Optimizing MiniSAT Variable Orderings for the Relational Model Finder Kodkod (Poster Presentation)

Markus Iser, Mana Taghdiri, Carsten Sinz

Karlsruhe Institute of Technology (KIT), Germany
markus.iser@student.kit.edu,
{mana.taghdiri, carsten.sinz}@kit.edu

Introduction. It is well-known that the order in which variables are processed in a DPLL-style SAT algorithm can have a substantial effect on its run-time. Different heuristics, such as VSIDS [2], have been proposed in the past to obtain good variable orderings. However, most of these orderings are general-purpose and do not take into account the additional structural information that is available on a higher problem description level. Thus, structural, problem-dependent strategies have been proposed (see, e.g., the work of Marques-Silva and Lynce on special strategies for cardinality constraints [1]).

In this paper, we propose a new, structural variable ordering heuristic that is specific to the relational model finder Kodkod [3]. Kodkod transforms first-order, relational logic formulas into equisatisfiable propositional formulas and solves them using a SAT solver. Structural properties of a Kodkod problem that can efficiently be extracted in its first-order relational representation get lost in its propositional encoding. Our proposed heuristic computes the "constrainedness" of relations, and gives priority to the propositional variables that stem from the most constrained relations. The constrainedness is computed from Kodkod's abstract syntax tree, and thus takes the structure of the original relational formula into account.

Kodkod is a model finder for a widely-used first-order relational logic—a constraint language that combines first-order logic and relational algebra, and is augmented with transitive closure, integer arithmetic, and cardinality operators. It has been used as the backend engine of several software analysis tools in order to find bugs in the design and implementation of various software. A *problem* in Kodkod consists of a universe declaration, a number of relation declarations, and a formula over those relations. The universe defines a finite set of uninterpreted atoms that can occur in the models of the problem. A relation declaration specifies both an upper and a lower bound on the value of that relation. An upper bound of a relation denotes all the tuples that the relation *may* contain, whereas a lower bound denotes all the tuples that it *must* contain. Relation bounds are used to specify various information such as partitioning the universe into types, or defining a partial model for the problem.

Method. Based on the observation that shuffling variables in the SAT encoding produced by Kodkod can have a tremendous effect on the run-time of the SAT solver, we developed strategies to obtain better, Kodkod-specific orderings. We experimented with two ways to modify MiniSAT's standard variable ordering:

1. **Initializing VSIDS:** Instead of using MiniSAT's default initialization that assigns variables in the same order in which they occur in the CNF file, we used a Kodkod-specific initial order.
2. **Overriding VSIDS:** Here, we partition variables into subsets $(S_1, \ldots, S_k)$, and assign variables in S_i before any variable occurring in a subset S_j, $j > i$. Within each subset, we use VSIDS scores to order variables.

Kodkod-specific variable orderings are computed based on the *constraining effect* that a subformula exerts on a relation's possible values. For example, a subset formula $R \subseteq S$, written as "`R in S`" in Kodkod's language, forces the entries of the relation R to be *false*, as soon as the corresponding entries in S are *false*. Similarly, for a cardinality restriction, written as "`#R <= c`" in Kodkod, where the constant c is small, many entries in R have to be set to *false*. Thus, the constraining effect of a cardinality restriction is usually high. We therefore try to assign variables corresponding to such relations first. The intuition is that the number of unit propagations on the SAT level can be maximized thereby. In our approach, we iteratively compute the effect of *highly constraining operators* (like subset or cardinality restrictions) on the relations that occur in the constrained relational expressions. The effect is summarized as a *weight* that we assign to each relation.

Conclusion. We have implemented several variants of the heuristic outlined above in a modified version of MiniSAT. Experiments show that the initialization-based approach is superior to score-overriding. The initial scores blur rapidly while search advances, but, since MiniSAT is a learning solver, the impact of score-initialization lasts longer than the induced priorities remain intact.

Experiments with score-overriding reveal no evident trend regarding runtime. However, using an initialization-based strategy the SAT solving run-time can be improved considerably in many cases. On a test set of 91 Kodkod problems, run-times could be improved by a factor of two or more for 26 instances (only for 6 problems, the runtime got worse by a factor of two or more); the maximal speed-up was over 800, while the worst deterioration was less than a factor of 4. Run-time was improved on 58 instances and deteriorated on 33.

References

1. Marques-Silva, J.P., Lynce, I.: Towards robust CNF encodings of cardinality constraints. In: CP'07. pp. 483–497 (2007)
2. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an efficient SAT solver. In: DAC'01. pp. 530–535 (2001)
3. Torlak, E., Jackson, D.: Kodkod: A relational model finder. In: TACAS'07. pp. 632–647 (2007)