Unifying hierarchies of fast SAT decision and knowledge compilation

Matthew Gwynne and Oliver Kullmann

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

Špindlerův Mlýn, January 28, 2013, SOFSEM 2013

We want to speed up SAT solving via
“SAT knowledge compilation”.

Our main tools are
unit-clause propagation and generalisations.

Outline

Outline

Outline

1. Introduction
2. $\mathcal{UC} + \mathcal{SLUR}$
3. $\mathcal{UC}_k$ and $\mathcal{SLUR}_k$

Outline

1. Introduction
2. $\mathcal{UC}$ + $\mathcal{SLUR}$
3. $\mathcal{UC}_k$ and $\mathcal{SLUR}_k$
4. Properties of $\mathcal{UC}_k$ and $\mathcal{SLUR}_k$

Outline

Preliminaries: clause-sets

We will consider **clause-sets**, e.g.,

$$F = \{ \{a, b\}, \{\overline{b}, c\}, \{\overline{a}, \overline{c}\} \}$$

as **C**onjunctive **N**ormal **F**orm (CNF) formulas, e.g.,

$$F \sim (a \vee b) \wedge (\neg b \vee c) \wedge (\neg a \vee \neg c).$$

Two fundamental clause-sets are

- $\top := \emptyset$ (empty conjunction – basic satisfiable clause-set)
- $\{\bot\} := \{\emptyset\}$ (conjunction with 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 φ s.t $\varphi * F = \top$ (i.e., we set at least one literal in every clause to true);
- **unsatisfiable** if for all total assignments φ we have $\bot \in \varphi * F$.

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

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.
- Using the notation $\mathbf{r_1}$ for UCP we have

$$r_1(F) := \begin{cases} \{\bot\} & \text{if } \bot \in F \\ r_1(\langle x \to 1 \rangle * F) & \text{if } \exists x \in \operatorname{lit}(F) : \bot \in \langle x \to 0 \rangle * F \\ F & \text{otherwise} \end{cases}.$$

$\mathcal{UC}$

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 assignment φ such that $\varphi * F$ is unsatisfiable
we have $\bot \in r_1(\varphi * F)$.

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

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

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. 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 (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 "$F \in \mathcal{UC}$?".

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

$\boldsymbol{\mathcal{UC}} \subseteq \boldsymbol{\mathcal{SLUR}}$:

1. SLUR probes ahead using r_1 and GIVES UP if it detects that it ended up in an unsatisfiable branch.
2. For $F \in \mathcal{UC}$ it never GIVES UP, as r_1 is sufficient to detect any unsatisfiable branch.

$\boldsymbol{\mathcal{SLUR}} \subseteq \boldsymbol{\mathcal{UC}}$ (we show $F \notin \mathcal{UC} \implies F \notin \mathcal{SLUR}$):

1. If $F \notin \mathcal{UC}$ then there is some φ s.t $\varphi * F$ is unsatisfiable but $\mathrm{r}_1(\varphi * F) \neq \{\bot\}$.
2. Via *transitions* of the $\mathcal{SLUR}$ algorithm, one can reach $\mathrm{r}_1(\varphi * F)$.
3. But then $\mathcal{SLUR}$ will GIVE UP later.

$\mathcal{SLUR}$ 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 $k \in \mathbb{N}_0$:

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

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

Example: r_2 is more powerful r_1

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

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

We actually obtain a strict hierarchy:

- We can use exponentially smaller clause-sets while maintaining the ability to decide clausal entailment in poly-time.
- See Gwynne and Kullmann [10] for a proof of this.

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

Results

We show

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

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.
- We show the following inclusion properties:
  - $\mathcal{HO} \subset \mathcal{UC}_1$ (more generally, $\mathcal{RHO} \subset \mathcal{UC}_1$).
  - 2–$\mathcal{CLS} \subset \mathcal{UC}_2$.
  - $\mathcal{QHO} \subset \mathcal{UC}_2$.
  - $\mathcal{HO}_k \subset \mathcal{UC}_k$ (generalised Horn clause-sets Kleine Büning [11]).
  - $\Pi_k \subset \mathcal{UC}_{k+1}$ and $\Upsilon_k \subset \mathcal{UC}_{k+2}$ (Čepek and Kučera [3]).

Existing hierarchies

$\mathcal{SLUR}$-based hierarchies:

1. $\mathcal{SLUR}(k)$ strengthens SLUR by the ability to choose k variables at once, rather than just 1 (see Čepek et al. [4], Balyo, Štefan Gurský, Kučera, and Vlček [1]).
2. $\mathcal{SLUR}*(k)$ strengthens $\mathcal{SLUR}(k)$ by interleaving r_1 with every choice (see Čepek et al. [4]).

Hierarchies based on restricted resolution:

- $\mathrm{CANON}(k)$ is the class of clause-sets for which all implied clauses are derivable by a resolution tree of height at most k.

In general, we have that:

1. $\mathcal{SLUR}(k) \subset \mathcal{SLUR}*(k) \subset \mathcal{SLUR}_{k+1} = \mathcal{UC}_{k+1}$.
2. $\mathrm{CANON}(k) \subset \mathcal{UC}_k = \mathcal{SLUR}_k$.
3. $\mathrm{CANON}(2) \not\subseteq \mathcal{SLUR}*(k)$ for every $k \in \mathbb{N}_0$.

Generalised input resolution

The strict inclusion $\mathrm{CANON}_k \subset \mathcal{UC}_k$ holds since CANON_k uses *bounded-height resolution*, where $\mathcal{UC}_k$ uses ***k*-times nested input resolution** (which is the same as space complexity of tree-resolution):

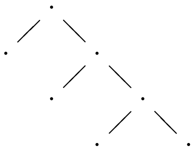

Figure: Tree-res proofs in $\mathcal{UC}_1$

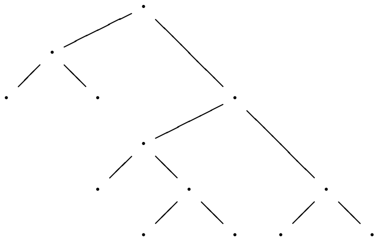

Figure: Tree-res proofs in $\mathcal{UC}_2$

So we have here *unbounded height*.

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

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

- A stricter 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/AcceptedP

[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. 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:XXX [cs.LO], arXiv, February 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.