# Towards a theory of good SAT representations

Matthew Gwynne and Oliver Kullmann

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

Bath, March 26, 2013, BCTCS 2013

We want to speed up SAT solving via
"SAT knowledge compilation".

Our main tools are
unit-clause propagation and generalisations.

# Outline

# Translating to SAT

1. Formulate problem in terms of a set $P$ of constraints.
2. **Encode** each constraint $c$ as a boolean function $f_c$.
3. **Translate** each boolean function to a CNF clause-set $F_{f_c}$.
4. Use **SAT solver** to determine satisfiability for $F := \bigcup_{c \in P} F_{f_c}$.

We focus on **step 3**: translating boolean functions to CNF.

# Preliminaries: clause-sets

We consider **clause-sets** as **C**onjunctive **N**ormal **F**orm (CNF) formulas, e.g.,

$$F = \underbrace{\{\{a, b\}, \{\overline{b}, c\}, \{\overline{a}, \overline{c}\}\}}_{\textbf{clause-set}} \quad \sim \quad \underbrace{(a \vee b) \wedge (\neg b \vee c) \wedge (\neg a \vee \neg c)}_{\textbf{CNF}}.$$

- A literal is a (boolean) variable or its negation.
- A clause is a set of literals (representing a disjunction).
- A clause-set is a set of clauses (representing a conjunction of clauses).

The two most fundamental clause-sets are

- $\top := \emptyset$ (empty conjunction – basic satisfiable clause-set)
- $\{\bot\}$ where $\bot := \emptyset$ is the empty clause (empty disjunction – basic unsatisfiable clause-set).

# Preliminaries: partial assignments

Fundamental to satisfiability is the concept of **partial assignment**:

$$\langle b \to 1 \rangle * \{ \{\overline{a}, b\}, \{\overline{b}, c\}, \{\overline{a}, c\} \} = \{ \{c\}, \{\overline{a}, c\} \}$$

- Setting a literal to true removes all clauses containing it.
- Setting a literal to false removes the literal from all clauses.

We say that a clause-set $F$ is

- **satisfiable** if there is a partial assignment $\varphi$ s.t $\varphi * F = \top$ (i.e., we set at least one literal in every clause to true);
- **unsatisfiable** if for all total assignments $\varphi$ we have $\bot \in \varphi * F$.

The classes of all sat resp. unsat clause-sets are $\mathcal{SAT}$ resp. $\mathcal{USAT}$.

# Hard unsatisfiable sub-instances

- Typically we solve SAT problems by building up partial assignments $\varphi$ and backtracking when we detect $\bot$.
- Ideally, we want to backtrack whenever $\varphi * F \in \mathcal{USAT}$, but in general detecting this is a coNP-complete problem.

**Instead**, we consider translations of boolean functions $f_c$ to clause-sets $F_{f_c}$ such that

for all partial assignments $\varphi$ we at least have that $\varphi * F_{f_c} \in \mathcal{USAT}$ is "easily" checked for all constraints $c$.

# Preliminaries: unit 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, obvious) **forced assignments**.
- Possible in linear time, and is confluent.
- Using the notation $\mathbf{r_1}$ for UCP we have

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

# Good representations

Can we construct clause-set translations $F$
such that for all partial assignments $\varphi$
if $\varphi * F \in \mathcal{USAT}$ then $\underbrace{\mathrm{r}_1(\varphi * F) = \{\bot\}}_{\textbf{UCP detects unsatisfiability}}$ ?

# $\mathcal{UC}$

In 1994, del Val [5] introduced the class of **unit-refutation complete** clause-sets:

A clause-set $F$ is unit-refutation complete iff
for all partial assignments $\varphi$ such that $\varphi * F$ is unsatisfiable,
$\bot$ follows via *unit-resolution* (i.e., $\bot \in \mathrm{r}_1(\varphi * F)$).

The set of all unit-refutation complete clause-sets is denoted by $\mathcal{UC}$.

In $\mathcal{UC}$ we can decide the **clausal entailment** problem via UCP.

# $\mathcal{SLUR}$

In 1995, Schlipf, Annexstein, Franco, and Swaminathan [14] introduced the SLUR (Single Lookahead Unit Resolution) algorithm.

- It was known for certain classes (Horn, renamable Horn, balanced clause-sets, ...) of *unsatisfiable* clause-sets that UCP was sufficient to prove unsatisfiability.
- The key insight in [14] was that there is a simple algorithm (SLUR) for deciding satisfiability for these classes in general — *without needing to know that the clause-sets is in this class*.
- $\mathcal{SLUR}$ is the class of clause-sets where this incomplete non-deterministic algorithms always succeeds.

Čepek, Kučera, and Vlček [4] show that *deciding membership* for the class $\mathcal{SLUR}$ is coNP-complete.

# SLUR algorithm

The non-deterministic SLUR-algorithm:

1. Take as input a clause-set $F$.
2. Compute $F := \mathrm{r}_1(F)$.
3. If $F = \{\bot\}$ then return UNSATISFIABLE.
4. While $F \neq \top$:
   1. Non-deterministically choose a literal $x$ such that $\mathrm{r}_1(\langle x \to 1\rangle * F) \neq \{\bot\}$ and set $F := \mathrm{r}_1(\langle x \to 1\rangle * F)$.
   2. If no such $x$ exists then return GIVE-UP.
5. Return SATISFIABLE.

This is possible in linear time as described by Franco and Gelder [6].

The $\mathcal{SLUR}$ class is the class of clause-sets for which the above (non-deterministic) algorithm never returns GIVE-UP.

# $\mathcal{SLUR} = \mathcal{UC}$

A fundamental insight ([9]; hasn't been realised until now!):

$$\mathcal{SLUR} \quad = \quad \mathcal{UC}.$$

This provides both an *algorithmic* and *semantic* perspective, and allows results and intuitions from both classes to be combined.

For example, Čepek et al. [4] show that deciding "$F \in \mathcal{SLUR}$?" is coNP-complete, and we now have this also for "$F \in \mathcal{UC}$?".

$\mathcal{SLUR}$ captures various classes of clause-set with poly-time SAT algorithms.

Can we capture more,
by generalising these classes?

# Generalised unit-clause propagation

Kullmann [12] introduced the notion of

**generalised unit-clause propagation**.

For $k \in \mathbb{N}_0$:

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

$$\mathrm{r}_1(F) = \begin{cases} \mathrm{r}_1(\langle x \to 1\rangle * F) & \text{if } \exists\, x \in \mathrm{lit}(F) : \mathrm{r}_0(\langle x \to 0\rangle * F) = \{\bot\} \\ F & \text{otherwise} \end{cases}$$

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

Rather than linear-time, this is now possible in time $n^{2k-1}$.

# Example: $\mathsf{r}_2$ is more powerful $\mathsf{r}_1$

Consider

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

We have that

1. $\mathsf{r}_1(F) = F$ (UCP does nothing).
2. $\mathsf{r}_2(F) = \mathsf{r}_2(\langle a \to 1\rangle * F) = \{\bot\}$, since

$$\mathsf{r}_1(\langle a \to 0\rangle * F) = \mathsf{r}_1(\{\, \{b\}, \{\overline{b}\} \,\}) = \{\bot\}.$$

# $\mathcal{UC}_k$ and $\mathcal{SLUR}_k$

By generalising $r_1$ to $r_k$ we allow more powerful inference methods at the expense of increasing time-complexity.

Definitions (for $k \in \mathbb{N}_0$):

1. $\boldsymbol{\mathcal{UC}_k}$ is the set of clause-sets $F$ such that under any partial assignment $\varphi$ for which $\varphi * F$ is unsatisfiable we have that $\bot \in r_k(\varphi * F)$.
2. $\boldsymbol{\mathcal{SLUR}_k}$ is the set of clause-sets $F$ such that either
   - $F$ is unsatisfiable and $r_k(F) = \{\bot\}$, or
   - making non-deterministic choices using lookahead + $r_k$ *always* eventually yields $\top$.

In the limit $\mathcal{UC}_k$ and $\mathcal{SLUR}_k$ cover all clause-sets.

# Results

We show

$$\mathcal{SLUR}_k = \mathcal{UC}_k.$$

Again, yielding both *algorithmic* and *semantic* perspectives.

- By "pumping up" the result from $\mathcal{SLUR}$ we get that deciding membership for $\mathcal{SLUR}_k$ resp. $\mathcal{UC}_k$ is coNP-complete.
- Many existing classes and hierarchies for poly-time satisfiability are subsumed as levels of $\mathcal{UC}_k$ (Horn, 2–$\mathcal{CLS}$, and numerous generalisations ([11, 15, 3, 4, 1]) of these classes (see [9, 8]))
- Strong connections to resolution complexity, e.g., if clause-set $F \in \mathcal{UC}_k \setminus \mathcal{UC}_{k-1}$ is unsatisfiable then

$$2^k \leq \underbrace{\mathrm{Comp}^*_{\mathrm{R}}(F)}_{\textbf{tree resolution complexity}} \leq (n(F)+1)^k.$$

# $\mathcal{UC}_k$: a representation hierarchy without new variables

We want to represent boolean functions $f$ with clause-sets $F$ in $\mathcal{UC}_k$.

Without new variables (i.e, $F \cong f$, "equivalent representations"):

- Every function has a representation in $\mathcal{UC}_0$ (its "prime implicates").
- But in terms of poly-size representability, the hierarchy is **strict**.
- That is, for all $k \geq 0$ we get that

  there are sequences $(F_n)_{n \in \mathbb{N}_0}$ such that all $F_n \in \mathcal{UC}_{k+1}$,
  but all equivalent poly-size $(F'_n)_{n \in \mathbb{N}_0}$ have some $F'_n \notin \mathcal{UC}_k$.

  See Gwynne and Kullmann [10] for details.

Equivalent representations are important, as in general this is the only case where we can **search** for and **optimise** representations.

# $\mathcal{UC}_k$: a representation hierarchy with new variables?

With new variables ($\varphi * f = 1 \Leftrightarrow \varphi * F \in \mathcal{SAT}$):

- If $f$ has a poly-size DNF, then there is a poly-size representation in $\mathcal{UC}_1$ via a variant of the Tseitin translation.
- However, there are boolean functions with small CNFs but only exponential size DNFs.
- We conjecture the strictness of $\mathcal{UC}_k$ when allowing new variables.

# Outlook: SAT knowledge compilation and $\mathcal{UC}_k$

We consider the following as natural developments from $\mathcal{UC}_k$:

- An interleaving hierarchy based on ability to detect forced assignments (“**propagation complete**” clause-sets, see Bordeaux and Marques-Silva [2] and [7]).
- A hierarchy based on **width-restricted *full*-resolution** (see [7]).
- Generalised hierarchies based on extending $\mathcal{UC}_k$ via unsatisfiability **oracles** (see [12, 13, 7]).
- **Optimisation** of the size of $\mathcal{UC}_k$ representations (we show NP-completeness in [7]).

All hierarchies above offer target classes for **SAT knowledge compilation** —

“compiling” knowledge into the clause-sets
to make SAT-solving easier.

# End

- Conference version is [9].
- Underlying report is [7].
- Journal version is [8].

(references on the remaining slides).

# Bibliography I

[1] Tomáš Balyo, Štefan Gurský, Petr Kučera, and Václav Vlček. On hierarchies over the SLUR class. In *Twelfth International Symposium on Artificial Intelligence and Mathematics (ISAIM 2012)*, January 2012. Available at http://www.cs.uic.edu/bin/view/Isaim2012/AcceptedPapers.

[2] 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.

[3] Ondřej Čepek and Petr Kučera. Known and new classes of generalized Horn formulae with polynomial recognition and SAT testing. *Discrete Applied Mathematics*, 149:14–52, 2005.

# Bibliography II

[4] 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.

[5] 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.

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

# Bibliography III

[7] Matthew Gwynne and Oliver Kullmann. Generalising unit-refutation completeness and SLUR via nested input resolution. Technical Report arXiv:1204.6529v5 [cs.LO], arXiv, January 2013.

[8] Matthew Gwynne and Oliver Kullmann. Generalising unit-refutation completeness and SLUR via nested input resolution. *Journal of Automated Reasoning*, 2013. doi: 10.1007/s10817-013-9275-8. To appear.

[9] 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.

# Bibliography IV

[10] Matthew Gwynne and Oliver Kullmann. Towards a theory of good SAT representations. Technical Report arXiv:1302.4421v3 [cs.AI], arXiv, March 2013.

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

[12] 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.

[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] 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.

[15] Hans van Maaren. A short note on some tractable cases of the satisfiability problem. *Information and Computation*, 158(2): 125–130, May 2000.