How to translate into SAT such that SAT solvers have a good time?! 

Oliver Kullmann
Computer Science Department
Swansea University

Fourth Workshop on Formal and Automated Theorem Proving and Applications
Belgrade, February 5, 2011
Solving hard “combinatorial” problems via SAT

- CNF-SAT solvers work relatively well.
- We believe not only in the beauty, but also in the power and usefulness of CNF.
- I consider the question of translating problems into CNF such that SAT solvers can succeed.
- Our focus is on intrinsically hard problems.
Two dimensions

The basic dimensions I am considering in this talk are:

1. The problem instance is already given naturally in some form of *non-boolean* CNF, and the task is to make a *boolean* CNF out of it. The fundamental problem here is that of translating non-boolean values into boolean values.

2. The problem instance is given in the form of the boolean combination of various boolean *black boxes* (i.e., as a generalised circuit, allowing arbitrary gates), and we have to “flatten” the boxes to CNF. The fundamental problem here is that of presenting complex computations via CNFs.
How to translate into SAT such that SAT solvers have a good time?!

Oliver Kullmann

Outline

1. Introduction
2. The generic boolean translation
3. Attacking AES via SAT
4. Towards a general theory of good translations
Non-boolean clause-sets

The “true” generalisation of boolean CNF to non-boolean CNF seems to be the following:

1. variables $v$ have (finite) domains $D_v$
2. literals are of the form “$v \neq \varepsilon$” for some $\varepsilon \in D_v$;
3. these clauses are called “no-goods” in constraint solving.

For a systematic investigation see [Kullmann, 2009, Kullmann, 2011a, Kullmann, 2011b].

With these non-boolean clause-sets for example hypergraph colouring problems and Ramsey-type problems now have a canonical representation.
The general idea of the “generic translation”

Consider a variable \( v \) with domain \( D_v = \{ \varepsilon_1, \ldots, \varepsilon_m \} \).

- So there are \( m \) literals, namely \((v, \varepsilon_1), \ldots, (v, \varepsilon_m)\).
- And for assignment \( \langle v \rightarrow \varepsilon_i \rangle \) exactly \( m - 1 \) of these literals become true, while \((v, \varepsilon_i)\) becomes false.
- It wouldn’t matter w.r.t. satisfiability if it would be possible to set more than one literal to false.

The idea now is to represent these literals by clauses from a clause-set \( F_v \).

- We need to choose \( m \) clauses \( C_1, \ldots, C_m \in F_v \).
- Since we must not be able to make all literals to true, \( F_v \) must be unsatisfiable.
- We demand all clauses \( C_i \) to be necessary for \( F_v \), that is, removal renders \( F_v \) satisfiable — in this way we model that all other literals become true.
More details

The *generic boolean translation* $F \rightsquigarrow T(F)$ for a non-boolean clause-set $F$ is as follows (using $m := |D_v|$):

- For each variable $v$, choose unsatisfiable variable-disjoint boolean clause-sets $F_v$ with at least $m$ clauses.
- Choose different clauses $C_1, \ldots, C_m \in F_v$.
- Literals “$v \neq \varepsilon_i$” are replaced by the clauses $C_i$.
- The “remainder clauses” in $R_v := F_v \setminus \{C_1, \ldots, C_m\}$ are all added to the translation.

Note that

$$n(T(F)) = \sum_{v \in \operatorname{var}(F)} n(F_v)$$

$$c(T(F)) = c(F) + \sum_{v \in \operatorname{var}(F)} c(R_v).$$
How to translate into SAT such that SAT solvers have a good time?!

Oliver Kullmann

Introduction

The generic boolean translation

Attacking AES via SAT

Towards a general theory of good translations

Example: The direct translations

Here we choose

\[ F_v = \{ \{ v_1 \}, \ldots, \{ v_m \}, \{ \overline{v}_1, \ldots, \overline{v}_m \} \}, \]

and we choose the unit-clauses to correspond to the values.

1. For the weak form (using only ALO-clauses) that's it (so we have one remainder clause).

2. For the strong form we add all positive binary clauses to (the remainder of) \( F_v \) (so obtaining the AMO-clauses).
Example: The simple logarithmic translation

- If \( m = 2^p \), then choose the (minimally) unsatisfiable clause-set \( F_v \) with \( p \) variables and \( 2^p \) clauses (which are all the full clauses using all variables).

- If \( m \) is not a power of two, then for the simple case just use the smallest \( p \) with \( m < 2^p \), use the same \( F_v \), and choose \( m \) of these clauses (the remaining clauses become remainder-clauses).
The weak nested translation

Here we use $p := m - 1$ (boolean) variables $v_1, \ldots, v_p$ and

$$F_v = \{ \{ v_1 \}, \{ \overline{v}_1, v_2 \}, \ldots, \{ v_1, \ldots, \overline{v}_{p-1}, v_p \}, \{ \overline{v}_1, \ldots, \overline{v}_p \} \}.$$ 

There are no remainder clauses.
Yet we tested these (and other, related) translations only on Green-Tao instances ([Kullmann, 2010]), but this we did rather extensively.

**Big surprise:**

For “large” $m$ the logarithmic translation was best, and for all other $m$ the weak nested translation — for all solver types.

“Best” often means by orders of magnitudes.
Attacking AES

- AES (“Advanced Encryption Standard”) is the successor of DES.
- AES is a “block cipher”, a basic cryptographic building block.

AES is a map

$$AES : \{0, 1\}^{128} \times \{0, 1\}^{128} \rightarrow \{0, 1\}^{128}$$

such that for every key $k \in \{0, 1\}^{128}$ the map $AES(\cdot, k) : \{0, 1\}^{128} \rightarrow \{0, 1\}^{128}$ is a permutation.

1. Given only a message $m \in \{0, 1\}^{128}$ and its encryption $AES(m, k)$, it should be hard to find a key $k' \in \mathbb{K}$ with $AES(m, k') = AES(m, k)$.
2. We attack precisely this.
AES clause-sets

The basic task is to construct

\[ F_{\text{AES}} \text{ in } 3 \cdot 128 = 384 \text{ variables} \]

representing the AES-relation.

After substituting \(2 \cdot 128 = 256\) (boolean) values for plain text \(m\) and cipher text \(\text{AES}(m, k)\), the satisfying assignments of the resulting clause-set

\[ (\varphi_m \cup \varphi_{\text{AES}(m,k)}) \ast F_{\text{AES}} \]

in 128 variables are exactly the possible keys \(k\).

By “\(\varphi\)” we typically denote partial (boolean) assignments, while by \(\varphi \ast F\) for a clause-set \(F\) we denote the result of applying \(\varphi\) to \(F\).
The basic structure of a block cipher

Let
- \( \mathbb{M} \) be the set of “messages”
- \( \mathbb{K} \) be the set of “keys”.

A block cipher is a map

\[
f : \mathbb{M} \times \mathbb{K} \rightarrow \mathbb{M}
\]

such that for each fixed key \( k \in \mathbb{K} \) the map

\[
m \in \mathbb{M} \mapsto f(m, k) \in \mathbb{M}
\]

is a bijection.
The basic structure of an iterated block cipher

The computation of \( f \) proceeds in rounds, so instead of \( f(m, k) \) we write \( f_p(m, k) \), using the round parameter \( p \in \{0, \ldots, N\} \) with

\[
f_0(m, k) = m, \quad f_N(m, k) = f(m, k).
\]

For simplicity from now on we assume \( \mathbb{M} = \mathbb{K} = \{0, 1\}^n \). The recursive equation now is

\[
f_{p+1}(m, k) = R(f_p(m, k) + k_p)
\]

where

- \( R : \mathbb{M} \to \mathbb{M} \) is the “round bijection”
- \( k_p \) is given by the “key schedule”:
  1. \( k_0 := k \)
  2. \( k_{p+1} = S(k_p) \)

for the “key bijection” \( S : \mathbb{M} \to \mathbb{M} \).
Patching up boolean functions

For AES, the round bijection and the key bijection are defined in terms of “boxes”, which are certain permutations

\[ S : \{0, 1\}^8 \rightarrow \{0, 1\}^8. \]

These boxes yield boolean functions in 16 variables,

- which are represented by clause-sets (using possibly additional (different) variables),
- and which are just put together, yielding \( F_{AES} \).
Representations of boolean functions

A clause-set $F$, understood as CNF, represents a boolean function $f : \{0, 1\}^V \rightarrow \{0, 1\}$ if

- $V \subseteq \text{var}(F)$, and
- the set of satisfying total assignments of $F$, projected to $V$, is exactly the set of boolean vectors $x : V \rightarrow \{0, 1\}$ with $f(x) = 1$.

A representation $F$ for $f$ has the unique extension property if

for every $x : V \rightarrow \{0, 1\}$ with $f(x) = 1$ there is (only) exactly one assignment $\varphi : \text{var}(F) \rightarrow \{0, 1\}$ with $\varphi \ast F = \top$. 
For clause-sets $F, F'$ the relation $F \supseteq \mapsto F'$ holds if for all $C \in F$ there is $C' \in F'$ with $C' \subseteq C$; we say that $F'$ strengthens $F$.

A reduction in this context is a map $r : \mathcal{CLS} \rightarrow \mathcal{CLS}$ such that for all $F, F' \in \mathcal{CLS}$ we have

1. $r(F)$ is satisfiability-equivalent to $F$;
2. if $\bot \in r(F)$ and $F'$ strengthens $F$ then $\bot \in r(F')$.

A reduction $r$ discovers unsatisfiability of $F$ if $\bot \in r(F)$. 
Generalised unit-clause propagation

In [Kullmann, 1999, Kullmann, 2004] a hierarchy of reductions $r_k$ has been studied, given by

$$\begin{align*}
r_0(F) &:= \begin{cases} 
\{\bot\} & \text{if } \bot \in F \\
F & \text{else}
\end{cases} \\
r_{k+1}(F) &:= \begin{cases} 
\langle v \rightarrow \varepsilon \rangle \ast F & \text{if } \exists v \in \text{var}(F), \varepsilon \in \{0, 1\} : r_k(\langle v \rightarrow \varepsilon \rangle \ast F) = \{\bot\} \\
F & \text{else}
\end{cases}
\end{align*}$$

- $r_1$ is unit-clause propagation.
- $r_2$ is (complete) elimination of “failed literals”.
- Solving SAT by applying $r_0, r_1, r_2, \ldots$ is the true core of the (infamous) Stalmarck method.
Restricted deduction power

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 $\varphi_C \ast F$ (that is, $\bot \in r(\varphi_C \ast F)$ for $\varphi_C = \langle x \mapsto 0 : x \in C \rangle$).
Consider a reduction $r : \mathcal{LS} \rightarrow \mathcal{LS}$.

- A clause-set $F$ is $r$-generated if for all clauses $C$ with $F \models C$ we have $F \vdash_r C$.

- More generally, a clause-set $F$ is $r$-generating for a boolean function $f$ if $F$ represents $f$, and if for all clauses $C$ with $f \models C$ we have $F \vdash_r C$.

- $F$ is $r$-generated iff $F$ is $r$-generating for the CNF $F$.

- $F$ is an $r$-base for $f$ if $F$ is minimally $r$-generating for $f$ w.r.t. elimination of clauses and literals.

- $F$ is $r$-based if $F$ is an $r$-base for $F$. 
How to translate into SAT such that SAT solvers have a good time?!

Oliver Kullmann

Introduction

The generic boolean translation

Attacking AES via SAT

Towards a general theory of good translations

The SAT Representation Hypothesis

The “SRH” is the (not fully specified) statement that the task of a

“good” representation
of a boolean function $f$ or a clause-set $F_0$,

for the purpose of SAT solving or of refuting $F_0$, both in polynomial time, is fully captured by

finding an $r_k$-generating clause-set $F$
for $f$ resp. $F_0$ for some $k$. 
The smaller $k$ the lower the exponent for the polynomial in the run-time estimation, but the larger $F$ is, so a balance is to be sought.

If $f$ is only some part of a bigger function (like for example the S-box in AES), then $f$ should be made as large as possible (again, a balance is to be sought).

The SRH states that the whole business of Extended Resolution and its various uses is to construct for a given clause-set $F$ some $r_k$-base for appropriate $k \geq 1$ (while the construction of an $r_0$-base is too expensive).
How to translate into SAT such that SAT solvers have a good time?! 

Oliver Kullmann

Introduction

The generic boolean translation

Attacking AES via SAT

Towards a general theory of good translations

Outlook

I The generic translation offers the possibility to translate each variable individually — for that we need to really understand what’s going on.

II Attacking AES, we are currently investigating various kinds of decompositions of the AES-computation, the various “boxes” resulting, and their effect on SAT solving.

III Regarding SRH, likely one can prove various generalities.
Bibliography I

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

Upper and lower bounds on the complexity of generalised resolution and generalised constraint satisfaction problems.
How to translate into SAT such that SAT solvers have a good time?!

Oliver Kullmann

Introduction

The generic boolean translation

Attacking AES via SAT

Towards a general theory of good translations

---

**Bibliography II**


Constraint satisfaction problems in clausal form: Autarkies and minimal unsatisfiability.

Technical Report TR 07-055, version 02, Electronic Colloquium on Computational Complexity (ECCC).


Green-Tao numbers and SAT.


ISBN-13 978-3-642-14185-0.


Constraint satisfaction problems in clausal form I: Autarkies and deficiency.

*Fundamenta Informaticae*, 109(1):27–81.
How to translate into SAT such that SAT solvers have a good time?!

Oliver Kullmann

Introduction

The generic boolean translation

Attacking AES via SAT

Towards a general theory of good translations