On SAT representations of XOR constraints (towards a theory of good SAT representations)

Oliver Kullmann

Computer Science Department
Swansea University
http://cs.swan.ac.uk/~csoliver/papers.html

Theoretical Foundations of Applied SAT Solving
January 24, 2014

Banff International Research Station

XOR

XOR-constraints are important for SAT solving and proof theory:

- Many SAT problems contain them (especially cryptographic ones).
- Many lower bounds on proof systems use them (in some form).

Only very recently have investigations started, whether the standard form of SAT translation can be improved.

Based on various **hardness measures**,
we start a systematic investigation.

That is, we consider all possible “representations” F of the boolean function given by a system S of XOR-constraints, and minimise “hardness” of F.

Changing some views I

This talk concentrates on the fundamental ideas:

There are various rather subtle
but crucial distinctions to be made.

To start with:

Translation = encoding + CNF-representation.

- The "encoding" maps the non-boolean to boolean variables.
- The "representation" maps the boolean function to a clause-set.
- We consider only representations here.

Using a "wild" encoding, every constraint can be trivialised!

Changing some views II

From a CSP-perspective it is not natural to think of

- **auxiliary variables** (additional variables in CNF-representation)
- as well as **constraint scopes of arbitrary size.**

Compared with this, our approach makes a strong distinction between

with/without auxiliary variables

(both have their advantages).
And allows naturally to handle

clauses of arbitrary size.

Changing some views III

A representation of a boolean function f

- is not just sat-equivalent to f,
- but must be either **logically equivalent** to f (without auxiliary variables),
- or, when using auxiliary variables, then the satisfying assignments of the presentation, when projected to the variables of f, must yield precisely the satisfying assignments of f.

Only in this way can the representation replace f, in the context of other clauses.

Changing some views IV

Finally, we consider a boolean function f and a CNF-representation.

So there is nothing than the representation.

The “other clauses”, which come from different constraints (making up the whole SAT-problem), are not here —

this belongs to another part of the theory,
the combination of CNF-representations.

We study the CNF-representations here *on their own*.

Hardness measures and hierarchies

The hardness measures $h : \mathcal{CLS} \to \mathbb{N}_0$ correspond to hierarchies:

- the sets of the hierarchy are $\{F \in \mathcal{CLS} : h(F) \leq k\}$;
- conversely, the hardness of F is the index of the first layer with F.

Sometimes it is more intuitive to think in terms of these hierarchies:

- These hierarchies are hierarchies for polytime SAT solving.
- **However**, we consider them under a different point of view, namely regarding representation of boolean functions.
- So for example we are interested in the best combination of hardness $h(F)$ and size amongst

  all clause-sets (logically) equivalent to F.

The hardness-considerations distinguish the approach from KC (Knowledge Compilation) — "hardness" must be relevant for SAT solving. One could speak of "SAT-KC".

Extension to SAT

We typically start with a measure

$$h_0 : \mathcal{USAT} \to \mathbb{N}_0$$

and extend it to $h : \mathcal{CLS} \to \mathbb{N}_0$ via

considering the worst case of $h_0(\varphi * F)$
for partial assignments φ such that $\varphi * F \in \mathcal{USAT}$.

That is, $h(F)$ for satisfiable F is the maximum of $h_0(F')$ for F' obtained from F by (partial) instantiation.

Link to proof theory

$h_0(F)$ measures proof complexity of unsatisfiable F.
$h(F)$ measures how bad arbitrary instantiations can be
(this can happen when running a SAT solver!).

New point of view for proof theory

The current task of proof theory is, to over-simplify it:

Create artificial examples which are "hard".

These examples are all unsatisfiable, and thus can be replaced by $\perp$. This arbitrariness is now turned into necessity as follows:

- Consider a representation F of a boolean function.
- We want to prove a lower bound on the size of a "good" F.
- So "hard" structures should show up in F which are too small.
- Thus the task now is to show, that those hard (artificial) unsatisfiable instances are **necessarily** hidden in F (somehow).

Good representations never create "hardness".

Intelligent representations

Yet typical for SAT translation:

- Either direct (simple) translation of each sub-constraint (XOR, cardinality, pseudo-boolean) — no "intelligence"
- or "DPLL(something)" — all intelligence outside of SAT.

We want to change that game:

We use intelligence to produce the translation —
(a) possibly considering larger junks (e.g., several XOR-constraints),
(b) and/or different hardness of the representation.

(a) Conjecture:

For lumping together (creating larger junks),
treewidth etc. is also of practical importance.

(b) We can show (yet for artificial examples): allowing a bit more "hardness" can save exponentially many clauses.

The main results reported here I

I report here on

lower and upper bounds for
"good" SAT-representations of XOR-clause-sets,

using various "hardness" measures to measure what "good" means.

We have a LATA 2014 paper Gwynne and Kullmann [9] (click), while the underlying (arXiv) report is Gwynne and Kullmann [7] (click).

The main results reported here II

Combining

- a translation of SAT-translations into monotone circuits, motivated by Bessiere, Katsirelos, Narodytska, and Walsh [2],
- with the lower bound on monotone span programs in Babai, Gál, and Wigderson [1]

we show that there is **no** polynomial-size SAT representation of arbitrary XOR-clause-sets, using the well-known notion(?!?) of quality, which we call **AC-representation.**

"AC-representation" — a CNF-representation
where every forced assignment after any partial instantiation
is detected by unit-clause propagation.

The main results reported here III

On the positive side:

- We show that computing an **AC-representation** is fpt in the number m of XOR-clauses.
- Considering the strongest criterion,

  representation via **propagation-complete clause-sets** $\mathcal{PC}$
  (introduced in Bordeaux and Marques-Silva [3])
  "PC = absolute AC" — now taking also
  the auxiliary variables into account

  we obtain various "intelligent" translations:

  1. The default representation X_1 for $m = 1$ is in $\mathcal{PC}$.
  2. With a more intelligent representation X_2 for $m = 2$ we also get $\mathcal{PC}$.

The main results reported here IV

We also start an analysis of the default representation $X_1(S)$ regarding various hardness measures, showing

- already for two XOR-clauses this is very bad considering **hardness** $\mathrm{hd}(X_1(S))$ (for unsat the same as clause-space of tree-resolution minus 1),
- while at least for two XOR-clauses it could be taken as "alright" when considering **w-hardness** $\mathrm{whd}(X_1(S))$ (using a generalised notion of width).

More precisely, for $m = 2$, $\mathrm{hd}(X_1(S))$ is up to $n - 2$ for n variables, while $\mathrm{whd}(X_1(S)) = 3$.

We don't know whether the (generalised) width only grows as a function of m (and not of n — recall $m \le n$, and in general m is much smaller than n).

The main results reported here V

Remark: So the standard representation $X_1(S)$ is very bad(!) (already for $m = 2$) for look-ahead solvers:

hard unsatisfiable instances have *precisely* $2^n \pm x$ nodes,
so already $n = 30$ is out of scope,

while CDCL-solvers handle $n = 10000$.
However with the new improved translation $X_2(S)$ (available yet only for $m = 2$):

Now also very easy for look-ahead solvers!

So here we have is an enormous improvement for look-ahead solvers (while a modest improvement for CDCL).

Other approaches at intelligent XOR-translations

- While we show fpt in the number m of XOR-clauses, the weaker parameter n, the number of variables, was show fpt in Laitinen, Junttila, and Niemelä [18].
- Practical results (SAT benchmarks) for translating XOR-clause-sets into CNF-clause-sets are in Laitinen, Junttila, and Niemelä [17].
- These authors also introduced the DPLL(XOR) framework, for integrating dedicated XOR-reasoning into SAT solving (Laitinen, Junttila, and Niemelä [15, 16]).

The project: a theory of SAT representations

See

- SOFSEM 2013 (click) and JAR (click) for the basic "hardness measures", measuring the "quality" of a representation: Gwynne and Kullmann [5, 8]
- Trading quality for size, showing that the various hardness measures yield hierarchies for the representation of boolean function,

  considering clause-sets **up to equivalence**
  (which yields much stronger hierarchies):

  Gwynne and Kullmann [6] (arXiv; click)
- These "hardness measures" for proof complexity: Kullmann [14] (arXiv; click).

Outline

The trivial representation of XOR-constraints

Let's assume we want to construct a "SAT representation" of something, which includes an XOR-constraint

$$x_1 \oplus \cdots \oplus x_n = \varepsilon,\ x_i \in \mathcal{LIT},\ \varepsilon \in \{0,1\}.$$

To make life easier, we assume $\varepsilon = 0$, and we represent that XOR-constraint simply as an **XOR-clause** $C := \{x_1, \ldots, x_n\} \in \mathcal{CL}$.

There is precisely one CNF-clause-set, which is equivalent to this XOR-clause, and we denote it by $\boldsymbol{X_0(C)} \in \mathcal{CLS}$.

- $X_0(C)$ has 2^{n-1} clauses of length n.
- For example $X_0(\{a,b\}) = \{\{a,\overline{b}\},\{\overline{a},b\}\}$.
- $X_0(C)$ is perfect for small n.

The standard representation of XOR-constraints I

We can use X_0 piecewise, obtaining the first general translation:

$$X_0 : \mathcal{CLS} \to \mathcal{CLS}$$
$$X_0(F) := \bigcup_{C \in F} X_0(C).$$

Now, to obtain a **small** translation for arbitrary XOR-clauses C, we use new variables. We split up C, using new variables y_i for partial sums, e.g. for $\{x_1, \dots, x_4\}$:

$$x_1 \oplus x_2 = y_2, \; y_2 \oplus x_3 = y_3, \; y_3 \oplus x_4 = 0.$$

In general C is split into an XOR-clause-set F' with $n - 1$ XOR-clauses, and we obtain the representation

$$\boldsymbol{X_1(C)} := X_0(F') \in 3\text{–}\mathcal{CLS},$$

The standard representation of XOR-constraints II

where we apply X_0 to the members of F'.

We got

$$\boldsymbol{X_1} : \mathcal{CL} \to 3\text{–}\mathcal{CLS}.$$

So we can represent a single XOR-constraint. If we have many of them, we apply the translation piecewise, obtaining

$$\boldsymbol{X_1} : \mathcal{CLS} \to 3\text{–}\mathcal{CLS}.$$

That is, for a general XOR-clause-set $F \in \mathcal{CLS}$ we get the representation

$$\boldsymbol{X_1(F)} := \bigcup_{C \in F} X_1(C) \in 3\text{–}\mathcal{CLS},$$

where new variables are used for the different XOR-clauses in F.

How good is this representation?

We now have a representation $X_1(F) \in 3\text{–}\mathcal{CLS}$ for arbitrary sets F of XOR-clauses.

- This is the default representation, used nearly everywhere.
- But is it “good” ?
- And can we do it “better” ?!

Measuring “hardness”

We have developed various hardness measures

$$\mathrm{phd}, \mathrm{hd}, \mathrm{whd} : \mathcal{CLS} \to \mathbb{N}_0$$

which measure the effort, in some way, to derive “everything” for all instantiations.

- The basis is $\mathrm{hd}, \mathrm{whd} : \mathcal{USAT} \to \mathbb{N}_0$.
- Both measures use resolution (tree/dag resolution).
- phd is a variation on hd.

Unit-clause propagation

A basic mechanism in determining satisfiability is

unit-clause propagation (UCP).

For example:

$$\{ \underbrace{\{a\}}_{\text{unit-clause}}, \{\overline{a}, b\}, \{\overline{b}\} \} \xrightarrow{\langle a \to 1 \rangle} \{ \{b\}, \{\overline{b}\} \} \xrightarrow{\langle b \to 1 \rangle} \{ \bot \}.$$

- Detects and sets some forced assignments, repeatedly.
- Possible in linear time, and is confluent.
- Using the map $\mathbf{r_1} : \mathcal{CLS} \to \mathcal{CLS}$ for UCP we have

$$\mathrm{r}_1(F) := \begin{cases} \{\bot\} & \text{if } \bot \in F \\ \mathrm{r}_1(\langle x \to 1 \rangle * F) & \text{if } \exists\, x \in \mathrm{lit}(F) : \bot \in \langle x \to 0 \rangle * F \\ F & \text{otherwise} \end{cases}.$$

Generalised unit-clause propagation

Kullmann [11, 13] introduced the notion of

generalised unit-clause propagation

$\mathrm{r}_k : \mathcal{CLS} \to \mathcal{CLS}$, $k \in \mathbb{N}_0$.

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

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

$\mathrm{r}_k(F)$ can be computed in time $\ell(F) \cdot n(F)^{2k-2}$.

Example: r_2 is more powerful r_1

$r_2 : \mathcal{CLS} \to \mathcal{CLS}$ is (full) *failed literal elimination*.

Consider

$$F := \{ \{a, b\}, \{a, \overline{b}\}, \{\overline{a}, b\}, \{\overline{a}, \overline{b}\} \}.$$

We have that

1. $r_1(F) = F$ (UCP does nothing).
2. $r_2(F) = r_2(\langle a \to 1\rangle * F) = \{\bot\}$, since

$$r_1(\langle a \to 0\rangle * F) = r_1(\{ \{b\}, \{\overline{b}\} \}) = \{\bot\}.$$

Hardness via r_k

For $F \in \mathcal{USAT}$ we define **hd(*F*)** as the

$$\text{minimum } k \in \mathbb{N}_0 \text{ such that } \mathrm{r}_k(F) = \{\bot\}.$$

And $\boldsymbol{\mathcal{UC}_k} := \{F \in \mathcal{CLS} : \mathrm{hd}(F) \le k\}$.

- $F \in \mathcal{UC}_0$ iff F is the set of all prime implicates of some boolean function (mod subsumption).
- $\mathcal{UC}_1 = \mathcal{UC}$ was introduced in del Val [4].
- $\mathcal{UC} = \mathcal{SLUR}$ ([5, 8]).

For unsatisfiable F, $\mathrm{hd}(F) + 1$ equals resolution tree-space.

Forced assignments

An assignment $\langle x \to 1\rangle$ for a literal x and $F \in \mathcal{CLS}$ is called **forced**, if $\langle x \to 0\rangle * F \in \mathcal{USAT}$.

Thus $\langle x \to 1\rangle * F$ is sat-equivalent to F.

- So we can (and should!) apply the partial assignment $\langle x \to 1\rangle$.
- Detection of a forced assignment is coNP-complete.
- So special cases need to be considered.
- The r_k detect and eliminate some forced assignments.
- With $k = n(F)$ we get all forced assignments.

Btw, for a forced assignment $\langle x \to 1\rangle$ also the literal x is called **forced**.

Propagation hardness

Let $\mathrm{r}_\infty(F) := \mathrm{r}_{n(F)}(F)$, that is, r_∞ applies all forced assignments.

Now $\mathrm{phd}(F)$ for $F \in \mathcal{CLS}$ is the

smallest k such that for all partial assignments φ
we have $\mathrm{r}_\infty(\varphi * F) = \mathrm{r}_k(\varphi * F)$.

Let $\boldsymbol{\mathcal{PC}_k} := \{F \in \mathcal{CLS} : \mathrm{phd}(F) \le k\}$.

- $\mathcal{PC}_k \subset \mathcal{UC}_k$.
- $\mathcal{PC}_1 = \mathcal{PC}$ was introduced in Pipatsrisawat and Darwiche [19], Bordeaux and Marques-Silva [3] (**unit-propagation complete**).

Width-hardness

[11, 13] and Kullmann [12] introduced a generalised notion of *width*, further studied in [14] (under the name of **asymmetric width** or **width-hardness**) — can handle long clauses!

Kleine Büning [10] introduced ***k*-resolution**:

$\boldsymbol{F} \vdash^{\boldsymbol{k}} \bot$ iff there is a resolution refutation of F, where for each resolution step at least one parent clause has length at most k.

For $F \in \mathcal{USAT}$ we define **whd(*F*)** as the

$$\text{minimum } k \in \mathbb{N}_0 \text{ such that } F \vdash^k \bot.$$

And $\boldsymbol{\mathcal{WC}_k} := \{F \in \mathcal{CLS} : \text{whd}(F) \leq k\}$.

- $\mathcal{WC}_0 = \mathcal{UC}_0$, $\mathcal{WC}_1 = \mathcal{UC}_1$
- $\mathcal{UC}_k \subset \mathcal{WC}_k$ for $k \geq 2$.

Relative hardness

For a set V of variables we define the relative hardness's

$$\mathbf{phd}^{V}(F), \mathbf{hd}^{V}(F), \mathbf{whd}^{V}(F)$$

by considering only partial assignments φ with $\mathrm{var}(\varphi) \subseteq V$ (for the extensions to satisfiable clause-sets).

- An **AC-representation** F of a boolean function f is a CNF-representation F of f with

$$\mathrm{phd}^{\mathrm{var}(f)}(F) \leq 1.$$

- "CNF-representation" is defined in the usual way ($\mathrm{var}(f) \subseteq \mathrm{var}(F)$, and the satisfying assignments of F projected to $\mathrm{var}(f)$ are precisely the satisfying assignments of f).

Monotonisation of boolean functions

Consider a boolean function f.

We want partial assignments to f,
handled by a boolean function $\widehat{f}$.

- Every variable is doubled.
- So we can encode "not assigned".

Now

$\widehat{f} = 0$ iff
the corresponding partial assignment
makes f unsatisfiable.

Example: the monotonisation of the bijective PHP_m^m function is the matching function (essentially).

Monotone circuits

Theorem

Consider a boolean function f and a representation F with

$$\mathrm{hd}^{\mathrm{var}(f)}(F) \leq 1.$$

From F we can compute in time $O(\ell(F) \cdot n(F)^2)$ a monotone circuit computing $\widehat{f}$.

Corollary

Boolean functions f_n have a CNF-representation F_n with $\mathrm{hd}^{\mathrm{var}(f_n)}(F_n) \leq 1$ and $\ell(F_n) = n^{O(1)}$ if and only if $\widehat{f_n}$ can be computed by monotone circuits of size polynomial in n.

No polysize AC for XOR's

Exploiting Babai et al. [1] (monotone span programs):

Theorem

The size of AC-representations of systems of XOR-constraints is super-polynomial in the number of constraints.

AC: FPT in number of XOR-constraints

Theorem

By adding all implied XOR-clauses, and translating each of them via X_1, we obtain an AC-representation of a system of m XOR-clauses with running time fixed-parameter tractable (fpt) in m (i.e., 2^m).

We believe that this can be strengthened in two dimensions:

- Instead of the "relative condition" AC, we can obtain the "absolute condition" $\mathcal{PC}$.
- Instead of fpt in m, we can obtain fpt in the treewidth of the incidence graph.

As a preliminary result in this direction, we can handle $m = 2$:

Lemma

By factoring out the common part of two XOR-clauses, we obtain a translation for $m = 2$ to $\mathcal{PC}$ in linear time.

Analysis of X_1

With our hardness measures we can also measure what X_1 is doing:

1. Already for two XOR-clauses, hardness hd can be very high, meaning very hard (unsatisfiable) problems for tree-resolution and look-ahead solvers (after appropriate instantiations).
2. However (symmetric as well as asymmetric) width for two XOR-clauses is 3, and indeed the unsatisfiable problems obtained by instantiations are very easy for conflict-driven solvers.
3. But the width-hardness whd must grow with the number of XOR-clauses (at least) — all Tseitin formulas can be created by instantiations.
4. A more precise and complete analysis is needed.

Summary and outlook

I We believe there is a whole world to be discovered.

II Hopefully a theory of “good SAT representations” will emerge which truly brings theory (proof theory) and practice (SAT solving) together.

III The translation of XOR-systems is a good first test-case: Despite the bad news “no poly-size AC-representation”, there seem to be a lot of opportunities for good representations (under various circumstances).

End

(references on the remaining slides).

For my papers see
http://cs.swan.ac.uk/~csoliver/papers.html.

Bibliography I

[1] László Babai, Anna Gál, and Avi Wigderson. Superpolynomial lower bounds for monotone span programs. *Combinatorica*, 19 (3):301–319, March 1999.

[2] Christian Bessiere, George Katsirelos, Nina Narodytska, and Toby Walsh. Circuit complexity and decompositions of global constraints. In *Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 412–418, 2009.

[3] Lucas Bordeaux and Joao Marques-Silva. Knowledge compilation with empowerment. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, *SOFSEM 2012: Theory and Practice of Computer Science*, volume 7147 of *Lecture Notes in Computer Science*, pages 612–624. Springer, 2012.

Bibliography II

[4] Alvaro del Val. Tractable databases: How to make propositional unit resolution complete through compilation. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 551–561, 1994.

[5] Matthew Gwynne and Oliver Kullmann. Generalising and unifying SLUR and unit-refutation completeness. In Peter van Emde Boas, Frans C. A. Groen, Giuseppe F. Italiano, Jerzy Nawrocki, and Harald Sack, editors, *SOFSEM 2013: Theory and Practice of Computer Science*, volume 7741 of *Lecture Notes in Computer Science (LNCS)*, pages 220–232. Springer, 2013. doi: 10.1007/978-3-642-35843-2_20.

[6] Matthew Gwynne and Oliver Kullmann. Trading inference effort versus size in CNF knowledge compilation. Technical Report arXiv:1310.5746v2 [cs.CC], arXiv, November 2013.

Bibliography III

[7] Matthew Gwynne and Oliver Kullmann. On SAT representations of XOR constraints. Technical Report arXiv:1309.3060v4 [cs.CC], arXiv, December 2013.

[8] Matthew Gwynne and Oliver Kullmann. Generalising unit-refutation completeness and SLUR via nested input resolution. *Journal of Automated Reasoning*, 52(1):31–65, January 2014. doi: 10.1007/s10817-013-9275-8.

[9] Matthew Gwynne and Oliver Kullmann. On SAT representations of XOR constraints. In Adrian-Horia Dediu, Carlos Martín-Vide, José-Luis Sierra, and Bianca Truthe, editors, *LATA 2014: Language and Automata Theory and Applications, 8th International Conference*, volume 8370 of *Lecture Notes in Computer Science (LNCS)*, pages 409–420. Springer, 2014.

[10] Hans Kleine Büning. On generalized Horn formulas and k-resolution. *Theoretical Computer Science*, 116:405–413, 1993.

Bibliography IV

[11] 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.

[12] Oliver Kullmann. An improved version of width restricted resolution. In *Electronical Proceedings of Sixth International Symposium on Artificial Intelligence and Mathematics*, January 2000. 11 pages; http://rutcor.rutgers.edu/~amai/aimath00/AcceptedCont.htm.

[13] 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.

Bibliography V

[14] Oliver Kullmann. Hardness measures and resolution lower bounds. Technical Report arXiv:1310.7627v2 [cs.CC], arXiv, February 2014.

[15] Tero Laitinen, Tommi Junttila, and Ilkka Niemelä. Extending clause learning DPLL with parity reasoning. In Helder Coelho, Rudi Studer, and Michael Wooldridge, editors, *ECAI 2010 – 19th European Conference on Artificial Intelligence*, pages 21–26. IOS Press, 2010.

[16] Tero Laitinen, Tommi Junttila, and Ilkka Niemelä. Extending clause learning SAT solvers with complete parity reasoning. In *ICTAI 2012 – 24th International Conference on Tools with Artificial Intelligence*, pages 65–72, 2012.

Bibliography VI

[17] Tero Laitinen, Tommi Junttila, and Ilkka Niemelä. Classifying and propagating parity constraints. In Michela Milano, editor, *Principles and Practice of Constraint Programming – CP 2012*, volume 7514 of *Lecture Notes in Computer Science*, pages 357–372. Springer, 2012.

[18] Tero Laitinen, Tommi Junttila, and Ilkka Niemelä. Simulating parity reasoning. In *Logic for Programming Artificial Intelligence and Reasoning – LPAR 2013*, LNCS Advanced Research in Computing and Software Science. Springer, 2013.

[19] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, 2011.