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

1. Introduction
2. $UC + SLUR$
3. $UC_k$ and $SLUR_k$
4. Properties of $UC_k$ and $SLUR_k$
5. Outlook
Preliminaries: clause-sets

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

\[
F = \{ \{a, b\}, \{\overline{b}, c\}, \{\overline{a}, \overline{c}\} \}
\]

as **Conjunctive Normal Form (CNF)** formulas, e.g.,

\[
F \sim (a \lor b) \land (\neg b \lor c) \land (\neg a \lor \neg c).
\]

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**: 

$$\langle b \rightarrow 1 \rangle \ast \{ \{ \bar{a}, b \}, \{ \bar{b}, c \}, \{ \bar{a}, c \} \} = \{ \{ c \}, \{ \bar{a}, c \} \}$$

- 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 $\varphi \ast F = \top$ (i.e., we set at least one literal in every clause to true);
- **unsatisfiable** if for all total assignments $\varphi$ we have $\bot \in \varphi \ast F$.

The classes of all sat resp. unsat clause-sets are **SAT** resp. **USAT**.
Preliminaries: unit propagation

A basic mechanism in determining satisfiability is **unit-clause propagation** (UCP)

For example:

\[
\begin{align*}
\{a\}, \{\bar{a}, b\}, \{\bar{b}\} & \xrightarrow{\langle a \rightarrow 1 \rangle} \{b\}, \{\bar{b}\} & \xrightarrow{\langle b \rightarrow 1 \rangle} \{\bot\}.
\end{align*}
\]

- Detects and sets (some, obvious) **forced assignments**.
- Possible in linear time.
- Using the notation \(r_1\) for UCP we have

\[
r_1(F) := \begin{cases} 
\{\bot\} & \text{if } \bot \in F \\
 r_1(\langle x \rightarrow 1 \rangle \ast F) & \text{if } \exists x \in \text{lit}(F) : \bot \in \langle x \rightarrow 0 \rangle \ast F \\
 F & \text{otherwise}
\end{cases}
\]
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 $\varphi$ such that $\varphi \ast F$ is unsatisfiable we have $\bot \in r_1(\varphi \ast F)$.

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.

- 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.

- 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 SLUR is coNP-complete.
SLUR algorithm

SLUR-algorithm:

1. Take as input a clause-set $F$.
2. Compute $F := r_1(F)$.
3. If $F = \{ \bot \}$ then return UNSATISFIABLE.
4. While $F \neq \top$:
   1. Choose a literal $x$ such that $r_1(\langle x \rightarrow 1 \rangle \ast F) \neq \{ \bot \}$ and set $F := r_1(\langle x \rightarrow 1 \rangle \ast F)$.
   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 SLUR class is the class of clause-sets for which the above (non-deterministic) algorithm never returns GIVE-UP.
A fundamental insight (hasn’t been realised until now!):

\[ \text{SLUR} = \text{UC}. \]

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 “\( F \in \text{SLUR} \)” is coNP-complete, and we now have this also for “\( F \in \text{UC} \)”. 
$\text{SLUR} = \text{UC}$

$\text{UC} \subseteq \text{SLUR}$:

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

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

1. If $F \notin \text{UC}$ then there is some $\varphi$ s.t $\varphi \ast F$ is unsatisfiable but $r_1(\varphi \ast F) \neq \{\bot\}$.
2. Via transitions of the $\text{SLUR}$ algorithm, one can reach $r_1(\varphi \ast F)$.
3. But then $\text{SLUR}$ will GIVE UP later.
SLUR captures various classes of clause-set with poly-time SAT algorithms.

Can we capture more, by generalising these classes?
Kullmann [12] introduced the notion of generalised unit-clause propagation.

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

$$
\begin{align*}
  r_0(F) & := \begin{cases} 
  \{\bot\} & \text{if } \bot \in F \\
  F & \text{otherwise}
  \end{cases} \\
  r_1(F) & = \begin{cases} 
  r_1(\langle x \rightarrow 1 \rangle \ast F) & \text{if } \exists x \in \text{lit}(F) : r_0(\langle x \rightarrow 0 \rangle \ast F) = \{\bot\} \\
  F & \text{otherwise}
  \end{cases} \\
  r_k(F) & := \begin{cases} 
  r_k(\langle x \rightarrow 1 \rangle \ast F) & \text{if } \exists x \in \text{lit}(F) : r_{k-1}(\langle x \rightarrow 0 \rangle \ast F) = \{\bot\} \\
  F & \text{otherwise}
  \end{cases}.
\end{align*}
$$

Rather than linear-time, this is now possible in time $n^{2k-1}$. 
Example: $r_2$ is more powerful than $r_1$

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. $r_2(F) = r_2(\langle a \rightarrow 1 \rangle \ast F) = \{ \bot \}$, since

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

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.
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. $UC_k$ is the set of clause-sets $F$ such that under any partial assignment $\varphi$ for which $\varphi \ast F$ is unsatisfiable we have that $\bot \in r_k(\varphi \ast F)$.

2. $SLUR_k$ 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 $\top$. 
We show

\[ \text{SLUR}_k = \text{UC}_k. \]

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

- By “pumping up” the result from SLUR we get that deciding membership for \( \text{SLUR}_k \) resp. \( \text{UC}_k \) is coNP-complete.
- We show the following inclusion properties:
  - \( \mathcal{HO} \subset \text{UC}_1 \) (more generally, \( \mathcal{RHO} \subset \text{UC}_1 \)).
  - \( 2\text{-CLS} \subset \text{UC}_2. \)
  - \( \mathcal{QHO} \subset \text{UC}_2. \)
  - \( \mathcal{HO}_k \subset \text{UC}_k \) (generalised Horn clause-sets Kleine Büning [11]).
  - \( \Pi_k \subset \text{UC}_{k+1} \) and \( \Upsilon_k \subset \text{UC}_{k+2} \) (Čepek and Kučera [3]).
Existing hierarchies

**SLUR**-based hierarchies:

1. **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. **SLUR***(k)** strengthens **SLUR**(k) by interleaving *r_1* with every choice (see Čepek et al. [4]).

Hierarchies based on restricted resolution:

- **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. **SLUR**(k) ⊂ **SLUR***(k) ⊂ **SLUR***(k+1) = **UC***(k+1).
2. **CANON**(k) ⊂ **UC***(k) = **SLUR***(k).
3. **CANON**(2) ⊄ **SLUR***(k) for every *k* ∈ ℤ₀.
Generalised input resolution

The strict inclusion $\text{CANON}_k \subset \mathcal{UC}_k$ holds since $\text{CANON}_k$ 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 \( \text{UC}_k \)

We consider the following as natural developments from \( \text{UC}_k \):

- 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 \( \text{UC}_k \) via unsatisfiability oracles (see [12, 13, 7]).
- Optimisation of the size of \( \text{UC}_k \) 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


