selected algorithms underlying the decision procedures. We illustrate using Z3 from Jupyter notebooks that run in an Azure service. The aim of offering programmatic interaction with Z3 is to ease development of custom applications. We use an encoding of bounded model checking and unweighted MaxSAT as illustrative examples.

Finally, we point to newer developments: powering search using clouds and neurons. We describe our preliminary experiences distributing search in Azure using Cube & Conquer techniques. Formulas from applications, such as scheduling and timetabling, are significantly larger and have wildly different structural properties. In spite of these differences, we found that the Cube & Conquer methodology can make a substantial difference as fixing even a limited number of variables can drastically reduce the overhead solving subformulas. We describe a distributed version of Z3 that scales with Azure's elastic cloud. It integrates recent advances in lookahead and distributed SAT solving for Z3's engines for SMT. A multi-threaded version of the Cube & Conquer solver is also available parallelizing SAT and SMT queries. Finally, we mention new developments to appear in SAT 2019 to speed up SAT search using DNNs for predicting which literals to branch on.

The topics covered in the talk were derived from joint work with many collaborators, including Marijn Heule, Rahul Kumar, Leonardo de Moura, Lev Nachmanson, Nina Narodytska, Miguel Angelo Da Terra Neves, Daniel Selsam, and Christoph Wintersteiger.