

STRUCTure: A parallel boolean satisfiability solver

Alexandru Moşoi
ami650@few.vu.nl
VU University Amsterdam

March 2, 2011

Introduction

STRUCTure is a novel look-ahead parallel boolean satisfiability solver running on the Constellation parallel framework [6]. The main goal of STRUCTure is scalability. Both STRUCTure and Constellation are developed at VU University Amsterdam¹ using the DAS-4 super computer². STRUCTure is available freely at <https://github.com/brtzsnr/structure>

Constellation's parallel programming model is based on activity spawning. Communication is done through message passing among running activities. There is no shared memory available, though activities running on the same machine could make use of the common address space. For the SAT Competition 2011 STRUCTure uses a single machine with up to number of cores specified activities running simultaneously.

STRUCTure is divided in two parts: a sequential preprocessor and distributed solver. In the preprocessing phase the instance is minimized using expensive reasonings such as XOR clauses extraction, dependent variable removal [4] (removal of variables that appear in a single XOR clause) and blocked clause

¹<http://www.cs.vu.nl/>

²<http://www.cs.vu.nl/das4/>

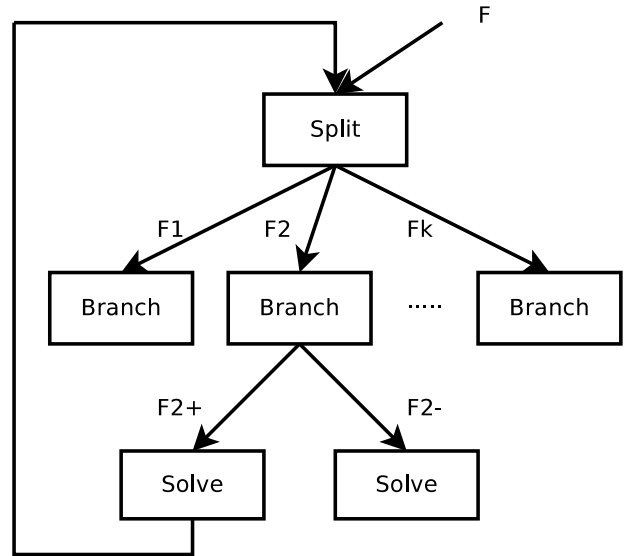


Figure 1: Activities workflow

elimination [5].

The goal of an activity in STRUCTure is to verify the satisfiability of an input SAT formula F . There are three types of activities listed below and pictured in figure 1.

- **SplitActivity** detects if F is composed of several smaller formulas F_1, \dots, F_k and solves the smaller formulas instead.
- **BranchActivity** picks a branching variable b and spawns two activities to solve formulas $F_- = F \wedge \{\neg b\}$ and $F_+ = F \wedge \{b\}$.

The solutions of F_- and F_+ are combined to answer F . The branching decision is similar to the one done in `March_eq` [2], with the observation that literals' scores are calculated based on the length of the clauses in which they appear and the graph of implications. Magic constants were fine tuned for the easy instances from SAT Competition 2009.

- **SolveActivity** tries to simplify the input formula and find a solution for it if it is simple enough (e.g. $\in 2SAT$). The simplifications include unit propagation, hyper-binary resolution [1], subsumming, pure literals propagation and binary reasoning. Binary reasoning is performed using the graph of implications and is composed of equality reduction, necessary assignment (u is a necessary assignment if $\neg u \rightarrow v \wedge \neg u \rightarrow \neg v$), hidden tautology elimination [3] and binary (self-)subsuming ((self-)subsuming when one clause has only two literals).

Scalability comes at a huge time and memory penalty of spawning new activities due to serialization and deserialization of SAT formulas therefore branching should be avoided whenever possible. The simplifications performed by **SolveActivity** are important and must be cheap. **STRUCTure** comes with a new fast algorithm for hyper-binary resolution which is the most expensive reasoning performed.

Future work includes 1. finding new branching decision heuristics 2. cheaper branching methods to reduce marshaling 3. back jumping 4. new preprocessing algorithms (e.g. Gaussian elimination on XOR clauses) 5. clause learning when unsatisfiable formula are identified.

References

- [1] F. Bacchus and J. Winter. Effective preprocessing with hyper-resolution and equality reduction.
- [2] Marijn Heule, Mark Dufour, Joris van Zwieten, and Hans van Maaren. `March_eq`: Implementing additional reasoning into an efficient look-ahead sat solver. pages 345–359. 2005.
- [3] Marijn Heule, Matti Järvisalo, and Armin Biere. Clause elimination procedures for cnf formulas. In *Proceedings of the 17th international conference on Logic for programming, artificial intelligence, and reasoning, LPAR'10*, pages 357–371, Berlin, Heidelberg, 2010. Springer-Verlag.
- [4] Marijn J.H. Heule. *SmArT solving: Tools and techniques for satisfiability solvers*. PhD thesis, TU Delft, 2008.
- [5] Matti Järvisalo, Armin Biere, and Marijn Heule. Blocked clause elimination. In *TACAS*, 2010.
- [6] Jason Maassen, Frank J. Seinstra, and Henri E. Bal. Context aware many-task computing with ibis/constellation. 2011.