Karlsruhe Institute of Technology

Institute for Theoretical Computer Science
Automated Software Analysis Group
JProf. Dr. Mana Taghdiri
Research Group Verification meets Algorithm Engineering
Dr. Carsten Sinz

Optimizing MiniSAT Variable Orderings for the Relational Model Finder Kodkod

Theory and Applications of Satisfiability Testing 2012 (SAT'12), Trento, Italy

Markus Iser, Mana Taghdiri, Carsten Sinz

Kodkod

- Constraint solver for Bounded Relational First Order Logic
- Translation into equisatisfiable propositional formula
- Interfaces with several SAT solvers (e.g. MiniSAT)
- Core API for Alloy Analyzer
- Core component of many other tools (e.g. JForge, TACO, Nitpick)

Primary Variables

- Each *primary variable* expresses whether or not a tuple belongs to a relation
- A problem is satisfiable iff there exists a satisfying assignment for its primary variables
- They usually make a few percent of the total amount of variables generated in SAT translation
- In literature such a subset of the variables is usually referred to as the input variables

Influencing MiniSAT's Variable Ordering

MiniSAT's branching heuristic orders variables by their *activity* which is initialized with zero and dynamically adjusted during runtime. At each decision the variable with the highest activity value is picked. We present three approaches that interfere with MiniSAT's native ordering heuristic.

- **Uniform Overriding**
  External priorities are used to *override* variable's activity. In the presented approach they are used to restrain the solver to primary variables. The within order of that subset is determined by each variable's activity.
- **Uniform Initialization**
  External priorities are used to *initialize* variable's activity. In the presented approach they are used to uniformly initialize activity of primary variables. The effect blurs since MiniSAT adjusts scores over time.
- **Structural Initialization**
  By structural analysis of Kodkod's Abstract Syntax Tree (AST) several subsets of primary variables receive individual weights. They are used to individually initialize the activity of those subsets of variables.

Uniform Activity Overriding

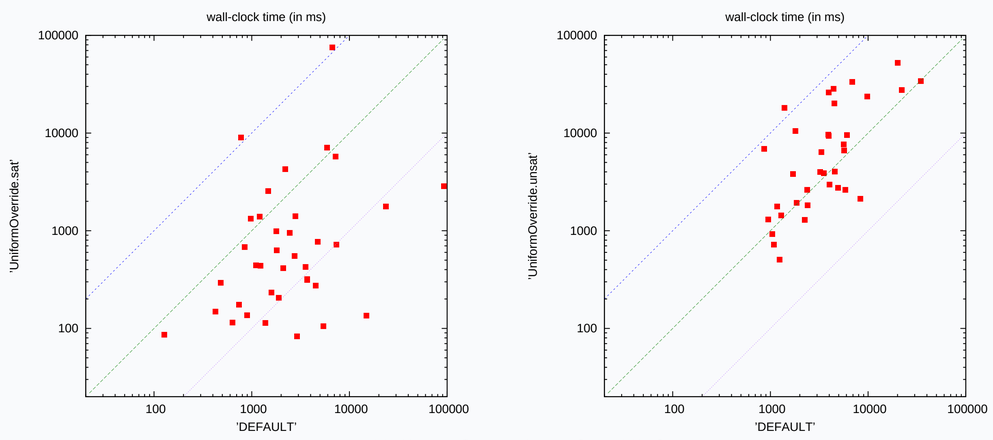

Figure: Uniform Overriding (SAT/UNSAT)

Search was restricted to primary variables. This is usually referred to as Input Restricted Branching. The results show a deterioration of solver performance on unsatisfiable problems. By contrast, increased performance is encountered with satisfiable problems, which is consistent with the intuition of search space reduction.

Uniform Activity Initialization

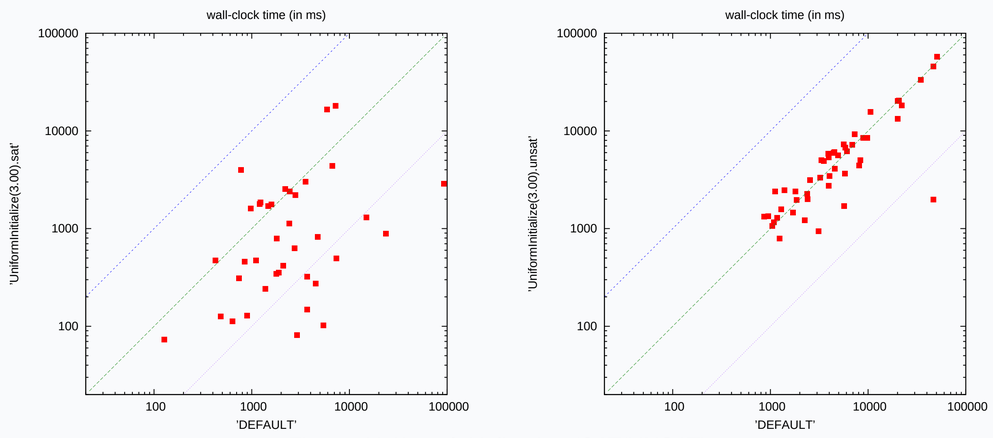

Figure: Uniform Initialization (SAT/UNSAT)

Activity of all primary variables was initialized with a value of 3. That method also works very well on satisfiable problems, and in addition does not suffer the runtime deterioration on unsatisfiable problems like in the overriding approach.

Structural Analysis of the Abstract Syntax Tree

- Kodkod's relations induce a partition $\{V_0, \ldots, V_n\}$ of the primary variables such that V_i denotes the variables corresponding to relation R_i
- Our algorithm distributes *scores* to relations participating in *highly constraining* formulas, the sums of those individual scores form a weight w that induces a partial ordering among the relations such that $w(R_i) \leq w(R_{i+1})$
- A formula is classified as being highly constraining if exponential decay in the number of possible solutions is encountered under application of that formula
- The idea is to first assign the primary variables that stem from the relation that is most highly constrained, thereby increasing the number of unit propagations

Highly Constraining Formulas

constraint category	formula	solution space without constraint	solution space with constraint
relational comparison	'R ⊆ S'	4^n	3^n
	'R = S'	4^n	2^n
cardinality bounds	'#R = c'	2^n	$\binom{n}{c}$
	'#R ≤ c'	2^n	$\sum_{k=0}^{c} \binom{n}{k}$

Table: Some highly constraining formulas and their effects on the size of the solution space (assuming a universe of size n, unary relations 'R' and 'S', and a constant number 'c').

Structural Activity Initialization

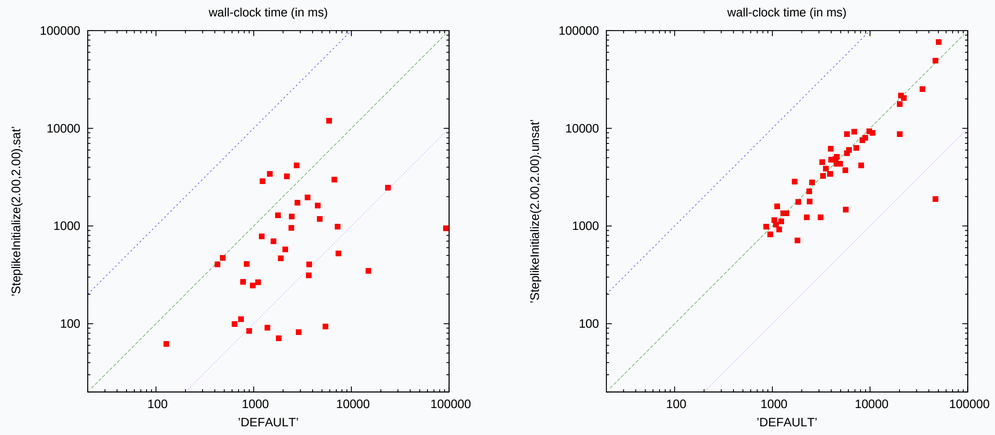

Figure: Structural Initialization (SAT/UNSAT)

Our algorithm generates for each relation R_i a weight $w(R_i)$ that induces an ordering on the relations by $w(R_i) \geq w(R_{i+1})$. Those weights are normalized with respect to the biggest weight such that $n(R_i) = w(R_i)/w(R_0)$. The normalized weights are used as an input to a linear function to calculate a weight-dependent activity $act(v) = b + f \cdot n(R_i)$. In the presented results the activity of each primary variable $v \in V_i$ is initialized with $act(v) = 2 + 2 \cdot n(R_i)$.

Uniform vs. Structural Initialization

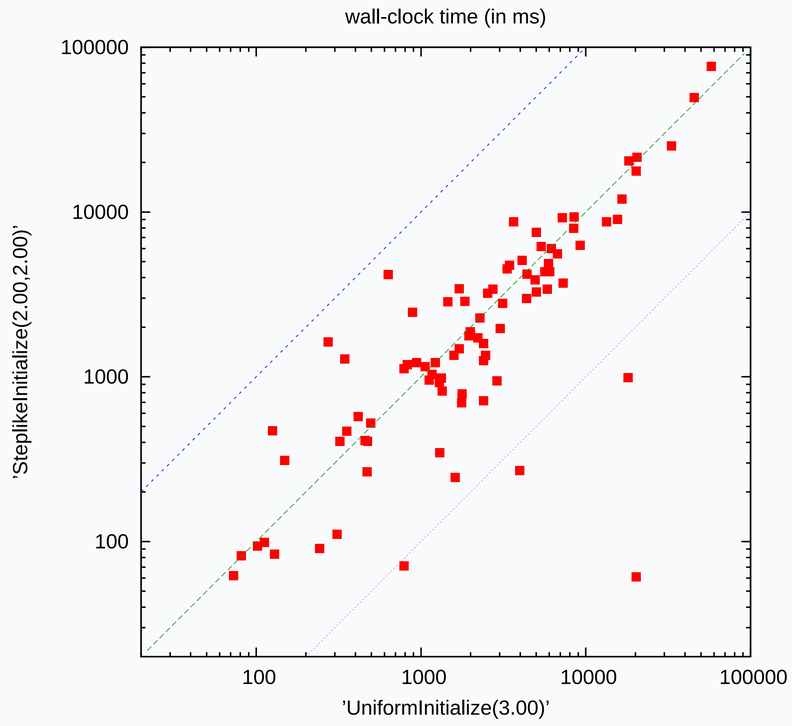

Figure: Uniform vs. Structural Initialization

Structure dependent activity initialization of primary variables shows slight runtime improvements compared to uniform activity initialization.