Towards a Theory of Good SAT Representations

Oliver Kullmann

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

University of Leicester, May 11, 2012, Computer Science Seminars

From its head to its feet

In the last decade, SAT solving has become a tool used for many applications; see [BHvMW09] for an overview.

The notion of "NP-completeness" has been turned from its head to its feet.

This talk is about "good SAT translations":

- A sketch of a general framework.
- The first general tool, the notion of "hardness".

Collaboration with my student Matthew Gwynne.

Project home page

Rough outline

1. The simplest form of our approach, only considered here, starts from a **boolean function** $f \in \mathcal{BF}$.
2. We consider a natural notion of **representation of** f **via a clause-set** $F \in \mathcal{CLS}$.
3. We develop a general notion of **hardness hd** : $\mathcal{CLS} \to \mathbb{N}_0$.
4. We want F to be as **soft** as feasible, that is, $\mathrm{hd}(F)$ as small as feasible.

The fragments on general representations are (yet) mostly of conceptual novelty. The "provable beef" of this talk is in the notion of "hardness", based on [GK12].

It reveals that "unit refutation completeness" and "propagation completeness" are not isolated notions, but there is an underlying general measurement principle.

Outline

TOC: Five basic notions

- boolean functions
- clause-sets
- partial assignments
- forced literals/assignments
- prime implicates.
- UCP (unit-clause propagation)

Boolean functions

Typically, a boolean function is considered as

$$f : \{0,1\}^n \to \{0,1\}.$$

However, it is useful to generalise this to "positional parameters", that is, using an infinite set $\mathcal{VA}$ of variables and consider

$$f : \{0,1\}^V \to \{0,1\}$$

for some finite $V \subset \mathcal{VA}$.

- $\mathrm{var}(f) := V$.
- $\mathcal{BF}$ is the set of all boolean functions.
- Via $\mathbb{N} \subseteq \mathcal{VA}$ we can handle positional parameters.

Semantics

- A map $\varphi : V \to \{0,1\}$ is a **partial assignment**.
- The set of all partial assignments is $\mathcal{PASS}$.
- $\mathrm{var}(\varphi) := V$.
- $f(\varphi) \in \{0,1\}$ is defined iff $\mathrm{var}(f) \subseteq \mathrm{var}(\varphi)$.

The relation $\mathbf{f} \models \mathbf{g}$ holds iff

$$\forall\, \varphi \in \mathcal{PASS} : \mathrm{var}(\varphi) \supseteq \mathrm{var}(f) \cup \mathrm{var}(g) \Rightarrow f(\varphi) \leq g(\varphi).$$

Furthermore f, g are **equivalent**, in signs $\mathbf{f} \cong \mathbf{g}$, iff $f \models g$ and $g \models f$.

$\mathcal{BF}/\cong$ is the boolean algebra freely generated
by the boolean functions given by the variables.

SEMANTICS!

- In the SAT world, there is typically a syntactical point of view, plus algorithmic transformations via equivalence and sat-equivalence.
- The notion of a "constraint" from CSP is often (not always) a strange hybrid between semantics, syntax and algorithmics.

With our notion of boolean functions we want to provide a true semantical underpinning of SAT.

Study the space of **all representations** of a boolean function!

Clause-sets

- Literals are variables $v \in \mathcal{VA}$ and their **complements** $\overline{\mathbf{v}}$.
- A clause C is a finite and clash-free set of literals, i.e., $C \cap \overline{C} = \emptyset$.
- A **clause-set** is a finite set of clauses.
- The set of all clause-sets is $\boldsymbol{\mathcal{CLS}}$.

For example

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

is a clause-set. The default interpretation is

$$\mathrm{CNF}(F) = (a \vee b) \wedge (\neg a \vee c) \in \mathcal{BF}.$$

Remark: Clause-sets should be considered as (precise) combinatorial objects, as generalised hypergraphs.

Applying partial assignments

The application of a partial assignment $\varphi \in \mathcal{PASS}$ to a clause-set $F \in \mathcal{CLS}$ is denoted by

$$\varphi * F \in \mathcal{CLS}.$$

Satisfied clauses are removed, then falsified literals.

This yields an operation of the monoid $\mathcal{PASS}$ on the set $\mathcal{CLS}$.

A special clause-set is $\top := \emptyset$, a special clause is $\bot := \emptyset$.

- F is **satisfiable** iff there is $\varphi \in \mathcal{PASS}$ with $\varphi * F = \top$.
- $\top$ is satisfiable.
- $\{\bot\}$ is unsatisfiable.
- More generally, every F with $\bot \in F$ is unsatisfiable.

F is unsatisfiable iff $\mathrm{CNF}(F) \cong 0$.

Forced literals and assignments

- A literal x is **forced** for a boolean function f iff $f \models x$.
- For a clause-set F this is equivalent to $\langle x \to 0\rangle * F \notin \mathcal{SAT}$.
- Accordingly we say that the assignment $\langle x \to 1\rangle$ is **forced**.

By the way, we use throughout the basic logical law

$$f \models g \Leftrightarrow f \wedge \neg g \models 0.$$

Note that if g is a clause, then $\neg g$ is basically a partial assignment.

Prime implicates

- An **implicate** of a boolean function f is a CNF-clause C with $f \models C$.
- It is **prime** iff no literal can be removed (recall, a CNF-clause get the stronger the shorter it is).

The set $\mathrm{prc}_0(f)$ is a very strong representation of f, basically the representations of hardness 0 — but it's in general also very large.

The whole effort of "good representations" can (could) be understood as compression of $\mathrm{prc}_0(f)$.

UCP

By

$$\mathsf{r}_1 : \mathcal{CLS} \to \mathcal{CLS}$$

unit-clause propagation is denoted, that is,

- applying $F \rightsquigarrow \langle x \to 1 \rangle * F$ as long as there are unit-clauses $\{x\} \in F$,
- and reducing $F \rightsquigarrow \{\bot\}$ in case of $\bot \in F$.

For example

$$\mathsf{r}_1(\{\{a\}, \{a, b\}, \{\overline{a}, c\}, \{c, d\}, \{d, e\}\}) = \{\{d, e\}\}.$$

Simple properties

1. $f \cong 0$ iff every literal is forced for f.
2. If $f \ncong 0$ then x is forced for f iff $\{x\}$ is a prime implicate for f.
3. f is uniquely satisfiable iff every prime implicate of f is of length 1.
4. If x is forced for F, then $\langle x \to 1 \rangle * F$ has the same number of satisfying assignments as F.
5. If x is forced for F and y is forced for $\langle x \to 1 \rangle * F$, then y is forced for F.
6. If $\{x\} \in F$ then x is forced for F.
7. So all assignments applied by UCP, i.e., assignments for $F \rightsquigarrow \mathrm{r}_1(F)$, are forced for F.

TOC: Connecting boolean functions and clause-sets

- QCNF representations
- CNF representations
- with and without new variables
- circuit representations
- UCP

Very powerful: QCNF representations

QCNFs (quantified conjunctive normal forms) F yield a natural framework for representations of boolean functions:

1. The free variables of F are the variables of the boolean function f.
2. A satisfying assignment for f is one making F true.

Our "representations" are precisely the Σ_1-CNF representations (allowing only existential quantifiers).

So our evaluation problem is not in PSPACE.
Still, for evaluating our representations, we need to solve NP-problems.
So some further conditions will be needed.

Representing boolean functions

Spelling it out, and also pushing the existential quantifiers to the background:

- A clause-set F is a **(CNF-)representation** of the boolean function f if $\mathrm{var}(f) \subseteq \mathrm{var}(F)$ and the satisfying assignments of F projected to $\mathrm{var}(f)$ are (precisely) the satisfying assignments of f.
- Important special case: $\mathrm{var}(f) = \mathrm{var}(F)$, i.e., *without using new variables* (so F is equivalent to f).

We have $F \models f$, while in the other direction we need an extension of the satisfying assignment.
Using QCNF we can say that F is a representation of f iff

$$f = \exists_{v \in \mathrm{var}(F) \setminus \mathrm{var}(f)} F.$$

Examples

- $\{\{a, b\}, \{\overline{a}, c\}\}$ is a representation of $(a \vee b) \wedge (\neg a \vee c)$.
- Also $\{\{a, b\}, \{\overline{a}, c\}, \{d, e\}\}$ is a representation of the same function.
- Also $\{\{a, b\}, \{\overline{a}, c\}, \{b, c\}\}$ is.
- While $\{\{a, b, c\}\}$ is not.
- Neither is $\{\{a\}, \{b\}, \{c\}\}$

Why new variables?

When the boolean functions are used as "constraints", that is, we patch together various constraints to obtain our translations, then the additional variables need to be **new variables**.

So well, what's now their point? Without them evaluation would be trivial!

It gives us more space, for better representations.

For example, very simple boolean functions (given by short DNFs) don't have short representations without new variables!

Remark: Allowing that a class $\mathcal{C} \subseteq \mathcal{CLS}$ of clause-sets is allowed to represent boolean functions via new variables is called "existential closure" in the knowledge-compilation literature.

Less powerful: Circuits

In the SAT context, a natural restriction on (CNF-)representations is to demand that evaluation is efficient.

- For example evaluation must be possible by unit-clause propagation.
- More precisely, the representation F of f is "1-effective" iff for every $\varphi \in \mathcal{PASS}$ with $\mathrm{var}(\varphi) = \mathrm{var}(f)$ we have $\mathrm{r}_1(\varphi * F) \in \{\{\bot\}, \top\}$.
- Such representations are basically "the same" as circuits.

We will actually consider here only further restricted forms of representations.

TOC: Two new dimensions

1. PC, UC
2. relative, absolute.

"Good" representations

Translating a computational problem into a SAT-problem (should!) mean:

1. The building blocks are boolean functions ("constraints") like "at most one" or cryptographic boxes.
2. For them "good" (often misunderstood as just being short) representations are sought (possibly using new variables).
3. The whole translation is the union of these representations.

"Good" representation for SAT however does not mean just "short":

- It's about how the solver can "handle" it.
- Experience with SAT-cryptanalysis of DES/AES suggests that for small boolean functions, actually shortest representations are best, however this changes for bigger boolean functions (there the small representations perform badly).

GAC (= relative PC)

The works in SAT being concerned about "good" representations F for f borrowed the concept of "Generalised Arc Consistence" (**GAC**) from CSP in the following way:

> If after applying a partial assignment φ to f we obtain a forced assignment, then we obtain it from $\varphi * F$ via UCP.

More precisely: Let $\mathrm{r}_\infty : \mathcal{CLS} \to \mathcal{CLS}$ denote full elimination of forced assignments. Then the condition is

$$\forall\, \varphi \in \mathcal{PASS} : \mathrm{var}(\varphi) \subseteq \mathrm{var}(f) \Rightarrow \mathrm{r}_\infty(\varphi * F) = \mathrm{r}_1(\varphi * F).$$

Finally, this amounts to compression of the prime implicates of f into F via UCP.

PC ("absolute")

We cared about the prime implicates of f.

What about the prime implicates of F ?!
In other words, what about *arbitrary* partial assignments?!

We argue that we need also the prime implicates of F. There is a related concept in the literature ([BMS12]), namely "Propagation Completeness" (**PC**), being investigated rather recently and employed under different circumstances:

A clause-set F is PC iff whenever $\varphi * F$ has a forced assignment, then we obtain it from $\varphi * F$ via UCP.

More precisely:

$$\mathcal{PC} := \{F \in \mathcal{CLS} \mid \forall\, \varphi \in \mathcal{PASS} : \mathrm{r}_\infty(\varphi * F) = \mathrm{r}_1(\varphi * F)\}.$$

UC more fundamental than PC

We do *not* take $\mathcal{PC}$ as our starting point and generalise it to $\mathcal{PC}_k$ — this is the next step!

Instead we start with "unit-refutation complete", called UC here, as first investigated in [dV94].

$$\mathcal{UC} := \{C \in F \mid \forall\, \varphi \in \mathcal{PASS} : \varphi * F \notin \mathcal{SAT} \Rightarrow \mathsf{r}_1(\varphi * F) = \{\bot\}\}.$$

This condition is simpler (and by definition $\mathcal{PC} \subset \mathcal{UC}$), and directly related to proof complexity, as shown in [Kul99, Kul04].

- In [dV94] methods for computing representations $F' \in \mathcal{UC}$ of $F \in \mathcal{CLS}$ are given, however only *without new variables*.
- This is reasonable, since only "sporadic" F are considered, and for such "random" inputs no algorithmic methods are known (yet), which could handle new variables.
- However in the general context of *representation* new variables are important.

TOC: Some discussions

- relative versus absolute
- knowledge compilation
- succinctness
- possibilities other than hardness.

Relative versus absolute

It seems that the interaction of new variables and the PC/UC-condition has not been discussed yet:

- From a knowledge compilation point of view it makes good sense to consider UC with new variables, but using the relative condition, as pointed out by [BJMSM12].
- They favour the relative over the absolute condition, though they can not prove it to be more succinct.
- Fair enough! They only want to represent knowledge, and then one doesn't need queries using the new variables.
- However SAT solvers should not be impeded!

Knowledge compilation?

The area of knowledge compilation (AI) has some relations to the theory of "good representations F of a boolean function f",

- since "all knowledge" about f should be presented in F,
- and this in an "accessible form",
- but the question is *how* the "retrieval" works!

"Good" SAT representations needs to be "understandable" by SAT solvers. This is rather different from knowledge compilation, where one has much more freedom in the retrieval mechanism.

And furthermore, although the prime implicates of f are surely "prime knowledge" of f, that's not all — absolute access is needed.

Size is not everything

Under appropriate complexity-theoretical assumptions:

- QCNF is more succinct than Σ_1-CNF.
- Σ_1-CNF is more succinct than circuits.

Provably [Juk12] (Section 9.11):

Circuits are more succinct than monotone circuits.

As shown in [BKNW09]:

relative UC $\sim$ relative PC $\sim$ monotone circuits.

Also our hierarchy collapses under the relative view-point.

This total collapse of everything to monotone circuits is due to the unaccountable handling of new variables.

This (we conjecture!) changes with the absolute condition.

All conclusions must be "nicely" derivable

Our general framework for a "good" representation F of f is:

All conclusions $F \models C$ must be "easily derivable".

Note again, that we consider all implication of F, not just those of f,

since splitting on new variables is important (the absolute condition).

So reasonable first measures are

- the binary logarithm of the maximum of (dag-)resolution complexity of deriving $F \models C$,
- the binary logarithm of the maximum of tree-resolution complexity of deriving $F \models C$.

A "good representation" then has logarithmically bounded measures.

We believe that "hardness" has nicer properties, but
especially the full resolution point of view might have potential.

TOC: A hierarchy of reductions

$$\mathsf{r}_k : \mathcal{CLS} \to \mathcal{CLS}.$$

Generalised UCP

Definition

The maps $\mathbf{r_k} : \mathcal{CLS} \to \mathcal{CLS}$ for $k \in \mathbb{N}_0$ are defined as follows:

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

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

Lemma

The maps are well-defined (don't depend on the choices), and apply (only) forced assignments. The bigger k the more forced assignments are found, and r_∞ applies all forced assignments.

We have $\mathsf{r}_\infty(F) = \mathsf{r}_{n(F)}(F)$ for all $F \in \mathcal{CLS}$.

Practical applications

- r_1 is UCP (unit-clause propagation).
- r_2 is failed-literal reduction (full).
- Running $r_0, r_1, \ldots$ achieves quasi-automatisation of tree-resolution.
- Remark: In [Kul99, Kul04] there is also an algorithmic extension of hardness to satisfiable clause-sets, and so we have also "hardness" for finding satisfying assignments.
- This approach is also the kernel of the Stalmarck-approach (at CNF-level).

TOC: The measure

$$\mathsf{hd} : \mathcal{CLS} \to \mathbb{N}_0.$$

"Hardness"

The hierarchy of r_k-reductions yields the following notion of **hardness**:

Definition

The **hardness** $\mathrm{hd}(F)$ of unsatisfiable F is the minimal k with $r_k(F) = \{\bot\}$.

1. $2^{\mathrm{hd}(F)} \leq \mathrm{Comp}^*_R(F) \leq (n(F)+1)^{\mathrm{hd}(F)}$.
2. $\mathrm{hd}(F)+1$ is the space complexity of tree-resolution.
3. $\mathrm{hd}(F)$ is the level of nested input-resolution needed.

[Kul99, Kul04] allow also for oracles for SAT and UNSAT decisions. This will become important in future investigations, but here we only use UNSAT and the trivial oracle for it, just looking for the empty clause.

Hardness generalised

As mentioned, in [Kul99, Kul04] hardness is also extended to satisfiable clause-sets, motivated by SAT-decision purposes. Here we use a stronger measurement (hard to evaluate):

- First mentioned in [ABLM08].
- For a clause-set F the hardness is the maximum of $\mathrm{hd}(\varphi * F)$ for such partial assignments φ where $\varphi * F$ is unsatisfiable.
- I.e., the maximum of $\mathrm{hd}(\langle x \to 0 : x \in C \rangle * F)$ over $F \models C$.
- This is the maximum level of nested input resolution needed to derive all $F \models C$.

Hardness — to be softened, not measured

[ABLM08] proposed "hardness" (for unsatisfiable clause-sets) as measure of solution hardness.

- Let's call a clause-set *k-soft* if $\mathrm{hd}(F) \leq k$.
- We think of the main role of hardness for *satisfiable* clause-sets as being a **target**, to construct soft clause-sets (representing relevant boolean functions).

Using [ČKV12] (where $k = 1$ is handled) we get:

Lemma

For $k \geq 1$ the decision $\mathrm{hd}(F) \leq k$ is coNP-complete.

(Recall that for unsatisfiable F computation of $\mathrm{hd}(F)$ can be done in time $O(n^{2\,\mathrm{hd}(F)})$.)

TOC: UC via hardness

- $\mathcal{UC}_1 = \mathcal{UC}$
- $\mathcal{UC}_k$
- $\mathcal{SLUR}$
- $\mathcal{SLUR}_k$

Recognising UC

We have

$$\mathcal{UC} = \{F \in \mathcal{CLS} : \mathrm{hd}(F) \leq 1\}.$$

This is just the well-known refutational equivalence of unit-resolution and input-resolution.

- So we can generalise!
- As explained in [Kul99, Kul04], for higher levels we have to stick to input-resolution, which "parameterises" tree-resolution, while generalised unit-resolution "parameterises" dag-resolution.

(Regarding the already mentioned possibility of considering dag-resolution, it might be interesting in the future to actually consider generalised unit-resolution (see [Kul99, Kul04]) and the hardness based on it.)

The generalised UC hierarchy

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

$$\mathcal{UC}_k := \{F \in \mathcal{CLS} : \mathrm{hd}(F) \leq k\}.$$

- $\mathcal{UC}_0$ is the set of clause-sets F which contain all their prime implicates (i.e., $\mathrm{prc}_0(F) \subseteq F$).
- $\mathcal{UC}_1 = \mathcal{UC}$.
- Membership decision for $\mathcal{UC}_0$ is polynomial-time, while it is coNP-complete for $k \geq 1$.

The fundamental lemma:

$F \in \mathcal{UC}_k$ if and only if for every $C \in \mathrm{prc}_0(F)$ there is a derivation of C from F via k-times nested input resolution.

SLUR

In [SAFS95] the SLUR-algorithm was introduced:

1. "Single lookahead unit resolution".
2. Input $F \in \mathcal{CLS}$, output "SAT", "UNSAT" or "don't know".
3. If $\mathrm{r}_1(F) = \{\bot\}$, then output "UNSAT".
4. Now only outputs "SAT" and "don't know" are possible, in the following loop:
   1. If $F = \top$, then output "SAT".
   2. Otherwise consider a literal x.
   3. If $\mathrm{r}_1(\langle x \to 1\rangle * F) = \{\bot\}$, then simplify $F \rightsquigarrow F\langle x \to 0\rangle * F$.
   4. Otherwise simplify $F \rightsquigarrow F\langle x \to 1\rangle * F$.
   5. If $F = \{\bot\}$, then output "don't know".
5. $\mathcal{SLUR}$ is the class of clause-sets F where SLUR never outputs "don't know".

SLUR properly turned into a hierarchy

To obtain the **right** generalisation of $\mathcal{SLUR}$,

just replace r_1 by r_k for $k \in \mathbb{N}_0$!

See [GK12] why this is better than the previous (three) approaches in [ČKV12, BvGKV12].

It all boils down to the fundamental lemma.

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

Once everything is in place, then it's not hard to show that

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

for all $k \in \mathbb{N}_0$.

- So for $\mathcal{UC}_k$ a simple SAT-decision algorithm is available, which outputs also a satisfying assignment (by a form of self-reduction).
- And this simple SAT-decision algorithm also characterises $\mathcal{UC}_k$.

Conclusions

Contributions:

- Started a general reflection on SAT translation, with the two dimensions "relative versus absolute" and "hardness" (see below!).
- Revealed that the hardness-measure underlies $\mathcal{UC}$, obtaining a natural hierarchy $\mathcal{UC}_k$.
- $\mathcal{UC}_k = \mathcal{SLUR}_k$.

Next steps:

- $\mathrm{phd}(F)$ and $\mathcal{PC}_k$
- We get: $\mathcal{PC}_0 \subset \mathcal{UC}_0 \subset \mathcal{PC}_1 \subset \mathcal{UC}_1 \subset \mathcal{PC}_2 \subset \mathcal{UC}_2 \ldots$.
- Perhaps we should use hardness-values $-\frac{1}{2}, 0, \frac{1}{2}, 1, 1\frac{1}{2}, 2, \ldots$.
- **Major conjecture**: proper representation-hierarchy (in two versions, with and without new variables).
- Relativise, obtaining $\mathrm{hd}_{\mathcal{U}}$ and $\mathrm{phd}_{\mathcal{U}}$, using oracle $\mathcal{U}$ for unsatisfiability.

End

(references on the remaining slides).

Bibliography I


Carlos Ansótegui, María Luisa Bonet, Jordi Levy, and Felip Manyà.
Measuring the hardness of SAT instances.
In Dieter Fox and Carla Gomes, editors, *Proceedings of the 23th AAAI Conference on Artificial Intelligence (AAAI-08)*, pages 222–228, 2008.


Armin Biere, Marijn J.H. Heule, Hans van Maaren, and Toby Walsh, editors.
Handbook of Satisfiability, volume 185 of *Frontiers in Artificial Intelligence and Applications*.
IOS Press, February 2009.


Lucas Bordeaux, Mikoláš Janota, Joao Marques-Silva, and Pierre Marquis.
On unit-refutation complete formulae with existentially quantified variables.
In *Knowledge Representation 2012 (KR 2012)*, 2012.

Bibliography II


Christian Bessiere, George Katsirelos, Nina Narodytska, and Toby Walsh.
Circuit complexity and decompositions of global constraints.
In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 412–418, 2009.


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.

Bibliography III


Tomáš Balyo, Štefan Gurský, Petr Kučera, and Václav Vlček.
On hierarchies over the SLUR class.
International Symposium on Artificial Intelligence and Mathematics, January 2012.
http://www.cs.uic.edu/bin/view/Isaim2012/AcceptedPapers.


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.

Bibliography IV

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.

Matthew Gwynne and Oliver Kullmann.
Generalising unit-refutation completeness and SLUR via nested input resolution.
Technical Report arXiv:1204.6529v1 [cs.LO], arXiv, April 2012.

Bibliography V


Stasys Jukna.
Boolean Function Complexity: Advances and Frontiers, volume 27 of *Algorithms and Combinatorics*.
Springer, 2012.
ISBN 978-3-642-24507-7.


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.

Bibliography VI


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.


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.