

Generalising Unit-Refutation Completeness and SLUR via Nested Input Resolution

Matthew Gwynne · Oliver Kullmann

Received: 3 July 2012 / Accepted: 28 January 2013 / Published online: 9 March 2013
© Springer Science+Business Media Dordrecht 2013

Abstract The class $SLUR$ (*Single Lookahead Unit Resolution*) was introduced in Schlipf et al. (Inf Process Lett 54:133–137, 1995) as an umbrella class for efficient (poly-time) SAT solving, with linear-time SAT decision, while the recognition problem was not considered. Čepek et al. (2012) and Balyo et al. (2012) extended this class in various ways to hierarchies covering all of CNF (all clause-sets). We introduce a hierarchy $SLUR_k$ which we argue is the natural “limit” of such approaches. The second source for our investigations is the class UC of *unit-refutation complete clause-sets*, introduced in del Val (1994) as a target class for knowledge compilation. Via the theory of “hardness” of clause-sets as developed in Kullmann (1999), Kullmann (Ann Math Artif Intell 40(3–4):303–352, 2004) and Ansótegui et al. (2008) we obtain a natural generalisation UC_k , containing those clause-sets which are “unit-refutation complete of level k ”, which is the same as having hardness at most k . Utilising the strong connections to (tree-)resolution complexity and (nested) input resolution, we develop basic methods for the determination of hardness (the level k in UC_k). A fundamental insight now is that $SLUR_k = UC_k$ holds for all k . We can thus exploit both streams of intuitions and methods for the investigations of these hierarchies. As an application we can easily show that the hierarchies from Čepek et al. (2012) and Balyo et al. (2012) are strongly subsumed by $SLUR_k$. Finally we consider the problem of “irredundant” clause-sets in UC_k . For 2-CNF we show that strong minimisations are possible in polynomial time, while already for (very special) Horn clause-sets minimisation is NP-complete. We conclude with an extensive discussion of open problems and future directions. We envisage the concepts investigated here to be

M. Gwynne · O. Kullmann (✉)
Computer Science Department, College of Science, Swansea University,
Swansea, SA2 8PP, UK
e-mail: O.Kullmann@Swansea.ac.uk
URL <http://cs.swan.ac.uk/~csoliver>

M. Gwynne
e-mail: csmg@swansea.ac.uk
URL <http://cs.swan.ac.uk/~csmg/>

the starting point for a theory of good SAT translations, which brings together the good SAT-solving aspects from *SLUR* together with the knowledge-representation aspects from *UC*, and expands this combination via notions of “hardness”.

Keywords SAT (satisfiability) · Generalised unit-propagation · Unit-refutation completeness · Hardness · Nested input resolution · SLUR · Knowledge compilation · Polynomial time SAT decision

1 Introduction

The boolean satisfiability problem, SAT for short, in its core version is the problem of deciding satisfiability of a conjunctive normal form (clause-set) F ; see the handbook [6] for further information. An important theme is the search for relevant classes \mathcal{C} of clause-sets F for which one can (at least) decide satisfiability in *polynomial time* (that is, deciding whether F logically implies the empty clause); see Section 1.19 in [22] for some basic information. For the task of *knowledge compilation* one wants more from the target-class \mathcal{C} , namely that the clausal entailment problem (deciding whether F logically implies some given clause) can be decided in polynomial time; see [17] for an overview. In this article now we bring together two previously unconnected streams of research from these two areas:

- SLUR** The SLUR algorithm is an incomplete linear-time SAT-decision algorithm, based on look-ahead via unit-clause propagation.
- UC** The class UC of unit-refutation complete clause-sets enables clausal-entailment decision in linear time via unit-clause propagation.

That both streams are based on unit-clause propagation, which is also the basic tool for efficient SAT solving, we consider as an essential feature. It means that actually we have some form of “SAT knowledge compilation”, where the “knowledge” is compiled in such a way that a SAT solver can “understand” it! In Sections 1.1 and 1.2 we will discuss these two streams in turn, while their unification is outlined in Section 1.3, and applications to SAT knowledge compilation are discussed in Section 1.4. This is the journal version of the conference paper [28], while the underlying report is [27].

1.1 The Quest for SLUR Hierarchies

In the year 1995 in [43] the SLUR algorithm was introduced, a simple incomplete non-deterministic SAT-decision algorithm, which always succeeded on various classes with polynomial-time SAT decision where previously only rather complicated algorithms were known. The computation is divided into two phases for input-clause-set F : First we check via unit-clause propagation (UCP) for unsatisfiability. If this check fails, then we assume F is satisfiable, and guess a satisfying assignment, using UCP-look-ahead for the guessed assignments to avoid obviously false assignments. The class *SLUR* contains those F where this algorithm always succeeds (i.e., determines unsatisfiability in the first phase, or always finds a satisfying assignment in the second phase).

So recognition of $SLUR$ seems a non-trivial problem, while SAT decision for $F \in SLUR$ can be done in linear time. The natural question arises, whether $SLUR$ can be turned into a hierarchy, covering in the limit all clause-sets. A generalisation of SLUR has been considered in [23] under the name “ISLUR” (improved SLUR), allowing a polynomial number $p(\ell(F))$ of backtracks (for a fixed polynomial p , in the input-size $\ell(F)$), in the unsatisfiability as well as in the satisfiability phase of the SLUR algorithm, before giving up. It is mentioned that ISLUR gives up on every large enough “sparse” clause-set (which are “typical” as random k -CNF clause-sets), when no variable occurs “too often”. This was considered to be “disappointing”—but from our point of view the value of the class $SLUR$ lies not in being a “big” class of clause-sets with polynomial-time SAT solving, but in establishing a basic target class for representations of boolean functions with very strong properties via clause-sets; see Section 1.4 for further discussions. For all fixed k there exists a polynomial p such that the k -th level of our hierarchy, $SLUR_k$, is contained in the class ISLUR (those clause-sets where the ISLUR algorithm never gives up). So all levels are negligible when considering the above sparse clause-sets, but as we will argue in Section 1.4, nevertheless this hierarchy is proper regarding good representations of boolean functions, and the parameter k is meaningful and robust (not just a numerical parameter like the polynomial p).

In [2, 12] the authors finally proved that membership decision of $SLUR$ is coNP-complete, and presented three hierarchies, $SLUR(k)$, $SLUR^*(k)$ and $CANON(k)$. It still seemed that none of these hierarchies is the final answer, though they all introduce a certain natural intuition. We now present what seems the natural “limit hierarchy”, which we call $SLUR_k$, and which unifies the two basic intuitions embodied in $SLUR(k)$, $SLUR^*(k)$ on the one hand and $CANON(k)$ on the other hand.

In order to do so we need a precise analysis of the $SLUR$ -class. We introduce the SLUR transition relation $F \xrightarrow{SLUR} F'$ between clause-sets F, F' , which makes precise one non-deterministic step of the SLUR-algorithm. This transition from F to F' happens when assigning a (single) literal in such a way that UCP does not create the empty clause. The core of the classes $SLUR(k)$ and $SLUR^*(k)$ is to strengthen the transition relation by requesting that not just one literal is choosable, but actually k literals can be chosen, while the difference between them is that $SLUR^*(k)$ performs UCP inbetween the choices, while the weaker class $SLUR(k)$ does not.

Before we can describe our solution, the $SLUR_k$ -hierarchy, we need to discuss the second source of our approach, the class UC of “unit-refutation complete clause-sets”, which is related to the stream embodied by $CANON(k)$.

1.2 Unit-Refutation Completeness and “Hardness”

In the year 1994 in [20] the class UC was introduced, containing clause-sets F such that clausal entailment, that is, whether $F \models C$ holds (clause C follows logically from F , i.e., C is an implicate of F), can be decided by unit-clause propagation. The motivation was knowledge compilation, that is, to have a more succinct alternative to the use of the set of all prime implicates of a given clause-set F_0 (clausal database), for which one seeks an equivalent F such that clausal entailment can be decided quickly.

A second development is important here, namely the development of the notion of “hardness” in [1, 36, 37]. The first source [36] from 1999 introduced the notion of

hardness as a measure $hd_0 : \mathcal{CLS} \rightarrow \mathbb{N}_0$, assigning natural numbers to clause-sets in the following way (using $\mathcal{SAT} \subset \mathcal{CLS}$ for the satisfiable clause-sets, and $\mathcal{USAT} := \mathcal{CLS} \setminus \mathcal{SAT}$):

- $hd_0(F) := 0$ for the simplest clause-sets $F \in \mathcal{CLS}$ regarding SAT decision, containing the empty clause (i.e., $\perp \in F$) or being empty (i.e., $F = \top$).¹
- $hd_0(F) = k \geq 1$ iff there is a literal x such that for $F' := \langle x \rightarrow 0 \rangle * F$ (setting x to 0) we have $hd_0(F') \leq k - 1$ and either $F' \in \mathcal{USAT}$ and $hd_0(\langle x \rightarrow 1 \rangle * F) \leq k$, or $F' \in \mathcal{SAT}$.

The second source [37] from 2004 generalised this approach to constraint satisfaction problems (and beyond). The third source [1] from 2008 considered $hd_0(F)$ on unsatisfiable clause-sets $F \in \mathcal{USAT}$, relating it to backdoors, cycle-cutsets and treewidth, and performing an experimental study on random instances. Also in [1] we find a different extension of $hd_0 : \mathcal{USAT} \rightarrow \mathbb{N}_0$ to a measure $hd : \mathcal{CLS} \rightarrow \mathbb{N}_0$, using for satisfiable instances $F \in \mathcal{SAT}$ the maximisation over all unsatisfiable sub-instances obtained by applying partial assignments. This hardness notion is harder to measure: as we show in this article, determining whether $hd(F) \leq k$ holds for a fixed $k \geq 1$ is coNP-complete, while $hd_0(F) \leq k$ can be decided in polynomial time (for fixed k). Nevertheless it is the central measure for this article, and we consider it as measuring “representation hardness”, while hd_0 measures “solver hardness”.²

As we show in Theorem 5.7, $hd(F) \leq k$ is equivalent to the property of F , that all implicates of F (i.e., all clauses C with $F \models C$) can be derived by k -times nested input resolution from F , a generalisation of input resolution as introduced and studied in [36, 37].³ So we obtain that \mathcal{UC} is precisely the class of clause-sets F with $hd(F) \leq 1$! It is then natural to define the hierarchy \mathcal{UC}_k via the property $hd(F) \leq k$. The hierarchy $\text{CANON}(k)$ is based on resolution trees of height at most k , which is a special case of k -times nested input resolution, and so we have $\text{CANON}(k) \subset \mathcal{UC}_k$.

1.3 Bringing SLUR and UC Together

In order to get back to SLUR, we need to emphasise the two-sided nature of the hardness measure, as developed in [36, 37]. In Section 1.2 we discussed the *proof-theoretic side* of it. The *algorithmic side* is given by the reductions $r_k : \mathcal{CLS} \rightarrow \mathcal{CLS}$ (introduced in [36]), which perform certain forced assignments:

1. r_1 is UCP, assigning $x \rightarrow 1$ for unit-clauses $\{x\}$ until all are eliminated.
2. r_2 is (complete) failed-literal elimination, assigning, while possible, $x \rightarrow 1$ for literals x such that the assignment $x \rightarrow 0$ yields a contradiction via r_1 ; see

¹Actually a two-dimensional family $hd_{\mathcal{U},\mathcal{S}}$ of such measures was introduced, based on oracles $\mathcal{U} \subseteq \mathcal{USAT}$, $\mathcal{S} \subseteq \mathcal{SAT}$ for deciding unsatisfiability resp. satisfiability, and setting $hd_{\mathcal{U},\mathcal{S}}(F) := 0$ for $F \in \mathcal{U} \cup \mathcal{S}$. In this article we consider only the simplest base case $hd_0 = hd_{\mathcal{U}_0, \mathcal{S}_0}$, where $\mathcal{U}_0 := \{F \in \mathcal{CLS} : \perp \in F\}$ and $\mathcal{S} := \{\top\}$. Oracle \mathcal{S} does not play a role in the setting of this article, which is fully unsatisfiability-based. See Section 6.3 for more information on these hierarchies, and see Section 9.4 for an outlook on relativised hardness.

² $hd(F)$ actually captures tree-like resolution (in a sense). In Section 9.5 we discuss a width-based measure of hardness, which captures dag-like resolution. We consider the tree-hardness as the natural starting point.

³Equivalently, as shown in [36, 37], one can say that all implicates C have a tree-resolution proof using space at most $k + 1$.

Section 5.2.1 in [32] for the usage of failed literals in SAT solvers (so-called “look-ahead solvers”), and see Section 7.2.2 in [39] for the general explanation of r_2 being the “look-ahead version” of r_1 .

3. In general r_{k+1} is the “look-ahead version” of r_k , assigning, while possible, $x \rightarrow 1$ for literals x such that the assignment $x \rightarrow 0$ yields a contradiction via r_k .

For unsatisfiable F the hardness $hd(F)$ is equal to the minimal k such that $r_k(F)$ detects unsatisfiability of F , i.e., $r_k(F) = \{\perp\}$. This yields the basic observation $UC \subseteq SLUR$ —and actually we have $UC = SLUR$!

So by replacing the use of r_1 in the SLUR algorithm by r_k (using our analysis via the transition relation) we obtain a natural hierarchy $SLUR_k$, which includes the previous SLUR-hierarchies $SLUR(k)$ and $SLUR^*(k)$, and where we have $SLUR_k = UC_k$. This equality of these two hierarchies is our argument that we have found the “limit hierarchy” for SLUR.

1.4 Outlook on Good Representations of Boolean Functions

The ideas presented in Sections 1.1–1.3 are the main thrust for the results of this paper (Sections 3–7), while in the final Section 8 (and also in the outlook in Section 9) we touch upon what we consider as the main application area and the main area for future developments of the theory, namely a theory of good representations of boolean functions. More precisely, in Section 8 we consider the complexity of finding short equivalent clause-sets of bounded hardness for the most basic CNF classes, 2-CNF and Horn clause-sets, and we show feasibility for the former, NP-completeness for the latter. We roughly outline now the basic ideas on “good representations” in general, while in Section 9 some more details are presented.

SAT algorithms have seen an astounding development in the last two decades. Especially efficient algorithms, data structures and heuristics have been developed. The main bottleneck currently is that the underlying constraint problem needs to be represented via boolean CNF, and it is not clear at all how to do this so that SAT solving becomes as easy as possible. “SAT modulo Theories” (SMT; see [3]) boosts the representation by extending the general method, however it does not yield insights into how to construct the basic representations by CNFs. What is needed is a systematic investigation into “good representations” of boolean functions f by clause-sets F , with the aim of “intelligent” SAT translations.

As a first answer, we consider the classes UC_k as the most basic target classes, that is, $F \in UC_k$ for k “as small as possible” is the (basic) fundamental guideline. The motivation for UC was that of a “good representation”, while the motivation for $SLUR$ was “good SAT solving”—the hierarchies $UC_k = SLUR_k$ bring these two aspects together, and this in a parameterised way, so that k can be traded against the size of F . So the theory of good representations F of boolean functions f can be considered as “SAT knowledge representation”, where the “knowledge”, the boolean function f , must be represented by a clause-set F such that all “aspects” of f (most fundamental the prime implicates) are represented in such a way that a SAT solver can “understand” this representation.

What is now the precise relation between the boolean function f to be represented, and the representation F , a clause-set? The most basic idea is to consider that F as a CNF is equivalent to f , which we write as $F \cong f$ (more precisely, $CNF(F) \cong f$). Good representations in this (restricted) setting then amount to consider subsets

$F \subseteq \text{prc}_0(f)$ of the set of prime implicates of f , such that $F \cong f$ and such that $\text{hd}(F)$ and $\ell(F)$ (the size of F) are in a “reasonable” relationship (the lower $\text{hd}(F)$ the higher $\ell(F)$, and so a balance is to be sought). The basic conjecture then states that allowing larger hardness yields more possibilities for short representations:

Conjecture 1.1 *For every $k \in \mathbb{N}_0$ there exists a sequence $(f_n)_{n \in \mathbb{N}}$ of boolean functions, such that no polysize-sequence $(F_n)_{n \in \mathbb{N}}$ (i.e., where $(\ell(F_n))_{n \in \mathbb{N}}$ is polynomially bounded in n) exists with*

- $F_n \cong f_n$
- $\text{hd}(F_n) \leq k$

for all n , but where such a sequence $(F_n)_{n \in \mathbb{N}}$ exists when allowing $\text{hd}(F_n) \leq k + 1$.

Conjecture 9.4 extends this conjecture to include the use of new variables, and also refines it by introducing intermediate levels between the hardness-levels.⁴

The algorithmic approach for such representations (not using new variables) is to systematically search for small F with a given hardness upper-bound. In Section 8 one finds the most basic considerations. In [26] we presented some initial experimental results on using this approach for the (small) building-blocks like the S-boxes in block ciphers like AES and DES, for their SAT-based cryptanalysis (see Section 9.3 for more information).

1.5 The Schaefer Classes

We conclude by some remarks on the four main classes from Schaefer’s dichotomy result (see Section 12.2 in [16] for an introduction, and see [15] for an in-depth overview on recent developments). Our point of view here is that we consider a boolean function f which is either Horn, dual Horn, bijunctive or affine, and we ask for a good representation $F \in \mathcal{CLS}$ of f :

- If f is Horn or dual Horn, then there is a (dual) Horn clause-set F equivalent to f , and by Part 4 of Lemma 6.5 we have $\text{hd}(F) \leq 1$. So obtaining a representation $F \in \mathcal{UC}$ is trivial; however optimising the size of F is NP-complete (see Theorem 8.4).
- If f is bijunctive, then there is a 2-CNF F equivalent to f , and by Part 3 of Lemma 6.5 we have $\text{hd}(F) \leq 2$. Moreover, by Theorem 8.3 we can reduce the hardness to 0 or 1 (as we wish) in polynomial time, and that by optimal (shortest) such F .
- If f is affine, that is, f is the conjunction of m linear equations $x_1 \oplus \cdots \oplus x_p = 0$ over $\{0, 1\}$ viewed as a 2-element field, with addition \oplus as exclusive-or, then the situation regarding the existence of a representation of bounded hardness is not fully understood yet:
 1. If $m = 1$, then there is precisely one CNF-representation of f without new variables, containing 2^{p-1} clauses and being (trivially) of hardness 0. So without new variables we have a polysize representation of bounded hardness iff p is bounded.

⁴In [29] we have meanwhile established that Conjecture 1.1 is true.

2. While when allowing new variables, then for $m = 1$ there is a representation $F \in \mathcal{UC}$, as will be shown in [29].
3. For arbitrary m there is definitely no small representation without new variables when the clause-length p is unbounded. When bounding p , or when allowing new variables, then the existence of a polysize $F \in \mathcal{UC}_k$ for some fixed k seems to be an interesting open problem; for some partial results see [40]. Perhaps no polysize representations $F \in \mathcal{UC}$ exist, even for the “relative condition”, where propagation-conditions are posed only for the variables in the XOR-clauses; see [5] for general tools for such lower bounds, and see Sections 9.2 and 9.4 for more discussions.

1.6 Overview

After discussing basic terminology in Section 2, in Section 3 we discuss SLUR and existing extensions. We give a precise (mathematical) definition of the class $SLUR$, achieving a conceptually clear understanding, and based on these concepts we give precise (mathematical) definitions of the various SLUR hierarchies from the literature. In Section 4 we provide the background about generalised unit-clause propagation, that is, the reductions $r_k : \mathcal{CLS} \rightarrow \mathcal{CLS}$, where \mathcal{CLS} is the set of all clause-sets and r_1 is unit-clause propagation. Section 5 then introduces the hardness $\text{hd} : \mathcal{CLS} \rightarrow \mathbb{N}_0$ and defines the classes $\mathcal{UC}_k \subset \mathcal{CLS}$ of “unit-refutation complete clause-sets of level k ” as those F with $\text{hd}(F) \leq k$. The first main result is Theorem 5.7, which states that the elements of \mathcal{UC}_k are precisely the clause-sets F where every prime implicate of F can be derived by k -times nested input resolution from F . In Section 6 we develop various tools to determine hardness. First we consider various constructions in Section 6.1. Then in Section 6.2 we provide tools to show that classes of clause-sets have bounded hardness, with applications to common classes and to stability properties of the classes \mathcal{UC}_k . Alternative and generalised hardness-notions are considered in Section 6.3. We conclude by considering algorithmic ways to determine the hardness-measure in Section 6.4. Section 7 introduces the $SLUR_k$ hierarchy. Our second major result is Theorem 7.4, showing that $\mathcal{UC}_k = SLUR_k$ holds. From this characterisation we derive in Theorem 7.5 the coNP-completeness of membership decision for \mathcal{UC}_k when $k \geq 1$. And in Theorems 7.6 and 7.7 we show that the previous hierarchies are (strictly) included in the $SLUR_k$ hierarchy, which we consider as a kind of “completion”, where both approaches, based on SLUR and UC, meet. In Section 8 we turn towards the problem of finding short equivalent clause-sets of low hardness for a given clause-set F . In Theorem 8.3 we show that for F in 2-CNF we can compute optimal equivalent clause-sets (of low hardness) in polynomial time. While in Theorem 8.4 we show that already for Horn clause-sets F , even when all prime implicates are given as part of the input, the decision whether there is an equivalent clause-set (of low hardness) using at most a given number of clauses is NP-complete. We conclude in Section 9 with the summary and an extensive discussion of future directions.

2 Preliminaries

We follow the general notions and notations as outlined in [35]. We use $\mathbb{N} = \{1, \dots\}$ and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Based on an infinite set \mathcal{VA} of variables, we form the set $\mathcal{LIT} := \mathcal{VA} \cup \overline{\mathcal{VA}}$ of positive and negative literals, using complementation. A clause

$C \subset \mathcal{LIT}$ is a finite set of literals without clashes, i.e., $C \cap \overline{C} = \emptyset$, where for $L \subseteq \mathcal{LIT}$ we set $\overline{L} := \{\overline{x} : x \in L\}$. The set of all clauses is denoted by \mathcal{CL} . A clause-set $F \subset \mathcal{CL}$ is a finite set of clauses, and the set of all clause-sets is denoted by \mathcal{CLS} . For $k \in \mathbb{N}_0$ we use $k\text{-}\mathcal{CLS} := \{F \in \mathcal{CLS} \mid \forall C \in F : |C| \leq k\}$ for the set of clause-sets where all clauses have length at most k .

A special clause is the empty clause $\perp := \emptyset \in \mathcal{CL}$, and a special clause-set is the empty clause-set $\top := \emptyset \in \mathcal{CLS}$. By $\text{lit}(F) := \bigcup F \cup \bigcup \overline{F}$ we denote the set of literals occurring at least in one polarity in F .

We use $\text{var} : \mathcal{LIT} \rightarrow \mathcal{VA}$ for the underlying variable of a literal, $\text{var}(C) := \{\text{var}(x) : x \in C\} \subset \mathcal{VA}$ for the set of variables in a clause, and $\text{var}(F) := \bigcup_{C \in F} \text{var}(C)$ for the set of variables in a clause-set. So $\text{lit}(F) = \text{var}(F) \cup \overline{\text{var}(F)}$. The number of variables in a clause-set is $n(F) := |\text{var}(F)| \in \mathbb{N}_0$, the number of clauses is $c(F) := |F| \in \mathbb{N}_0$, and the number of literal occurrences is $\ell(F) := \sum_{C \in F} |C| \in \mathbb{N}_0$.

A *full clause-set* is a clause-set F such that each clause contains all variables, that is, for all $C \in F$ we have $\text{var}(C) = \text{var}(F)$. The set of Horn clause-sets is $\mathcal{HO} \subset \mathcal{CLS}$, where every clause contains at most one positive literal, while $\mathcal{HO}^+ \subset \mathcal{HO}$ is the set of pure Horn clause-sets, where every clause contains exactly one positive literal. $\mathcal{HO} \subset \mathcal{RHO} \subset \mathcal{CLS}$ is the set of renamable (“hidden”) Horn clause-sets, which by flipping signs can be turned into a Horn clause-set.

A partial assignment $\varphi : V \rightarrow \{0, 1\}$ maps a finite $V \subset \mathcal{VA}$ to truth-values, the set of all partial assignments is \mathcal{PASS} . A special partial assignment is the empty partial assignment $\langle \rangle := \emptyset \in \mathcal{PASS}$. We can construct partial assignments via $(v_1 \rightarrow \varepsilon_1, \dots, v_n \rightarrow \varepsilon_n) \in \mathcal{PASS}$ for $v_i \in \mathcal{VA}$ and $\varepsilon_i \in \{0, 1\}$ (which must be consistent). We use $\text{var}(\varphi) := V = \text{dom}(\varphi)$ for the variables in the domain of φ , and by $\mathcal{TASS}(V)$ we denote the set of all “total assignments” for V , that is, the $\varphi \in \mathcal{PASS}$ with $\text{var}(\varphi) = V$. And $n(\varphi) := |\text{var}(\varphi)| \in \mathbb{N}_0$ is the number of variables assigned by φ .

For a partial assignment $\varphi \in \mathcal{PASS}$ and a clause-set $F \in \mathcal{CLS}$ the application of φ to F is denoted by $\varphi * F \in \mathcal{CLS}$, which results from F by removing all satisfied clauses (containing at least one satisfied literal), and removing all falsified literals from the remaining clauses. A class $\mathcal{C} \subseteq \mathcal{CLS}$ of clause-sets is *stable under (application of) partial assignments* if for all $F \in \mathcal{C}$ and $\varphi \in \mathcal{PASS}$ holds $\varphi * F \in \mathcal{C}$.

A clause-set F is *satisfiable* (i.e., $F \in \mathcal{SAT} \subset \mathcal{CLS}$) if there exists a partial assignment φ with $\varphi * F = \top$, otherwise F is *unsatisfiable* (i.e., $F \in \mathcal{USAT} := \mathcal{CLS} \setminus \mathcal{SAT}$). For a clause C the partial assignment $\varphi_C \in \mathcal{PASS}$ is defined as $\varphi_C := \langle x \rightarrow 0 : x \in C \rangle$, that is, it sets precisely the literals of C to 0 (and leaves all other variables unassigned). For example $\varphi_\perp = \langle \rangle$ and $\varphi_{\{x\}} = \langle x \rightarrow 0 \rangle$.

Two clauses $C, D \in \mathcal{CL}$ are *resolvable* if they clash in exactly one literal x , that is, $C \cap \overline{D} = x$, in which case their resolvent is $(C \cup D) \setminus \{x, \overline{x}\}$ (with resolution literal x). A resolution tree is a binary tree formed by the resolution operation. We write $\mathbf{T} : \mathbf{F} \vdash \mathbf{C}$ if T is a resolution tree with axioms (the clauses at the leaves) all in F and with derived clause (at the root) C . By $\mathbf{Comp}_R^*(F)$ for unsatisfiable F the minimum number of leaves in a tree-resolution-refutation $T : F \vdash \perp$ is denoted.

A boolean function f is a map $f : \mathcal{TASS}(V) \rightarrow \{0, 1\}$ for some finite $V =: \text{var}(f)$; we can also use $f(\varphi) \in \{0, 1\}$ for $\varphi \in \mathcal{PASS}$ with $\text{var}(f) \subseteq \text{var}(\varphi)$, in which case φ is restricted to $\text{var}(f)$. Special boolean functions are 0^V and 1^V for the constant-0 resp. constant-1 functions with domain V . We write $f \models g$ for boolean functions f, g if for all partial assignments φ with $\text{var}(\varphi) \supseteq \text{var}(f) \cup \text{var}(g)$ we have $f(\varphi) = 1 \Rightarrow g(\varphi) = 1$. Equivalence of boolean functions f, g means $f \models g$ and $g \models f$ (so all 0^V are equivalent, and all 1^V are equivalent).

The interpretation of clauses C and clause-sets F as boolean functions is explicitly denoted by $\text{CNF}(C)$ and $\text{CNF}(F)$, using the CNF-interpretation (a clause as a disjunction of literals, a clause-set as a conjunction of clauses), and happens in this article typically implicitly.

For a boolean function f the set of prime implicates is denoted by $\text{prc}_0(f)$, the set of all clauses C with $f \models C$ while for $C' \subset C$ holds $f \not\models C'$. (The “0” in $\text{prc}_0(f)$ resp. $\text{prc}_0(F)$ in the set of prime implicates of a boolean function or a clause-set (interpreted as CNF) shall remind at “false” or “unsatisfiable”, since CNF have “falsity” at the core.) So a boolean function f is equivalent to $\text{prc}_0(f)$, that is, more explicitly, to $\text{CNF}(\text{prc}_0(f))$. As it is well-known, by considering any clause-set F equivalent to f and computing the resolution-closure of F , followed by subsumption-elimination, we obtain precisely $\text{prc}_0(f)$.

We denote by $\text{CNF}(f)$ the “distinguished canonical normal form”, or the set of “minterms of f ”, that is, the set of clauses $C \in \mathcal{CL}$ with $\text{var}(C) = \text{var}(f)$ and $f \models C$ (that is, $f \models \text{CNF}(C)$). Dually, by $\text{DNF}(f)$ we denote the set of clauses $C \in \mathcal{CL}$ with $\text{var}(C) = \text{var}(f)$ and $\text{DNF}(C) \models f$ (the “maxterms of f ”; note that for us a clause is a combinatorial object, and the logical interpretation has to be added). In the DNF-interpretation a clause is the conjunction of its literals, and a clause-set is the disjunction of its clauses.

Finally, by $r_1 : \mathcal{CLS} \rightarrow \mathcal{CLS}$ unit-clause propagation is denoted, that is applying $F \rightsquigarrow \langle x \rightarrow 1 \rangle * F$ as long as there are unit-clauses $\{x\} \in F$, and reducing $F \rightsquigarrow \{\perp\}$ in case of $\perp \in F$. In Definition 4.3 the general $r_k : \mathcal{CLS} \rightarrow \mathcal{CLS}$ is defined.

3 The SLUR Class and Extensions

The SLUR-algorithm and the class $\mathit{SLUR} \subset \mathcal{CLS}$ have been introduced in [43]. The SLUR-algorithm for input $F \in \mathcal{CLS}$ is an incomplete polynomial-time SAT algorithm, which either returns “SAT”, “UNSAT” (in both cases correctly) or gives up. This algorithm is non-deterministic, and SLUR is the class of clause-sets where it never gives up (and thus SAT-decision for $F \in \mathit{SLUR}$ can be done in polynomial time). Due to an observation attributed to Truemper in [21], the SLUR-algorithm can be implemented such that it runs in linear time. Decision of membership, that is whether $F \in \mathit{SLUR}$ holds, by definition is in coNP, but only in [12] it was finally shown that this decision problem is coNP-complete.

The original motivation was that SLUR contains several other classes, including renamable Horn, extended Horn, hidden extended Horn, simple extended Horn and CC-balanced clause-sets, where for each class it was known that the SAT problem is solvable in polynomial time, but with in some cases rather complicated proofs, while it is trivial to see that the SLUR-algorithm runs in polynomial time. In [21, 24] probabilistic properties of SLUR have been investigated.⁵

⁵At this point a popular misunderstanding should be avoided: The well-known dichotomy result of Schaefer (see Section 1.5) states that under certain conditions there are precisely six classes of problem instances with polytime SAT solving (unless P=NP). However this has no bearing on the classes considered here, since they do not fall within the restricted framework of Schaefer’s theorem.

In this section we first give a semantic definition of $SLUR$ in Section 3.1. In a nutshell, $SLUR$ is the class of clause-sets where either UCP (unit-clause propagation aka r_1) creates the empty clause, or where otherwise iteratively making assignments followed by UCP will always yield a satisfying assignment, given that these transitions do not obviously create unsatisfiable results, i.e., do not create the empty clause. In order to understand this definition (and its various extensions) clearly, we present a precise mathematical (non-algorithmic) definition, based on the transition relation $F \xrightarrow{SLUR} F'$ (Definition 3.3), which represents one non-deterministic step of the SLUR algorithm: If r_1 on input $F \in \mathcal{CLS}$ does not determine unsatisfiability (in which case we have $F \in SLUR$), then $F \in SLUR$ iff \top can be reached by this transition relation, while everything else reachable from F is not an end-point of this transition relation.

In [2, 12] recently three approaches towards generalising $SLUR$ have been considered, and we discuss them in Section 3.2. Our generalisation, called $SLUR_k$, which we see as the natural completion of these approaches, will be presented in Section 7.

3.1 SLUR

The SLUR-algorithm (“Single Lookahead Unit Resolution”) from [43] is described for input $F \in \mathcal{CLS}$ as follows:

1. First run UCP, that is, reduce $F \rightsquigarrow r_1(F)$.
2. If now $\perp \in F$ then we determined F unsatisfiable.
3. If not, then the algorithm guesses a satisfying assignment for F , by repeated transitions $F \xrightarrow{SLUR} F'$, where F' is obtained by assigning one variable and then performing UCP, i.e., $F' = r_1(\langle x \rightarrow 1 \rangle * F)$ for some literal x .
4. The “lookahead” means that a transition with $F' = \{\perp\}$ is avoided.
5. The algorithm might find a satisfying assignment in this way, or it gets stuck, that is, for the chosen literal both assignments $x \rightarrow 1$ and $\bar{x} \rightarrow 1$ yield $\{\perp\}$, in which case it “gives up”.

The SLUR class is defined as the class of clause-sets where this algorithm never gives up. The precise details are as follows. First we define the underlying transition relation (one non-failing transition from F to F'):

Definition 3.1 For clause-sets $F, F' \in \mathcal{CLS}$ the relation $F \xrightarrow{SLUR} F'$ holds if there is $x \in \text{lit}(F)$ such that $F' = r_1(\langle x \rightarrow 1 \rangle * F)$ and $F' \neq \{\perp\}$. The transitive-reflexive closure is denoted by $F \xrightarrow{SLUR}_* F'$.

Example 3.2 Considering when we have $F \xrightarrow{SLUR}_* F'$ and when not:

1. $F \xrightarrow{SLUR}_* \top$ iff $F \in \mathcal{SAT}$.
2. $\{C\} \xrightarrow{SLUR} \top$ precisely for all clauses $C \neq \perp$.
3. $\{\{x, y\}, \{x, \bar{y}\}\} \xrightarrow{SLUR} \top$.
4. $\{\{\bar{x}, y\}, \{\bar{y}, z\}\} \xrightarrow{SLUR} \top$ (due to e.g. $r_1(\langle x \rightarrow 1 \rangle * \{\{\bar{x}, y\}, \{\bar{y}, z\}\}) = \top$).

5. $F \xrightarrow{\text{SLUR}} F'$ does not hold if there is no literal to set, or if r_1 detects unsatisfiability of F' . That is, there are **no** clause-sets F, F' such that any of the following hold:
- (a) $\top \xrightarrow{\text{SLUR}} F$.
 - (b) $\{\perp\} \xrightarrow{\text{SLUR}} F$.
 - (c) $F \xrightarrow{\text{SLUR}} F$.
 - (d) $F \xrightarrow{\text{SLUR}} F'$ where $r_1(F') = \{\perp\}$.

Via the transition-relation $F \xrightarrow{\text{SLUR}} F'$ we can now easily define the class \mathcal{SLUR} , which will find a natural generalisation in Definition 7.1 to \mathcal{SLUR}_k for $k \in \mathbb{N}_0$ (where $\mathcal{SLUR} = \mathcal{SLUR}_1$):

Definition 3.3 The set of all fully reduced clause-sets reachable from $F \in \mathcal{CLS}$ is denoted by

$$\text{slur}(F) := \left\{ F' \in \mathcal{CLS} \mid F \xrightarrow{\text{SLUR}}_* F' \wedge \neg \exists F'' \in \mathcal{CLS} : F' \xrightarrow{\text{SLUR}} F'' \right\}.$$

Finally the class of all clause-sets which are either identified by UCP to be unsatisfiable, or where by SLUR-reduction always a satisfying assignment is found, is denoted by $\mathcal{SLUR} := \{F \in \mathcal{CLS} : r_1(F) \neq \{\perp\} \Rightarrow \text{slur}(F) = \{\top\}\}$.

We could define $\xrightarrow{\text{SLUR}}$ as $F \xrightarrow{\text{SLUR}} \langle x \rightarrow 1 \rangle * F$ iff $r_1(\langle x \rightarrow 1 \rangle * F) \neq \perp$, and this would yield the same class \mathcal{SLUR} but a different transition relation (one would not be forced to immediately make forced assignments).

Example 3.4 Computing $\text{slur}(F)$ for clause-sets F :

1. $\text{slur}(F) \neq \emptyset$ (in the “worst” case we have $F \in \text{slur}(F)$).
2. $\text{slur}(\{\perp\}) = \{\{\perp\}\}$.
3. $\text{slur}(\top) = \{\top\}$.
4. $\text{slur}(\{C\}) = \{\top\}$ iff $C \neq \perp$.
5. If $r_1(F) = \top$ then $\text{slur}(F) = \{\top\}$.
6. $\text{slur}(\{\{x, y\}, \{x, \bar{y}\}\}) = \{\top\}$.
7. $\text{slur}(\{\{\bar{x}, y\}, \{\bar{y}, z\}\}) = \{\top\}$.
8. For $F := \{\{x, y\}, \{x, \bar{y}\}, \{\bar{x}, y\}, \{\bar{x}, \bar{y}\}\}$ we have $\text{slur}(F) = \{F\}$.
9. For $F' := \{\{z, x, y\}, \{z, x, \bar{y}\}, \{z, \bar{x}, y\}, \{z, \bar{x}, \bar{y}\}\}$ we have $\top, F \in \text{slur}(F')$.

3.2 Previous Approaches for SLUR Hierarchies

In [2, 12] three hierarchies $\mathcal{SLUR}(k), \mathcal{SLUR}^*(k)$ ($k \in \mathbb{N}$) and $\text{CANON}(k)$ ($k \in \mathbb{N}_0$) have been introduced. In Section 4 of [2] it is shown that $\mathcal{SLUR}(k) \subset \mathcal{SLUR}^*(k)$ for all $k \in \mathbb{N}$ and so we restrict our attention to $\mathcal{SLUR}^*(k)$ and $\text{CANON}(k)$.

$\text{CANON}(k)$ is defined to be the set of clause-sets F such that every $C \in \text{pre}_0(F)$ can be derived from F by a resolution tree of height at most k . Note that basically by definition (using stability of resolution proofs under application of partial assignments) we get that each $\text{CANON}(k)$ is stable under application of partial assignments and under variable-disjoint union.

The $SLUR^*(k)$ hierarchy is derived in [2] from the $SLUR$ class by extending the reduction r_1 . We provide an alternative formalisation here, in the same manner as in Section 3.1. The main question is the transition relation $F \rightsquigarrow F'$. The $SLUR^*(k)$ -hierarchy provides stronger and stronger witnesses that F' might be satisfiable, by longer and longer assignments (making “ k decisions”) not yielding the empty clause:

Definition 3.5 That partial assignment $\varphi \in \mathcal{PASS}$ makes k decisions for some $k \in \mathbb{N}_0$ w.r.t. $F \in \mathcal{CLS}$ is defined recursively as follows: For $k = 0$ this relation holds if $\varphi * F = r_1(F)$, while for $k > 0$ this relation holds if either there is $k' < k$ such that φ makes k' decision w.r.t. F and $\varphi * F = \top$, or there exists $x \in \text{lit}(F)$ and a partial assignment φ' making $k - 1$ decision for $r_1((x \rightarrow 1) * F)$, and where $\varphi * F = \varphi' * r_1((x \rightarrow 1) * F)$.

Now $F \xrightarrow{SLUR^*k} F'$ for $k \geq 1$ by definition holds if there is a partial assignment φ making k decision w.r.t. F with $F' = \varphi * F$, where $F' \neq \{\perp\}$. The reflexive-transitive closure is $\xrightarrow{SLUR^*k}_*$.

Finally we can define the hierarchy:

$$\begin{aligned} \text{slur}^*(k)(F) &:= \left\{ F' \in \mathcal{CLS} \mid F \xrightarrow{SLUR^*k}_* F' \wedge \neg \exists F'' : F' \xrightarrow{SLUR^*k} F'' \right\} \\ \mathcal{SLUR}^*(k) &:= \{ F \in \mathcal{CLS} : \text{slur}^*(k)(F) \neq \{F\} \Rightarrow \text{slur}^*(k)(F) = \{\top\} \}. \end{aligned}$$

The unsatisfiable elements of $SLUR^*(k)$ are those $F \neq \top$ with $\text{slur}^*(k)(F) = \{F\}$. By definition each $SLUR^*(k)$ is stable under application of partial assignments, but not stable under variable-disjoint union, since the number of decision variables is bounded by k (in Lemma 6.7 we will see that our hierarchy is stable under variable-disjoint union, which is natural since it strengthens the $CANON(k)$ -hierarchy).

Example 3.6 Some examples for $CANON(k)$ and $SLUR^*(k)$ ($k \in \mathbb{N}$):

1. Consider the unsatisfiable clause-set $F := \{ \{x, y\}, \{x, \bar{y}\}, \{\bar{x}, y\}, \{\bar{x}, \bar{y}\} \}$.
 - (a) $F \notin SLUR$ because F is unsatisfiable but $r_1(F) \neq \{\perp\}$.
 - (b) $F \in \mathcal{SLUR}^*(1)$ because $r_1((x' \rightarrow 1) * F) = \{\perp\}$ for all $x' \in \text{lit}(F)$ and so $\text{slur}^*(1)(F) = \{F\}$.
 - (c) This establishes $SLUR \subset \mathcal{SLUR}^*(1)$.
 - (d) $F \in \mathcal{CANON}(2) \setminus \mathcal{CANON}(1)$ because actually all tree-resolution refutations of F are full binary trees of height 2.
2. Consider the satisfiable clause-set $F' := \{ \{x_1, \dots, x_k\} \cup C \mid C \in F \}$.
 - (a) $F' \notin \mathcal{SLUR}^*(k)$ because $F' \xrightarrow{SLUR^*k}_* F$, where F is unsatisfiable and thus $\neg(F \xrightarrow{SLUR^*k}_* \top)$, whence $\text{slur}^*(k)(F') \neq \{\top\}$.
 - (b) $F' \in \mathcal{SLUR}^*(k + 1)$ because we have $r_1(\varphi * F') \in \{\top, \{\perp\}\}$ for all partial assignments φ of length $k + 1$ on variables of F' hence $\text{slur}^*(k)(F_1) = \{\top\}$.
 - (c) $F' \in \mathcal{CANON}(2)$ because the only prime implicate is $\{x_1, \dots, x_k\}$ and actually all its tree-resolution proofs are full binary trees of height 2.

4 Generalised Unit-Clause Propagation

In this section we review the approximations of forced assignments as computed by the hierarchy of reductions $r_k : \mathcal{CLS} \rightarrow \mathcal{CLS}$ from [36, 37] for $k \in \mathbb{N}_0$. First we introduce the semantical notion of forced literals/assignments in Section 4.1 together with the limit-reduction $r_\infty : \mathcal{CLS} \rightarrow \mathcal{CLS}$, which eliminates *all* forced assignments. In Section 4.2 then the r_k -reductions themselves (eliminating some forced assignments) are defined and basic properties discussed. In Section 4.3 finally we introduce generalised (nested) input resolution and its main parameter, the ‘‘Horton–Strahler number’’ of the corresponding resolution tree, generalising the well-known refutational equivalence between unit resolution and input resolution, and providing the proof-theoretic background.

For further discussions of these reductions, in the context of SAT decision and in their relations to various consistency and width-related notions, see [36, 37] and Section 3 in [38]. It seems to us that the r_k -reductions establish the SAT-counterpart to consistency-notions from the constraint literature (see [4] for an overview). We have the following basic distinction between SAT and CSP: SAT has the extremely ‘‘thin’’ clauses, enabling the global point of view (‘‘no (or flat) hierarchies’’), while CSP has ‘‘fat’’ constraints, the ‘‘lumping together’’ of clauses. In the SAT world, the r_k -reductions approximate global consistency via approaching all assignments of r_∞ , while in the CSP world, consistency means making the constraints stronger and stronger (lumping more and more clauses together), until only one constraint is left. Thus the (stronger) consistency-notions of CSP are more related to width-restricted resolution, while, as shown in [36, 37], the r_k -reductions are much weaker (each only using linear space). Making a clause-set F ‘‘consistent’’ in the SAT world thus means (to us) to find a ‘‘representation’’ F' of F (see Section 9.2 for some discussion on ‘‘representations’’), where via r_k for some $k \in \mathbb{N}_0$ we can derive ‘‘everything’’, which is embodied in its most elementary form in the \mathcal{UC}_k -hierarchy, that is, via the condition $F' \in \mathcal{UC}_k$ (Definition 5.6).

4.1 Forced Literals/Assignments

Fundamental is the notion of a ‘‘forced literal’’ of a boolean function resp. a clause-set,⁶ which are literals which must be set to true in order to satisfy the function resp. clause-set:

Definition 4.1 A literal x is **forced** for a boolean function f if $f \models x$, and the set of forced literals for f is $\text{fl}(f) \subseteq \mathcal{LIT}$. A literal is forced for a clause-set F if it is forced for $\text{CNF}(F)$, and we set $\text{fl}(F) := \text{fl}(\text{CNF}(F))$.

Every literal is forced for every 0^V . In fact a boolean function f is constant zero iff $\text{fl}(f) = \mathcal{LIT}$ iff there is a literal x with $x, \bar{x} \in \text{fl}(f)$. No literal is forced for any 1^V (i.e., $\text{fl}(1^V) = \emptyset$). We have for every boolean function f that

$$\text{fl}(f) = \bigcap_{\mathcal{LIT}} \text{DNF}(f)$$

⁶We prefer this logical (and common) terminology over ‘‘backbone literal’’, which is only used in a special context.

(the index “ \mathcal{LIT} ” in the intersection is the “universe” of the sets considered in the intersection, which becomes the result if there are no sets to intersect, that is, if f is unsatisfiable). More directly we can read off the forced literals from the prime clauses, namely x is forced for f iff $\text{prc}_0(f) \cap \{\perp, \{x\}\} \neq \emptyset$.

Example 4.2 Here are some basic determinations of $\text{fl}(F)$:

1. $\text{fl}(\{\perp\}) = \mathcal{LIT}$.
2. $\text{fl}(\top) = \emptyset$.
3. $\text{fl}(\{x_1, \dots, x_n\}) = \{x_1, \dots, x_n\}$.
4. $\text{fl}(\{x, \bar{y}\}, \{\bar{x}, y\}) = \emptyset$.
5. $\text{fl}(\{x, y\}, \{x, \bar{y}\}) = \{x\}$.

If x is a forced literal for F , then the **forced assignment** $\langle x \rightarrow 1 \rangle$ yields the clause-set $\langle x \rightarrow 1 \rangle * F$ which is satisfiability-equivalent to F . We denote by $\mathbf{r}_\infty(F) \in \mathcal{CLS}$ the result of applying all forced assignments to F . Note that F is unsatisfiable iff $\mathbf{r}_\infty(F) = \{\perp\}$ (while F is uniquely satisfiable after discarding variables without influence iff $\mathbf{r}_\infty(F) = \top$).

4.2 A Hierarchy of Reductions

We now review the hierarchy $\mathbf{r}_k : \mathcal{CLS} \rightarrow \mathcal{CLS}$, $k \in \mathbb{N}_0$, of reductions [36], which achieves approximating \mathbf{r}_∞ by poly-time computable functions. The basic idea is that unit-clause propagation in a sense computes the most direct forced assignments (at “level $k = 1$ ”), and generalisations like failed-literal elimination (level $k = 2$) find more forced assignments.

Definition 4.3 [36] The maps $\mathbf{r}_k : \mathcal{CLS} \rightarrow \mathcal{CLS}$ for $k \in \mathbb{N}_0$ are defined as follows (for $F \in \mathcal{CLS}$):

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

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

\mathbf{r}_1 is unit-clause propagation, \mathbf{r}_2 is (full) failed literal elimination. We call \mathbf{r}_k **generalised unit-clause-propagation of level k** . In [36] one finds the following basic observations proven (for $k \in \mathbb{N}_0$, $F \in \mathcal{CLS}$ and $\varphi \in \text{PASS}$):

- The map $\mathbf{r}_k : \mathcal{CLS} \rightarrow \mathcal{CLS}$ is well-defined (does not depend on the choices).
- \mathbf{r}_k applies only forced assignments (and so $\mathbf{r}_k(F)$ is satisfiability-equivalent to F).
- $\mathbf{r}_k(F)$ is computable in time $O(\ell(F) \cdot n(F)^{2(k-1)})$ and linear space.
- $\mathbf{r}_k(F) = \{\perp\}$ implies $\mathbf{r}_k(\varphi * F) = \{\perp\}$.
- $\mathbf{r}_k(\varphi * \mathbf{r}_k(F)) = \mathbf{r}_k(\varphi * F)$.

Quasi-automatisation of tree-resolution is achieved for inputs $F \in \text{USAT}$ by applying $\mathbf{r}_0(F), \mathbf{r}_1(F), \dots$ until unsatisfiability has been achieved [36]. Also satisfiable

instances are handled in [36], however in this paper we do not consider these algorithmical aspects.

Actually, a more general form was introduced in [36], namely $r_k^{\mathcal{U}}$ for some oracle \mathcal{U} deciding unsatisfiability at level 0. We believe that this generalisation is important for further progress (see Section 9.4), however in this article we mostly consider only the trivial oracle $\mathcal{U} = \{F \in \mathcal{CLS} : \perp \in F\}$, which recognises unsatisfiability at level 0 iff the empty clause occurs (see Section 6.3 for some discussion of this choice). A further generalisation to constraint-like systems (via an abstract, axiomatic approach) was achieved in [37], however in this initial study we do only consider boolean values and CNF-representations.

Example 4.4 Computing some $r_k(F)$ (using literals x_1, \dots, x_n, x, y with pairwise different underlying variables):

1. $r_k(\{\perp\}) = \{\perp\}$ for $k \geq 0$.
2. $r_k(\top) = \top$ for $k \geq 0$.
3. For $F := \{\{x_1\}, \dots, \{x_n\}\}$: $r_0(F) = F$, $r_k(F) = \top$ for $k \geq 1$.
4. For $F' := F \cup \{\{x, y\}\}$: $r_0(F') = F'$, $r_k(F') = \{\{x, y\}\}$ for $k \geq 1$ (note that $\{\{x, y\}\}$ has no forced assignments).
5. For $F := \{\{x, y\}, \{x, \bar{y}\}\}$: $r_k(F) = F$ for $k \leq 1$, $r_k(F) = \top$ for $k \geq 2$.
6. For $F := \{\{x, y\}, \{x, \bar{y}\}, \{\bar{x}, y\}, \{\bar{x}, \bar{y}\}\}$: $r_k(F) = F$ for $k \leq 1$, $r_k(F) = \{\perp\}$ for $k \geq 2$.

Via the reductions r_k we can approximate the implication relation $F \models C$ as follows:

Definition 4.5 [36, 37] For $k \in \mathbb{N}_0$, clause-sets F and clauses C the relation $F \models_k C$ holds if $r_k(\varphi_C * F) = \{\perp\}$.

As it is well-known, $F \models_1 C$ iff some subclause of C follows from F via input resolution.

Example 4.6 Consider $k \in \mathbb{N}_0$ and literals x, y, w :

1. For all $k \geq 0$ and all clauses C we have:
 - (a) $F \models_k C$ if there is $D \in F$ with $D \subseteq C$ (note $\perp \in \varphi_C * F$).
 - (b) $\{\perp\} \models_k C$ and $\top \not\models_k C$.
2. $\{\{x, y\}, \{x, \bar{y}\}\} \models_k \{x\}$ iff $k \geq 1$.
3. For $F := \{\{\bar{x}, y\}, \{\bar{y}, z\}\}$ we have $F \models_k \{\bar{x}, z\}$ iff $k \geq 1$.
4. For $F := \{\{\bar{x}, y, w\}, \{\bar{y}, z, w\}, \{\bar{x}, y, \bar{w}\}, \{\bar{y}, z, \bar{w}\}\}$ we have $F \models_k \{\bar{x}, z\}$ iff $k \geq 2$ (note that $\langle x \rightarrow 1, z \rightarrow 0 \rangle * F \in 2\text{-}\mathcal{CLS}$).

4.3 Generalised Input Resolution



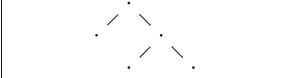
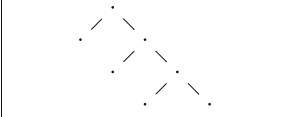
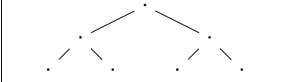
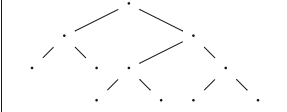
In [36], Chapter 4, the *levelled height* “ $h(T)$ ” of branching trees T has been introduced, which was further generalised in [37], Chapter 3 (to a general form of constraint satisfaction problems). It handles satisfiable as well as unsatisfiable clause-sets. In this article we will only use the unsatisfiable case. In this case the

measure reduces to a well-known measure which only considers the structure of the tree. As discussed in Sections 4.2 and 4.3 of [36], this case, the levelled height of splitting trees for unsatisfiable clause-sets, appeared at many places in the literature. Ansótegui et al. [1] used the term ‘‘Horton–Strahler number’’ (sometimes also ‘‘Strahler number’’): it seems the oldest source (from 1945), however disconnected from its various (re-)inventions in computer science. As in [1], the Horton–Strahler number of the trivial tree is 0.

Definition 4.7 Consider a resolution tree T . The **Horton–Strahler number** $hs(T) \in \mathbb{N}_0$ is defined as $hs(T) := 0$, if T is trivial (consists only of one node), while otherwise we have two subtrees T_1, T_2 , and we set $hs(T) := \max(hs(T_1), hs(T_2))$ if $hs(T_1) \neq hs(T_2)$, while in case of $hs(T_1) = hs(T_2)$ we set $hs(T) := \max(hs(T_1), hs(T_2)) + 1$.

See Sections 4.2 and 4.3 in [36] for various characterisations of $hs(T)$.

Example 4.8 Examples of trees with their Horton–Strahler numbers. We denote by T_1 and T_2 in each example the left and right sub-trees of the root.

<i>Tree T</i>	$hs(T)$	<i>Explanation</i>
	0	trivial tree
	1	$hs(T_1) = 0,$ $hs(T_2) = 0.$
	1	$hs(T_1) = 0,$ $hs(T_2) = 1.$
	1	$hs(T_1) = 0,$ $hs(T_2) = 1.$
	2	$hs(T_1) = 1,$ $hs(T_2) = 1.$
	2	$hs(T_1) = 1,$ $hs(T_2) = 2.$

In [36], Section 7 (generalised in [37], Section 5), *generalised input resolution* was introduced. We use the notation ‘‘ \vdash_k ’’ for it:

Definition 4.9 [36, 37] For a clause-set $F \in \mathcal{CLS}$ and a clause $C \in \mathcal{CL}$ the relation $F \vdash_k C$ (C can be derived from F by k -times nested input resolution) holds if there exists a resolution tree T and $C' \subseteq C$ with $T : F \vdash C'$ and $hs(T) \leq k$.

By parts 1 and 2 of Theorem 7.5 in [36], generalised in Corollary 5.12 in [37]:

Lemma 4.10 [36, 37] For clause-sets F , clauses C and $k \in \mathbb{N}_0$ we have $F \models_k C$ if and only if $F \vdash_k C$.

5 Hardness

This section is devoted to the discussion of $\text{hd} : \mathcal{CLS} \rightarrow \mathbb{N}_0$. It is the central concept of the paper, from which the hierarchy \mathcal{UC}_k is derived (Definition 5.6). The basic idea is to start with some measurement $h : \mathcal{USAT} \rightarrow \mathbb{N}_0$ of “the complexity” of unsatisfiable F . This measure is extended to arbitrary $F \in \mathcal{CLS}$ by maximising over all “sub-instances” of F , that is, over all unsatisfiable $\varphi * F$ (arbitrary) partial assignments φ . A first guess for $h : \mathcal{USAT} \rightarrow \mathbb{N}_0$ is to take something like the logarithm of the tree-resolution complexity of F . However this measure is too fine-grained, and doesn’t yield a hierarchy like \mathcal{UC}_k , where each level brings a qualitative enhancement. Another approach is algorithmical, measuring how far F is from being refutable by unit-clause propagation. As shown in [36, 37], actually these two lines of thought can be brought together by the hardness measure $\text{hd} : \mathcal{USAT} \rightarrow \mathbb{N}_0$. Why only tree-resolution, and not dag-resolution (i.e., full resolution)? The tree-resolution approach is the natural starting point, and what is easy for tree-resolution is also easy for dag-resolution. Our basic approach towards the more complicated handling of dag-resolution is shown in Section 9.5.

The outline of this section is as follows. $\text{hd}(F)$ is defined and discussed for unsatisfiable F in Section 5.1. The general case (arbitrary F) is handled in Section 5.2 by reduction to the unsatisfiable cases within F (as produced by applying partial assignments). The central result of this section can be seen in Theorem 5.7, which shows that $F \in \mathcal{UC}_k$ (i.e., $\text{hd}(F) \leq k$) is equivalent to the condition that all prime implicates of F can be derived by some resolution tree with a Horton–Strahler number at most k . In this way some form of geometric intuition is gained, and a machinery becomes available. The first applications are given by the various lemmas in Section 6 for determining hardness under various circumstances.

We remark that, when considering only unsatisfiable clause-sets F , in [36, 37] actually a general concept of “hardness” was introduced, parameterised by an oracle $\mathcal{U} \subseteq \mathcal{USAT}$ for (“easy”) detection of special cases of unsatisfiability. In this article only $\mathcal{U} = \{F \in \mathcal{CLS} : \perp \in F\}$ is used, but we expect the general theory to become important in the future. See Section 9.4 for some further discussions.

5.1 Hardness of Unsatisfiable Clause-Sets

In [36] the following hardness parameter was introduced and investigated (further generalised in [37]):

Definition 5.1 [36, 37] The **hardness** $\text{hd}(F)$ of an unsatisfiable $F \in \mathcal{CLS}$ is the minimal $k \in \mathbb{N}_0$ such that $r_k(F) = \{\perp\}$.

As shown in [36], $\text{hd}(F) + 1$ is precisely the clause-space complexity of F regarding tree-resolution (see [41] for a recent overview on space complexity of resolution). In [36, 37] the notation “ $h(F)$ ” was used (resp., more generally, “ $h_{\mathcal{U},\mathcal{S}}(F)$ ”, using oracles for unsatisfiability and satisfiability detection), which seems now to us too unspecific. From [31] we gain the insight that for $F \in \mathcal{USAT}$ holds $\text{hd}(F) \leq 1$ iff there exists $F' \subseteq F$ which is an unsatisfiable renamable Horn clause-set (i.e.,

$F' \in \mathcal{RHO} \cap \mathcal{USAT}$). By Theorem 7.8 (and Corollary 7.9) in [36] (or, more generally, Theorem 5.14 in [37]) we have for $F \in \mathcal{USAT}$:

$$2^{\text{hd}(F)} \leq \text{Comp}_R^*(F) \leq (n(F) + 1)^{\text{hd}(F)}.$$

Example 5.2 Some basic determinations of $\text{hd}(F)$ for unsatisfiable F :

1. $\text{hd}(F) = 0$ iff $\perp \in F$.
2. $\text{hd}(\{x, \bar{x}\}) = 1$.
3. $\text{hd}(\{x, \bar{x}, y, \bar{y}, z, \bar{z}\}) = 1$.
4. $\text{hd}(\{x, y, \bar{x}, \bar{y}\}) = 2$.
5. $\text{hd}(\{x, \bar{y}, \bar{x}, y, \bar{z}, z, \bar{z}\}) = 2$.

By Lemma 4.10 we get:

Lemma 5.3 [36, 37] *For an unsatisfiable clause-set F and $k \in \mathbb{N}_0$ we have $\text{hd}(F) \leq k$ iff $F \models_k \perp$ iff $F \vdash_k \perp$.*

By applying partial assignments we can reach all hardness-levels in a clause-set, as the following lemma shows.

Lemma 5.4 *For an unsatisfiable clause-set F and every $0 \leq k \leq \text{hd}(F)$ there exists a partial assignment φ with $n(\varphi) = k$ and $\text{hd}(\varphi * F) = \text{hd}(F) - k$.*

Proof We proceed by induction on $n(F)$. As $k \leq \text{hd}(F) \leq n(F)$, for the base case we consider $n(F) = k$. If $n(F) = k$ then all φ with $n(\varphi) = k$ have $\text{hd}(\varphi * F) = \text{hd}(\{\perp\}) = 0 = \text{hd}(F) - k$. For $n(F) > k$, we make a case distinction on the value of k . If $k = 0$ then choose $\varphi = \langle \rangle$. If $k = 1$ then:

1. Assume for the sake of contradiction that there is no $x \in \text{lit}(F)$ such that $\text{hd}(\langle x \rightarrow 1 \rangle * F) = \text{hd}(F) - 1$; otherwise we are done.
2. If for all $x \in \text{lit}(F)$ we had $\text{hd}(\langle x \rightarrow 1 \rangle * F) \leq \text{hd}(F) - 2$ then by Definition 5.1 we would have $\text{hd}(F) \leq k - 1$, a contradiction.
3. Therefore there must exist an $x \in \text{lit}(F)$ such that

$$\text{hd}(F) = \text{hd}(\langle x \rightarrow 1 \rangle * F) > \text{hd}(\langle x \rightarrow 0 \rangle * F) + 1.$$

4. By induction hypothesis we have a partial assignment φ with $n(\varphi) = 1$ such that $\text{hd}(\varphi * (\langle x \rightarrow 1 \rangle * F)) = \text{hd}(F) - 1$.
5. Application of partial assignments doesn't increase hardness (Lemma 3.11 of [36]) and so we have

$$\text{hd}(\varphi * F) \geq \text{hd}(\langle x \rightarrow 1 \rangle * (\varphi * F)) = \text{hd}(F) - 1.$$

6. By our choice of x we have

$$\text{hd}(\langle x \rightarrow 1 \rangle * (\varphi * F)) = \text{hd}(F) - 1$$

$$\text{hd}(\langle x \rightarrow 0 \rangle * (\varphi * F)) \leq \text{hd}(F) - 2,$$

therefore by Definition 5.1 we have $\text{hd}(\varphi * F) \leq \text{hd}(F) - 1$.

7. Thus we have that $\text{hd}(\varphi * F) = \text{hd}(F) - 1$.

Finally, for $k > 1$, we apply induction using the $k = 1$ case; once we can reduce by 1 we can reduce by k . □

5.2 Hardness of Arbitrary Clause-Sets

The hardness $\text{hd}(F)$ of arbitrary clause-sets can now be defined as the maximum hardness over all unsatisfiable instances obtained by partial assignments.

Definition 5.5 The **hardness** $\text{hd}(F) \in \mathbb{N}_0$ for $F \in \mathcal{CLS}$ is the minimal $k \in \mathbb{N}_0$ such that for all clauses C with $F \models C$ we have $F \models_k C$ (recall Definition 4.5; by Lemma 4.10 this is equivalent to $F \vdash_k C$).

In other words, if $F \neq \top$ then $\text{hd}(F)$ is the maximum of $\text{hd}(\varphi * F)$ for partial assignments φ such that $\varphi * F \in \mathcal{USAT}$. To our knowledge, the measure $\text{hd}(F)$ for satisfiable F was mentioned the first time in the literature in [1], Definition 8 (the only result there concerning this measure is Lemma 9, relating it to another hardness-alternative for satisfiable F). Note that one can restrict attention in Definition 5.5 to $C \in \text{prc}_0(F)$. Hardness 0 means that all prime clauses are there, i.e., $\text{hd}(F) = 0$ iff $\text{prc}_0(F) \subseteq F$. Especially $\text{hd}(\top) = 0$.

Lemma 5.4, stating that $\text{hd}(\varphi * F)$ takes exactly the values from 0 to $\text{hd}(F)$, extends by definition to satisfiable $F \in \mathcal{CLS}$, when adding to the size of the partial assignment φ the minimum size of a partial assignment ψ with $\psi * F \in \mathcal{USAT}$ and $\text{hd}(\psi * F) = \text{hd}(F)$.

Definition 5.6 For $k \in \mathbb{N}_0$ let $\mathcal{UC}_k := \{F \in \mathcal{CLS} : \text{hd}(F) \leq k\}$ (the class of **unit-refutation complete clause-sets of level k**).

The class \mathcal{UC}_1 has been introduced in [20] for knowledge compilation. Various (resolution-based) algorithms computing for clause-sets F some equivalent set $F' \in \mathcal{UC}_1$ of prime implicates are discussed there. Based on the results from [36, 37], we can now give a powerful proof-theoretic characterisation for all classes \mathcal{UC}_k :

Theorem 5.7 For $k \in \mathbb{N}_0$ and $F \in \mathcal{CLS}$ we have

$$F \in \mathcal{UC}_k \iff \forall C \in \text{prc}_0(F) : F \vdash_k C.$$

Thus if every $C \in \text{prc}_0(F)$ has a tree-resolution refutation using at most $2^{k+1} - 1$ leaves (i.e., $\text{Comp}_R^*(\varphi_C * F) < 2^{k+1}$), then $\text{hd}(F) \leq k$.

Proof The equivalence $F \in \mathcal{UC}_k \iff \forall C \in \text{prc}_0(F) : F \vdash_k C$ follows from Lemma 4.10. And if $\text{hd}(F) > k$, then there is $C \in \text{prc}_0(F)$ with $F \not\vdash_k C$, and then every tree-resolution derivation of C from F needs at least 2^{k+1} leaves due to $2^{\text{hd}(\varphi_C * F)} \leq \text{Comp}_R^*(\varphi_C * F)$ (as stated before). □

Example 5.8 Here are some basic calculations of hardness for satisfiable clause-sets (for unsatisfiable F see Example 5.2), using Theorem 5.7:

1. $\text{hd}(\top) = 0$.
2. $\text{hd}(\{\{x\}\}) = 0$.

3. For $F := \{\{x, y\}, \{x, \bar{y}\}\}$ we have $\text{hd}(F) = 1$:
 - (a) $\text{prc}_0(F) = \{\{x\}\}$.
 - (b) $\text{hd}((x \rightarrow 0) * F) = \text{hd}(\{\{y\}, \{\bar{y}\}\}) = 1$.
4. For $F := \{\{\bar{x}, y\}, \{\bar{y}, z\}\}$ we have $\text{hd}(F) = 1$:
 - (a) $\text{prc}_0(F) = \{\{\{\bar{x}, y\}, \{\bar{y}, z\}, \{\bar{x}, z\}\}\}$.
 - (b) $\text{hd}((x \rightarrow 1, y \rightarrow 0) * F) = \text{hd}(\{\perp\}) = 0$.
 - (c) $\text{hd}((y \rightarrow 1, z \rightarrow 0) * F) = \text{hd}(\{\perp\}) = 0$.
 - (d) $\text{hd}((x \rightarrow 1, z \rightarrow 0) * F) = \text{hd}(\{\{y\}, \{\bar{y}\}\}) = 1$.
5. For $F := \{\{z, x, y\}, \{z, x, \bar{y}\}, \{z, \bar{x}, y\}, \{z, \bar{x}, \bar{y}\}\}$ we have $\text{hd}(F) = 2$:
 - (a) $\text{prc}_0(F) = \{\{z\}\}$.
 - (b) $\text{hd}((z \rightarrow 0) * F) = \text{hd}(\{\{x, y\}, \{x, \bar{y}\}, \{\bar{x}, y\}, \{\bar{x}, \bar{y}\}\}) = 2$.

6 Fundamental Properties of \mathcal{UC}_k

In Section 6.1 we determine hardness for various constructions. In Section 6.2 we consider various classes contained in some \mathcal{UC}_k together with stability properties of \mathcal{UC}_k . Relations to alternative hierarchies from the literature are discussed in Section 6.3. We conclude our discussion of basic properties of hardness in Section 6.4, considering the most basic cases of precise hardness-computations. We stress that (algorithmic) computation of hardness for arbitrary instances is less important here,⁷ since we aim more at constructing “soft” (low hardness) representations than measuring hardness of given instances. What is needed is a theory to identify general constructions.

6.1 Some Basic Hardness Determinations

The following basic lemma follows directly by definition:

Lemma 6.1 *If two clause-sets F and F' are variable-disjoint, then we have:*

1. *If $F, F' \in \mathcal{SAT}$, then $\text{hd}(F \cup F') = \max(\text{hd}(F), \text{hd}(F'))$.*
2. *If $F \in \mathcal{SAT}$ and $F' \in \mathcal{USAT}$, then $\text{hd}(F \cup F') = \text{hd}(F')$.*
3. *If $F, F' \in \mathcal{USAT}$, then $\text{hd}(F \cup F') = \min(\text{hd}(F), \text{hd}(F'))$.*

Via full clause-sets A_n with n variables and 2^n clauses we obtain (unsatisfiable, simplest) examples with $\text{hd}(A_n) = n$, and when removing one clause for $n \geq 1$, then we obtain satisfiable examples A'_n with $\text{hd}(A'_n) = n - 1$:

Lemma 6.2 *Consider a full clause-set $F \in \mathcal{CLS}$ (i.e., each clause contains all variables).*

1. $\text{hd}(\top) = 0$.
2. *If F is unsatisfiable then $\text{hd}(F) = n(F)$.*

⁷Decision of membership in \mathcal{UC}_k for $k \geq 1$ is coNP-complete, as shown in Theorem 7.5, which seems natural for classes with strong expressive power.

3. If $F \neq \top$, then $\text{hd}(F) = n(F) - \min_{C \in \text{pre}_0(F)} |C|$.
4. If for F no two clauses are resolvable, then $\text{hd}(F) = 0$.

Proof Part 1 follows by definition, Part 2 is Lemma 3.18 in [36], while Part 4 follows from Part 3. It remains to show Part 3. If F is unsatisfiable, then we get Part 2. For satisfiable F and a partial assignment φ with $\text{var}(\varphi) \subseteq \text{var}(F)$ it is $\varphi * F$ a full clause-set with $n(\varphi * F) = n(F) - n(\varphi)$, and so the assertion follows by reduction to the unsatisfiable case. \square

The following lemma yields a way of pumping up hardness:

Lemma 6.3 Consider $F \in \mathcal{CLS}$ and $v \in \mathcal{VA} \setminus \text{var}(F)$. Let $F' := \{C \cup \{v\} : C \in F\} \cup \{C \cup \{\bar{v}\} : C \in F\}$. Then we have $\text{hd}(F') = \text{hd}(F) + 1$.

Proof We have $\text{hd}(F') \leq \text{hd}(F) + 1$ by definition (if v is not set by the test-assignment, then it can be set to an arbitrary value, yielding a forced assignment at level $\text{hd}(F)$). Now consider a partial assignment φ with $\text{var}(\varphi) \subseteq \text{var}(F)$, $\varphi * F \in \mathcal{USAT}$ and $\text{hd}(\varphi * F) = \text{hd}(F)$. Now also $\varphi * F' \in \mathcal{USAT}$ holds, where $\varphi * F' = \{C \cup \{v\} : C \in \varphi * F\} \cup \{C \cup \{\bar{v}\} : C \in \varphi * F\}$. Thus we have reduced the assertion of the lemma to the special case where $F \in \mathcal{USAT}$, and where $\text{hd}(F') \geq \text{hd}(F) + 1$ is left to be shown. This now follows easily by induction on the number of variables. \square

6.2 Containment and Stability Properties

The following fundamental lemma is obvious from the definition:

Lemma 6.4 Consider $\mathcal{C} \subseteq \mathcal{CLS}$ stable under application of partial assignments and $k \in \mathbb{N}_0$. If $\mathcal{C} \cap \mathcal{USAT} \subseteq \mathcal{UC}_k$ then $\mathcal{C} \subseteq \mathcal{UC}_k$.

We apply Lemma 6.4 to various well-known classes \mathcal{C} (stating in brackets the source for the bound on the unsatisfiable cases).

Lemma 6.5 Consider $F \in \mathcal{CLS}$.

1. For $\varphi \in \mathcal{PASS}$ we have $\text{hd}(\varphi * F) \leq \text{hd}(F)$ (by Lemma 3.11 in [36]).
2. $\text{hd}(F) \leq n(F)$ (by Lemma 3.18 in [36]).
3. If $F \in 2\text{-}\mathcal{CLS} = \{F \in \mathcal{CLS} \mid \forall C \in F : |C| \leq 2\}$, then $\text{hd}(F) \leq 2$ (by Lemma 5.6 in [36]).
4. If $F \in \mathcal{HO} = \{F \in \mathcal{CLS} \mid \forall C \in F : |C \cap \mathcal{VA}| \leq 1\}$ (Horn clause-sets), then $\text{hd}(F) \leq 1$ by (Lemma 5.8 in [36]).
5. More generally, if $F \in \mathcal{QHO}$, the set of q -Horn clause-sets (see Section 6.10.2 in [14], and [44]), then $\text{hd}(F) \leq 2$ (by Lemma 5.12 in [36]).
6. Generalising Horn clause-sets to the hierarchy \mathcal{HO}_k from [34] (with $\mathcal{HO}_1 = \mathcal{HO}$): if $F \in \mathcal{HO}_k$ for $k \in \mathbb{N}$, then $\text{hd}(F) \leq k$ (by Lemma 5.10 in [36]).

Obviously Part 4 of Lemma 6.5 can be generalised to $F \in \mathcal{RHO}$ (see Lemma 6.7, Part 3). And considering Part 3, by a standard autarky-argument for $2\text{-}\mathcal{CLS}$ (see [35]) we can sharpen the hardness-upper-bound 2 for *satisfiable* clause-sets:

Lemma 6.6 For $F \in 2\text{-}\mathcal{CLS} \cap \mathcal{SAT}$ we have $\text{hd}(F) \leq 1$.

Proof Consider a partial assignment φ with unsatisfiable $\varphi * F$. Now we have $r_1(\varphi * F) = \{\perp\}$, since otherwise $r_1(\varphi * F) \subseteq F$, and thus $r_1(\varphi * F)$ would be satisfiable. \square

We have the following stability properties:

Lemma 6.7 Consider $k \in \mathbb{N}_0$.

1. \mathcal{UC}_k is stable under application of partial assignments (with Lemma 6.5, Part 1; this might reduce hardness).
2. \mathcal{UC}_k is stable under variable-disjoint union (with Lemma 6.1).
3. \mathcal{UC}_k is stable under renaming variables and switching polarities (by definition).
4. \mathcal{UC}_k is stable under subsumption-elimination (by basic properties of resolution).
5. \mathcal{UC}_k is stable under addition of inferred clauses (by definition; this might reduce hardness).

Example 6.8 Examples for non-stability:

1. \mathcal{UC}_0 is obviously not stable under removal of clauses.
2. \mathcal{UC}_0 is not stable under removal of literal occurrences, for example $\{\{x, y\}, \{\bar{x}, \bar{y}\}\} \in \mathcal{UC}_0$, but $\{\{x\}, \{\bar{x}, \bar{y}\}\} \notin \mathcal{UC}_0$.
3. \mathcal{UC}_0 is not stable under crossing out of variables, e.g. $\{\{x, y\}, \{\bar{x}, \bar{y}\}\} \in \mathcal{UC}_0$, but when crossing out variable x we obtain $\{\{y\}, \{\bar{y}\}\} \notin \mathcal{UC}_0$.
4. \mathcal{UC}_0 is not stable under addition of clauses, for example $\{\{x\}\} \in \mathcal{UC}_0$, but $\{\{x\}, \{\bar{x}\}\} \notin \mathcal{UC}_0$.
5. \mathcal{UC}_0 is not stable under addition of literal occurrences, e.g. $\{\{x\}, \{y\}\} \in \mathcal{UC}_0$, but $\{\{x, \bar{y}\}, \{y\}\} \notin \mathcal{UC}_0$.

6.3 Alternative Hierarchies

No class \mathcal{UC}_k is stable under removal of clauses. We will see in this subsection that this boils down to the class \mathcal{U}_0 of clause-sets containing the empty clauses not being stable under removal of clauses. Some classes contained in \mathcal{UC}_1 however are stable under removal of clauses, for examples renamable Horn clause-sets (\mathcal{RHO}), and in [11] hierarchies based on this more restricted class have been considered. To understand the connection to our approach, some comments on the use of “oracles” in this setting are needed (see Section 9.4 for future developments).

In [36, 37] the hierarchy $G_k(\mathcal{U}, \mathcal{S}) \subseteq \mathcal{CLS}$ ($k \in \mathbb{N}_0$) has been introduced, using oracles $\mathcal{U} \subseteq \mathcal{USAT}$ for unsatisfiability detection and $\mathcal{S} \subseteq \mathcal{SAT}$ for satisfiability detection:

1. The minimal oracles considered there are $\mathcal{U}_0 := \{F \in \mathcal{CLS} : \perp \in F\}$ and $\mathcal{S}_0 := \{\top\}$.
2. One uses $G_k^0(\mathcal{U}, \mathcal{S}) := G_k(\mathcal{U}, \mathcal{S}) \cap \mathcal{USAT}$ and $G_k^1(\mathcal{U}, \mathcal{S}) := G_k(\mathcal{U}, \mathcal{S}) \cap \mathcal{SAT}$. Since $G_k^0(\mathcal{U}, \mathcal{S})$ does not depend on \mathcal{S} , one writes $G_k^0(\mathcal{U}) := G_k^0(\mathcal{U}, \mathcal{S})$.
3. For all $k \in \mathbb{N}_0$ holds $G_k^0(\mathcal{U}_0) = \mathcal{UC}_k \cap \mathcal{USAT}$. On satisfiable instances in general the hierarchies are incomparable.
4. If $\mathcal{C} \subseteq \mathcal{CLS}$ is stable under application of partial assignments, then each class $G_k(\mathcal{C}) := G_k(\mathcal{C} \cap \mathcal{USAT}, \mathcal{C} \cap \mathcal{SAT})$ (for $k \in \mathbb{N}_0$) is also stable under partial assignments (Lemma 4.2 in [37]). So if $\mathcal{C} \cap \mathcal{USAT} \subseteq \mathcal{UC}_{k'}$ for some $k' \in \mathbb{N}_0$, then

we have $G_k(\mathcal{C}) \subseteq \mathcal{UC}_{k+k'}$ (using Lemma 6.4). This is the basis of all inclusion-relations of Section 6.

5. In [36, 37] it is assumed that $\mathcal{U}_0 \subseteq \mathcal{U}$ holds. This ensures that $\mathcal{UC}_k \cap \mathcal{USAT} \subseteq G_k^0(\mathcal{C})$ always holds, but in most cases makes classes $G_k(\mathcal{U}, \mathcal{S})$ unstable under elimination of clauses.

In [11] two hierarchies $(\Pi_k)_{k \in \mathbb{N}_0}, (\Upsilon_k)_{k \in \mathbb{N}_0}$ have been introduced; the basic motivations and the relations to our hierarchies are as follows:

1. We have $\Pi_k \cap \mathcal{USAT} = G_k^0(\mathcal{RHO})$ and $\Pi_k \cap \mathcal{SAT} \subseteq G_k^1(\mathcal{RHO})$ (with $\Pi_0 = \mathcal{RHO}$). Note that we do not have $\mathcal{U}_0 \subseteq \mathcal{RHO}$ here.
2. It is $\mathcal{RHO} \cap \mathcal{USAT} \subset G_1^0(\mathcal{U}_0)$ (Lemma 6.5, Part 4), while $\mathcal{RHO} \cap \mathcal{SAT}$ is not included in any $G_k^1(\mathcal{U}, \mathcal{S}_0)$. More generally we have $\Pi_k \cap \mathcal{USAT} \subset G_{k+1}^0(\mathcal{U}_0)$ for all $k \geq 0$.
3. So the choice of the oracle \mathcal{RHO} is less powerful on unsatisfiable instances than the choice of \mathcal{U}_0 (when going up one level in the hierarchy), while the special recognition of satisfiability for \mathcal{RHO} is (naturally) not captured by any level of the G_k -hierarchy, when using only the trivial satisfiability-oracle \mathcal{S}_0 (even using $\mathcal{U} = \mathcal{USAT}$ does not change this, since this only yields full handling of all forced assignments, while a satisfiable instance in \mathcal{RHO} might not have any forced assignment).
4. For $k \geq 1$ we have $\Pi_k \cap \mathcal{SAT} \subset G_k^1(\mathcal{RHO})$, where an example for $F \in G_k^1(\mathcal{RHO}) \setminus \Pi_k$ is given by $F := \{\{v\} \cup C : C \in F'\}$ for some $F' \in \mathcal{CLS} \setminus \Pi_k$ and $v \in \mathcal{VA} \setminus \text{var}(F')$. The point is that recognition for the $G_k(\mathcal{U}, \mathcal{S})$ -hierarchy includes satisfiability-decision at lower levels, and if one branch, here $\langle v \rightarrow 1 \rangle$, yields a satisfiable instance, then the other branch ($\langle v \rightarrow 0 \rangle$) is not inspected—which however is the case for Π_k .
5. \mathcal{RHO} is stable under application of partial assignments, and, that is its main feature, stable under removal of clauses. This yields that all Π_k are stable under removal of clauses, which is the main motivation for this choice of the base oracle.
6. \mathcal{U}_0 is not contained in any Π_k , and thus there are unsatisfiable clause-sets of hardness 0 not contained in any given Π_k .
7. Čepek and Kučera [11] considered also (shortly) the hierarchy $\Upsilon_k \subset \mathcal{CLS}$ ($k \in \mathbb{N}_0$), with $\Upsilon_k \cap \mathcal{USAT} = G_k^0(\mathcal{QHO})$ and $\Upsilon_k \cap \mathcal{SAT} \subseteq G_k^1(\mathcal{QHO})$, based on the stronger oracle $\mathcal{QHO} \supset \mathcal{RHO}$ of q-Horn clause-sets (again stable under application of partial assignments and removal of clauses). We have $\Upsilon_k \cap \mathcal{USAT} \subset G_{k+2}^0(\mathcal{U}_0)$ for all $k \geq 0$ (Lemma 6.5, Part 5).

By Lemma 6.4 we get:

Lemma 6.9 *For all $k \in \mathbb{N}_0$ we have $\Pi_k \subset \mathcal{UC}_{k+1}$ and $\Upsilon_k \subset \mathcal{UC}_{k+2}$ for the hierarchies Π_k, Υ_k introduced in [11].*

6.4 Determining Hardness Computationally

By the well-known computation of $\text{prc}_0(F)$ via resolution-closure we obtain:

Lemma 6.10 *Whether for $F \in \mathcal{CLS}$ we have $\text{hd}(F) = 0$ or not can be decided in polynomial time, namely $\text{hd}(F) = 0$ holds if and only if F is stable under resolution*

modulo subsumption (which means that for all resolvable $C, D \in F$ with resolvent R there exists $E \in F$ with $E \subseteq R$).

Thus if the hardness is known to be at most 1, we can compute it efficiently:

Corollary 6.11 Consider a class $\mathcal{C} \subseteq \mathcal{CLS}$ of clause-sets where $\mathcal{C} \subseteq \mathcal{UC}_1$ is known. Then for $F \in \mathcal{C}$ one can compute $\text{hd}(F) \in \{0, 1\}$ in polynomial time.

Examples for \mathcal{C} are given by $\mathcal{HO} \subset \mathcal{UC}_1$ (Lemma 6.5) and in Section 3.1. Another example class with known hardness is given by $2\text{-}\mathcal{CLS} \subset \mathcal{UC}_2$ (Lemma 6.5), and also here we can compute the hardness efficiently:

Lemma 6.12 For $F \in 2\text{-}\mathcal{CLS}$ one can compute $\text{hd}(F) \in \{0, 1, 2\}$ in polynomial time.

Proof One method is to observe that for elements of $2\text{-}\mathcal{CLS}$ the set of prime-implicates can be determined in polynomial time, while SAT-decision can be done in linear time. More efficient is the following:

1. Determine first whether F is satisfiable or not.
2. If F is satisfiable, then $\text{hd}(F) \in \{0, 1\}$ by Lemma 6.6, and whether $\text{hd}(F) = 0$ or not can be determined by Lemma 6.10.
3. If F is unsatisfiable, then it suffices to compute $r_0(F)$ and $r_1(F)$. □

See Theorem 7.5 for coNP-completeness of determining an upper bound on hardness.

7 The SLUR Hierarchy

We now define the $SLUR_k$ hierarchy, generalising $SLUR$ (recall Section 3.1) in a natural way, by replacing r_1 with r_k . In Section 7.1 we show $SLUR_k = \mathcal{UC}_k$, and as application obtain coNP-completeness of membership decision for \mathcal{UC}_k for $k \geq 1$. In Section 7.2 we determine the relations to the previous hierarchies $SLUR^*(k)$ and $CANON(k)$ as discussed in Section 3.2.

Definition 7.1 Consider $k \in \mathbb{N}_0$. For clause-sets $F, F' \in \mathcal{CLS}$ the relation $F \xrightarrow{SLUR_k} F'$ holds if there is $x \in \text{lit}(F)$ such that $F' = r_k(\langle x \rightarrow 1 \rangle * F)$ and $F' \neq \{\perp\}$. The transitive-reflexive closure is denoted by $F \xrightarrow{SLUR_k}_* F'$. The set of all fully reduced clause-sets reachable from F is denoted by

$$\text{slur}_k(F) := \left\{ F' \in \mathcal{CLS} \mid F \xrightarrow{SLUR_k}_* F' \wedge \neg \exists F'' \in \mathcal{CLS} : F' \xrightarrow{SLUR_k} F'' \right\}.$$

Finally the class of all clause-sets which are either identified by r_k to be unsatisfiable, or where by k -SLUR-reduction always a satisfying assignment is found, is denoted by $\mathcal{SLUR}_k := \{F \in \mathcal{CLS} : r_k(F) \neq \{\perp\} \Rightarrow \text{slur}_k(F) = \{\top\}\}$.

We have $SLUR_1 = SLUR$ (recall Definition 3.3). Note also the following simple properties for $F \in \mathcal{CLS}$:

1. $\top \in \text{slur}_k(F) \Leftrightarrow F \in \mathcal{SAT}$.

2. For $F' \in \text{slur}_k(F) \setminus \{\top\}$ we have $F' \in \text{USAT}$, and if $F \in \text{SAT}$, then $r_k(F') \neq \{\perp\}$.
3. If $F \in \text{SLUR}_k$, then $F \in \text{SAT}$ and $F \xrightarrow{\text{SLUR}_k}_* F'$ implies $F' \in \text{SAT}$.

Again we could define the transition relation in a less restricted way, as $F \xrightarrow{\text{SLUR}_k} \langle x \rightarrow 1 \rangle * F$ iff $r_k(\langle x \rightarrow 1 \rangle * F) \neq \perp$, and this would yield the same class SLUR_k .

Example 7.2 Some examples for $\text{SLUR}_2 \setminus \text{SLUR}_1$:

1. Consider the unsatisfiable clause-set $F := \{\{x, y\}, \{x, \bar{y}\}, \{\bar{x}, y\}, \{\bar{x}, \bar{y}\}\}$.
 - (a) $F \notin \text{SLUR}_1$ because F is unsatisfiable but $r_1(F) \neq \{\perp\}$.
 - (b) $F \in \text{SLUR}_2$ because $r_2(F) = \{\perp\}$.
2. Consider the satisfiable clause-set $F' := \{\{x_1, x_2\} \cup C \mid C \in F\}$.
 - (a) $F' \notin \text{SLUR}_1 = \text{SLUR}$ because $F' \xrightarrow{\text{SLUR}}_* F = \langle x_1, x_2 \rightarrow 0 \rangle * F'$, where $\text{slur}(F) = \{F\}$ and so $F \in \text{slur}(F')$.
 - (b) $F' \in \text{SLUR}_2$ because for any φ such that $F' \xrightarrow{\text{SLUR}_2}_* \varphi * F'$ and $F' \neq \top$ we have one of the following two cases:
 - i. $\varphi * F'$ is satisfiable, and so $\varphi * F' \notin \text{slur}_2(F)$.
 - ii. $\varphi * F'$ is unsatisfiable and so $\langle x_1 \rightarrow 0, x_2 \rightarrow 0 \rangle \subseteq \varphi$, but this contradicts the fact that $F' \xrightarrow{\text{SLUR}_2}_* \varphi * F'$. That is, after setting either x_1 or x_2 to 0, lookahead with r_2 detects unsatisfiability of $\varphi * F'$ and so one can never transition to $\varphi * F'$ from F' .

Therefore $\text{slur}_2(F') = \{\top\}$.

More generally we have $\{\{x_1, \dots, x_k\} \cup C \mid C \in F\} \in \text{SLUR}_2 \setminus \text{SLUR}_*(k)$ (recall Example 3.6).

Lemma 7.3 *We have for $F \in \text{CLS}$, $k \in \mathbb{N}_0$ and a partial assignment φ with $r_k(\varphi * F) \neq \{\perp\}$ that $F \xrightarrow{\text{SLUR}_k}_* r_k(\varphi * F)$ holds.*

Proof The assignments of φ can be performed via SLUR- k -transitions. □

7.1 SLUR = UC

For $F \in \text{UC}_k$ there is the following polynomial-time SAT decision: F is unsatisfiable iff $r_k(F) = \{\perp\}$. And a satisfying assignment can be found for satisfiable F via self-reduction, that is, probing variables, where unsatisfiability again is checked for by means of r_k . For $k = 1$ this means exactly that the nondeterministic “SLUR”-algorithm will not fail. And that implies that $F \in \text{SLUR}$ holds, where SLUR is the class of clause-sets where that algorithm never fails. So $\text{UC}_1 \subseteq \text{SLUR}$. Now it turns out, that actually this property characterises UC_1 , that is, $\text{UC}_1 = \text{SLUR}$ holds, which makes available the results on SLUR .

We now show that this equality between UC and SLUR holds in full generality for the UC_k and SLUR_k hierarchies.

Theorem 7.4 *For all $k \in \mathbb{N}_0$ holds $\text{SLUR}_k = \text{UC}_k$.*

Proof Consider $F \in \mathcal{CLS}$. We have to show $F \in \mathcal{SLUR}_k \Leftrightarrow \text{hd}(F) \leq k$. For $F \in \mathcal{USAT}$ this follows from the definitions, and thus we assume $F \in \mathcal{SAT}$.

First consider $F \in \mathcal{SLUR}_k$. Consider a partial assignment φ such that $\varphi * F \in \mathcal{USAT}$. We have to show $r_k(\varphi * F) = \{\perp\}$, and so assume $r_k(\varphi * F) \neq \{\perp\}$. It follows $F \xrightarrow{\text{SLUR}_k} r_k(\varphi * F)$ by Lemma 7.3, whence $r_k(\varphi * F) \in \mathcal{SAT}$ contradicting $\varphi * F \in \mathcal{USAT}$.

Now assume $\text{hd}(F) \leq k$, and we show $F \in \mathcal{SLUR}_k$, i.e., $\text{slur}_k(F) = \top$. Assume there is $F' \in \text{slur}_k(F) \setminus \{\top\}$. By Property 2 for Definition 7.1 we get $F' \in \mathcal{USAT}$ and $r_k(F') \neq \{\perp\}$. However by Lemma 6.5, Part 1 we get $\text{hd}(F') \leq k$, and thus $r_k(F') = \{\perp\}$. □

It seemed an essential feature of the class \mathcal{SLUR} , that its most natural definition is by the SLUR-algorithm; for example in [23] we find the quote “I find it interesting that the algorithm seems simpler than the conditions under which it is a decision procedure.” By Theorem 7.4 now we have a simple characterisation of these conditions, namely that unsatisfiability after instantiation is always detected by unit-clause propagation. Using the characterisation $\mathcal{SLUR} = \mathcal{UC}$, we can show coNP-completeness of hardness-determination:

Theorem 7.5 *For fixed $k \in \mathbb{N}$ the decision whether $\text{hd}(F) \leq k$ (i.e., whether $F \in \mathcal{UC}_k$, or, by Theorem 7.4, whether $F \in \mathcal{SLUR}_k$) is coNP-complete.*

Proof The decision whether $F \notin \mathcal{SLUR}_k$ is in NP by definition of \mathcal{SLUR}_k (or use Lemma 5.4). By Theorem 3 in [12] we have that \mathcal{SLUR} is coNP-complete, which by Lemma 6.3 can be lifted to higher k . □

7.2 Comparison to the Previous Hierarchies

The alternative hierarchies $\mathcal{SLUR}^*(k)$ and $\text{CANON}(k)$ (recall Section 3.2) do not generalise r_1 by r_k , but extend r_1 in various ways (maintaining linear-time computation for the (non-deterministic) transitions). In this way in [2, 12] rather complicated argumentations arise, in contrast to our elegant characterisation of the classes \mathcal{UC}_k in Theorem 5.7. As a consequence, we can give short proofs that the alternative hierarchies are subsumed by our hierarchy, while already the second level of our hierarchy is (naturally) not contained in any levels of these two hierarchies (naturally, since the time-exponent for deciding whether a (non-deterministic) transition can be done w.r.t. hierarchy \mathcal{SLUR}_k depends on k).

First we simplify and generalise the main result of [2], that $\text{CANON}(1) \subseteq \mathcal{SLUR}$. By definition we have $\text{CANON}(0) = \mathcal{UC}_0$.

Theorem 7.6 *For all $k \in \mathbb{N}_0$ we have:*

1. $\text{CANON}(k) \subseteq \mathcal{UC}_k$.
2. $\mathcal{UC}_1 \not\subseteq \text{CANON}(k)$ (and thus $\text{CANON}(k) \subset \mathcal{UC}_k$ for $k \geq 1$).

Proof By Theorem 5.7 and the fact, that the Horton–Strahler number of a tree is at most the height, we see that $\text{CANON}(k) \subseteq \mathcal{UC}_k$. That $\mathcal{UC}_1 \not\subseteq \text{CANON}(k)$ can be seen by observing that there are formulas in $\mathcal{HO} \cap \mathcal{USAT}$ with arbitrary

resolution-height complexity and so $\mathcal{HO} \not\subseteq \text{CANON}(k)$. By $\mathcal{HO} \subset \mathcal{UC}_1$ we get $\mathcal{UC}_1 \not\subseteq \text{CANON}(k)$. \square

Also the other hierarchy $\text{SLUR}^*(k)$ is strictly contained in our hierarchy:

Theorem 7.7 *For all $k \in \mathbb{N}_0$ we have:*

1. $\text{SLUR}^*(k) \subset \text{SLUR}_{k+1}$.
2. $\text{SLUR}_2 \not\subseteq \text{SLUR}^*(k)$.

Proof Part 1 follows most easily by using Lemma 6.4 together with the simple fact that $\text{slur}^*(k)(F) = \{F\}$ for $F \neq \top$ implies $r_{k+1}(F) = \{\perp\}$; for the strictness of the inclusion use Part 2. Part 2 follows from $\text{CANON}(2) \not\subseteq \text{SLUR}^*(k)$ (Lemma 13 in [2]), while by Theorem 7.6 we have $\text{CANON}(2) \subseteq \text{SLUR}_2$. \square

Part 1 of Theorem 7.7 can not be improved, since $\text{SLUR}^*(k)$ and SLUR_k are incomparable:

Lemma 7.8 *For $k \geq 2$ holds $\text{SLUR}^*(k) \not\subseteq \text{SLUR}_k$ and $\text{SLUR}_k \not\subseteq \text{SLUR}^*(k)$.*

Proof That $\text{SLUR}_k \not\subseteq \text{SLUR}^*(k)$ follows by Part 2 of Theorem 7.7. That $\text{SLUR}^*(k) \not\subseteq \text{SLUR}_k$ follows from the fact that for the full unsatisfiable clause-set F_k on k variables (i.e., containing all 2^k clauses of length k) we have $F_{k+1} \in \text{SLUR}^*(k)$ by Lemma 10 in [2] but $F_{k+1} \notin \text{SLUR}_k$ by Part 2 of Lemma 6.2. \square

8 Optimisation

We conclude by considering the question of finding, for an input-clause-set F , short equivalent clause-sets $F' \in \mathcal{UC}_k$ for fixed k . Definition 8.1 provides the appropriate notion of “irredundancy” via the notion of a “ k -base”, where irredundancy refers to both removal of literal occurrences and removal of clauses. In Theorem 8.3 we show that the problem is solvable in polynomial time for inputs $F \in 2\text{-}\mathcal{CLS}$, while in Theorem 8.4 we show that the problem is NP-complete even when restricting the input to Horn clause-sets with very few prime implicates.

Definition 8.1 A clause-set F is a k -base for some $k \in \mathbb{N}_0 \cup \{+\infty\}$ if $\text{hd}(F) \leq k$, and after removing any literal occurrence or any clause from F , the result F' is either not equivalent to F or has $\text{hd}(F') > k$.

Remarks:

1. Every k -base F is primal, that is, $F \subseteq \text{prc}_0(F)$.
2. A clause-set F is a 0-base iff $F = \text{prc}_0(F)$, while F is an ∞ -base iff F is primal and irredundant (removal of any clause yields a clause-set not equivalent to F).
3. For a given clause-set F , we consider the problem of computing a shortest (w.r.t. the number of clauses or the number of literal occurrences) equivalent k -base F' , which we call a **k -base for F** :
 - (a) By [42] for $k = \infty$ this problem is Σ_2 -complete.

- (b) A special case of interest here is when $F = \text{prc}_0(F)$, in which case $F' \subseteq F$ must hold. Since all prime implicates are given as input, for $k < \infty$ the decision problem whether F has a k -base of size at most k (k is part of the input) is now in NP. In Theorem 8.4 we will see that this decision problem is actually NP-complete, even under rather restricted circumstances.

Example 8.2 Consider the clause-set

$$F := \left\{ \underbrace{\{v_1, \bar{v}_3, \bar{v}_4\}}_{C_1}, \underbrace{\{v_2, v_3, \bar{v}_4\}}_{C_2}, \underbrace{\{v_2, \bar{v}_3, v_4\}}_{C_3}, \underbrace{\{\bar{v}_2, v_3, v_4\}}_{C_4}, \underbrace{\{v_1, v_3, v_4\}}_{C_5}, \underbrace{\{v_1, v_2\}}_{C_6} \right\}.$$

and clause-sets $F_1 := F \setminus \{C_5\}$ and $F_2 := F \setminus \{C_6\}$. We have that:

1. F is a 0-base, that is, $\text{prc}_0(F) = F$.
We have to show that F is closed under resolution modulo subsumption. We have the following possible resolutions in F with the associated subsuming clauses: $C_1 \diamond C_2 \supset C_6, C_1 \diamond C_3 \supset C_6, C_2 \diamond C_5 \supset C_6, C_3 \diamond C_5 \supset C_6, C_4 \diamond C_6 = C_5$.
2. F, F_1 and F_2 are the only k -bases ($k \in \mathbb{N}_0$) that are equivalent to F .
To show that there are no other k -bases equivalent to F we must show that all other subsets of F are not equivalent to F . It suffices to show that the clauses C_1, C_2, C_3, C_4 are irredundant (i.e., occur in all primal clause-sets equivalent to F) and the clause-set $F_3 := F \setminus \{C_5, C_6\}$ is not equivalent to F . The irredundancy of C_1, C_2, C_3, C_4 is seen by the fact that they are not obtained as resolvents. That F_3 is not equivalent to F follows from the fact that F_3 does not contain positive clauses while F does.
3. F_1 is a 1-base (and 2-base) and is equivalent to F but is not a 0-base.
We have $C_4 \diamond C_6 = C_5$ and thus $F_1 \models C_5$. To see $\text{hd}(F_1) = 1$, observe $\text{hd}(\varphi_{C_5} * F_1) = \text{hd}(\{\{\bar{v}_2\}, \{v_2\}\}) = 1$.
4. F_2 is a 2-base and is equivalent to F but is not a 1-base.
We have $(C_1 \diamond C_3) \diamond (C_2 \diamond C_5) = C_6$ and thus $F_2 \models C_6$. Furthermore $\text{hd}(\varphi_{C_6} * F_2) = \text{hd}(\{\{\bar{v}_3, \bar{v}_4\}, \{v_3, \bar{v}_4\}, \{\bar{v}_3, v_4\}, \{v_3, v_4\}\}) = 2$.
5. Thus F is neither a 1-base nor a 2-base.

Theorem 8.3 For clause-sets $F \in 2\text{-CLS}$ we can compute shortest-size (minimum number of clauses or minimum number of literal occurrences) equivalent k -bases F' for all $k \in \mathbb{N}_0 \cup \{+\infty\}$ in polynomial time as follows:

1. If F is unsatisfiable, then the best possibility is $F' := \{\perp\}$. So assume in the sequel that F is satisfiable.
2. If $F = \top$, then $F' := \top$. So assume in the sequel that $F \neq \top$.
3. If F has a forced literal x , then any k -base for F contains $\{x\}$, and we can split off x by considering an optimal k -base for $(x \rightarrow 1) * F$. So we can assume w.l.o.g. in the sequel that F has no forced literals. (Thus F as well as $\text{prc}_0(F)$ contains only clauses of length equal 2.)
4. Since all k -bases of F without new variables are subsets of $\text{prc}_0(F)$, when considering “shortest k -bases” now there is no differences between the measures

- c (number of clauses) and ℓ (number of literal occurrences), and we can just speak of “shortest k -bases”.
5. The (unique) 0-base of F , the set $\text{prc}_0(F) \in 2\text{-CLS}$ of all prime-implicates, can be computed in polynomial time by the methods discussed in Section 5.8 in [14].
 6. Every ∞ -base of F without new variables is a 1-base (Lemma 6.6), and thus w.r.t. k -bases for $k \in \mathbb{N}_0 \cup \{+\infty\}$ only the determination of shortest 1-bases is left, where the shortest 1-bases are precisely the smallest subsets of $\text{prc}_0(F)$ equivalent to F .
 7. Finally in Chapter 9 of [13] (affirmed in [30]) it is shown how to compute shortest equivalent sets of prime-implicates, and thus shortest 1-bases can be computed in polynomial time.

Theorem 8.4 Consider $k \in \mathbb{N}_0 \cup \{+\infty\}$.

1. Assume $k \geq 1$. The decision problem “For inputs $F \in \mathcal{HO}^+ \cap 3\text{-CLS}$ with $\text{prc}_0(F) = F$ and $m \in \mathbb{N}_0$, decide whether there is a k -base F' of F with $c(F') \leq m$.” (note that here $F' \subseteq F$ must hold) is NP-complete.
2. For $k = 0$ the decision problem “For input $F \in \mathcal{HO}$ and $m \in \mathbb{N}_0$, decide whether there is a k -base F' of F with $c(F) \leq m$.” is in P.

Proof For Part 2 one enumerates with polynomial delay the prime implicates of F (see Section 6.5 in [14] for efficient methods): if this process stops with at most m prime implicates found, then the answer is “yes”, otherwise the answer is “no”.

For Part 1 we first note that the problem is in NP, since all prime clauses are given, and $\text{hd}(F) \leq 1$. The heart of the completeness is Theorem 6.18 in [14], which states that “Horn minimisation w.r.t. the number of clauses remains NP-complete even if the input is restricted to cubic pure Horn expressions.”, plus the fact from the underlying report [9], that for the considered $G \in \mathcal{HO}^+ \cap 3\text{-CLS}$ all prime implicates are also of length at most 3, and thus we can take as input $F := \text{prc}_0(G) \in \mathcal{HO}^+ \cap 3\text{-CLS}$ (which can be computed in polynomial time). □

9 Conclusion and Outlook

We brought together two streams of research, one started by [20] in 1994, introducing \mathcal{UC} for knowledge compilation, and one started by [43] in 1995, introducing \mathcal{SLUR} for polytime SAT decision. Two natural generalisations, \mathcal{UC}_k and \mathcal{SLUR}_k have been provided, and the (actually surprising) identity $\mathcal{SLUR}_k = \mathcal{UC}_k$ provides both sides of the equation with additional tools. Various basic lemmas have been shown, providing a framework for elegant and powerful proofs. Regarding computational problems, we solved the most basic questions.

Our main future application, which brings the \mathcal{UC} -perspective and the \mathcal{SLUR} -perspective together, is in the area of “good SAT representations”; see Section 9.2 for more information. We consider the approach of representing a boolean function f via a clause-set $F \in \mathcal{UC}_k$ as the first beginning of what we envisage as a theory of good SAT representations.

We outline now what seems to us the most promising directions for future investigations (and where we already have partial results).

9.1 Propagation-Hardness

Complementary to “unit-refutation completeness” there is the notion of “propagation completeness”, as investigated in [7, 18]. This will be captured and generalised by a corresponding measure $\text{phd} : \mathcal{CLS} \rightarrow \mathbb{N}_0$ of “propagation-hardness”, defined as follows:

Definition 9.1 For $F \in \mathcal{CLS}$ we define the **propagation-hardness** (for short “p-hardness”) $\text{phd}(F) \in \mathbb{N}_0$ as the minimal $k \in \mathbb{N}_0$ such that for all partial assignments $\varphi \in \mathcal{PASS}$ we have

$$r_k(\varphi * F) = r_\infty(\varphi * F).$$

Now the class \mathcal{PC} of “propagation-complete clause-sets” can be properly generalised:

Definition 9.2 For $k \in \mathbb{N}_0$ let $\mathcal{PC}_k := \{F \in \mathcal{CLS} : \text{phd}(F) \leq k\}$ (the class of **propagation-complete clause-sets of level k**).

We have $\mathcal{PC} = \mathcal{PC}_1$. These classes lie (strictly) between the \mathcal{UC}_k -classes:

Lemma 9.3 For $k \in \mathbb{N}_0$ we have $\mathcal{PC}_k \subset \mathcal{UC}_k \subset \mathcal{PC}_{k+1}$.

9.2 Good Representations of Boolean Functions

The real power of SAT representations comes with **new variables**. Expressive power and limitations of “good representations” have to be studied. In the SAT-context the most useful notion of “representation” of a boolean function f seems to be Σ_1 -QCNF-representations, that is, clause-sets F with $\text{var}(f) \subseteq \text{var}(F)$, where the new variables (in $\text{var}(F) \setminus \text{var}(f)$) are implicitly existentially quantified—in other words, the satisfying assignments of F projected to the variables of f are precisely the satisfying assignments of f ; see [10] for some general results. The restricted representations we already considered in Section 1.4 are those without new variables, that is, where $\text{var}(F) = \text{var}(f)$.

Additional conditions on F are needed to get “effective” representations, since in general the evaluation of F for a total assignment for f is an NP-problem. Strong representations are those with bounded hardness. Strengthening Conjecture 1.1 from the introduction, we conjecture that also with new variables the power of representing boolean functions increases when allowing higher hardness:

Conjecture 9.4 For every $k \in \mathbb{N}_0$ the set of sequences $(f_n)_{n \in \mathbb{N}}$ of boolean functions having sequences $(F_n)_{n \in \mathbb{N}}$ of polysize-representations of p -hardness at most k (i.e., $\text{phd}(F_n) \leq k$ for all n) is strictly smaller than those having polysize-representations of hardness at most k (i.e., $\text{hd}(F_n) \leq k$ for all n), which in turn is strictly smaller than those having polysize-representations of p -hardness at most $k + 1$ (i.e., $\text{phd}(F_n) \leq k + 1$ for all n).

We wish to remind the reader of the open problem mentioned in Section 1.5 about the existence of a polysize-representation of bounded hardness for affine boolean functions.

We need to emphasise here that representations F of boolean functions f with $\text{hd}(F) \leq k$ fulfil an **absolute condition**, that is, we can determine unsatisfiability by r_k for *arbitrary* partial assignments, not just those using only the variables of f . When only asking for this **relative condition** (currently the standard, posing conditions only on variables occurring in the represented boolean function f , ignoring the new variables of F), then by generalising [5] we can show that the hierarchies collapse to the first level. This is due to the “uncontrolled” use of the new variables (the relative condition doesn’t pose conditions on them). See [8] for a study on \mathcal{UC} together with the relative condition.

9.3 Applications to Cryptanalysis

As an application of the theory of “good representations” we consider cryptanalytic problems, especially attacking AES/DES, as preliminary discussed in [25, 26]. For the experimental evaluation we consider the various boolean functions (“constraints”) used by these ciphers, most prominently the “S-boxes”, and systematically search for short representations of hardness 0, 1, 2 and p-hardness 1, 2. Various solvers are then run on the SAT-problems obtained by plaintext-/ciphertext pairs (where the task is to determine the key). The strengthened inference power seems especially interesting for the combination of look-ahead (“tree-resolution based”) and conflict-driven (“dag-resolution based”) SAT solvers as introduced in [33].

9.4 Relativised Hardness

Generalising [5] we can show that for example the satisfiable pigeonhole formulas PHP_m^m do not have polysize representations of bounded hardness even for the relative condition. One way to overcome this barrier is to generalise the theory started here via the use of oracles as in [36, 37] (recall Section 6.3), and then employing oracles which can handle pigeonhole formulas. The basic definitions are as follows.

Definition 9.5 A **valid oracle** for generalised unit-clause propagation is some $\mathcal{U} \subseteq \mathcal{USAT}$ with $\{\perp\} \in \mathcal{U}$ which is stable under application of partial assignments. The oracle is **strong** if $\mathcal{U}_0 \subseteq \mathcal{U}$, where $\mathcal{U}_0 := \{F \in \mathcal{CLS} : \perp \in F\}$.

Consider $k \in \mathbb{N}_0$. In [36] the reduction $r_k^{\mathcal{U}} : \mathcal{CLS} \rightarrow \mathcal{CLS}$ has been defined. An equivalent definition (generalising Definition 4.3) is as follows for $F \in \mathcal{CLS}$:

$$r_0^{\mathcal{U}}(F) := \begin{cases} \{\perp\} & \text{if } F \in \mathcal{U} \\ F & \text{otherwise} \end{cases}$$

$$r_{k+1}^{\mathcal{U}}(F) := \begin{cases} r_{k+1}^{\mathcal{U}}((x \rightarrow 1) * F) & \text{if } \exists x \in \text{lit}(F) : r_k^{\mathcal{U}}((x \rightarrow 0) * F) = \{\perp\} \\ F & \text{otherwise} \end{cases} .$$

Note $r_k = r_k^{\mathcal{U}_0}$. Generalising Definitions 5.1 and 5.5:

Definition 9.6 Consider a valid oracle \mathcal{U} . The **hardness** $\text{hd}_{\mathcal{U}}(F) \in \mathbb{N}_0$ (“hardness with oracle \mathcal{U} ”) of an unsatisfiable $F \in \mathcal{CLS}$ is the minimal $k \in \mathbb{N}_0$ such that $r_k^{\mathcal{U}}(F) = \{\perp\}$. And for general $F \in \mathcal{CLS}$ we define $\text{hd}_{\mathcal{U}}(\top) := 0$, while for $F \neq \top$ let

$$\text{hd}_{\mathcal{U}}(F) := \max \{ \text{hd}_{\mathcal{U}}(\varphi * F) : \varphi \in \mathcal{PASS} \wedge \varphi * F \in \mathcal{USAT} \} \in \mathbb{N}_0.$$

We have $\text{hd} = \text{hd}_{\mathcal{U}_0}$, and if \mathcal{U} is strong then for all F holds $\text{hd}_{\mathcal{U}}(F) \leq \text{hd}(F)$. An interesting oracle \mathcal{U} (with polytime membership decision) is given by the class of unsatisfiable clause-sets defined in [19] via semidefinite programming, for which we get $\text{hd}_{\mathcal{U}}(\text{PHP}_m^m) = 0$.

9.5 Width-Based Hardness

The basic idea is to use width-restricted resolution instead of nested input resolution, in order to increase inference power from tree-resolution to dag-resolution. A basic weakness of the standard notion of width-restricted resolution, which demands that *both* parent clauses must have length at most k for some fixed $k \in \mathbb{N}_0$ (the “width”), is that even Horn clause-sets require unbounded width in this sense. The correct solution, as investigated and discussed in [36, 37], is to use the notion of “ k -resolution” as introduced in [34], where only *one* parent clause needs to have length at most k (thus properly generalising unit-resolution).

Definition 9.7 Consider $k \in \mathbb{N}_0$.

- Two resolvable clauses C, D are **k -resolvable** if $|C| \leq k$ or $|D| \leq k$.
- We use $F \vdash^k C$ if there is a resolution proof R of some $C' \subseteq C$ from F such that all resolutions in R are k -resolutions.

This allows us now to define “width-hardness” (accordingly the “hardness” only studied in this paper can be called “tree-hardness”):

Definition 9.8 For $F \in \mathcal{USAT}$ let $\text{whd}(F) \in \mathbb{N}_0$ be the minimal $k \in \mathbb{N}_0$ such that $F \vdash^k \perp$ holds. And for $F \in \mathcal{CLS}$ let $\text{whd}(F) \in \mathbb{N}_0$ be the minimal $k \in \mathbb{N}_0$ such that for all partial assignments φ holds $\varphi * F \in \mathcal{USAT} \Rightarrow \varphi * F \vdash^k \perp$.

We have $\text{whd}(F) = k \Leftrightarrow \text{hd}(F) = k$ for $k \in \{0, 1\}$, while in general $\text{whd}(F) \leq \text{hd}(F)$ holds (for all $F \in \mathcal{CLS}$).

Conjecture 9.9 For every $k \in \mathbb{N}_0$ the set of families of boolean functions having polysize representations of width-hardness at most k is strictly smaller than those having polysize-representations of width-hardness at most $k + 1$. For $k \geq 1$ families showing the separation can be chosen such that they have unbounded hardness.

Finally we mention that, as in Section 9.4, we also have a relativised version $\text{whd}_{\mathcal{U}}$, based on relativised k -resolution as studied in [36, 37].

References

1. Ansótegui, C., Bonet, M.L., Levy, J., Manyà, F.: Measuring the hardness of SAT instances. In: Fox, D., Gomes, C. (eds.) *Proceedings of the 23th AAAI Conference on Artificial Intelligence (AAAI-08)*, pp. 222–228 (2008)
2. Balyo, T., Gurský, Š., Kučera, P., Vlček, V.: On hierarchies over the SLUR class. In: *Twelfth International Symposium on Artificial Intelligence and Mathematics (ISAIM 2012)* (2012). Available at <http://www.cs.uic.edu/bin/view/Isaim2012/AcceptedPapers>
3. Barrett, C., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds) *Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications*, vol. 185, chapter 26, pp. 825–885. IOS Press (2009). ISBN 978-1-58603-929-5
4. Bessiere, C.: Constraint propagation. In: Rossi, F., van Beek, P., Walsh, T. (eds.) *Handbook of Constraint Programming. Foundations of Artificial Intelligence*, chapter 3, pp. 29–83. Elsevier (2006). ISBN 0-444-52726-5
5. Bessiere, C., Katsirelos, G., Narodytska, N., Walsh, T.: Circuit complexity and decompositions of global constraints. In: *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pp. 412–418 (2009)
6. Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds.): *Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications*, vol. 185. IOS Press (2009). ISBN 978-1-58603-929-5
7. Bordeaux, L., Marques-Silva, J.: Knowledge compilation with empowerment. In: Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., Turán, G. (eds.) *SOFSEM 2012: Theory and Practice of Computer Science. Lecture Notes in Computer Science*, vol. 7147, pp. 612–624. Springer (2012)
8. Bordeaux, L., Janota, M., Marques-Silva, J., Marquis, P.: On unit-refutation complete formulae with existentially quantified variables. In: *Knowledge Representation 2012 (KR 2012)*. Association for the Advancement of Artificial Intelligence (AAAI Press) (2012)
9. Boros, E., Čeppek, O.: On the complexity of Horn minimization. Technical Report RRR 1-94, Rutcor Research Report (1994)
10. Bubeck, U., Büning, H.K.: The power of auxiliary variables for propositional and quantified boolean formulas. *Stud. Log.* **3**(3), 1–23 (2010)
11. Čeppek, O., Kučera, P.: Known and new classes of generalized Horn formulae with polynomial recognition and SAT testing. *Discrete Appl. Math.* **149**, 14–52 (2005)
12. Čeppek, O., Kučera, P., Vlček, V.: Properties of SLUR formulae. In: Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., Turán, G. (eds.) *SOFSEM 2012: Theory and Practice of Computer Science. LNCS Lecture Notes in Computer Science*, vol. 7147, pp. 177–189. Springer (2012)
13. Chang, T.: Horn formula minimization. Master’s thesis, Rochester Institute of Technology (2004)
14. Crama, Y., Hammer, P.L.: Boolean functions: theory, algorithms, and applications. In: *Encyclopedia of Mathematics and Its Applications*, vol. 142. Cambridge University Press (2011). ISBN 978-0-521-84751-3
15. Creignou, N., Kolaitis, P., Vollmer, H. (eds.): *Complexity of Constraints: An Overview of Current Research Themes. Lecture Notes in Computer Science (LNCS)*, vol. 5250. Springer (2008). ISBN-10 3-540-92799-9
16. Dantsin, E., Hirsch, E.A.: Worst-case upper bounds. In: Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds.) *Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications*, vol. 185, chapter 12, pp. 403–424. IOS Press (2009). ISBN 978-1-58603-929-5
17. Darwiche, A., Marquis, P.: A knowledge compilation map. *J. Artif. Intell. Res.* **17**, 229–264 (2002)
18. Darwiche, A., Pipatsrisawat, K.: On the power of clause-learning SAT solvers as resolution engines. *Artif. Intell.* **175**(2), 512–525 (2011)
19. de Klerk, E., van Maaren, H., Warners, J.P.: Relaxations of the satisfiability problem using semidefinite programming. *J. Autom. Reason.* **24**, 37–65 (2000)
20. del Val, A.: 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)*, pp. 551–561 (1994)
21. Franco, J.: Relative size of certain polynomial time solvable subclasses of satisfiability. In: Du, D., Gu, J., Pardalos, P.M. (eds.) *Satisfiability Problem: Theory and Applications (DIMACS Workshop March 11–13, 1996)*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 35, pp. 211–223. American Mathematical Society (1997). ISBN 0-8218-0479-0

22. Franco, J., Martin, J.: A history of satisfiability. In: Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, vol. 185, chapter 1, pp. 3–74. IOS Press (2009). ISBN 978-1-58603-929-5
23. Franco, J., Schlipf, J.: 1997 final report: Describing new results under the research project entitled Complexity of algorithms for problems in propositional logic. Covering the period January 1, 1994–march 31, 1997. Technical report, University of Cincinnati and Office of Naval Research (1997). Available at <http://www.dtic.mil/docs/citations/ADA325949>
24. Franco, J., Van Gelder, A.: A perspective on certain polynomial-time solvable classes of satisfiability. *Discrete Appl. Math.* **125**, 177–214 (2003)
25. Gwynne, M., Kullmann, O.: Towards a better understanding of hardness. In: The Seventeenth International Conference on Principles and Practice of Constraint Programming (CP 2011): Doctoral Program Proceedings, pp. 37–42 (2011). Proceedings available at http://www.dmi.unipg.it/cp2011/downloads/dp2011/DP_at_CP2011.pdf
26. Gwynne, M., Kullmann, O.: Towards a better understanding of SAT translations. In: Berger, U., Therien, D. (eds.) Logic and Computational Complexity (LCC'11), as part of LICS 2011 (2011). 10 pp., available at <http://www.cs.swansea.ac.uk/lcc2011/>
27. Gwynne, M., Kullmann, O.: Generalising unit-refutation completeness and SLUR via nested input resolution. Technical Report [arXiv:1204.6529v5](https://arxiv.org/abs/1204.6529v5) [cs.LO], arXiv (2013)
28. Gwynne, M., Kullmann, O.: Generalising and unifying SLUR and unit-refutation completeness. In: van Emde Boas, P., Groen, F.C.A., Italiano, G.F., Nawrocki, J., Sack, H. (eds.) SOFSEM 2013: Theory and Practice of Computer Science. Lecture Notes in Computer Science (LNCS), vol. 7741, pp. 220–232. Springer (2013). doi:10.1007/978-3-642-35843-2_20
29. Gwynne, M., Kullmann, O.: Towards a theory of good SAT representations. Technical Report [arXiv:1302.4421](https://arxiv.org/abs/1302.4421) [cs.AI], arXiv (2013)
30. Hemaspaandra, E., Schnoor, H.: Minimization for generalized boolean formulas. In: Walsh, T. (ed.) Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, vol. 1, pp. 566–571. AAAI Press (2011)
31. Henschen, L.J., Wos, L.: Unit refutations and Horn sets. *J. Assoc. Comput. Mach.* **21**(4), 590–605 (1974)
32. Heule, M.J.H., van Maaren, H.: Look-ahead based SAT solvers. In: Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, vol. 185, chapter 5, pp. 155–184. IOS Press (2009). ISBN 978-1-58603-929-5
33. Heule, M.J.H., Kullmann, O., Wieringa, S., Biere, A.: Cube and conquer: guiding CDCL SAT solvers by lookaheads. In: Eder, K., Lourenço, J., Shehory, O. (eds.) Hardware and Software: Verification and Testing (HVC 2011). Lecture Notes in Computer Science (LNCS), vol. 7261, pp. 50–65. Springer (2012). doi:10.1007/978-3-642-34188-5_8. <http://cs.swan.ac.uk/~csoliver/papers.html#CuCo2011>
34. Kleine Büning, H.: On generalized Horn formulas and k -resolution. *Theor. Comput. Sci.* **116**, 405–413 (1993)
35. Kleine Büning, H., Kullmann, O.: Minimal unsatisfiability and autarkies. In: Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, vol. 185, chapter 11, pp. 339–401 (2009). ISBN 978-1-58603-929-5. doi:10.3233/978-1-58603-929-5-339
36. Kullmann, O.: 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) (1999)
37. Kullmann, O.: Upper and lower bounds on the complexity of generalised resolution and generalised constraint satisfaction problems. *Ann. Math. Artif. Intell.* **40**(3–4), 303–352 (2004)
38. Kullmann, O.: Present and future of practical SAT solving. In: Creignou, N., Kolaitis, P., Vollmer, H. (eds.): Complexity of Constraints: An Overview of Current Research Themes. Lecture Notes in Computer Science (LNCS), vol. 5250, pp. 283–319. Springer (2008). ISBN-10 3-540-92799-9. doi:10.1007/978-3-540-92800-3_11
39. Kullmann, O.: Fundaments of branching heuristics. In: Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, vol. 185, chapter 7, pp. 205–244. IOS Press (2009). ISBN 978-1-58603-929-5. doi:10.3233/978-1-58603-929-5-205
40. Laitinen, T., Junttila, T., Niemelä, I.: Classifying and propagating parity constraints. In: Milano, M. (ed.) Principles and Practice of Constraint Programming—CP 2012. Lecture Notes in Computer Science (LNCS), vol. 7514, pp. 357–372. Springer (2012)

41. Nordström, J.: Pebble games, proof complexity, and time-space trade-offs. In: Logical Methods in Computer Science (2013, to appear)
42. Schaefer, M., Umans, C.: Completeness in the polynomial-time hierarchy: a compendium. SIGACT News **33**(3), 32–49 (2002)
43. Schlipf, J.S., Annexstein, F.S., Franco, J.V., Swaminathan, R.P.: On finding solutions for extended Horn formulas. Inf. Process. Lett. **54**, 133–137 (1995)
44. van Maaren, H.: A short note on some tractable cases of the satisfiability problem. Inf. Comput. **158**(2), 125–130 (2000)