On the hardness of (satisfiable) conjunctive normal forms

Matthew Gwynne and Oliver Kullmann

Computer Science Department
Swansea University
csmg@swansea.ac.uk
http://cs.swansea.ac.uk/~csmg/

Swansea University, PCV Seminar, 5th May 2011
Introduction

Hardness

A notion of **hardness** is a function $h : \mathcal{CLS} \rightarrow \mathbb{N}_0$, assigning to every clause-set $F \in \mathcal{CLS}$ a natural number $h(F)$.

- Bounds for the complexity of proof systems (for instance resolution).
- Hierarchy of poly-time SAT decision.
- Measurement of hardness for SAT solving.

The fundamental notion, based on tree-resolution, introduced by Kullmann in [1, 2].
Overview

1. Hardness based on tree resolution
2. New hardness notion for SAT
3. The SAT representation hypothesis
4. Attacking AES and DES
5. Future work
A “tree-based” notion of hardness

Hierarchy of clause-set classes

Consider a clause-set $F \in \mathcal{CL}S$ and $k \in \mathbb{N}_0$.

1. $F \in G_k^0(U, S)$: “unsatisfiable in $k$ levels”.
2. $F \in G_k^1(U, S)$: “satisfiable in $k$ levels”.
3. $F \in G_k(U, S) := G_k^0(U, S) \cup G_k^1(U, S)$.

$U$ and $S$ are unsatisfiability and satisfiability oracles.

Definition (Hardness)

The hardness $h_{(U,S)}(F) \in \mathbb{N}_0$ of a clause-set $F \in \mathcal{CL}S$ is the minimum $k \in \mathbb{N}_0$ with $F \in G_k(U, S)$. 
Algorithm

For input $F_0 \in \mathcal{CLS}$: Is $F_0 \in G^1_k(U, S)$ or $F_0 \in G^0_k(U)$ or $F_0 \notin G_k(U, S)$?

1. $F := F_0$
2. If $k = 0$:
   1. If $F \in S$ then return $F_0 \in G^1_k(U, S)$.
   2. If $F \in U$ then return $F_0 \in G^0_k(U, S)$.
   3. Otherwise return $F_0 \notin G_k(U, S)$.
3. While there is a variable $v \in \text{var}(F)$ and $\varepsilon \in \{0, 1\}$ such that $F' := \langle v \rightarrow \varepsilon \rangle \ast F \in G_{k-1}(U, S)$ holds:
   1. If $F' \in G^1_{k-1}(U, S)$ then return $F_0 \in G^1_k(U, S)$.
   2. $F := F'$.
4. If $F = \top$ then return $F_0 \in G^1_k(U, S)$.
5. If $F = \{ot\}$ then return $F_0 \in G^0_k(U, S)$.
6. Otherwise return $F_0 \notin G_k(U, S)$.

Polynomial time ($O(n^{2k})$) — recognition and SAT decision.
Algorithm

For input $F_0 \in \mathbb{CFS}$: Is $F_0 \in G^1_k(U, S)$ or $F_0 \in G^0_k(U)$ or $F_0 \not\in G_k(U, S)$?

1. $F := F_0$
2. If $k = 0$
   1. If $F \in S$ then return $F_0 \in G^1_k(U, S)$.
   2. If $F \in U$ then return $F_0 \in G^0_k(U, S)$.
   3. Otherwise return $F_0 \not\in G_k(U, S)$.
3. While there is a variable $v \in \text{var}(F)$ and $\varepsilon \in \{0, 1\}$ such that $F' := \langle v \rightarrow \varepsilon \rangle \ast F \in G_{k-1}^1(U, S)$ holds:
   1. If $F' \in G^1_{k-1}(U, S)$ then return $F_0 \in G^1_k(U, S)$.
   2. $F := F'$.
4. If $F = \top$ then return $F_0 \in G^1_k(U, S)$.
5. If $F = \{\bot\}$ then return $F_0 \in G^0_k(U, S)$.
6. Otherwise return $F_0 \not\in G_k(U, S)$.

Polynomial time ($O(n^{2k})$) — recognition and SAT decision.
Algorithm

For input $F_0 \in \mathcal{CLS}$: Is $F_0 \in G^1_k(U, S)$ or $F_0 \in G^0_k(U)$ or $F_0 \notin G_k(U, S)$?

1. $F := F_0$
2. If $k = 0$:
   1. If $F \in S$ then return $F_0 \in G^1_k(U, S)$.
   2. If $F \in U$ then return $F_0 \in G^0_k(U, S)$.
   3. Otherwise return $F_0 \notin G_k(U, S)$.
3. While there is a variable $v \in \text{var}(F)$ and $\varepsilon \in \{0, 1\}$ such that $F' := \langle v \rightarrow \varepsilon \rangle \ast F \in G_{k-1}^1(U, S)$ holds:
   1. If $F' \in G^1_{k-1}(U, S)$ then return $F_0 \in G^1_k(U, S)$.
   2. $F := F'$.
4. If $F = \top$ then return $F_0 \in G^1_k(U, S)$.
5. If $F = \{\bot\}$ then return $F_0 \in G^0_k(U, S)$.
6. Otherwise return $F_0 \notin G_k(U, S)$.

Polynomial time ($O(n^{2k})$) — recognition and SAT decision.
A $U$-tree ($h(F) = 1$)

Figure: Example of $U$-tree for $F \in USAT$ with $h_U(F) = 1$.

At inner nodes the hardness of the subtree is shown.
Another $U$-tree ($h(F) = 2$)

Figure: Example of $U$-tree for $F \in USAT$ with $h_U(F) = 2$. 
(\(U, S\))-tree with \(h(F) = 3\)

Figure: \((U, S)\)-tree for parity function on 3 variables.

Note that the splitting trees for parity functions are independent of the oracles.
Properties

1. Provides hierarchy of clause-sets with polynomial-time satisfiability decision.
2. Yields quasi-automatisation of tree-resolution (with oracles).

Choices of oracle

1. $U$ and $S$ must be closed under *forced assignments*.
2. If $S$ is the set of all clause-sets with *linear autarkies* as satisfying assignments, $S_1$, then many polynomial-time SAT decision classes are in $G_k(U, S_1)$ for some $k$. 
$r_k$-reductions: Generalised UCP

Only considering the forced assignments, not the guesses, in the definition of the $(G_k(U,S))_{k \in \mathbb{N}_0}$ hierarchy:

1. $F \xrightarrow{0,U} \{\bot\}$ if $F \in U$

2. $F \xrightarrow{k+1,U} \langle v \rightarrow \varepsilon \rangle \ast F$ if there is $(v, \varepsilon) \in \text{var}(F) \times \{0, 1\}$ with

   \[ \langle v \rightarrow \varepsilon \rangle \ast F \in G^0_k(U). \]

$F \xrightarrow{k,U} \ast F'$ is the reflexive-transitive closure of $\xrightarrow{k,U}$.

All reductions $r_k$ are confluent:

\[ r^U_k(F) := F' \]

where $F \xrightarrow{k,U} \ast F'$, and there is no $F'' \neq F'$ with $F' \xrightarrow{k,U} \ast F''$.

$r_1$ is unit-clause propagation.
A new hardness for satisfiable clause-sets

Deductions using a reduction
Consider a reduction $r$. The relation $F \vdash_r C$ holds for a clause-set $F$ and a clause $C$, and we say $C$ is deducible from $F$ via $r$, if $r$ discovers unsatisfiability of $\langle x \mapsto 0 : x \in C \rangle * F$, that is, $\bot \in r(\langle x \mapsto 0 : x \in C \rangle * F)$.

Hardness of clause-sets
The hardness $h(F) \in \mathbb{N}_0$ for clause-set $F \in \mathcal{CLS}$ is the minimal $k \in \mathbb{N}_0$ such that for all clauses $C$ with $F \models C$ we have $F \vdash_{r_k} C$. 

Discussion

Key points

1. For unsatisfiable $F$ we have $h(F) = h_{U_0}(F)$.
2. Considers whole boolean function via the implicates.
3. Different for satisfiable $F$.

Upper and lower bounds

1. $F \in \mathcal{CLS}$ is $k$-soft if $h(F) \leq k$ (upper bound).
2. $F \in \mathcal{CLS}$ is $k$-hard if $h(F) \geq k$ (lower bound).
Finding satisfying assignments for $k$-soft $F$

Having a $k$-soft $F$ means that satisfiability of $\varphi \ast F$ for any partial assignment $\varphi$ can be decided in polynomial time, and we can also find a satisfying assignment as follows:

1. Let $F' := \varphi \ast F$.
2. If $r_k(F') = \{\bot\}$ then $F'$ is unsatisfiable.
3. Otherwise we know that $F'$ is satisfiable.
4. A satisfying assignment for $F'$ is found as follows (by self-reduction):
   1. Pick any variable $v \in \text{var}(F')$ and $\varepsilon \in \{0, 1\}$.
   2. If $r_k(\langle v \rightarrow \varepsilon \rangle \ast F') = \{\bot\}$ then apply $v \rightarrow \overline{\varepsilon}$.
   3. Otherwise apply $v \rightarrow \varepsilon$.
   4. Repeat this process.
Some properties

1. \( h(F) = 0 \) iff \( F \) contains all its prime implicates.
2. If \( F \) is a renamable Horn clause-set then \( h(F) \leq 1 \).
3. If \( F \) is in 2-CNF then \( h(F) \leq 2 \).
4. If \( F \subseteq F' \) and \( F' \) is equivalent to \( F \), then \( h(F') \leq h(F) \).
The SAT representation hypothesis

“SAT representation hypothesis”
Finding a “good” representation of a boolean function $f$, for deciding satisfiability in polynomial time, is captured by constructing a $k$-soft representation for $f$ for some small $k$.

A representation $F$ of $f$ is a clause-set $F$ with $\text{var}(f) \subseteq \text{var}(F)$ such that restricting the satisfying assignments of $F$ to $\text{var}(f)$ we get exactly the satisfying assignments of $f$.

Remarks
- Lower $k =$ lower runtime but larger representation $F$.
- If $f$ is only some part of a bigger boolean function, then $f$ should be made as large as possible.

In both cases, a balance must be sought!
How to construct a $k$-soft representation?

Consider a boolean function $f : \{0, 1\}^V \rightarrow \{0, 1\}$.

How to construct a $k$-soft representation $F$ of $f$ for some $k$?
Here are the basic possibilities:

- Take the set of prime implicates $F = \text{prc}_0(f)$ (0-soft).
- Compute some $k$-soft representation $F \subseteq \text{prc}_0(f)$.
- Introduce a representation $f'$ of $f$, using new variables, and use a $k$-soft representation of $f'$.
- Decompose $f$ into boolean functions $\{f_1, \ldots, f_m\}$ for $m \in \mathbb{N}$, and find $k_i$-soft representations for the $f_i$. 
A central question here is to understand the hardness of compositions of boolean functions.

- If we are lucky, then the functions combine “well”, and the overall hardness will not increase.
- However the overall hardness can also explode (becoming essentially the number of variables), when the functions “miss each other”.
Two 1-soft representations of boolean functions

1-bases

- Computational 1-soft representation \textit{without} new variables.
- Compute by:
  1. Starting with the prime implicates.
  2. Iteratively removing clauses.
  3. Check all prime implicates still follow by $r_1$ after each removal.
- Use the smallest such clause-set found.

Canonical translation

- General 1-soft representation \textit{with} new variables.
- Take the union of all prime implicates for each direct sub-formula of the following boolean function:

\[
\left( \bigwedge_{C \in \text{DNF}(f)} \text{vct}_f(C) \leftrightarrow \bigvee x \right) \land \bigvee_{x \in C} \text{vct}_f(C) \land \bigvee_{C \in \text{DNF}(f)} \text{vct}_f(C).
\]
The \{Advanced, Data\} encryption standard

We consider the \textbf{Advanced Encryption Standard} and \textbf{Data Encryption Standard} as examples.

Translation

- Fix boolean function / variant of the cipher.
- Decompose boolean function into small (8 or 16 bit) boolean functions using new variables.
  Additions, Sboxes and multiplications\(^a\).
- Translate small boolean functions using 1-based representations.

\(^a\)AES only

Central here (as usual) the relational point-of-view:

Treat boolean functions with \(n\) inputs and \(m\) outputs as a boolean function (with one output) in \(n + m\) variables.
Within the OKlibrary

http://www.ok-sat-library.org

we provide a general open-source framework for translating AES, generalised AES and DES to SAT, in many forms.

See [3] for an article explaining the general philosophy of the OKlibrary.

For an early approach to attack DES via SAT see [4].
Experimental results

Advanced Encryption Standard

1. Key discovery for small cases of small-scale variations of AES follows using only 1 or 2 decisions, compared with thousands for other representations.

2. For non-trivial instances 1-soft representations perform better compared to “small” representations (but more data needed - large space of variations).

Data Encryption Standard

1. Compared to known benchmarks, in the same time we find the key with around 3 - 4 fewer given key bits.

For some look-ahead solvers (such as the OKsolver), 1-soft representations can be the difference between not solving in 10 hours and solving in less than a minute.
Further Work

Composition  When can we compose boolean functions with $k$-soft representations and get a $k$-soft representation?

Extended resolution  New variables help! How? How does this relate to the complexity of extended resolution?


AES and DES  Finding good decompositions of AES and DES.
References I

Oliver Kullmann.
Investigating a general hierarchy of polynomially decidable classes of CNF’s based on short tree-like resolution proofs.

Oliver Kullmann.
Upper and lower bounds on the complexity of generalised resolution and generalised constraint satisfaction problems.
On the hardness of (satisfiable) conjunctive normal forms

Matthew Gwynne (Swansea)

Hardness based on tree resolution

New hardness notion for SAT

The SAT representation hypothesis

Attacking AES and DES

Future work

References II

Oliver Kullmann.
The OKlibrary: Introducing a "holistic" research platform for (generalised) SAT solving.

Fabio Massacci and Laura Marraro.
Logical cryptoanalysis as a SAT problem.
ISBN 1 58603 061 2.
On the hardness of (satisfiable) conjunctive normal forms

Matthew Gwynne (Swansea)

Hardness based on tree resolution

New hardness notion for SAT

The SAT representation hypothesis

Attacking AES and DES

Future work