Good representations via complete clause-learning

Oliver Kullmann

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

Rome, September 27, 2012, Limits of Theorem Proving
Good representations of boolean functions

- Contrary to what I believed in my Cambridge talk this year, there are no good representations of $\text{PHP}_m^n$.
- Doesn’t matter — interesting theory, relevant applications (and relativisation can overcome the PHP-barrier).

Collaboration with my student Matthew Gwynne.

Project home page
A clause-set $F$ is a \textbf{(CNF-)representation} of a boolean function $f$ if $\text{var}(f) \subseteq \text{var}(F)$ and the satisfying assignments of $F$ projected to $\text{var}(f)$ are (precisely) the satisfying assignments of $f$.

Important special case: $\text{var}(f) = \text{var}(F)$, i.e., \textit{without using new variables} (so $F$ is equivalent to $f$).

Using QCNF we can say that $F$ is a representation of $f$ iff

$$f = \exists v \in \text{var}(F) \setminus \text{var}(f) F.$$ 

Paradigmatic example for new variables: extension variables.
In the SAT context, a natural restriction on (CNF-)representations is to demand that evaluation of a *total* assignment for $f$ is “effective” for $F$.

- Evaluation must be possible by unit-clause propagation.
- **Lemma** Such representations are basically “the same” as circuits.

We will actually consider here only further restricted forms of representations.
Example: $\text{PHP}_m^m$

$\text{PHP}_m^m$ as boolean function:

- analogous to “all-different” constraint;
- for that we need to consider the “functional” form, which requires every pigeon to sit in at most one hole;
- so we consider $\text{FPHP}_m^m$ instead;
- useful as building block for CNF-representations (requiring some bijection).

We have an effective representation of $\text{FPHP}_m^m$ (as boolean function), namely the usual clause-set (including the injectivity-clauses), which even does not use new variables.
Consider a CNF-representation $F$ of $f$:

Now we want for all partial assignments $\varphi$ for $f$ such that $\varphi \ast f$ is unsatisfiable, that UCP for $\varphi \ast F$ yields the empty clause.

That’s needed for SAT solving!

We call that a 1-soft representation.

(More generally, $k$-soft means that for every clause $C$ with $f \models C$ there is a tree resolution derivation $F \vdash C$ using space at most $k + 1$.)
Partialising boolean function

To boolean function $f$ we associate boolean function $\hat{f}$, allowing to consider partial assignments:

- Variables $v \in \text{var}(f)$ are doubled: $v_0, v_1 \in \text{var}(\hat{f})$.
- That $v$ is unassigned is expressed by $v_0 = v_1 = 0$.
- If $v_0 = v_1 = 1$, then $\hat{f} = 1$.
- And if $v_\varepsilon = 1$ and $v_{\overline{\varepsilon}} = 0$, then $v = \varepsilon$.
- $\hat{f}$ true iff the corresponding partial assignment to $f$ makes $f$ unsatisfiable.

$\hat{f}$ is monotone.
Partialising $\text{FPHP}_m^m$

The boolean function $\text{FPHP}_m^m$ determines whether a bipartite graph with at most $m$ vertices on each side admits a perfect matching (setting the missing edges to false).

- So we have a short circuit for $\text{FPHP}_m^m$ (via perfect matching decision).
- However that’s not useful for SAT (the underlying algorithm can only be exploited by constraint solving), since this doesn’t translate back to $\text{FPHP}_m^m$.

SAT-solving uses partial evaluation.
Relative 1-softness $= \text{monotone circuits}$

From [BKNW09] follows:

Monotone circuits for $\hat{f}$ correspond effectively to CNF-representations of $f$ of relative softness 1.

Easiest to see is the direction from a monotone circuit for $\hat{f}$ to a CNF-representation of $f$ of relative softness 1: a monotone circuit which evaluates to 1 doesn’t bother about inputs equal to 0 (which here means missing assignments) — thus we can just plug in $v$ for $v_1$ and $\neg v$ for $v_0$, and translate the circuit into a CNF, with negated output.

The above theorem yields:

There is no poly-size CNF-representation of relative softness 1 for $\text{FPHP}_m^m$. 

O Kullmann (Swansea University)
The UC hierarchy

$\mathcal{UC}_k$ is the set of clause-sets $F$ such that for every $F \models C$ there is a tree-resolution derivation $F \vdash C$ using clause-space at most $k + 1$.

Many equivalent characterisations, for example using generalised UCP.

A(n) (absolute) $k$-soft representation $F$ for $f$ is a representation $F \in \mathcal{UC}_k$ for $f$. 
Hierarchy conjecture

For the relative condition everything collapses to $k = 1$, since there are no restrictions on the new variables (so they can absorb higher $k$).

**Conjecture** We have a true representation hierarchy for the absolute condition.

Partial result: can show this when not using new variables (here the relative and absolute condition coincide).

What is needed now is a hierarchy inside monotone circuits!
In [GK12, GK13] we show

\[ UC_k = SLUR_k \]

where \( SLUR \) is the class of clause-sets where “single look-ahead unit-resolution” is guaranteed to succeed, and \( SLUR_k \) generalises this via generalised UCP.

This means:

\( UC_1 \) is the class of clause-sets closed under clause learning, where the algorithm not just uses UCP, but also look-ahead with UCP (to set a variable).

Note that here satisfiable clause-sets are most interesting. Finally, we can also use standard clause-learning, arriving at the hierarchy \( PC_k \) (“propagation-completeness”).
End

(references on the remaining slides).
Christian Bessiere, George Katsirelos, Nina Narodytska, and Toby Walsh.

Matthew Gwynne and Oliver Kullmann.
Generalising unit-refutation completeness and SLUR via nested input resolution.
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 