Towards a better understanding of SAT translations (From Hardness to Softness)

Matthew Gwynne[1] Oliver Kullmann[2]

Computer Science, Swansea University, UK

[1]`http://cs.swan.ac.uk/~csmg`

[2]`http://cs.swan.ac.uk/~csoliver`

Proof Complexity
Banff, October 4, 2011

SAT translations: case studies and theory

Some remarks on the genesis of this research:

1. We started by translating AES and DES into SAT.
2. Trying to develop good translations, we came up with some general ideas.
3. In this talk, only this theory side is considered.
4. See the forthcoming technical report [Gwynne and Kullmann, 2011], where we will then also present extensive experimental data (and their analysis).

All software is available in the `OKlibrary` (`http://www.ok-sat-library.org`).

Outline

Generalised UCP

In [Kullmann, 1999, Kullmann, 2004] the following hierarchy of reductions $r_k : \mathcal{CLS} \to \mathcal{CLS}$ has been investigated:

$$r_0(F) := \begin{cases} \{\bot\} & \text{if } \bot \in F \\ F & \text{otherwise} \end{cases}$$

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

- r_1 is unit-clause propagation (UCP)
- r_2 is failed-literal reduction

Running time

$r_k(F)$ can be computed in time

$$\ell(F) \cdot O(n(F)^{2k-2})$$

for fixed $k \geq 1$.

- Using $\ell(F)$ for the length of F and $n(F)$ for the number of variables.
- This comes from linear-time computation of r_1 (which is optimal).
- It is not known whether for $k \geq 2$ this can be improved.

Hardness for unsatisfiable clause-sets

For unsatisfiable F the **hardness** is defined as

$$\mathbf{hd}(F) := \min\{k \in \mathbb{N}_0 : r_k(F) = \{\bot\}\}.$$

We call F k**-soft** if $\mathrm{hd}(F) \leq k$.

For the tree-resolution complexity $\mathbf{Comp}^*_{\mathbf{R}}(F)$ (minimum number of leaves in a tree representing a resolution refutation of F) we have

$$2^{\mathrm{hd}(F)} \leq \mathrm{Comp}^*_{\mathrm{R}}(F) \leq (n(F)+1)^{\mathrm{hd}(F)}.$$

Computing $r_0(F), r_1(F), \ldots$ achieves quasi-automisation of tree resolution.

The levelled height of trees

Let the *levelled height* $\mathrm{h_l}(T)$ of a rooted tree be defined as follows:

1. If T is trivial then $\mathrm{h_l}(T) := 0$.
2. Otherwise consider the subtrees $T_1, \dots, T_k$, $k \geq 1$, at the root, and let $m := \max_{i=1}^{k} \mathrm{h_l}(T_i)$.
3. If there is exactly one $i \in \{1, \dots, k\}$ with $\mathrm{h_l}(T_i) = m$, then $\mathrm{h_l}(T) := m$.
4. Otherwise $\mathrm{h_l}(T) := m + 1$.

We have the following equivalent descriptions:

- For binary trees T we have that $\mathrm{h_l}(T) + 1$ is the pebbling complexity of T in the black-pebbles game allowing shifting of pebbles.
- For arbitrary rooted trees T we have that $\mathrm{h_l}(T) + 1$ is the *Strahler number* of T.

Space complexity

- For an unsatisfiable F we have $\mathrm{hd}(F) \leq k$ iff there is a resolution tree refutation of F with $\mathrm{h}_{\mathrm{l}}(T) \leq k$.
- Thus $\mathrm{hd}(F)$ is the space-complexity of F w.r.t. tree resolution.
- As shown in [Kullmann, 2004], the characterisation of $\mathrm{hd}(F)$ in terms of $\mathrm{h}_{\mathrm{l}}(T)$ for resolution trees carries over to a very general form of constraint satisfaction problems (with non-boolean variables).
- However for non-boolean variables the characterisation via space-complexity breaks down.

Generalisation for all clause-sets

In [Kullmann, 1999, Kullmann, 2004] also an **algorithmically motivated** extension of $\mathrm{hd}(F)$ for all clause-sets F has been introduced and discussed.

Here now we investigate (for the first time) another extension which shall measure how good F is as a **representation** of some underlying boolean function:

For clause-set F the **hardness** $\mathrm{hd}(F)$ is the smallest $k \in \mathbb{N}_0$ such that for all clauses C with $F \models C$ this can be verified by means of r_k, i.e.,

$$\mathrm{hd}(\langle x \to 0 : x \in C \rangle * F) \le k.$$

(Using $F \models C \Leftrightarrow F \wedge \neg C \models \bot$.)

Generalised input resolution

$\mathrm{hd}(F) \le k$ if and only if
for every clause C with $F \models C$
there is a tree resolution derivation T
of $C' \subseteq C$ from F with $\mathrm{h_l}(T) \le k$.

- We have $\mathrm{hd}(F) \le 1$ iff for every clause C with $F \models C$ there is a input derivation of $C' \subseteq C$ from F.
- And in general we have $\mathrm{hd}(F) \le k$ iff for every clause C with $F \models C$ there is a k-times nested input derivation of $C' \subseteq C$ from F.

Here a *k-times nested input-resolution derivation* is just a resolution tree T with $\mathrm{h_l}(T) \le k$.

1. For $k = 1$ this is just input resolution.
2. And a $k + 1$-times nested derivation has the shape of an input resolution, where at the axiom-places we have k-times nested derivations.

Relations

- Likely decision whether $\mathrm{hd}(F) \leq k$ holds is Π_2-complete (for fixed k).
- Apparently the first time this extension (to satisfiable clause-sets) of the basic hardness measure (as introduced in [Kullmann, 1999, Kullmann, 2004]) was (briefly) mentioned in the literature is [Ansótegui et al., 2008].
- We consider $\mathrm{hd}(F)$ for satisfiable F not as a measure of solving-hardness (it would be asking too much!), but as

  target for **constructing** good representations.

Representing boolean functions by CNFs

A **boolean function** is a map $f : \{0, 1\}^V \to \{0, 1\}$ for some (finite) set V of variables.

A clause-set F **represents** f if

- $\text{var}(f) \subseteq \text{var}(F)$
- taking the set of satisfying total assignments for F and restricting it to V, we obtain exactly the set of satisfying assignments for f.

If F has exactly the same number of satisfying total assignments as f, then the representation has the **unique extension property** (uep).

Remark: In practice all representations seem to have uep — could there be a proof that we need only to consider representations with uep "without loss of power"?

A different point of view

For a clause-set F, the boolean functions represented by F are obtained as follows:

1. Let f_0 be the boolean function underlying F (with $\text{var}(f_0) = \text{var}(F)$).
2. Now the boolean functions represented by F are exactly the "1-projections" of f_0 to $V \subseteq \text{var}(F)$.
3. Such a 1-projection for an assignment to V yields 1 iff there exists an extension to a satisfying assignment of F.
4. So F represents $(0)_{x \in \{0,1\}^{\emptyset}}$ iff F is unsatisfiable.
5. And F represents $(1)_{x \in \{0,1\}^{\emptyset}}$ iff F is satisfiable.

The SAT Representation Hypothesis (SRH)

SRH is the following hypothesis under development:

A representation F of a boolean function f is "good"
for SAT solving if and only if
F has low hardness
(and F is not too large).

Two features:

1. A representation F of f with low hardness must allow to derive *all* clauses which follow from F — not just those which follow from f.
2. There is a tradeoff between hardness and the size of the representation.

Low hardness is "knowing the truth-table"

What is the meaning of having low hardness?

- "Knowing" a boolean function means "knowing the truth-table".
- Similarly, "knowing" a constraint means knowing the satisfying (and falsifying!) assignments.
- In the same vein, now "knowing" means "falsification can be detected by r_k-reduction".

So having a representation F of f with "low hardness" can be interpreted as a parameterised version of

"F acting as a constraint".

Hardness 1 versus “hyperarc consistency”

In the literature one finds the related notion of “(hyper)arc consistency”:

- This (seems) to mean that for every partial assignment *in the original variables* (that is, $\mathrm{var}(f)$) one can find all forced assignments by UCP.
- In contrast, our approach also takes the new variables into account (i.e., $\mathrm{var}(F)$).
- Instead of UCP (i.e., r_1) we now consider r_k.
- We treat as the central category the detection of mere falsification, not forced assignments.
- The term “(hyper)arc consistency” is not appropriate, since the notion of “constraint” is very fuzzy here.

So we propose to consider our notion of hardness as a good replacement of “hyperarc consistency” (of course, only for SAT translations).

Remarks on Extended Resolution (ER)

- The SRH says: The whole business of Extended Resolution is to construct some (poly-size) k-soft representation for appropriate (fixed!) $k \geq 1$.
- Later we will discuss this w.r.t. PHP.
- SRH needs only to consider tree-like resolution, since w.r.t. ER full resolution and tree-like resolution have the same power.

Two natural questions here:

- Can we make the application of our framework more powerful by looking at smaller boolean functions inside the "big" constant-0 function?
- Is splitting on the new variables of importance, or is the sole purpose of the new variables to enable compression of prime implicates via r_k-reduction?

Remarks on “too big” boolean functions

- We don’t know the truth-table of DES or AES.
- So we have to decompose the big function into small functions.
- We do not understand how to make a “good decomposition”.
- For this first phase of our investigations, we only considered the obvious decomposition, and apply SRH to the small functions.

Prime implicates I

- A **prime implicate** of a boolean function f is a clause C with $f \models C$ and $\forall\, C' \subset C : f \not\models C'$.
- And a prime implicate of a clause-set F is a prime implicate of the underlying boolean function.

By $\mathbf{prc_0(\mathit{f})}$ resp. $\mathbf{prc_0(\mathit{F})}$ we denote the set of all prime implicates ("0" for unsat – falsifying assignments).

$\mathrm{prc}_0(f)$ is the prototypical representation of f with hardness 0 — in the light of SRH, "all what remains" is to find suitable abbreviations for this set (which is mostly too large for SAT solving).

Prime implicates II

- “Smurfs” ([Franco et al., 2004]) yield representations of boolean functions comprising all prime implicates and all prime implicants via a BDD-like approach.
- We on the other hand “believe in CNF”.
- CNF offer the potential of breaking up the barriers between “constraints”.
- And representations by CNFs offer the potential of splitting on new variables.
- That is, we break up the black box.

Bases

A basic systematic approach for finding a k-soft representation of f is

1. Start with $F := \mathrm{prc}_0(f)$.
2. Repeatedly remove clauses $C \in F$ such that F remains k-soft.

A completed such computation yields a ***k*-base**.

- We have developed some heuristic improvements of this basic algorithm.
- Given the truth-table of f (which we always assume), decision of "F is k-base for f" is in polytime.
- So finding a k-base is a search problem in NP.
- The optimisation problem seems very tough, even for boolean functions with just, say, 8 variables.

The canonical translation: The idea

A class of alternative approaches for finding 1-soft representations of f is based on the following idea:

1. Consider the canonical DNF $\mathrm{DNF}(f)$, consisting of all prime implicants of f (i.e., all satisfying total assignments, as DNF-clauses).
2. Apply the Tseitin translation to $\mathrm{DNF}(f)$.

This yields a 1*-soft representation of* f*.*

- There is more to it than just "Tseitin translation applied to DNF", and we present a more systematic development.
- For DES/AES, the main boolean functions are the "boxes", which are permutations, and permutations have unique DNFs which are also small.

The semantics: 1-extensions

For a boolean function f and $C \in \mathrm{DNF}(f)$ we consider a new variable $\mathrm{vct}_f(C)$.

The **canonical 1-extension** of f is the *boolean function*

$$\mathbf{ce}(f) := f \wedge \bigwedge_{C \in \mathrm{DNF}(f)} \mathrm{vct}_f(C) \leftrightarrow \bigwedge_{x \in C} x.$$

A **general canonical representation** of f is a representation of $\mathrm{ce}(f)$ without new variables.

- We believe that it is important to start with the semantical side, the boolean function.
- And not directly jumping to syntactical manipulations — like the Tseitin translation.
- The point here is that there are many general canonical representations!
- And we can apply the ideas underlying the notion of a k-base to $\mathrm{ce}(f)$.

Recall PHP

For $m \in \mathbb{N}_0$ pigeons and $k \in \mathbb{N}_0$ holes we have the clause-sets PHP_k^m:

- variables are $p_{i,j}$ for $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, k\}$
- expressing "pigeon i sits in hole j"
- we have binary clauses expressing that no two pigeons sit in the same hole
- and we have m clauses of length k, expressing that every pigeon sits in one hole.

PHP_k^m is satisfiable iff $m \leq k$.

Hardness of PHP

In [Kullmann, 1999] it was established $\mathrm{hd}(\mathrm{PHP}_k^m) = k$ for $m > k$. This is generalised now by

$$\mathrm{hd}(\mathrm{PHP}_k^m) = \min(\max(m-1, 0), k)$$

for $m, k \in \mathbb{N}_0$.

- The upper bound is established by the observation that setting any variable to true and applying r_1 yields PHP_{k-1}^{m-1}.
- For the lower bound we additionally observe that when setting any variable to false, then setting any remaining variable to true we again obtain PHP_{k-1}^{m-1}.

Remarks on tree-resolution complexity

From $\mathrm{hd}(\mathrm{PHP}_k^m) = k$ for $m > k$ we get
$2^k \leq \mathrm{Comp}^*_{\mathrm{R}}(\mathrm{PHP}_k^m) \leq (m \cdot k + 1)^k$.

- This lower bound appeared first in [Buss and Pitassi, 1998].
- In [Iwama and Miyazaki, 1999] this was sharpened to $(\frac{k}{4})^{\frac{k}{4}} \leq \mathrm{Comp}^*_{\mathrm{R}}(\mathrm{PHP}_k^{k+1}) \leq O(k^2 \cdot k!)$.
- In [Dantchev and Riis, 2001] this was generalised to $k^{\Omega(k)} \leq \mathrm{Comp}^*_{\mathrm{R}}(\mathrm{PHP}_k^m) \leq m^{O(k)}$.
- Here actually the upper bound holds for *any* regular tree-resolution refutation.
- In [Beyersdorff et al., 2010] one finds a simpler proof for $k^{\Omega(k)} \leq \mathrm{Comp}^*_{\mathrm{R}}(\mathrm{PHP}_k^m)$.

The hardness parameter $\mathrm{hd}(F)$ in general does not yield very sharp bounds for tree-resolution, however it seems to be the simplest general method.

Reminder Extended Resolution

It seems best to us to split ER into two steps:

1. **Extension** The original clause-set F is extended to F' stepwise, by adding representations (without new variables) of
$$v \leftrightarrow f$$
where v is a new variable and f is a boolean function in the old variables.
2. **Resolution** F' is used for a resolution refutation.

ER is polynomially equivalent to Extended Frege with Substitution.

Extended Resolution for PHP

[Cook, 1976] introduced a specific extension EPHP_k of PHP_k^{k+1}:

- In this way the (very simple) inductive proof of "there is no injection from $\{1, \ldots, k+1\}$ to $\{1, \ldots, k\}$" can be simulated.
- And this by a polysize resolution refutation.

We wondered about the *tree*-resolution complexity of EPHP_k:

$$\text{Possibly } \mathrm{hd}(\mathrm{EPHP}_k) = k.$$

That is, tree-resolution can't do much with the extension.

Tree- versus full resolution for ER (?!?)

Given a clause-set F and a resolution refutation R of F, we get an extension $F'_R(F)$ by adding the equivalences

$$v \leftrightarrow C$$

for all the (different!) clauses in R (axioms and resolvents). Then

$$\mathrm{hd}(F'_R(F)) \leq 2.$$

In this sense ER-with-tree-resolution and ER-with-full-resolution are polynomially equivalent:

However this equivalence is **non-uniform**!
That is, given just F and an extension F', it is not known how to compute an extension F'' of F' in polytime, such that if F' has a polysize resolution refutation, then F'' has a polysize tree-resolution refutation.

Summary

I We investigated a general notion of “hardness” for clause-sets.

II We sketched the SRH, that is, “good representation” means “low hardness”.

III (We introduced two methods for constructing representations of low hardness.)

IV We applied hardness considerations to PHP.

V (We presented first data on attacking DES and AES using these methods.)

Bibliography I

Ansótegui, C., Bonet, M. L., Levy, J., and Manyà, F. (2008).
Measuring the hardness of SAT instances.
In Fox, D. and Gomes, C., editors, *Proceedings of the 23th AAAI Conference on Artificial Intelligence (AAAI-08)*.

Beyersdorff, O., Galesi, N., and Lauria, M. (2010).
A lower bound for the pigeonhole prinziple in tree-like resolution by asymmetric prover-delayer games.
Technical Report TR10-081, Electronic Colloquium in Computational Complexity (ECCC).

Bibliography II

Buss, S. and Pitassi, T. (1998).
Resolution and the weak pigeonhole principle.
In Nielsen, M. and Thomas, W., editors, *Computer Science Logic (CSL) 97*, volume 1414 of *Lecture Notes in Computer Science*, pages 149–156.

Cook, S. A. (1976).
A short proof of the pigeonhole principle using extended resolution.
SIGACT News, pages 28–32.

Dantchev, S. and Riis, S. (2001).
Tree resolution proofs of the weak pigeon-hole principle.
In *16th Annual IEEE Conference on Computational Complexity*, pages 69–75.

Bibliography III

Franco, J., Kouril, M., Schlipf, J., Weaver, S., Dransfield, M., and Vanfleet, W. M. (2004).
Function-complete lookahead in support of efficient SAT search heuristics.
Journal of Universal Computer Science, 10(12):1655–1692.

Gwynne, M. and Kullmann, O. (2011).
Attacking DES + AES via SAT: Constraints and boolean functions.
Technical Report arXiv:??? [cs.DM], arXiv.

Bibliography IV

Iwama, K. and Miyazaki, S. (1999).
Tree-like resolution is superpolynomially slower than dag-like resolution for the pigeonhole principle.
In Aggarwal, A. and Rangan, C. P., editors, *Algorithms and Computation*, volume 1741 of *Lecture Notes in Computer Science*, pages 133–142. Springer.

Kullmann, O. (1999).
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).

Bibliography V

Kullmann, O. (2004).
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.

End