SAT Solving: Present and Future (in two vignettes)

Oliver Kullmann

Swansea University, United Kingdom
http://cs.swan.ac.uk/~csoliver

Cambridge, March 27, 2012,
Logical Approaches to Barriers in Complexity II

From its head to its feet

In the last decade, SAT solving has become a tool used for many applications; see [BHvMW09] for an overview.

In my talk I want to discuss two problem areas

- which seem of importance to me for practical SAT solving,
- and where proof complexity should be able to offer solutions.

Sub-theme:

Tree-resolution has still a lot to offer, theoretically as well as practically.

Collaboration with my student Matthew Gwynne, and, recently started, with Olaf Beyersdorff.

Outline

Resolution parameters

We have three main resolution systems for SAT solving:

1. tree resolution
2. regular resolution
3. full (dag) resolution.

And there are three main parameters (with variations):

1. length
2. width
3. space

Finally there are two main types of complete SAT solvers:

- look-ahead ("DPLL"; close to tree resolution; effort on good branching and reduction)
- conflict-driven ("CDCL"; "approximates" dag resolution; effort on efficient learning).

Measuring "hardness"?

It is tempting to use one of the resolution-parameters as a hidden parameter to measure the difficulty for SAT solving:

- A basic problem is that these measures are "nearly non-computable" (especially for instances of interest).
- And obviously a SAT solver doesn't compute any of the measures — what degrees of "approximations" could be meaningful here?

What roles play the heuristics?

There are (important) heuristics for

- decision variable and decision value
- which clauses to learn
- which clauses to forget.

It is tempting to consider them as "approximating" resolution — what is the truth of this assumption?
A neglected aspect seems to me

1. complexity theory only performs asymptotical analysis
2. thus it can not be applied to any concrete instance.

(((Big-Oh as the downfall of computer science.)))

Furthermore, SAT solvers also go *beyond* resolution
(at least in the polynomial factors)!

Towards a structural proof complexity theory

Not all proofs are equal.
Length and other measures are far too rough.
But proofs have "shapes".

I believe "shapes" can become better guides for SAT solvers than numbers.

- "Shape" should yield the general structure.
- Numbers are then for fine-tuning.
- These numbers need to have a heuristical reality.

Tree resolution is nice!

We have much better tools for tree-resolution than for dag-resolution:

- We have the "hardness" ([Kul99, Kul04]), which links basic SAT algorithms with length and space of tree resolution.
- Since "hardness" is algorithmic, we get quasi-automatisation of tree-resolution.
- The Pudlák-Impagliazzo (prover-delayer) game ([PI00]) yields a game-theoretical interpretation of "hardness".
- The Beyersdorff-Galesi (asymmetric prover-delayer) game ([BGL10]) can yield precise(!) bounds for length.
- Last, but not least, we have a theory of branching heuristics ([Kul09]).

Reminder: Generalised UCP

Definition

The maps $\mathbf{r_k} : \mathcal{CLS} \to \mathcal{CLS}$ for $k \in \mathbb{N}_0$ are defined as follows:

$$\mathsf{r}_0(F) := \begin{cases} \{\bot\} & \text{if } \bot \in F \\ F & \text{otherwise} \end{cases}$$

$$\mathsf{r}_{k+1}(F) := \begin{cases} \mathsf{r}_{k+1}(\langle x \to 1\rangle * F) & \text{if } \exists\, x \in \mathsf{lit}(F) : \mathsf{r}_k(\langle x \to 0\rangle * F) = \{\bot\} \\ F & \text{otherwise} \end{cases}$$

The map $\mathbf{r_\infty} : \mathcal{CLS} \to \mathcal{CLS}$ is defined as $\mathsf{r}_\infty(F) := \mathsf{r}_{n(F)}(F)$.

Lemma

The maps are well-defined (don't depend on the choices), and apply (only) forced assignments. The bigger k the more forced assignments are found, and r_∞ applies all forced assignments.

Reminder: "Hardness"

The hierarchy of r_k-reductions, allowing also for oracles for SAT and UNSAT decisions (this will become important later), yields the following notion of **hardness** (here only using UNSAT and the trivial oracle):

Definition

The **hardness** $\mathrm{hd}(F)$ of unsatisfiable F is the minimal k with $r_k(F) = \{\bot\}$.

1. $2^{\mathrm{hd}(F)} \leq \mathrm{Comp}^*_R(F) \leq (n(F)+1)^{\mathrm{hd}(F)}$.
2. $\mathrm{hd}(F)+1$ is the space complexity of tree-resolution.
3. $\mathrm{hd}(F)$ is the level of nested input-resolution needed.
4. Considering the trivial oracle, the optimal value of the prover-delayer game is equal to the hardness.

Practical applications

- r_1 is UCP (unit-clause propagation).
- r_2 is failed-literal reduction (full).
- Running $r_0, r_1, \ldots$ achieves quasi-automatisation of tree-resolution.
- Remark: In [Kul99, Kul04] there is also an algorithmic extension of hardness to satisfiable clause-sets, and so we have also "hardness" for finding satisfying assignments.
- This approach is also the kernel of the Stalmarck-approach (at CNF-level).

Huuh — tree resolution is so out?!

What am I talking about — aren't we living in the age of the "modern SAT solver"?!
That is, we are approaching dag-resolution.

- I haven't seen yet any study at all, which would *show* (empirically) that it is "tree-like versus dag-like" what underlies the success of CDCL solvers.
- Though I accept it as a working hypothesis.

However there is another working hypothesis:

If $\mathrm{Comp}_R(F)$ and $\mathrm{Comp}^*_R(F)$ are close,
then look-ahead solvers are (far) better.

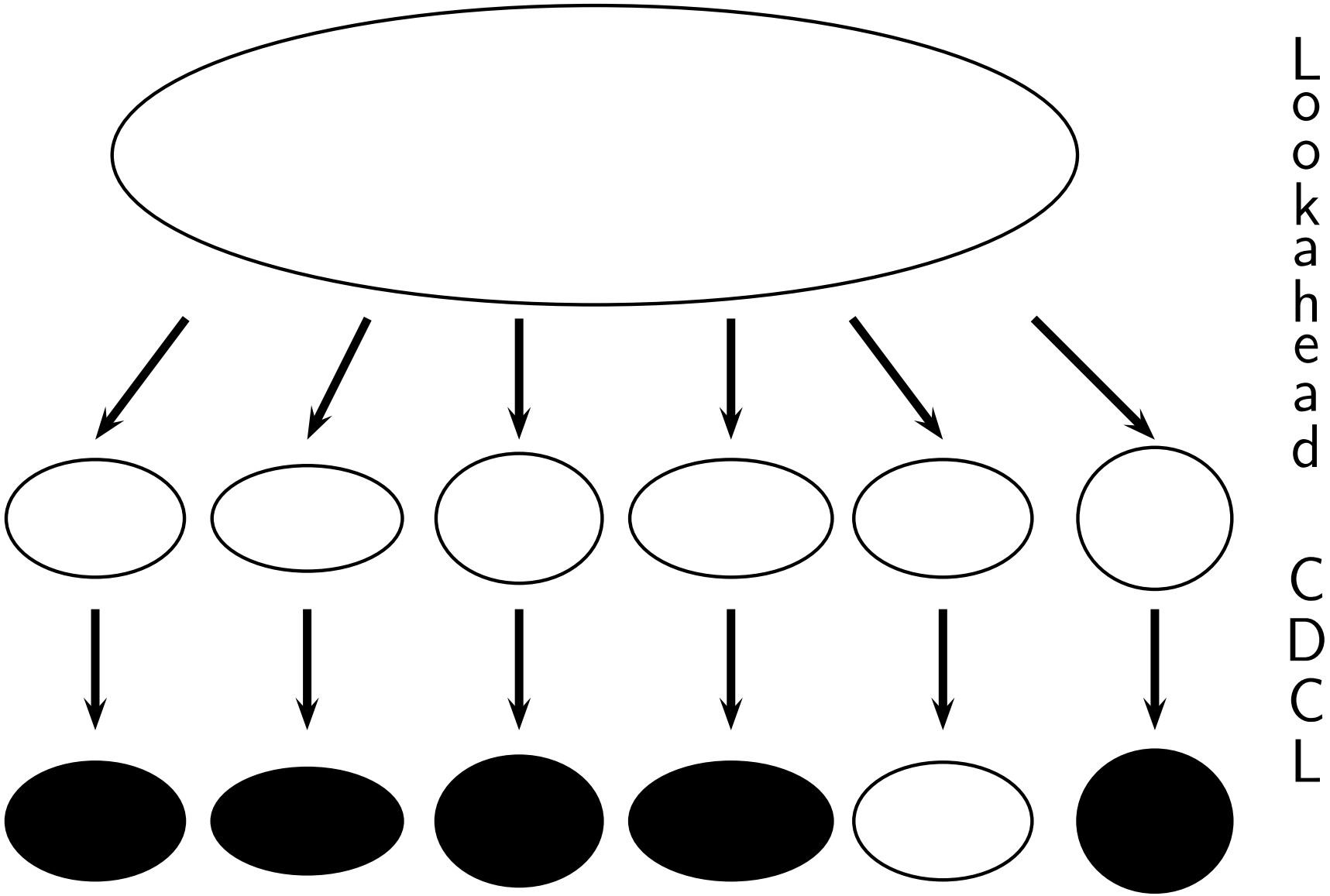
Cube and Conquer

Based on [AKS11] (experiments in exact Ramsey theory)

- I "invented" a combination of look-ahead and conflict-driven SAT solvers
- which works on various classes far better than *any* solver alone
- and this also on classes where conflict-driven works far better than look-ahead.

In [HKWB12] (see here) this was implemented in a more "industrial" setting, and shown to yield very good results on SAT-competition benchmarks.

Best of both worlds: Combining Lookahead and CDCL



Dags only at the leaves

Another common assumption on CDCL-solvers is needed here:

- they are not good at finding large proofs
- they have a "point of competence".

Then for instances which require dag-resolution, however not in an "entangled way", we can split off the tree part, and let the CDCL-solvers work on the smaller problems.

1. So good instances are those which have good resolution refutations with a tree-like part at the root.
2. With Olaf Beyersdorff we started investigating "mixed proof systems", based on [Kul99, Kul04], and now essentially employing the oracles at the leaves (using the CDCL solvers there).

Representing boolean functions

QBFs F yield a natural framework for representations of boolean functions f in the SAT context:

1. The free variables of F are the variables of the boolean function f.
2. A satisfying assignment for f is one making F true.

Real practical applications has Σ_1-CNF, which I consider as most natural:

- A clause-set F is a **CNF-representation** of the boolean function f if $\mathrm{var}(f) \subseteq \mathrm{var}(F)$ and the satisfying assignments of F projected to $\mathrm{var}(f)$ are (precisely) the satisfying assignments of f.
- Important special case: $\mathrm{var}(f) = \mathrm{var}(F)$, i.e., *without using new variables* (so F is equivalent to f).

We have $F \models f$, while in the other direction we need an extension.

"Good" representations

Translating a computational problem into a SAT-problem (should!) mean:

1. The building blocks are boolean functions ("constraints") like "at most one" or cryptographic boxes.
2. For them "good" (currently this typically means short) representations are sought (possibly using new variables).
3. The whole translation is the union of these representations.

"Good" representation for SAT however does not mean "short":

- It's about how the solver can "handle" it.
- Experience with SAT-cryptanalysis of DES/AES suggests that for small boolean functions, actually shortest representations are best, however this changes for bigger boolean functions (there the small representations perform badly).

Knowledge compilation?

The area of knowledge compilation (AI) has some relations to the theory of "good representations F of a boolean function f",

- since "all knowledge" about f should be presented in F,
- and this in an "accessible form",
- but the question is *how* the "retrieval" works!

"Good" SAT representations needs to be "understandable" by SAT solvers. This is rather different from knowledge compilation, where one has much more freedom in the retrieval mechanism.

And furthermore, although the prime implicates of prime implicants of f are surely "prime knowledge" of f, that's not all.

GAC and PC

The works in SAT being concerned about "good" representations borrowed the concept of "Generalised Arc Consistence" (**GAC**) from CP in the following way:

> If after applying a partial assignment φ to f we obtain a forced assignment, then we obtain it from $\varphi * F$ via UCP.

This amounts to compression of the prime implicates of f into F via UCP.

> What about the prime implicates of F ?!

We argue that we need also the prime implicates of F. There is a related concept in the literature, namely "Propagation Completeness" (**PC**), being investigated rather recently and employed under different circumstances:

> A clause-set F is PC iff whenever $\varphi * F$ has a forced assignment, then we obtain it from $\varphi * F$ via UCP.

Good representations of PHP_m^m

Before making "good" more precise, let's make things more concrete:

Is there a "good representation" of PHP_m^m *(as boolean function)?*

Conjecture I There is no good representation without using new variables.

Conjecture II There is a good representation using new variables.

All conclusions must be "nicely" derivable

Our general framework for a "good" representation F of f is:

All conclusions $F \models C$ must be "easily derivable".

Note that we consider all implication of F, not just those of f,

since splitting on new variables is important.

So reasonable first measures are

- the binary logarithm of the maximum of resolution complexity of deriving $F \models C$,
- the binary logarithm of the maximum of tree-resolution complexity of deriving $F \models C$.

A "good representation" then has logarithmically bounded measures.

Hardness generalised

We propose a measurement with better properties:

- First mentioned in [ABLM08].
- For a clause-set F the hardness is the maximum of $\mathrm{hd}(\langle x \to 0 : x \in C\rangle * F)$ over $F \models C$.
- This is the maximum level of nested input resolution needed to derive all $F \models C$.

A "good" presentation is then one of bounded hardness.

Hardness — to be softened, not measured

[ABLM08] proposed "hardness" (for unsatisfiable clause-sets) as measure of solution hardness.

- Let's call a clause-set *k-soft* if $\mathrm{hd}(F) \leq k$.
- We think of the main role of hardness for *satisfiable* clause-sets as being a **target**, to construct soft clause-sets (representing relevant boolean functions).

Using [ČKV12] (where $k = 1$ is handled) we get:

Lemma

For $k \geq 1$ the decision $\mathrm{hd}(F) \leq k$ is coNP-complete.

(Recall that for unsatisfiable F computation of $\mathrm{hd}(F)$ can be done in time $O(n^{2\,\mathrm{hd}(F)})$.)

SLUR properly turned into a hierarchy

The class SLUR ([SAFS95, Fra97, FG03]) is equal to

$$\{F \in \mathcal{CLS} : \mathrm{hd}(F) \leq 1\}.$$

Using "$\mathrm{hd}(F) \leq k$" we obtain a natural hierarchy.

Propagation-hardness

Now we have handled GAC properly. What about PC?

Definition

The **propagation-hardness** ("p-hardness") **phd(F)** of $F \in \mathcal{CLS}$ is the minimal $k \geq 1$ such that for all partial assignments φ we have

$$\mathsf{r}_k(\varphi * F) = \mathsf{r}_\infty(\varphi * F).$$

The class PC is equal to

$$\{F \in \mathcal{CLS} : \mathrm{phd}(F) \leq 1\}.$$

Using "$\mathrm{phd}(F) \leq k$" we obtain again a natural hierarchy (strengthening the SLUR-hierarchy).

Proper representation-hierarchies

Are the hierarchies useful for better representations, that is, does increasing the parameter (hardness or p-hardness) by 1 enable more clause-sets with short *and* "good" representations?

- We can show that going from k to $k+1$ enables exponentially shorter representations, when not using new variables.
- While the hierarchies collapse to the first level when allowing arbitrarily many new variables.

It is of interesting now to only allow a restricted amount of new variables. (However, lower bounds when allowing new variables don't seem easy.)

New perspectives on PHP

Now we have various possibilities to make the two conjectures precise:

Your turn!

Just to mention: For arbitrary m, n we have

$$\mathrm{hd}(\mathrm{PHP}_n^m) = \min(\max(m-1, 0), n).$$

(The task is now to consider arbitrary representations of PHP_n^m.)

Consider SAT!

As we have seen, satisfiable clause-set F become in this way an object of proof theory:

It's about the unsatisfiable clause-sets obtained
by applying partial assignments to F.

And a major perspective change is needed:

Not given F are to be studied,
but all F representing a given boolean function.

Conclusions

Understanding hardness of SAT solving:

- Study shape, not just numbers!
- Cube and Conquer is based on a two-phase proof search, combining tree- and dag-resolution.

Understanding SAT translations:

- Hardness and p-hardness as general measures of goodness of representations.
- New approach to a "proof theory of SAT" (taken literally).

Bibliography I

Carlos Ansótegui, María Luisa Bonet, Jordi Levy, and Felip Manyà.
Measuring the hardness of SAT instances.
In Dieter Fox and Carla Gomes, editors, *Proceedings of the 23th AAAI Conference on Artificial Intelligence (AAAI-08)*, pages 222–228, 2008.

Tanbir Ahmed, Oliver Kullmann, and Hunter Snevily.
On the van der Waerden numbers $w(2;3,t)$.
Technical Report arXiv:1102.5433v2 [math.CO], arXiv, October 2011.

Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria.
A lower bound for the pigeonhole principle in tree-like resolution by asymmetric prover-delayer games.
Information Processing Letters, 110(23):1074–1077, 2010.

Bibliography II


Armin Biere, Marijn J.H. Heule, Hans van Maaren, and Toby Walsh, editors.
Handbook of Satisfiability, volume 185 of *Frontiers in Artificial Intelligence and Applications*.
IOS Press, February 2009.


Ondřej Čepek, Petr Kučera, and Václav Vlček.
Properties of SLUR formulae.
In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, *SOFSEM 2012: Theory and Practice of Computer Science*, volume 7147 of *LNCS Lecture Notes in Computer Science*, pages 177–189. Springer, 2012.

Bibliography III

John Franco and Allen Van Gelder.
A perspective on certain polynomial-time solvable classes of satisfiability.
Discrete Applied Mathematics, 125:177–214, 2003.

John Franco.
Relative size of certain polynomial time solvable subclasses of satisfiability.
In Dingzhu Du, Jun Gu, and Panos M. Pardalos, editors, *Satisfiability Problem: Theory and Applications (DIMACS Workshop March 11-13, 1996)*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 211–223. American Mathematical Society, 1997.
ISBN 0-8218-0479-0.

Bibliography IV


Marijn J.H. Heule, Oliver Kullmann, Siert Wieringa, and Armin Biere.
Cube and conquer: Guiding CDCL SAT solvers by lookaheads.
To appear in LNCS, HVC 2011; available at
http://cs.swan.ac.uk/~csoliver/papers.html, January 2012.


Oliver Kullmann.
Investigating a general hierarchy of polynomially decidable classes of CNF's based on short tree-like resolution proofs.
Technical Report TR99-041, Electronic Colloquium on Computational Complexity (ECCC), October 1999.

Bibliography V

Oliver Kullmann.
Upper and lower bounds on the complexity of generalised resolution and generalised constraint satisfaction problems.
Annals of Mathematics and Artificial Intelligence, 40(3-4):303–352, March 2004.

Oliver Kullmann.
Fundaments of branching heuristics.
In Biere et al. [BHvMW09], chapter 7, pages 205–244.

Pavel Pudlák and Russell Impagliazzo.
A lower bound for DLL algorithms for k-SAT (preliminary version).
In *SODA*, pages 128–136. ACM/SIAM, 2000.

Bibliography VI

John S. Schlipf, Fred S. Annexstein, John V. Franco, and R.P. Swaminathan.

On finding solutions for extended Horn formulas.

Information Processing Letters, 54:133–137, 1995.

End