Trading inference effort versus size in CNF Knowledge Compilation

Matthew Gwynne and Oliver Kullmann

Abstract

Knowledge Compilation (KC) studies compilation of boolean functions \(f\) into some formalism \(F\), which allows to answer all queries of a certain kind in polynomial time. Due to its relevance for SAT solving, we concentrate on the query type “clausal entailment” (CE), i.e., whether a clause \(C\) follows from \(f\) or not, and we consider subclasses of CNF, i.e., clause-sets \(F \in \mathcal{CLS}\) with special properties (CNF itself is not suitable for CE queries unless \(P = NP\)). In this report we do not allow auxiliary variables (except of the Outlook), and thus \(F\) needs to be equivalent to \(f\).

We consider the hierarchies \(\mathcal{UC}_k \subseteq \mathcal{WC}_k \subset \mathcal{CLS}\) (\(k \in \mathbb{N}_0\)), which were introduced in \([26, 27]\), and where each level allows CE queries. The first two levels are well-known classes for KC, namely \(\mathcal{UC}_0 = \mathcal{WC}_0\) is the same as PI as studied in KC, that is, \(f\) is represented by the set of all prime implicates, while \(\mathcal{UC}_1 = \mathcal{WC}_1\) is the same as \(\mathcal{UC}\), the class of unit-refutation complete clause-sets introduced in \([20]\). We show that for each \(k\) there are (sequences of) boolean functions with polysize representations in \(\mathcal{UC}_{k+1}\), but with an exponential lower bound on representations in \(\mathcal{WC}_k\). Such a separation was previously only know for \(k = 0\). We also consider \(\mathcal{PC} \subset \mathcal{UC}\), the class of propagation-complete clause-sets introduced in \([52, 11]\). Strengthening \([2]\), we show that there are (sequences of) boolean functions with polysize representations in \(\mathcal{UC}\), while there is an exponential lower bound for representations in \(\mathcal{PC}\). These separations are steps towards a general conjecture determining the representation power of the hierarchies \(\mathcal{PC}_k \subset \mathcal{UC}_k \subset \mathcal{WC}_k\). The strong form of this conjecture also allows auxiliary variables, as discussed in depth in the Outlook.

Contents

1 Introduction
  1.1 Hierarchies for CNF Knowledge Compilation
  1.2 Mapping the hierarchies
  1.3 Understanding the structure of satisfiable clause-sets
  1.4 Finding relatively hard boolean functions
  1.5 Relevance of these hierarchies for SAT solving
  1.6 Overview on results
1 Introduction

Boolean functions \( f : \{0, 1\}^n \to \{0, 1\} \) are fundamental objects of computer science, and many fields are concerned with their representation. In Knowledge Compilation (KC; see [18] for a general overview), \( f \) is given by some propositional formula (theory), and is to be compiled (off-line, that is, complex computations are possible here) into some \( F \) belonging to some target language, such that a large number of queries of a certain kind can be answered efficiently (using \( F \)).

A natural target language is CNF (conjunctive normal forms), for which we write \( F \in CLS \), where \( CLS \) is the class of all clause-sets, interpreted as CNFs. A basic subclass is PI, that is \( F \) is the (precisely) the set of all prime implicates of some boolean function \( f \) (this is a true normal form for \( f \), since it is unique and identifies \( f \)). Now in general not all prime implicates are needed, if additional

1) See [17] for the basic theory, [16] for an overview on their applications, and [35] for the complexity theory of their circuit representations.
mechanisms are used to answer queries. This led to the introduction of the class $\mathcal{UC}$ of “unit-refutation complete clause-sets” in [23], where the defining property of $F \in \mathcal{UC}$ is that if instantiation, that is, applying a partial assignment $\varphi$ to $F$, resulting in the clause-set $\varphi \ast F$, yields an unsatisfiable $\varphi \ast F$, then this is detected by unit-clause propagation. Using $r_1 : \mathcal{CLS} \to \mathcal{CLS}$ for the process of unit-clause propagation, detection of unsatisfiability means $\perp \in r_1(\varphi \ast F)$, where $\perp$ is the empty clause (while the defining property of IP is that in that case we already have $\perp \in \varphi \ast F$). It is shown in [23] that there are short clause-sets in $\mathcal{UC}$ with an exponential number of prime implicates (and so the equivalent representation in IP is very large).

The well-known case of full failed-literal elimination is precisely $r_1$ as an umbrella class for efficient SAT solving. [14, 3] extended this class in various ways to hierarchies covering all of CNF (all clause-sets). These hierarchies were unified and strengthened in [26, 27] to the classes $\mathcal{SLUR}$, using generalised unit-clause propagation $r_k : \mathcal{CLS} \to \mathcal{CLS}$ as introduced in [10]. The well-known case of full failed-literal elimination is precisely $r_2$, which applies a reduction $F \dashv \vdash (x \to 1) \ast F$ as long as there is a literal $x$ with $\perp \in r_1((x \to 0) \ast F)$, and $r_k$ is the natural generalisation to arbitrary $k$. The class $\mathcal{SLUR}_k$ is the class of all clause-sets $F$, where either $\perp \in r_k(F)$, or else one is guaranteed to find a

\footnote{Note that $F \models C$ holds iff $\varphi_C \ast F$ is unsatisfiable, where $\varphi_C$ sets all literals in $C$ to 0. So by the definition of $\mathcal{UC}$, the query-type “clausal entailment” (CE) is directly handled by $r_1$.}

\footnote{See [5] for an overview.}

\footnote{To be fully precise, $\mathcal{UC}_0$ is the class of clause-sets such that after elimination of subsumed clauses we obtain an element of PI.}

1.1 Hierarchies for CNF Knowledge Compilation

A basic task of KC is to find shortest (or short) representations in the target class. This has also applications in the area of “SAT solving”, which is about deciding satisfiability of propositional formulas, mostly in CNF. Often the translation starts with a set of boolean constraints (in fact boolean functions), and size of the translation is a basic criterion to be optimised. Furthermore, the target class should be “easy” for SAT solving. The quest for such classes of clause-sets with polynomial-time SAT-decision led to the hierarchy $\mathcal{UC}_k \subset \mathcal{CLS}$, $k \in \mathbb{N}_0$, with $\mathcal{UC}_k \subset \mathcal{UC}_{k+1}$ and $\bigcup_k \mathcal{UC}_k = \mathcal{CLS}$, where $\mathcal{UC}_0 = \text{PI}$, and $\mathcal{UC}_1 = \mathcal{UC}$, by the following development.

The class $\mathcal{SLUR}$ (“Single Lookahead Unit Resolution”) was introduced in [23] as an umbrella class for efficient SAT solving. [4, 3] extended this class in various ways to hierarchies covering all of CNF (all clause-sets). These hierarchies were unified and strengthened in [26, 27] to the classes $\mathcal{SLUR}_k$, with $\mathcal{SLUR}_1 = \mathcal{SLUR}$, using generalised unit-clause propagation $r_k : \mathcal{CLS} \to \mathcal{CLS}$ as introduced in [10].
satisfying assignment by choosing any literal \( x \) with \( \perp \notin r_k(\langle x \rightarrow 1 \rangle \ast F) \), reducing \( F \sim \langle x \rightarrow 1 \rangle \ast F \), and repeating this process. Using \( r_k \), we can also define \( UC_k \) as the class of clause-sets \( F \) such that for each partial assignment \( \varphi \) with unsatisfiable \( \varphi \ast F \) holds \( \perp \in r_k(\varphi \ast F) \). A basic result of [24, 27] is that \( SLUR_k = UC_k \) holds for all \( k \in \mathbb{N} \), which further motivates the claim that representation of boolean functions via \( UC_k \) has special relevance for finding good SAT translations.

The basic hierarchy \( UC_k \) had two offsprings, the stricter hierarchy \( PC_k \) and the wider hierarchy \( WC_k \). Generalising the class \( PC \subseteq UC \) of “unit-propagation complete clause-sets”, introduced in [11] (using ideas from [22]), the interleaving hierarchy \( PC_k \), with \( PC_0 \subseteq UC_0 \subseteq PC_1 \subseteq UC_1 \subseteq \ldots \) was defined in [27], and further studied in [24]. The elements of \( PC_k \) are those clause-sets \( F \) such that for each partial assignment \( \varphi \) either \( \perp \in r_k(\varphi \ast F) \) holds or otherwise \( r_k(\varphi \ast F) \) does not have any forced assignments. The hierarchy \( WC_k \) with \( WC_0 = UC_0 \), \( WC_1 = UC_1 \) and \( WC_k \supset UC_k \) for \( k \geq 2 \), also defined in [27], and further studied in [30], is defined as the class of clause-sets \( F \) such that for each partial assignment \( \varphi \) with unsatisfiable \( \varphi \ast F \) the inconsistency of \( \varphi \ast F \) can be derived by \( k \)-resolution, that is, resolution where for each resolution step at least one parent clause has length at most \( k \).

In this report we consider these hierarchies \( PC_k, UC_k, WC_k \) for the purpose of KC, representing boolean functions by equivalent clause-sets in one of these classes. Conjecture 1.1 in [27] says that there are boolean functions with short equivalent clause-sets in \( UC_{k+1} \), but without short equivalent clause-sets in \( UC_k \), for each \( k \). While Conjecture 9.9 in [27] says, when considered for the case without auxiliary variables, the same for the hierarchy \( WC_k \). We show both separations together, in a stronger form, in Theorem 6.14, namely we show that there are short clause-sets in \( UC_{k+1} \) which have no short equivalent clause-sets in \( WC_k \). Furthermore we show that there are short clause-sets in \( UC \) without equivalent short clause-sets in \( PC \).

1.2 Mapping the hierarchies

Our separation results show parts of a general conjecture, which determines the relations between the classes of the three hierarchies \( PC_k, UC_k, WC_k \) regarding their expressive power w.r.t. equivalence. First we need some definitions:

- For a clause-set \( F \in CLS \) we use \( n(F) := \text{var}(F) \) for the number of variables and \( \ell(F) := \sum_{C \in F} |C| \) for the number of literal occurrences.

- A sequence \( (F_n)_{n \in \mathbb{N}} \) of clause-sets is equivalent to a sequence of \( (F_n)_{n \in \mathbb{N}} \), if \( F'_n \) is equivalent to \( F_n \) for each \( n \in \mathbb{N} \).

Now we can define precisely what it means that a class \( C \) of clause-sets can be more succinct than another class \( C' \):

**Definition 1.1** For \( C, C' \subseteq CLS \) the relation \( C' \rightleftharpoons C \) ("\( C' \) does not simulate \( C \"\)) holds if there is a sequence \( (F_n)_{n \in \mathbb{N}} \) in \( C \) (i.e., \( F_n \in C \)) such that \( n(F_n) = n \) and \( F_n \) is computable in time \( n^{O(1)} \), and such that there is no equivalent sequence \( (F'_n)_{n \in \mathbb{N}} \) in \( C' \) with \( \ell(F'_n) = n^{O(1)} \).

The main conjecture (weak form) now says, that the subset-relations between the classes we consider already determine their expressive power (while the strong form, Conjecture 9.11, also allows the use of auxiliary variables, and is discussed in the conclusions):
Conjecture 1.2 (Main Conjecture, weak form) For $C, C' \in \{PC_k, UC_k, WC_k : k \in \mathbb{N}_0\}$ we have $C \rightarrow C'$ if and only if $C' \not\subseteq C$.

If follows from Conjecture 1.2 that for these classes $C, C'$ there is a polytime-computable map translating every clause-set in $C$ into an equivalent clause-set in $C'$ if and only if $C \subseteq C'$ (where the map is just the identity). The relation $C \rightarrow C'$ is stronger than $C \not< C'$, where $C \not< C'$ is the relation “$C$ is at least as succinct as $C'$” as defined for example in [18], since we do not require that a single polynomial regulates the size-relation between representations via these classes (as in $C \leq C'$), but for every sequence there can be another polynomial (and moreover, we only consider “simple sequences”).

Our main result, Theorem 6.14, is $WC_k \not\rightarrow UC_{k+1}$ for every $k \in \mathbb{N}_0$, that is, there are polysize sequences in $UC_{k+1}$ such that no equivalent polysize sequences exist in $WC_k$ (moreover we show an exponential separation). We also show $PC \not\rightarrow UC$ (Theorem 8.1; again, we show in fact an exponential separation). The remaining open cases of Conjecture 1.2 are discussed in Subsection 9.1.

1.3 Understanding the structure of satisfiable clause-sets

To be able to prove properties about all equivalent representations of some clause-set $F$, we must be able to understand its combinatorial structure in relation to the set of all its prime implicates. The notion of minimal unsatisfiability (MU) and minimally unsatisfiable subsets (MUS) is important in understanding the combinatorics of unsatisfiable clause-sets (see [29, 50]). To understand the structure of satisfiable clause-sets and their associated boolean functions, we now consider the concept of “minimal premise sets” (MPS) introduced in [46]. The notion of MPS generalises that of MU by considering clause-sets $F$ which are minimal w.r.t implying any clause $C$ rather than just those implying $\perp$. And accordingly we consider the minimal-premise subsets (MPSS) of a clause-set $F$.

Every prime implicate $C$ of a clause-set $F$ has an associated MPSS (just consider the minimal sub-clause-set of $F$ that implies $C$), but not every MPSS of $F$ yields a prime implicate (e.g., consider the MPSS $\{C\}$ for some non-prime clause $C \in F$). However, by “doping” the clause-set, i.e., adding a new unique variable to every clause, every clause in an MPSS $F'$ makes a unique contribution to its derived clause $C$. This results in a new clause-set $D(F)$ which has an exact correspondence between its minimal premise sets (which are (essentially) also those of $F$) and its prime implicates. In this way, by considering clause-sets $F$ with a very structured set of minimal premise subsets, we can derive clause-sets $D(F)$ with very structured set of prime implicates.

1.4 Finding relatively hard boolean functions

A sequence $(f_h)_{h \in \mathbb{N}}$ of boolean functions, which separates $UC_{k+1}$ from $UC_k$ w.r.t. clause-sets equivalent to $f_h$ in $UC_{k+1}$ resp. $UC_k$, should have the following properties:

1. **A large number of prime implicates**: the number of prime implicates for $f_h$ should at least grow super-polynomially in $h$, since otherwise already the set of prime implicates is a small clause-set in $UC_0$ equivalent to $f_h$.

2. **Easily characterised prime implicates**: the prime implicates of $f_h$ should be easily characterised, since otherwise we can not understand how clause-sets equivalent to $f_h$ look like.

3. **Poly-size representations**: there must exist short clause-sets in $UC_{k+1}$ equivalent to $f_h$ for all $h \in \mathbb{N}$.
introduced a special type of boolean functions, called \textit{Non-repeating Unate Decision trees} (NUD) there, by adding new variables to each clause of clause-sets in $SMU_{\delta=1}$, which is the class of unsatisfiable hitting clause-sets of deficiency $\delta = 1$. These boolean functions have a large number of prime implicates (the maximum regarding the original number of clauses), and thus are natural to consider as candidates to separate the levels of $UC_k$. In Section 4 we show that the underlying $SMU_{\delta=1}$ clause-sets determine the structure. The clause-sets in $SMU_{\delta=1}$ are exactly those with the maximum number of minimal premise sets, and then doping elements of $SMU_{\delta=1}$ yields clause-sets with the maximal number of prime implicates. We utilise the tree structure of $SMU_{\delta=1}$ to prove lower bounds on the size of equivalent representations in $UC_k$ of doped $SMU_{\delta=1}$ clause-sets.

In Section 6 we introduce the basic method (see Theorem 6.4) for lower bounding the size of equivalent clause-sets of a given hardness, via the transversal number of “trigger hypergraphs”. The basic idea is very simple, namely if we want $F$ to have hardness at most $k$, then for every prime implicate $C$ of $F$ the (unsatisfiable) clause-set $\varphi_C \ast F$ must contain a clause of length at most $k$, in order to “trigger” the derivation of the empty clause from $\varphi_C \ast F$. This applies to w-hardness as well, and thus we actually obtain a lower bound on the w-hardness.

Using this lower-bound method, in Theorem 6.13 we show a lower bound on the matching number (the maximal number of disjoint hyperedges) of the trigger hypergraph of doped “extremal” $SMU_{\delta=1}$-clause-sets. From this follows immediately Theorem 6.14 that for every $k \in \mathbb{N}_0$ there are polysize clause-sets in $UC_{k+1}$, where every equivalent clause-set in $WC_k$ is of exponential size. Thus the $UC_k$ as well as the $WC_k$ hierarchy is strict regarding equivalence of polysize clause-sets.

1.5 Relevance of these hierarchies for SAT solving

The poly-time methods used to detect unsatisfiability of instantiations of clause-sets in $UC_k$ resp. $WC_k$ have a running-time with an exponent depending on $k$, and in the latter case also space-complexity depends in the exponent on $k$.

1. This seems a necessary condition for showing a separation result as in this paper. It is needed that the different levels are qualitatively different. And this seems very unlikely to be achievable with a parameter which would allow fixed-parameter tractability, and which thus would only be a quantitative parameter (like the number of variables), only expressing a gradual increase in complexity.\footnote{See Lemma 1.5 for an example of a collapsing hierarchy.}

2. The class $UC_k$ uses generalised unit-clause propagation, namely the reduction $r_k$. Especially $r_2$, which is (complete) failed-literal elimination, is used in lookahead SAT solvers (see \cite{DBLP:conf/sat/KrohmerW14} for an overview) such as Oksolver (\cite{DBLP:conf/ coerceo/FreuderD93}), march (\cite{DBLP:conf/sat/MeiburgMS12}) and \texttt{satz} (\cite{DBLP:journals/ai/SattingerP98}). Also conflict-driven solvers such as CryptoMiniSat (\cite{DBLP:conf/aaai/DinicDPR08}) and \texttt{PicoSAT} (\cite{DBLP:journals/tpls/SchederK05}) integrate $r_2$ during search, and solvers such as Lingeling (\cite{DBLP:conf/aaai/GaspersH10}) use $r_2$ as a preprocessing technique. Furthermore, in general $r_k$ is used, in even stronger versions, in the Stålmarck-solver (see \cite{DBLP:journals/iandc/HuangD11}), and see Section 3.5 of \cite{DBLP:journals/ai/FreuderH11} for a discussion of the connections to $r_k$), and via breadth-first “branch/merge” rules in \texttt{HeerHugo} (see \cite{DBLP:journals/ipl/JansenMS13}).

1.6 Overview on results

The preliminaries (Section 2) define the basic notions. The classes $UC_k$, $PC_k$ and $WC_k$ are defined in Section 3. In Section 4 we investigate minimal premise sets and doping in general, while in Section 5 we apply these notions to our source of hard
examples. In Section 6 we are then able to show the separation $WC_k \rightarrow UC_{k+1}$. In Section 7 we discuss the KC-queries supported by our three hierarchies. In Section 8 we show $PC \rightarrow UC$. Finally, in Section 9 one finds many open problems. We now list our mean results (marked as “theorems”, in contrast to “lemmas”, which are “small results”). The main results on minimal premise sets and doping are:

1. Theorem 4.18 shows the correlation between prime implicates of doped clause-sets and minimal premise-sets of the original (undoped) clause-sets.
2. Theorem 5.12 characterises unsatisfiable clause-sets where every non-empty sub-clause-set is a minimal premise set.
3. Theorem 5.22 gives basic characteristics of doped $SMU_0=1$-clause-sets.

The main results related to the three hierarchies are:

1. Theorem 6.4 introduces the basic method for lower bounding the size of equivalent clause-sets of a given w-hardness, via the transversal number of “trigger hypergraphs”.
2. Theorem 6.13 shows a lower bound on the matching number of the trigger hypergraph of doped “extremal” $SMU_0=1$-clause-sets.
3. Theorem 6.14 shows that for every $k \in \mathbb{N}_0$ there are polysize clause-sets in $UC_{k+1}$, where every equivalent clause-set in $WC_k$ is of exponential size.
4. Theorem 7.1 states KC queries supported by the three hierarchies.
5. Theorem 8.1 shows that there are polysize clause-sets in $UC$, where every equivalent clause-set in $PC$ is of exponential size.

Remarks on the history of this report Many results of this report were originally contained in [31]. That report, conceived as a starting point for a theory of SAT representations, had three topics: The separation results as in this paper, representation of XOR constraints, and the relations to SAT solving. The fifth version would have had over 80 pages, and so we decided to split it into three reports (which each contain substantial additions):

1. The representation of XOR constraints is now in [30].
2. Results regarding the separation of the hierarchies in this report.
3. While the SAT-related theory and experimentation is in [29] (to appear).

2 Preliminaries

We follow the general notations and definitions as outlined in [39]. We use $\mathbb{N} = \{1,2,\ldots\}$, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, and $\mathcal{P}(M)$ for the set of subsets of set $M$.

2.1 Clause-sets

Let $\mathcal{V}A$ be the infinite set of variables, and let $\mathcal{LIT} = \mathcal{V}A \cup \{ \overline{v} : v \in \mathcal{V}A \}$ be the set of literals, the disjoint union of variables as positive literals and complemented variables as negative literals. We use $\overline{L} := \{ \overline{x} : x \in L \}$ to complement a set $L$ of literals. A clause is a finite subset $C \subset \mathcal{LIT}$ which is complement-free, i.e., $C \cap \overline{C} = \emptyset$; the set of all clauses is denoted by $\mathcal{CL}$. A clause-set is a finite set of clauses, the set of all clause-sets is $\mathcal{CLS}$. By $\text{var}(x) \in \mathcal{V}A$ we denote the underlying variable of
a literal $x \in \mathcal{LIT}$, and we extend this via $\text{var}(C) := \{\text{var}(x) : x \in C\} \subset \mathcal{VA}$ for clauses $C$, and via $\text{var}(F) := \bigcup_{C \in F} \text{var}(C)$ for clause-sets $F$. The possible literals in a clause-set $F$ are denoted by $\text{lit}(F) := \text{var}(F) \cup \text{var}(\overline{F})$. Measuring clause-sets happens by $n(F) := |\text{var}(F)|$ for the number of variables, $c(F) := |F|$ for the number of clauses, and $\ell(F) := \sum_{C \in F} |C|$ for the number of literal occurrences. A special clause-set is $\top := \emptyset \in \mathcal{CLS}$, the empty clause-set, and a special clause is $\bot := \emptyset \in \mathcal{C}$, the empty clause.

A partial assignment is a map $\varphi : V \to \{0, 1\}$ for some finite $V \subset \mathcal{VA}$, where we set $\text{var}(\varphi) := V$, and where the set of all partial assignments is $\mathcal{PASS}$. For $v \in \text{var}(\varphi)$ let $\varphi(\overline{v}) := \overline{\varphi(v)}$ (with $\overline{0} = 1$ and $\overline{1} = 0$). We construct partial assignments by terms $\langle x_1 \to \varepsilon_1, \ldots, x_n \to \varepsilon_n \rangle \in \mathcal{PASS}$ for literals $x_1, \ldots, x_n$ with different underlying variables and $\varepsilon_i \in \{0, 1\}$. We use $\varphi_C := \langle x \to 0 : x \in C\rangle$ for the partial assignment setting precisely the literals in clause $C \in \mathcal{C}$ to false.

For $\varphi \in \mathcal{PASS}$ and $F \in \mathcal{CLS}$ we denote the result of applying $\varphi$ to $F$ by $\varphi \circ F$, removing clauses $C \in F$ containing $x \in C$ with $\varphi(x) = 1$, and removing literals $x$ with $\varphi(x) = 0$ from the remaining clauses. By $\mathcal{SAT} := \{F \in \mathcal{CLS} \mid \exists \varphi \in \mathcal{PASS} : \varphi \circ F = \top\}$ the set of satisfiable clause-sets is denoted, and by $\mathcal{USAT} := \mathcal{CLS} \setminus \mathcal{SAT}$ the set of unsatisfiable clause-sets.

So clausal entailment, that is the relation $F \models C$ for $F \in \mathcal{CLS}$ and $C \in \mathcal{C}$, which by definition holds true iff for all $\varphi \in \mathcal{PASS}$ with $\varphi \circ F = \top$ we have $\varphi \circ \{C\} = \top$, is equivalent to $\varphi_C \circ F \in \mathcal{USAT}$.

Two clauses $C, D \in \mathcal{C}$ are resolvable iff they clash in exactly one literal $x$, that is, $C \cap \overline{D} = \{x\}$, in which case their resolvent is $C \circ D := (C \cup D) \setminus \{x, \overline{x}\}$ (with resolution literal $x$). A resolution tree is a full binary tree formed by the resolution operation. We write $T : F \vdash C$ if $T$ is a resolution tree with axioms (the clauses at the leaves) all in $F$ and with derived clause (at the root) $C$. A resolution tree $T : F \vdash C$ is regular iff along each path from the root of $T$ to a leaf no resolution-variable is used more than once. In this article we use only resolution trees, even when speaking of unrestricted resolution, that is, we always unfold dag-resolution proofs to (full) binary resolution trees. Completeness of resolution means that $F \models C$ (semantic implication) is equivalent to $F \vdash C$, i.e., there is some $C' \subseteq C$ and some $T$ with $T : F \vdash C'$.

A prime implicate of $F \in \mathcal{CLS}$ is a clause $C$ such that a resolution tree $T$ with $T : F \vdash C$ exists, but no $T'$ exists for some $C' \subset C$ with $T' : F \vdash C'$; the set of all prime implicants of $F$ is denoted by $\text{prc}_0(F) \in \mathcal{CLS}$. The term “implicate” refers to the implicit interpretation of $F$ as a conjunctive normal form (CNF). Considering clauses as combinatorial objects one can speak of “prime clauses”, and the “0” in our notation reminds of “unsatisfiability”, which is characteristic for CNF. Two clause-sets $F, F' \in \mathcal{CLS}$ are equivalent iff $\text{prc}_0(F) = \text{prc}_0(F')$. A clause-set $F$ is unsatisfiable iff $\text{prc}_0(F) = \{\bot\}$. The set of prime implicants of a clause-set $F \in \mathcal{CLS}$ is denoted by $\text{prc}_1(F) \in \mathcal{CLS}$, and is the set of all clauses $C \in \mathcal{C}$ such that for all $D \in F$ we have $C \cap D \neq \emptyset$, while this holds for no strict subset of $C$.

### 2.2 On “good” equivalent clause-sets

A basic problem considered in this article is for a given $F \in \mathcal{CLS}$ to find a “good” equivalent $F' \in \mathcal{CLS}$. How “good” $F'$ depends in our context on two factors, which have to be balanced against each other:

- the size of $F'$: we measure $c(F')$, and the smaller the better;
- the inference power of $F'$: inference from $F'$ should be “as easy as possible”, and we consider two measures in this article, (tree-)hardness in Subsection 3.1, and width-hardness in Subsection 3.3, the smaller these measures, the easier inference w.r.t. tree resolution resp. (generalised) width-bounded resolution.
The basic size-lower-bound for \( F' \) is given by the essential prime implicates, which are those \( C \in \text{prc}_0(F) \) such that \( \text{prc}_0(F) \setminus \{C\} \) is not equivalent to \( F \):

**Lemma 2.1** Consider \( F \in \mathcal{CLS} \), and let \( P \subseteq \text{prc}_0(F) \) be the set of essential prime implicates of \( F \). Now for every \( F' \in \mathcal{CLS} \) equivalent to \( F \) there exists an injection \( i : P \to F' \) such that for all \( C \in P \) holds \( C \subseteq i(C) \). Thus \( c(F') \geq c(P) \).

**Proof:** For every \( C' \in F' \) there exists a \( C \in \text{prc}_0(F) \) such that \( C \subseteq C' \); replacing every \( C' \in F' \) by such a chosen \( C \) we obtain \( F'' \subseteq \text{prc}_0(F) \) with \( P \subseteq F'' \). \( \Box \)

Note that Lemma 2.1 crucially depends on not allowing auxiliary variables — when allowing new variable, then we currently do not have any overview on the possibilities for “better” \( F' \). The most powerful representation regarding inference alone (with or without new variables) is given by the set \( \text{prc}_0(F) \) of all prime implicates of \( F \), and will have “hardness” 0, as defined in the following section. (The problem is of course that in most cases this representation is too large, and thus higher hardness must be allowed.)

### 3 Measuring “SAT representation complexity”

In this section we define and discuss the measures \( \text{hd}, \text{phd}, \text{whd} : \mathcal{CLS} \to \mathbb{N}_0 \) and the corresponding classes \( \mathcal{UC}_k, \mathcal{PC}_k, \mathcal{WC}_k \subseteq \mathcal{CLS} \). It is mostly of an expository nature, explaining what we need from \([40, 45, 26, 28, 27]\), with some additional remarks.

#### 3.1 Hardness and \( \mathcal{UC}_k \)

First we turn to the most basic hardness measurement. It can be based on resolution refutation trees, as we do here, but it can also be defined algorithmically, via generalised unit-clause propagation (see Lemma 3.4).

**Definition 3.1** For a full binary tree \( T \) the height \( \text{ht}(T) \in \mathbb{N}_0 \) and the Horton-Strahler number \( \text{hs}(T) \in \mathbb{N}_0 \) are defined as follows:

1. If \( T \) is trivial (i.e., \( \#\text{nds}(T) = 1 \)), then \( \text{ht}(T) := 0 \) and \( \text{hs}(T) := 0 \).
2. Otherwise let \( T_1, T_2 \) be the two subtrees of \( T \):
   
   \( a \) \( \text{ht}(T) := 1 + \max(\text{ht}(T_1), \text{ht}(T_2)) \)
   
   \( b \) If \( \text{hs}(T_1) = \text{hs}(T_2) \), then \( \text{hs}(T) := 1 + \max(\text{hs}(T_1), \text{hs}(T_2)) \), otherwise \( \text{hs}(T) := \max(\text{hs}(T_1), \text{hs}(T_2)) \).

Obviously we always have \( \text{hs}(T) \leq \text{ht}(T) \).

**Example 3.2** For the tree \( T \) from Example 5.3 we have \( \text{ht}(T) = 3, \text{hs}(T) = 2 \). The Horton-Strahler numbers of the subtrees are as follows:
Definition 3.3 The hardness \( \text{hd} : \text{CLS} \rightarrow \mathbb{N}_0 \) is defined for \( F \in \text{CLS} \) as follows:

1. If \( F \in \text{USAT} \), then \( \text{hd}(F) \) is the minimum \( \text{hs}(T) \) for \( T : F \vdash \bot \).
2. If \( F = \top \), then \( \text{hd}(F) := 0 \).
3. If \( F \in \text{SAT} \setminus \{ \top \} \), then \( \text{hd}(F) := \max_{\varphi \in \text{PASS}} \{ \text{hd}(\varphi \ast F) : \varphi \ast F \in \text{USAT} \} \).

Hardness for unsatisfiable clause-sets was introduced in [40, 45], while this generalisation to arbitrary clause-sets was first mentioned in [1], and systematically studied in [26, 28, 27]. It is easy to see that the hardness of \( F \in \text{CLS} \) is the minimal \( k \in \mathbb{N}_0 \) such that for all prime implicates \( C \) of \( F \) there exists \( T : F \vdash C \) with \( \text{hs}(T) \leq k \).

Definition 3.3 defines hardness proof-theoretically: importantly, it can also be characterised algorithmically via necessary levels of generalised unit-clause propagation (see [26, 28, 27] for the details):

Lemma 3.4 Consider the reductions \( r_k : \text{CLS} \rightarrow \text{CLS} \) for \( k \in \mathbb{N}_0 \) as introduced in [26]; it is \( r_1 \) unit-clause propagation, while \( r_2 \) is (full, iterated) failed-literal elimination. Then \( \text{hd}(F) \) for \( F \in \text{CLS} \) is the minimal \( k \in \mathbb{N}_0 \) such that for all \( \varphi \in \text{PASS} \) with \( \varphi \ast F \in \text{USAT} \) holds \( r_k(\varphi \ast F) = \{ \bot \} \), i.e., the minimal \( k \) such that \( r_k \) detects unsatisfiability of any instantiation.

For \( F \in \text{CLS} \) there is a partial assignment \( \varphi \) with \( \varphi \ast F = r_k(F) \), where \( \varphi \) consists of certain “forced assignments” \( \langle x \rightarrow 1 \rangle \subseteq \varphi \), i.e., \( \langle x \rightarrow 0 \rangle \ast F \in \text{USAT} \). Another “localisation” of forced assignments has been considered in [19], namely “\( k \)-backbones”, which is a forced assignment \( \langle x \rightarrow 1 \rangle \) for \( F \) such that there is \( F' \subseteq F \) with \( c(F') \leq k \) and such that \( \langle x \rightarrow 1 \rangle \) is forced also for \( F' \). It is not hard to see that \( r_k \) for \( k \in \mathbb{N}_0 \) will set all \( k \)-backbones of \( F \in \text{CLS} \) (using that for \( F \in \text{USAT} \) we have \( \text{hd}(F) < c(F) \) by Lemma 3.18 in [40]).

We can now define our main hierarchy, the \( \text{UC}_k \)-hierarchy (with “UC” for “unit-refutation complete”) via (tree-)hardness:

Definition 3.5 For \( k \in \mathbb{N}_0 \) let \( \text{UC}_k := \{ F \in \text{CLS} : \text{hd}(F) \leq k \} \).

\( \text{UC}_1 = \text{UC} \) is the class of unit-refutation complete clause-sets, as introduced in [20]. In [24, 28, 27] we show that \( \text{UC} = \text{SLUR} \), where \( \text{SLUR} \) is the class of clause-sets solvable via Single Lookahead Unit Resolution (see [22]). Using [14] we then obtain [24, 28, 27] that membership decision for \( \text{UC}_k = \text{SLUR}_k \) is coNP-complete for \( k \geq 1 \). The class \( \text{UC}_2 \) is the class of all clause-sets where unsatisfiability for any partial assignment is detected by failed-literal reduction (see Section 5.2.1 in [34] for the usage of failed literals in SAT solvers).

A basic fact is that the classes \( \text{UC}_k \) are stable under application of partial assignments, in other words, for \( F \in \text{CLS} \) and \( \varphi \in \text{PASS} \) we have \( \text{hd}(\varphi \ast F) \leq \text{hd}(F) \). For showing lower bounds on the hardness for unsatisfiable clause-sets, we can use the methodology developed in Subsection 3.4.2 of [10]. A simplified version of Lemma 3.17 from [10] is sufficient for our purposes, is as follows (with a technical correction, as explained in Example 3.7):

Lemma 3.6 Consider \( C \subseteq \text{USAT} \) and a function \( h : C \rightarrow \mathbb{N}_0 \). For \( k \in \mathbb{N}_0 \) let \( C_k := \{ F \in C : h(F) \geq k \} \). Then \( \forall F \in C : \text{hd}(F) \geq h(F) \) holds if and only if \( \text{UC}_0 \cap C_1 = \emptyset \), and for all \( k \in \mathbb{N}_0 \), \( F \in C_k \) and \( x \in \text{lit}(F) \) there exist clause-sets \( F_0, F_1 \in \text{CLS} \) fulfilling the following three conditions:

(i) \( n(F_\varepsilon) < n(F) \) for both \( \varepsilon \in \{0, 1\} \);

(ii) \( \text{hd}(F_\varepsilon) \leq \text{hd}(\langle x \rightarrow \varepsilon \rangle \ast F) \) for both \( \varepsilon \in \{0, 1\} \);
(iii) $F_{0} \in C_{k}$ or $F_{1} \in C_{k-1}$.

**Proof:** The given conditions are necessary for $\forall F \in C : \text{hd}(F) \geq h(F)$, since we can choose $F_{\varepsilon} := (v \rightarrow \varepsilon) \ast F$ for $\varepsilon \in \{0, 1\}$. To see sufficiency, assume for the sake of contradiction that there is $F \in C$ with $\text{hd}(F) < h(F)$, and consider such an $F$ with minimal $n(F)$. If $\text{hd}(F) = 0$, so $h(F) = 0$ by assumption, and thus $\text{hd}(F) \geq 1$ would hold. So assume $\text{hd}(F) \geq 1$. It follows that there is a literal $x \in \text{lit}(F)$ with $\text{hd}(\langle x \rightarrow 1 \rangle \ast F) < \text{hd}(F)$. Let $k := h(F)$; so $F \in C_{k}$. By assumption there are $F_{0}, F_{1} \in \text{CLS}$ with $\text{hd}(F_{\varepsilon}) \leq \text{hd}(\langle x \rightarrow \varepsilon \rangle \ast F)$ for both $\varepsilon \in \{0, 1\}$, and $F_{0} \in C_{k}$ or $F_{1} \in C_{k-1}$. If $F_{0} \in C_{k}$, then $\text{hd}(F_{0}) \leq \text{hd}(F) < k \leq h(F_{0})$, while $n(F_{0}) < n(F)$, contradicting minimality of $F$. And if $F_{1} \in C_{k-1}$, then $\text{hd}(F_{1}) \leq \text{hd}(F) - 1 < k - 1 \leq h(F_{1})$, while $n(F_{1}) < n(F)$, contradicting again minimality of $F$. \hfill \Box

Lemma 3.17 in [10] doesn’t state the condition (i) from Lemma 3.4. The following example shows that this condition actually needs to be stated (that is, if we just have (ii) and (iii), then $h$ doesn’t need to be a lower bound for $\text{hd}$); fortunately in all applications in [10] this (natural) condition is fulfilled.

**Example 3.7** Consider $C := UC_{1} \cap USAT$. Define $h : C \rightarrow \{0, 1, 2\}$ as $h(F) = 0$ iff $\bot \in F$, and $h(F) = 1$ iff $\bot \notin F$ and there is $v \in \text{var}(F)$ with $\{v, \overline{v}\} \in F$. So we have $h(F) = 2$ if and only if for all literals $x \in \text{lit}(F)$ holds $\text{hd}(\langle x \rightarrow 0 \rangle \ast F) = \text{hd}(\langle x \rightarrow 0 \rangle \ast F) = 1$. By definition we have $UC_{0} \cap C_{1} = \emptyset$. Now consider $k \in \{1, 2\}$, $F \in C_{k}$ and $x \in \text{lit}(F)$. If $h(F) = 1$, then let $F_{\varepsilon} := (x \rightarrow \varepsilon) \ast F$, while otherwise $F_{\varepsilon} := F$ for $\varepsilon \in \{0, 1\}$. Now Conditions (ii), (iii) of Lemma 3.4 are fulfilled (if $h(F) = 1$, then for Condition (iii) always $F_{1} \in C_{k-1}$ holds, while in case of $h(F) = 2$ we always have $F_{0} \in C_{k}$). But by definition $h$ is not a lower bound on $\text{hd}$.

### 3.2 P-Hardness and $\mathcal{PC}_{k}$

Complementary to “unit-refutation completeness”, there is the notion of “propagation-completeness” as investigated in [28, 27], yielding the class $\mathcal{PC} \subseteq \mathcal{UC}$. This was captured and generalised by a measure $\text{phd} : \text{CLS} \rightarrow \mathbb{N}_{0}$ of “propagation-hardness” along with the associated hierarchy, defined in [28, 27] as follows:

**Definition 3.8** For $F \in \text{CLS}$ 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 $\varphi \in \text{PASS}$ we have $r_{k}(\varphi \ast F) = r_{\infty}(\varphi \ast F)$, where $r_{k} : \text{CLS} \rightarrow \text{CLS}$ is generalised unit-clause propagation $[14, 14]$, and $r_{\infty} : \text{CLS} \rightarrow \text{CLS}$ applies all forced assignments, and can be defined by $r_{\infty}(F) := r_{n(F)}(F)$. For $k \in \mathbb{N}_{0}$ let $\mathcal{PC}_{k} := \{F \in \text{CLS} : \text{phd}(F) \leq k\}$ (the **class of propagation-complete clause-sets of level** $k$).

Remarks:

1. We have $\mathcal{PC} = \mathcal{PC}_{1}$.
2. For $k \in \mathbb{N}_{0}$ we have $\mathcal{PC}_{k} \subseteq \mathcal{UC}_{k} \subseteq \mathcal{PC}_{k+1}$.
3. By definition (and composition of partial assignments) we have that all classes $\mathcal{PC}_{k}$ are stable under application of partial assignments.
4. For $F \in \text{CLS}$ a literal $x \in \text{LIT}$ is **forced for** $F$ (more precisely, the assignment $\langle x \rightarrow 1 \rangle$ is forced for $F$), iff $\langle x \rightarrow 0 \rangle \ast F \in \text{USAT}$. Note that for $F \in \text{USAT}$ all $x \in \text{LIT}$ are forced, while for $F \in \text{SAT}$ and a forced literal $x$ we have $x \in \text{lit}(F)$. Now for $k \in \mathbb{N}_{0}$ and $F \in \text{CLS}$ we have $F \in \mathcal{PC}_{k}$ iff for all $\varphi \in \text{PASS}$ the clause-set $F' := r_{k}(\varphi \ast F)$ has no forced literals $x$ with $x \in \text{lit}(F')$. 

11
3.3 W-Hardness and $WC_k$

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 $k \in \mathbb{N}_0$ ("width", denoted by $\text{wid}(F)$ below; see [14]), is that even Horn clause-sets require unbounded width in this sense. The correct solution, as investigated and discussed in [12, 15], is to use the notion of "$k$-resolution" as introduced in [28], where only one parent clause needs to have length at most $k$ (thus properly generalising unit-resolution). Nested input-resolution ([12, 15]) is the proof-theoretic basis of hardness, and approximates tree-resolution. In the same vein, $k$-resolution is the proof-theoretic basis of "w-hardness", and approximates dag-resolution (see Theorem 6.12 in [14]):

**Definition 3.9** The **$w$-hardness** $\text{whd} : \text{CLS} \rightarrow \mathbb{N}_0$ ("width-hardness", or "asymmetric width") is defined for $F \in \text{CLS}$ as follows:

1. If $F \in \text{USAT}$, then $\text{whd}(F)$ is the minimum $k \in \mathbb{N}_0$ such that $k$-resolution refutes $F$, that is, such that $T : F \vdash \bot$ exists where for each resolution step $R = C \lor D$ in $T$ we have $|C| \leq k$ or $|D| \leq k$ (this corresponds to Definition 8.2 in [14], and is a special case of $\text{wid}_0$ introduced in Subsection 6.1 of [14]).

2. If $F = \top$, then $\text{whd}(F) := 0$.

3. If $F \in \text{SAT} \setminus \{\top\}$, then $\text{whd}(F) := \max \{\text{whd}(\varphi \ast F) : \varphi \ast F \in \text{USAT}\}$.

For $k \in \mathbb{N}_0$ let $\text{WC}_k := \{F \in \text{CLS} : \text{whd}(F) \leq k\}$.

The **symmetric width** $\text{wid} : \text{CLS} \rightarrow \mathbb{N}_0$ is defined in the same way, only that for $F \in \text{USAT}$ we define $\text{wid}(F)$ as the minimal $k \in \mathbb{N}_0$ such that there is $T : F \vdash \bot$, where all clauses of $T$ (axioms and resolvents) have length at most $k$.

Remarks:

1. We have $\text{WC}_0 = \text{UC}_0$, $\text{WC}_1 = \text{UC}_1$, and for all $k \in \mathbb{N}_0$ holds $\text{UC}_k \subseteq \text{WC}_k$ (this follows by Lemma 6.8 in [14] for unsatisfiable clause-sets, which extends to satisfiable clause-sets by definition).

2. For $k \geq 3$ and $k' \geq 0$ we have $\text{WC}_k \cap \text{USAT} \not\subseteq \text{WC}_{k'}$; this follows from known resolution lower bounds for the symmetric width, for example in Subsection 10.2 of [20] a sequence $T_n$ of (short) unsatisfiable clause-sets with $\text{wid}(T_n) = 3$ and $\text{hd}(T_n) = n$ is given.

3. Thus for $k \geq 3$ we have $\text{UC}_k \subset \text{WC}_k$; Example 3.10 extends this to $k \geq 2$.

4. Obviously we have $\text{whd}(F) \leq \text{wid}(F)$ for all $F \in \text{CLS}$, where for $F \in \text{HO} \cap \text{USAT}$ the symmetric width $\text{wid}(F)$ is unbounded (actually it is precisely equal to the maximal clause-length of $F$), in contrast to $\text{whd}(F) \leq 1$.

**Example 3.10** An example for $F \in \text{USAT}$ with $\text{whd}(F) = 2$ and $\text{hd}(F) = 3$ is

$$F := \{(2, 3, 4), \{-4, 2\}, \{-2, 1, 5\}, \{-5, -2\}, \{-3, 1, 6\}, \{-6, -3\},$$
$$\{7, 8, 9\}, \{-9, 7\}, \{-7, -1, 10\}, \{-10, -7\}, \{-8, -1, 11\}, \{-11, -8\}\}.$$ 

We believe that this example can be extended:

**Conjecture 3.11** For $k \in \mathbb{N}_0$ holds $\text{WC}_2 \not\subseteq \text{UC}_k$. 

12
For unsatisfiable $F$, whether $\mathrm{whd}(F) = k$ holds for $k \in \{0, 1, 2\}$ can be decided in polynomial time; this is non-trivial for $k = 2$ ([13]) and unknown for $k > 2$. Nevertheless, the clausal entailment problem $F \models C$ for $F \in \mathcal{WC}_k$ and fixed $k \in \mathbb{N}_0$ is decidable in polynomial time, as shown in Subsection 6.5 of [15], by actually using a slight strengthening of $k$-resolution, which combines width-bounded resolution and input resolution. While space-complexity of the decision $F \models C$ for $F \in \mathcal{UC}_k$ space-complexity is $O((\ell(F) \cdot n(F))^{O(k)})$.

As a special case of Theorem 6.12 in [45] we obtain for $F \in \mathcal{USAT}$, $n(F) \neq 0$, the following general lower bound on resolution complexity:

$$\text{Comp}_R(F) > b^{\mathrm{whd}(F)^2},$$

where $b := e^{1.1331484\ldots}$, while $\text{Comp}_R(F) \in \mathbb{N}$ is the minimal number of different clauses in a (tree-)resolution refutation of $F$. Similar to Theorem 14 in [26] resp. Theorem 5.7 in [28, 27] we thus obtain:

**Lemma 3.12** For $F \in \mathcal{CLS}$ and $k \in \mathbb{N}_0$, such that for every $C \in \text{prc}_0(F)$ with $|C| < n(F)$ there exists a resolution proof of $C$ from $F$ using at most $b^{(k+1)\frac{\ell(F)^2}{|C|}}$ different clauses, we have $\mathrm{whd}(F) \leq k$.

## 4 Minimal premise sets and doped clause-sets

In this section we study “minimal premise sets”, “mps’s” for short, introduced in [46], together with the properties of “doped” clause-sets, generalising a construction used in [38]. Mps’s are generalisations of minimally unsatisfiable clause-sets stronger than irredundant clause-sets, while doping relates prime implicates and sub-mps’s.

Recall that a clause-set $F$ is minimally unsatisfiable if $F \in \mathcal{USAT}$, while for all $C \in F$ holds $F \setminus \{C\} \in \mathcal{SAT}$. The set of all minimally unsatisfiable clause-sets is $\mathcal{MU} \subset \mathcal{CLS}$; see [13] for more information. In other words, for $F \in \mathcal{CLS}$ we have $F \in \mathcal{MU}$ if and only if $F \models \bot$ and $F$ is minimal regarding this entailment relation. Now an mps is a clause-set $F$ which minimally implies some clause $C$, i.e., $F \models C$, while $F' \nvdash C$ for all $F' \subset F$. In Subsection 4.1 we study the basic properties of mps’s $F$, and determine the unique minimal clause implied by $F$ as $\text{puc}(F)$, the set of pure literals of $F$.

For a clause-set $F$ its doped version $D(F) \in \mathcal{CLS}$ receives an additional new (“doping”) variable for each clause. The basic properties are studied in Subsection 4.1, and in Theorem 4.18 we show that the prime implicates of $D(F)$ correspond 1-1 to the mps’s contained in $F$. In Subsection 4.3 we determine the hardness of doped clause-sets.

### 4.1 Minimal premise sets

In Section 4.1 in [46] basic properties of minimal premise sets are considered:

**Definition 4.1** A clause-set $F \in \mathcal{CLS}$ is a minimal premise set (“mps”) for a clause $C \in \mathcal{CL}$ if $F \models C$ and $\forall F' \subset F : F' \nvdash C$, while $F$ is a minimal premise set if there exists a clause $C$ such that $F$ is a minimal premise set for $C$. The set of all minimal premise (clause-)sets is denoted by $\mathcal{MPS}$.

Remarks:

1. $\top$ is not an mps (since no clause follows from $\top$).
2. An unsatisfiable clause-set is an mps iff it is minimally unsatisfiable, i.e., 
\[ \mathcal{MP} \cap \text{USAT} = \text{MU} \]. In Corollary 4.8 we will see that the minimally
unsatisfiable clause-sets are precisely the mps’s without pure literals.

3. Every minimal premise clause-set is irredundant (no clause follows from the
other clauses).

4. For a clause-set \( F \) and any implicate \( F \models C \) there exists a minimal premise
sub-clause-set \( F' \subseteq F \) for \( C \).

5. A single clause \( C \) yields an mps \( \{ C \} \).

6. Two clauses \( C \neq D \) yield an mps \( \{ C, D \} \) iff \( C, D \) are resolvable.

7. If \( F_1, F_2 \in \mathcal{MP} \) with \( \text{var}(F_1) \cap \text{var}(F_2) = \emptyset \), then \( F_1 \cup F_2 \notin \mathcal{MP} \) except in
case of \( F_1 = F_2 = \{ \bot \} \).

Example 4.2 \( \{\{a\}, \{b\}\} \) for variables \( a \neq b \) is irredundant but not an mps.

With Corollary 4.5 in [46] we see that no clause-set can minimally entail more
than one clause:

**Lemma 4.3** For \( F \in \mathcal{MP} \) there exists exactly one \( C \in \text{prc}_0(F) \) such that \( C \)
is a minimal premise set for \( C \), and \( C \) is the smallest element of the set of clauses for
which \( F \) is a minimal premise set.

We remark that Lemma 4.3 does not mean that \( |\text{prc}_0(F)| = 1 \) for \( F \in \mathcal{MP} \); indeed, \( F \)
can have many \( F' \subset F \) with \( F' \in \mathcal{MP} \), and each such \( F' \) might contribute a prime
implicate, as we will see later. We wish now to determine that unique prime
implicate \( C \) which follows minimally from an mps \( F \). It is clear that
\( C \) must contain all pure literals from \( F \), since all clauses of \( F \) must be used, and
we can not get rid off pure literals.

**Definition 4.4** For \( F \in \mathcal{CLS} \) the pure clause of \( F \), denoted by \( \text{puc}(F) \in \mathcal{CL} \),
is the set of pure literals of \( F \), that is, \( \text{puc}(F) := L \setminus (L \cap \overline{L}) \), where \( L := \bigcup F \) is the set of literals occurring in \( F \).

**Example 4.5** For \( F = \{\{a, b\}, \{\overline{a}, \overline{c}\}\} \) we have \( \text{puc}(F) = \{b, \overline{c}\} \).

The main observation for determining \( C \) is that the conclusion of a regular
resolution proof consists precisely of the pure literals of the axioms (this follows by
definition):

**Lemma 4.6** For a regular resolution proof \( T : F \vdash C \), where every clause of \( F \) is
used as an axiom in \( T \), we have \( C = \text{puc}(F) \).

Due to the completeness of regular resolution we thus see, that \( \text{puc}(F) \) is the
desired unique prime implicate:

**Lemma 4.7** For \( F \in \mathcal{MP} \) the unique prime implicate \( C \), for which \( F \) is a mini-
mal premise set (see Lemma 4.3), is \( C = \text{puc}(F) \).

**Proof:** Consider a regular resolution proof \( T : F \vdash C \) (recall that regular resolution
is complete); due to \( F \in \mathcal{MP} \) every clause of \( F \) must be used in \( T \), and thus the
assertion follows by Lemma 4.6.

**Corollary 4.8** If we have \( F \in \mathcal{MP} \) with \( \text{puc}(F) = \bot \), then \( F \in \text{MU} \).
By Lemma 4.4 in [46] we get the main characterisation of mps’s, namely that after elimination of pure literals they must be minimally unsatisfiable:

**Lemma 4.9** Consider a clause-set \( F \in \mathcal{CLS} \). Then \( F \in \mathcal{MPS} \) if and only if the following two conditions hold for \( \varphi := \varphi_{\text{puc}(F)} \) (setting precisely the pure literals of \( F \) to false):

1. \( \varphi \ast F \in \mathcal{MU} \) (after removing the pure literals we obtain a minimal unsatisfiable clause-sets).
2. \( \varphi \) is contraction-free for \( F \), that is, for clauses \( C, D \in F \) with \( C \neq D \) we have \( \varphi \ast \{C\} \neq \varphi \ast \{D\} \).

These two conditions are equivalent to stating that \( \varphi \ast F \) as a multi-clause-set (not contracting equal clauses) is minimally unsatisfiable.

Thus we obtain all mps’s by considering some minimally unsatisfiable clause-sets and adding new variables in the form of pure literals:

**Corollary 4.10** The following process generates precisely the \( F' \in \mathcal{MPS} \):

1. Choose \( F \in \mathcal{MU} \).
2. Choose a clause \( P \) with \( \text{var}(P) \cap \text{var}(F) = \emptyset \) ("P" like "pure").
3. Choose a map \( e : F \to \mathcal{P}(P) \) ("e" like "extension").
4. Let \( F' := \{C \cup e(C) : C \in F\} \).

For unsatisfiable clause-sets the set of minimally unsatisfiable sub-clause-sets has been studied extensively in the literature; see [50] for a recent overview. The set of subsets which are mps’s strengthen this notion (now for all clause-sets):

**Definition 4.11** For a clause-set \( F \in \mathcal{CLS} \) by \( \text{mps}(F) \subset \mathcal{CLS} \) the set of all minimal premise sub-clause-sets is denoted: \( \text{mps}(F) := \mathcal{P}(F) \cap \mathcal{MPS} \).

We have \(|\text{mps}(F)| \leq 2^{c(F)} - 1\) \( ^7 \) The minimal elements of \( \text{mps}(F) \) are \( \{C\} \in \text{mps}(F) \) for \( C \in F \). Since every prime implicate of a clause-set has some minimal premise sub-clause-set, we get that running through all sub-mps’s in a clause-set \( F \) and extracting the clauses with the pure literals we obtain at least all prime implicates:

**Lemma 4.12** For \( F \in \mathcal{CLS} \) the map \( F' \in \text{mps}(F) \mapsto \text{puc}(F') \subseteq \{C \in \mathcal{CL} : F \models C\} \) covers \( \text{prc}(F) \) (i.e., its range contains the prime implicates of \( F \)).

**Example 4.13** Examples where we have more minimal premise sub-clause-sets than prime implicates are given by \( F \in \mathcal{MU} \), where \( \text{prc}(F) = \{\bot\} \), while in the most extreme case every non-empty subset of \( F \) can be a minimal premise sub-clause-set (see Theorem 5.13).

\( ^7 \) There is a typo in Corollary 4.6 of [46], misplacing the "−1" into the exponent.
4.2 Doping clause-sets

“Doping” is the process of adding a unique new variable to every clause of a clause-set. It enables us to follow the usage of this clause in derivations:

Definition 4.14 For every clause-set $F \in \mathcal{CLS}$ we assume an injection $u^F : F \to \forall A \setminus \text{var}(F)$ in the following, assigning to every clause $C$ a different variable $u^F_C$. For a clause $C \in \mathcal{CL}$ and a clause-set $F \in \mathcal{CLS}$ we then define the doping $D_F(C) := C \cup \{u^F_C\} \in \mathcal{CL}$, while $D(F) := \{D_F(C) : C \in F\} \in \mathcal{CLS}$.

Remarks:
1. In the following we drop the upper index in “$u^F_C$”, i.e., we just use “$u_C$”.
2. We have $D : \mathcal{CLS} \to \mathcal{SAT}$.
3. For $F \in \mathcal{CLS}$ we have $n(D(F)) = n(F) + c(F)$ and $c(D(F)) = c(F)$.
4. For $F \in \mathcal{CLS}$ we have $\text{puc}(D(F)) = \text{puc}(F) \cup \{u_C : C \in F\}$.

We are interested in the prime implicants of doped clause-sets. It is easy to see that all doped clauses are themselves essential prime implicants:

Lemma 4.15 For $F \in \mathcal{CLS}$ we have $D(F) \subseteq \text{prc}_0(D(F))$, and furthermore all elements of $D(F)$ are essential prime implicants.

Proof: Every resolvent of clauses from $D(F)$ contains at least two doping variables, and thus the clauses of $D(F)$ themselves (which contain only one doping variable) are prime and necessary. □

Thus by Lemma 4.14 among all the clause-sets equivalent to $D(F)$ this clause-set itself is the smallest. Directly by Lemma 4.9 we get that a clause-set is a mps iff its doped form is an mps:

Lemma 4.16 For $F \in \mathcal{CLS}$ holds $F \in \mathcal{MPS} \iff D(F) \in \mathcal{MPS}$. Thus the map $F' \in \text{mps}(F) \mapsto D(F')$ is a bijection from $\text{mps}(F)$ to $\text{mps}(D(F))$.

For doped clause-sets the surjection of Lemma 4.12 is bijective:

Lemma 4.17 Consider a clause-set $F \in \mathcal{CLS}$, and let $G := D(F)$.

1. The map $F' \in \text{mps}(G) \mapsto \text{puc}(F') \in \mathcal{CL}$ is a bijection from $\text{mps}(G)$ to $\text{prc}_0(G)$.

2. The inverse map from $\text{prc}_0(G)$ to $\text{mps}(G)$ obtains from $C \in \text{prc}_0(G)$ the clause-set $F' \in \text{mps}(G)$ with $\text{puc}(F') = C$ as $F' = \{D(D) : D \in F \land u_D \in \text{var}(C)\}$.

Proof: By Lemma 4.13 it remains to show that the map of Part 1 is injective and does not have subsumptions in the image. Assume for the sake of contradiction there are $G', G'' \in \text{mps}(G)$, $G' \neq G''$, with $\text{puc}(G') \subseteq \text{puc}(G'')$. Since every clause of $F$ has a different doping-variable, $G' \subseteq G''$ must hold. Consider the $F', F'' \in \text{mps}(F)$ with $D(F') = G'$ and $D(F'') = G''$. We have $F' \subseteq F''$, and thus $\text{puc}(F') \subseteq \text{puc}(F'')$, since for every $F \in \mathcal{MPS}$ the clause $\text{puc}(F)$ is a prime implicate of $F$. It follows that $\text{puc}(G') \subseteq \text{puc}(G'')$, contradicting the assumption. □

By Lemma 4.16 and Lemma 4.17 we obtain:

Theorem 4.18 Consider $F \in \mathcal{CLS}$. Then the map $F' \in \text{mps}(F) \mapsto \text{puc}(D(F')) \in \mathcal{CL}$ is a bijection from $\text{mps}(F)$ to $\text{prc}_0(D(F))$. 
Theorem 4.18 together with the description of the inversion map in Lemma 4.17 yields computation of the set $\text{mps}(F)$ for $F \in \mathcal{CLS}$ via computation of $\text{prc}_0(D(F))$.

**Corollary 4.19** For $F \in \mathcal{CLS}$ we obtain a map from $\text{prc}_0(D(F))$ to the set of implicates of $F$ covering $\text{prc}_0(F)$ by the mapping $C \in \text{prc}_0(D(F)) \rightarrow C \setminus V$ for $V := \{u_C : C \in F\}$.

**Proof:** The given map can be obtained as a composition as follows: For $C \in \text{prc}_0(D(F))$ take (the unique) $F' \in \text{mps}(F)$ with $\text{puc}(D(F')) = C$, and we have $C \setminus V = \text{puc}(F')$. □

### 4.3 Hardness of doped clause-sets

The hardness of a doped clause-set is the maximal hardness of sub-clause-sets of the original clause-set:

**Lemma 4.20** For $F \in \mathcal{CLS}$ we have $\text{hd}(D(F)) = \max_{F' \subseteq F} \text{hd}(F')$.

**Proof:** We have $\text{hd}(F') \leq \text{hd}(D(F))$ for all $F' \subseteq F$, since via applying a suitable partial assignment we obtain $F'$ from $F$, setting the doping-variables in $F'$ to false, and the rest to true. And if we consider an arbitrary partial assignment $\varphi$ with $\varphi \ast D(F) \in \mathcal{USAT}$, then w.l.o.g. all doping variables are set (we can set the doping-variables not used by $\varphi$ to true, since these variables are all pure), and then we have a partial assignment making $F'$ unsatisfiable for that $F' \in \mathcal{USAT}$ given by all the doping variables set by $\varphi$ to false. □

**Example 4.21** For an example of a clause-set $F \in \mathcal{USAT}$ with $\text{hd}(D(F)) > \text{hd}(F)$ consider any clause-set $F' \in \mathcal{CLS}$ with $\text{hd}(F') > 0$, and then take $F := F' \cup \{\bot\}$ (note that $\bot \notin F'$). Thus $\text{hd}(F) = 0$. And by Part 1 of Lemma 6.5 in [28, 27], all $\mathcal{UC}_k$ are closed under partial assignments, so for $\varphi := \langle u_{\bot} \rightarrow 1 \rangle \cup \langle u_C \rightarrow 0 \mid C \in F' \rangle$ we have $\text{hd}(D(F)) \geq \text{hd}(\varphi \ast D(F)) = \text{hd}(F') > \text{hd}(F) = 0$.

### 5 Doping tree clause-sets

As explained in Subsection 7.4, we want to construct boolean functions (given by clause-sets) with a large number of prime implicates, where we have strong control over these prime implicates. For this purpose we dope “minimally unsatisfiable clause-sets of deficiency 1”, that is the elements of $\mathcal{SMU}_{k=1}$. First we review in Subsection 5.1 the background (for more information see [39]). In Subsection 5.2 we show that these clause-sets are the core of “total minimal premise sets”, which have as many minimal-premise sub-clause-sets as possible. In Theorem 5.12 we show that $F \in \mathcal{SMU}_{k=1}$ are precisely the unsatisfiable clause-sets such that every non-empty subset is an mps. Then in Subsection 5.3 we consider doping of these special clause-sets, and in Theorem 5.22 we determine basic properties of $D(F)$.

#### 5.1 Preliminaries on minimal unsatisfiability

A minimally unsatisfiable $F \in \mathcal{MU}$ is *saturated minimally unsatisfiable* iff for all clauses $C \in F$ and for every literal $x$ with $\text{var}(x) \notin \text{var}(C)$ the clause-set $(F \setminus C) \cup \{C \cup \{x\}\}$ is satisfiable. The set of all saturated minimally unsatisfiable clause-sets is denoted by $\mathcal{SMU} \subset \mathcal{MU}$. By $\mathcal{SMU}_{k=1}$ we denote the set of $F \in \mathcal{SMU}$ with $\delta(F) = k$, where the *deficiency* of a clause-set $F$ is given by $\delta(F) := c(F) - n(F)$. 

17
In [11] (generalised in [10]) it is shown that the elements of $SMU_{6=1}$ are exactly the clause-sets introduced in [13]. The details are as follows. For rooted trees $T$ we use $nds(T)$ for the set of nodes and $lvs(T) \subseteq lvs(T)$ for the set of leaves, and we set $\#nds(T) := |nds(T)|$ and $\#lvs(T) := |lvs(T)|$. In our context, the nodes of rooted trees are just determined by their positions, and do not have names themselves. Another useful notation for a tree $T$ and a node $w$ is $T_w$, which is the sub-tree of $T$ with root $w$; so $lvs(T) = \{ w \in nds(T) : \#nds(T_w) = 1 \}$. Recall that for a full binary tree $T$ (every non-leaf node has two children) we have $\#nds(T) = 2 \#lvs(T) - 1$.

**Definition 5.1** Consider a full binary tree $T$ and an injective vertex labelling $u : (nds(T) \setminus lvs(T)) \to VA$ for the inner nodes; the set of all such pairs is denoted by $T_1$. The induced edge-labelling assigns to every edge from an inner node $w$ to a child $w'$ the literal $u(w)$ resp. $u(w)$ for a left resp. right child. We define the **clause-set representation** $F^1(T, u)$ (where “1” reminds of deficiency 1 here; see Lemma 5.3) to be $F^1(T, u) := \{ C_w : w \in lvs(T) \}$, where clause $C_w$ consists of all the literals (i.e., edge-labels) on the path from the root of $T$ to $w$.

By Lemma C.5 in [11] we know that via this tree-construction we obtain exactly the clause-sets in $SMU_{6=1}$:

**Lemma 5.2** $F^1 : T_1 \to SMU_{6=1}$ is a bijection.

By $T^1 : SMU_{6=1} \to T_1$ we denote the inversion of $F^1$. Typically we identify $(T, u) \in T_1$ with $T$, and let the context determine $u$. So $T^1(F)$ is the full binary tree, where the variable $v$ labelling the root (for $F \neq \{ \bot \}$) is the unique variable occurring in every clause of $F$, and the clause-sets determining the left resp. right subtree are $\langle v \to 0 \rangle \ast F$ resp. $\langle v \to 1 \rangle \ast F$. By $w_C$ for $C \in F$ we denote the leaf $w$ of $T^1(F)$ such that $C_w = C$. Furthermore we identify the literals of $F$ with the edges of $T^1(F)$. Note that $c(F) = \#lvs(T^1(F))$ and $n(F) = \#nds(T^1(F)) - \#lvs(T^1(F))$.

**Example 5.3** Consider the following labelled binary tree $T$ (using additionally labels 1, . . . , 6 for the leaves):

```
                      v1
                     /\   \
                    v2   v3
                   /   \  \    \
                  v4   v5  v6
                 /     \  \
                v7     v8
```

Then $F^1(T) = \{ \{ v_1, v_2, v_3 \}, \{ v_1, v_2, v_6 \}, \{ v_1, v_2, v_4 \}, \{ v_1, v_5, v_6 \}, \{ v_5, v_6 \}, \{ v_1, v_5 \} \}$, where for example $C_3 = \{ v_1, v_2, v_4 \}$ and $w_{v_1, v_5} = 6$.

We note in passing, that those $F^1(T)$ with $hs(T) \leq 1$ can be easily characterised as follows. A clause $C \in F$ for $F \in CLS$ is called **full** if $\text{var}(C) = \text{var}(F)$, that is, $C$ contains all variables of $F$.

**Lemma 5.4** $F \in SMU_{6=1}$ contains a full clause if and only if $hs(T^1(F)) \leq 1$.

See Example 6.10 for more on these special clause-sets. The effect of applying a partial assignment to some element of $SMU_{6=1}$ is easily described as follows:
Lemma 5.5 Consider $F \in SMU_{6=1}$ and $x \in \text{lit}(F)$, and let $F' := (x \rightarrow 1) \ast F$. We have:

1. $F' \in SMU_{6=1}$.

2. Let $T := T^1(F)$ and $T' := T^1(F')$. The tree $T'$ is obtained from $T$ as follows:
   (a) Consider the node $w \in T$ labelled with $\var(x)$. Let $T_x, T_{\overline{x}}$ be the two subtrees hanging at $w$, following the edge labelled with $x$ resp. $\overline{x}$.
   (b) Now $T'$ is obtained from $T'$ by removing subtree $T_x$, and attaching $T_{\overline{x}}$ directly at position $w$.

Example 5.6 Consider the labelled binary tree $T$ from Example 5.3 where

$$F^1(T) = \{ \{v_1, v_2, v_3\}, \{v_1, v_2, v_3\}, \{v_1, \overline{v}_2, v_4\}, \{v_1, \overline{v}_2, \overline{v}_4\}, \{v_1, v_5\}, \{v_1, \overline{v}_5\} \}$$

Now consider the application of the partial assignment $(v_2 \rightarrow 1)$ to $F^1(T)$:

1. Clauses $C_1$ and $C_2$ are satisfied, and so are removed (both contain $v_2$).

2. Clauses $C_3$ and $C_4$ both contain $\overline{v}_2$ and so this literal is removed.

This yields:

$$\langle v_2 \rightarrow 1 \rangle \ast F^1(T) = \{ \{v_1, v_4\}, \{v_1, \overline{v}_4\}, \{v_1, v_5\}, \{v_1, \overline{v}_5\} \}$$

The satisfaction (removal) of clauses and removal of literals is illustrated directly on $T$ in Figure 2 with dotted and dashed lines for clause and literal removal respectively. The tree corresponding to $\langle v_2 \rightarrow 1 \rangle \ast F^1(T)$ is illustrated in Figure 3.

Figure 1: Illustration of application of $\langle v_2 \rightarrow 1 \rangle$ to $F^1(T)$. Dotted lines indicate that the clauses corresponding to the effected leaves are satisfied; dashed lines indicate that the corresponding literal is falsified and therefore removed from all clauses.

Corollary 5.7 $SMU_{6=1}$ is stable under application of partial assignments, that is, for $F \in SMU_{6=1}$ and $\varphi \in \text{PASS}$ holds $\varphi \ast F \in SMU_{6=1}$.

From Lemma 5.2 follows $SMU_{6=1} \subseteq \text{UHIT}$, where $\text{HIT} \subseteq \text{CLS}$ is the set of hitting clause-sets, that is, those $F \in \text{CLS}$ where every two clauses clash in at least one literal, i.e., for all $C, D \in F$, $C \neq D$, we have $|C \cap \overline{D}| \geq 1$, and $\text{UHIT} := \text{HIT} \cap \text{USAT}$. It is well-known that $\text{UHIT} \subseteq SMU$ holds (for a proof see Lemma 2 in [18]).
5.2 Total minimal premise sets

We are interested in clause-sets which have as many sub-mps’s as possible:

Definition 5.8 A clause-set $F \neq \top$ is a total mps if $\text{mps}(F) = \mathbb{P}(F) \setminus \{\top\}$.

Every total mps is an mps.

Example 5.9 $\{\{a, b\}, \{\overline{a}, b\}\}$ is a total mps, while $\{\{a, b\}, \{\overline{a}\}\}$ is an mps (since minimally unsatisfiable), but not a total mps.

To determine all total mps’s, the central task to determine the minimally unsatisfiable total mps’s. Before we can prove that these are precisely the saturated minimally unsatisfiable clause-sets of deficiency 1, we need to state a basic property of these clause-sets, which follows by definition of $T^1(F)$ for $F \in S\mathcal{M}U_{d=1}$ (recall Subsection 5.1):

Lemma 5.10 Consider $F \in S\mathcal{M}U_{d=1}$ and $F' \subseteq F$. Let $T := T^1(F)$. The set $\text{puc}(F')$ of pure literals of $F'$ can be determined as follows:

1. Let $W_F := \{w_C : C \in F'\} \subseteq \text{lvs}(T)$ be the set of leaves corresponding to the clauses of $F'$.

2. For a literal $x \in \text{lit}(F)$ let $w \in \text{nds}(T)$ be the node labelled with $\text{var}(x)$, and let $T_x$ the the subtree of $w$ reached by $x$, and let $T_{\overline{x}}$ be the subtree of $w$ reached by $\overline{x}$.

3. Now $x \in \text{puc}(F')$ if and only if $W_{F'} \cap \text{lvs}(T_x) \neq \emptyset$ and $W_{F'} \cap \text{lvs}(T_{\overline{x}}) = \emptyset$.

Example 5.11 Consider the clause-set

$$F := \{\{v_1, v_2, v_3\}, \{v_1, v_2, \overline{v_3}\}, \{v_1, \overline{v_2}, v_4\}, \{v_1, \overline{v_2}, \overline{v_4}\}, \{v_1, \overline{v_5}, v_6\}, \{v_1, \overline{v_5}, \overline{v_6}\}, \{v_1, \overline{v_5}\}\}$$

and the subset $F' := \{C_1, C_3, C_4, C_7\}$. The tree $T^1(F)$ is as follows, with the dashed
edges representing literals not in $\bigcup F' = \{v_1, v_2, v_3, v_4, \overline{v_1}, \overline{v_2}, \overline{v_3}\}$:

We have $W_{F'} = \{1, 3, 4, 7\}$ and

$$\text{puc}(F') = \bigcup F' \setminus \{\overline{v_2}, \overline{v_1}, v_4, \overline{v_1}\} = \{v_3, \overline{v_5}\}.$$ 

Now consider $x \in \text{lit}(F)$:

1. For $x = v_3$ holds $\text{lvs}(T_{v_3}) \cap W_{F'} = \{1\}$ and $T_{\overline{v_3}} \cap W_{F'} = \emptyset$, thus $v_3 \in \text{puc}(F')$.
2. For $x = \overline{v_5}$ holds $\text{lvs}(T_{\overline{v_5}}) \cap W_{F'} = \{7\}$ and $T_{v_5} \cap W_{F'} = \emptyset$, thus $\overline{v_5} \in \text{puc}(F')$.
3. Considering for example $x = v_1$, we have $\text{lvs}(T_{v_1}) \cap W_{F'} = \{1, 3\}$ and $\text{lvs}(T_{\overline{v_1}}) \cap W_{F'} = \emptyset$, thus $v_1 \notin \text{puc}(F')$, while for $x = v_6$ we have $\text{lvs}(T_{v_6}) \cap W_{F'} = \emptyset$ and $\text{lvs}(T_{\overline{v_6}}) \cap W_{F'} = \emptyset$, thus $v_6 \notin \text{puc}(F')$.

**Theorem 5.12** An unsatisfiable clause-set $F \in \text{USAT}$ is a total mps if and only if $F \in \text{SMU}_{\delta=1}$.

**Proof:** First assume that $F$ is a total mps. Then every two clauses $C, D \in F$, $C \neq D$, clash in exactly one literal (otherwise $\{C, D\} \notin \text{MPS}$). In [44], Corollary 34, it was shown that an unsatisfiable clause-sets $F$ has precisely one clash between any pair of different clause-sets iff $F \in \text{SMU}_{\delta=1}$ holds (an alternative proof was found in [45]). Now assume $F \in \text{SMU}_{\delta=1}$, and we have to show that $F$ is a total mps. So consider $F' = \mathbb{P}(F) \setminus \{\top\}$, and let $C := \text{puc}(F)$, $\varphi := \varphi_C$. Since $F'$ is a hitting clause-set, $\varphi$ is contraction-free for $F'$, and according to Lemma 4.9 it remains to show that $F'' := \varphi \ast F'$ is unsatisfiable (recall that hitting clause-sets are irredundant). Assume that $F''$ is satisfiable, and consider a partial assignment $\psi$ with $\psi \ast F'' = \top$ and $\text{var}(\psi) \cap \text{var}(\varphi) = \emptyset$. We show that then $\varphi \cup \psi$ would be a satisfying assignment for $F$, contradicting the assumption. To this end it suffices to show that for all $D \in F \setminus F'$ holds $C \cap D = \emptyset$. Consider $T := T^1(F)$, and let $W_{F'}$ be defined as in Lemma 5.10. Starting from the leaf $w_D$, let $w$ be the first node on the path to the root of $T$ such that one of the two subtrees of $w$ contains a leaf of $W_{F'}$. Let $\overline{x}$ be the literal at $w$ on the path to $w_D$. So by Lemma 5.10, we have $x \in C$, while by definition $\overline{x} \in D$. \qed

**Corollary 5.13** For a clause-set $F \in \text{CLS}$ the following properties are equivalent:

1. $F$ is a total mps.
2. $\varphi_{\text{puc}(F)} \ast F \in \text{SMU}_{\delta=1}$, and $\varphi_{\text{puc}(F)}$ is contraction-free for $F$.

*In [44] the notation “$UHIT$” was used to denote “uniform hitting clause-sets”, which is now more appropriately called “(conflict-)regular hitting clause-sets”, while “$U$” now stands for “unsatisfiable”. \footnote{In [44] the notation “$UHIT$” was used to denote “uniform hitting clause-sets”, which is now more appropriately called “(conflict-)regular hitting clause-sets”, while “$U$” now stands for “unsatisfiable.”} \footnote{In [44] the notation “$UHIT$” was used to denote “uniform hitting clause-sets”, which is now more appropriately called “(conflict-)regular hitting clause-sets”, while “$U$” now stands for “unsatisfiable.”}
**Proof** Let $F' := \varphi_{\text{puc}(F)} \ast F$. If $F$ is a total mps, then by Lemma 4.9 follows $F' \in \mathcal{M}U$, where $\varphi_{\text{puc}(F)}$ is contraction-free for $F$. Also by Lemma 4.9 follows then, that $F' \in \mathcal{MPS}$, and thus by Theorem 5.12 we obtain $F' \in \mathcal{S}\mathcal{M}\mathcal{U}_2$. For the other direction, if $F' \in \mathcal{S}\mathcal{M}\mathcal{U}_{\delta=1}$ holds, where $\varphi_{\text{puc}(F)}$ is contraction-free for $F$, then by Theorem 5.12 follows that $F'$ is a total mps, which by Lemma 4.13 yields that $F$ is a total mps. 

Thus we can precisely construct all total mps’s, if we start the process described in Corollary 4.10 not with an arbitrary $F \in \mathcal{M}U$, but with an $F \in \mathcal{S}\mathcal{M}\mathcal{U}_{\delta=1}$.

**Example 5.14** That every 2-element sub-clause-set of $F \in \mathcal{CLS}$ is an mps, that is, every two (different) clauses of $F$ clash in precisely one literal, says that $F$ is 1-regular hitting in the terminology of [4], Section 6. For $F \in \mathcal{USAT}$ the proof of Theorem 5.12 shows, that $F$ is a total mps iff $F$ is 1-regular hitting. However for $F \in \mathcal{SAT}$ this is not true, and the simplest example is $F := \{\overline{a}, b\}, \{\overline{c}, c\}, \{\overline{a}, a\}$: $F$ is 1-regular hitting, but has no pure literal and is satisfiable, and thus $F \notin \mathcal{MPS}$.

In this case we have $\delta(F) = 0$. For an interesting example with deficiency 1 see Section 5 in [4].

We arrive at a simple and perspicuous proof of the main result of [56], that the clause-sets $F$ with $|\text{prc}_0(F)| = 2^{c(F)} - 1$ are precisely the clause-sets $D(F)$ for $F \in \mathcal{S}\mathcal{M}\mathcal{U}_{\delta=1}$ when allowing to replace the single doping variable of a clause by any non-empty set of new (pure) literals:

**Lemma 5.15** For $F \in \mathcal{CLS} \setminus \{\top\}$ holds $|\text{prc}_0(F)| = 2^{c(F)} - 1$ if and only if the following two conditions hold:

1. $F$ is a total mps.
2. For every clause $C \in F$ there is $x \in C$ such that $\text{var}(x) \notin \text{var}(F \setminus \{C\})$.

**Proof** First assume $|\text{prc}_0(F)| = 2^{c(F)} - 1$. Thus the map $F' \in \text{mps}(F) \mapsto \text{puc}(F') \subseteq \{C \in \mathcal{CLS} : F \models C\}$, which according to Lemma 4.12 covers $\text{prc}_0(F)$, must indeed be a bijection from $\text{mps}(F)$ to $\text{prc}_0(F)$, and hence $F$ is a total mps (here we need $F \neq \top$). If there would be $C \in F$ such that for all $x \in C$ we have $\text{var}(x) \in \text{var}(F \setminus \{C\})$, then $\text{puc}(F) \subseteq \text{puc}(F \setminus \{C\})$, and thus $F \setminus \{C\}$ could not yield a prime implicate different from the prime implicate obtained from $F$.

The inverse direction follows by the observation, that the existence of the unique “doping literals” $x \in C$ has the consequence, that for $\top \subseteq F', F'' \subseteq F$ with $F' \neq F''$ we get $\text{puc}(F') \neq \text{puc}(F'')$, since these doping literals make a difference. \qed

### 5.3 Doping $\mathcal{S}\mathcal{M}\mathcal{U}_{\delta=1}$

We are turning now our attention to a closer understanding of the prime implicates $C$ of doped $F \in \mathcal{S}\mathcal{M}\mathcal{U}_{\delta=1}$. We start with their identification with non-empty sub-clause-sets $F'$ of $D(F)$:

**Lemma 5.16** Consider a clause-set $F \in \mathcal{S}\mathcal{M}\mathcal{U}_{\delta=1}$. By Theorem 5.12 each non-empty subset yields a minimal premise set. Thus by Theorem 4.13 we have:

1. $\text{prc}_0(D(F)) = \{\text{puc}(F') \mid \top \neq F' \subseteq D(F)\}$.
2. $|\text{prc}_0(D(F))| = 2^{c(F)} - 1$. 

22
Since the clauses of $D(F)$ can be identified with leaves of the tree $T^1(F)$, we obtain a bijection between non-empty sets $V$ of leaves of the tree $T^1(F)$ and prime implicates of $D(F)$:

**Definition 5.17** For $F \in SMU_{d=1}$ and $\emptyset \neq V \subseteq lvs(T^1(F))$ the clause $C_V$ is the prime implicate $puc(\{C_w \in F \mid w \in V\})$ of $D(F)$ according to Lemma 5.16. For $w \in lvs(T^1(F))$ we furthermore set $u_w := u_{C_w}$.

By Lemma 5.16

**Lemma 5.18** For $F \in SMU_{d=1}$ holds $prc_0(D(F)) = \{C_V \mid \emptyset \neq V \subseteq lvs(T^1(F))\}$.

How precisely from $V \subseteq lvs(T^1(F))$ the prime implicate $C_V$ is constructed shows the following lemma:

**Lemma 5.19** Consider $F \in SMU_{d=1}$ and $\emptyset \neq V \subseteq lvs(T^1(F))$. We have $C_V = U_V \cup P_V$, $U_V \cap P_V = \emptyset$, where

1. $U_V := \{u_w \mid w \in V\}$, and
2. $P_V := puc(F')$ for $F' := \{C_w : w \in V\}$ as given in Lemma 5.10, that is, $P_V$ is the set of literals $x$ such that $V \cap lvs(T_x) \neq \emptyset$ and $V \cap lvs(T_x^2) = \emptyset$.

**Example 5.20** Consider the clause-set

$$F := \{\{v_1, v_2\}, \{v_1, \overline{v_2}\}, \{v_3\}, \{\overline{v_1}, v_3\}\} \in SMU_{d=1}$$

corresponding to the tree

```
   v1
  /   \
 v2   v3
 /\   /\  \
1 2 3 4
```

with the doped clause-set

$$D(F) = \{\{v_1, v_2, u_1\}, \{v_1, \overline{v_2}, u_2\}, \{\overline{v_1}, v_3, u_3\}, \{\overline{v_1}, \overline{v_3}, u_4\}\}.$$  

Now consider the set $V := \{1, 3\}$. According to Definition 5.13 we have that $C_V = puc(\{v_1, v_2, u_1\}, \{\overline{v_1}, v_3, u_3\}) = \{v_2, v_3, u_1, u_3\}$. By Lemma 5.13 we have that $C_V = U_V \cup P_V$, where $U_V = \{u_1, u_3\}$ and $P_V = puc(\{v_1, v_2\}, \{\overline{v_1}, v_3\}) = \{v_2, v_3\}$. Note that for both $x \in \{v_2, v_3\} = P_V$ we have that $lvs(T_x) \cap V \neq \emptyset$ and $lvs(T_x^2) \cap V = \emptyset$, but we do not have this for $x \in \text{lit}(F) \setminus \{v_2, v_3\}$.

The hardness of $F$ as well as $D(F)$ is the Horton-Strahler number of $T^1(F)$:

**Lemma 5.21** Consider $F \in SMU_{d=1}$, and let $k := \text{hs}(T^1(F))$. Then we have $\text{hd}(F) = \text{hd}(D(F)) = k$.

**Proof:** Let $T := T^1(F)$. First we show $\text{hd}(F) = k$. We have $\text{hd}(F) \leq k$, since $T$ is by definition of $F = F^1(T)$ already a resolution tree (when extending the labelling of leaves to all nodes), deriving $\perp$ from $F$. To show $\text{hd}(F) \geq k$, we use Lemma 3.5 with $C := SMU_{d=1}$ and $h(F) := \text{hs}(T^1(F))$. Based on Lemma 5.3, we consider the effect on the Horton-Strahler number of assigning a truth value to one variable $v \in \text{var}(F)$. Let $w \in \text{nds}(T)$ be the (inner) node labelled with $v$, and let $T^w_0, T^w_1$ be
the left resp. right subtree hanging at \( w \). Now the effect of assigning \( \varepsilon \in \{0,1\} \) to \( v \) is to replace \( T_w \) with \( T_w^{e} \). Let \( T_v \) be the (whole) tree obtained by assigning \( \varepsilon \) to \( v \), that is, \( T_v := T^1(\langle v \rightarrow \varepsilon \rangle + F) \). If \( \text{hs}(T_w^e) = \text{hs}(T_w) \), then we have \( \text{hs}(T_v) \geq k - 1 \), since at most one increase of the Horton-Strahler number for subtrees is missed out now. Otherwise we have \( \text{hs}(T_0) = \text{hs}(T) \) or \( \text{hs}(T_1) = \text{hs}(T) \), since removal of the subtree with the smaller Horton-Strahler number has no influence on the Horton-Strahler number of the whole tree. So altogether Lemma 5.6 is applicable, which concludes the proof of \( \text{hd}(F) = k \).

For showing \( \text{hd}(D(F)) = k \) we use Lemma 4.24, so consider \( F' \subseteq F \) and \( \varphi \in \mathcal{P}_\mathcal{ASS} \) with \( \varphi \ast F' \in \mathcal{USAT} \), let \( F'' := \varphi \ast F' \), and we have to show \( \text{hd}(F'') \leq k \). W.l.o.g. \( \text{var}(\varphi) \subseteq \text{var}(F') \). By Corollary 5.4 we have that \( \varphi \ast F \in \mathcal{SMU}_{h_{0}=1} \), and thus \( \varphi \ast F = F'' \) must hold, and \( \text{hd}(F'') = \text{hs}(T^1(F'')) \) (by the first part). By Lemma 6.5 \( T^1(F'') \) results from \( T \) by a sequence of removing subtrees, and it is easy to see, that thus \( \text{hs}(T^1(F'')) \leq k \) holds. \( \square \)

We summarise what we have learned about \( D(F) \) for \( F \in \mathcal{SMU}_{h_{0}=1} \):

**Theorem 5.22** Consider \( F \in \mathcal{SMU}_{h_{0}=1} \).

1. For each clause-set \( F' \) equivalent to \( D(F) \) there is an injection \( i : D(F) \rightarrow F' \) with \( \forall C \in D(F) : C \subseteq i(C) \) (by Lemma 4.13).
2. \( D(F) \) is a total mps (by Corollary 5.13).
3. The prime implicates of \( D(F) \) are given by Lemmas 5.18, 5.14.
4. \( \text{hd}(D(F)) = \text{hs}(T^1(F)) \) (by Lemma 5.24).

## 6 Separating \( \mathcal{UC}_{k+1} \) from \( \mathcal{WC}_k \)

This section proves the main result of this article, Theorem 6.14, which exhibits for every \( k \geq 0 \) sequences \( (F^k_h)_{h \in \mathbb{N}} \) of small clause-sets of hardness \( k + 1 \), where every equivalent clause-set of hardness \( k \) (indeed of w-hardness \( k \)) is of exponential size. In this way we show that the \( \mathcal{UC}_k \) hierarchy as well as the \( \mathcal{WC}_k \) hierarchy is useful, i.e., equivalent clause-sets with higher \( \omega \)-hardness can be substantially shorter. These \( F^k_h \) are doped versions of clause-sets from \( \mathcal{SMU}_{h_{0}=1} \) (recall Theorem 5.22), which are “extremal”, that is, their underlying trees \( T^1(F^k_h) \) are for given Horton-Strahler number \( k + 1 \) and height \( h \) as large as possible.

The organisation of this section is as follows: In Subsection 6.1 the main tool for showing size-lower-bounds for equivalent clause-sets of a given \( \omega \)-hardness is established in Theorem 6.14. Subsection 6.2 introduces the “extremal trees”. Subsection 6.3 shows the main lower bound in Theorem 6.13 and applies it to show the separation Theorem 6.14.

### 6.1 Trigger hypergraphs

Our goal is to construct clause-sets \( F^k_h \) of hardness \( k + 1 \), which have no short equivalent clause-set \( F \) with \( \text{whd}(F) \leq k \), where w.l.o.g. \( F \subseteq \text{prc}_0(F^k_h) = \text{prc}_0(F) \). This subsection is about the general lower-bound method. How are we going to find a lower bound on the number of clauses of \( F \)? The property \( \text{whd}(F) \leq k \) means, that for every \( C \in \text{prc}_0(F) \) the unsatisfiable clause-set \( \varphi_C \ast F \) can be refuted by \( k \)-resolution. In order for \( k \)-resolution to have a chance, there must be at least one clause of length at most \( k \) in \( \varphi_C \ast F \) — and this necessary condition is all we consider. So our strategy is to show that every \( F \subseteq \text{prc}_0(F^k_h) \), such that for all \( C \in \text{prc}_0(F^k_h) \) there is a clause of length at most \( k \) in \( \varphi_C \ast F \), is big.
Definition 6.1 Consider \( k \in \mathbb{N}_0 \) and \( F \in \mathcal{CLS} \). The trigger hypergraph \( T_k(F) \) is the hypergraph with the prime implicates of \( F \) as its vertices, and for every prime implicate \( C \) of \( F \) a hyperedge \( E^k_C \). The hyperedge \( E^k_C \) contains all prime implicants \( C' \in \text{prc}_0(F) \) which are not satisfied by \( \varphi_C \) and yield a clause of size at most \( k \) under \( \varphi_C \). That is,

1. \( V(T_k(F)) := \text{prc}_0(F) \), and
2. \( E(T_k(F)) := \{ E^k_C | C \in \text{prc}_0(F) \} \),

where \( E^k_C := \{ C' \in \text{prc}_0(F) | C' \cap \overline{C} = \emptyset \land |C' \setminus C| \leq k \} \).

Note that the trigger hypergraph of \( F \in \mathcal{CLS} \) depends only on the underlying boolean function of \( F \), and thus for every equivalent \( F' \) we have \( T_k(F') = T_k(F) \).

Example 6.2 Consider the clause-set

\[
F := \{ \{v_1, \overline{v_3}, v_2\}, \{v_2, v_3, \overline{v_4}\}, \{v_2, \overline{v_3}, v_4\}, \{v_1, v_3, v_4\}, \{v_1, v_2\} \}.
\]

As shown in Example 8.2 of [28, 27] we have \( \text{prc}_0(F) = F \). The trigger hypergraph \( T_0(F) \) is (as always) the hypergraph with all singleton sets, i.e., \( E(T_0(F)) = \{ \{C_1\}, \ldots, \{C_6\} \} \). The hypergraphs \( T_k(F) \) for \( k \in \{1, 2\} \) are represented by Figures 3 and 4.

![Figure 3: \( T_1(F) \)](image)

![Figure 4: \( T_2(F) \)](image)

To interpret the diagrams:

1. An arrow from a clause \( C \) to a clause \( D \) represents that \( C \in E^k_D \).
2. A dotted arrow from \( C \) to \( D \) represents that \( |D \setminus C| > k \) (so \( C \notin E^k_D \)), but \( C \cap \overline{D} = \emptyset \), and thus for some large enough \( k' > k \) we will have \( C \in E^{k'}_D \).
3. No arrow between \( C \) and \( D \) indicates that \( C \cap \overline{D} \neq \emptyset \) (i.e., for all \( k' \) we have \( C \notin E^{k'}_D \) and \( D \notin E^{k'}_C \)).
4. The size of a hyperedge \( E^k_D \) is the in-degree of the vertex \( D \).

Consider \( E^1_{C_6} = \{C_6\} \) and \( E^2_{C_6} = \{C_1, C_2, C_3, C_5, C_6\} \). As we will see in Lemma 6.3, therefore every \( F' \subseteq F \) equivalent to \( F \) such that \( F' \in \mathcal{UC}_1 \) must have \( C_6 \in F' \). However, \( E^2_{C_6} \) contains more clauses than \( E^1_{C_6} \), and for example \( F \setminus \{C_6\} \in \mathcal{UC}_2 \setminus \mathcal{UC}_1 \) as shown in Example 8.2 of [28, 27]. Using the above diagrammatic notation, we can also see that for all \( k' \geq 2 \) we have \( T_k(F) = T_k(F') \), as there are no dotted lines for \( T_2(F) \) (i.e., no clauses \( C \) and \( D \) such that \( |D \setminus C| > 2 \) but \( C \cap \overline{D} = \emptyset \)).
The point of the trigger hypergraph $T_k(F)$ is, that every clause-set equivalent to $F$ and of w-hardness at most $k$ must be a transversal of it:

**Lemma 6.3** Consider $k \in \mathbb{N}_0$ and $F \in \mathcal{CLS}$ with $\text{whd}(F) \leq k$. Then there is a clause-set $F'$ such that

1. $F' \subseteq \text{prc}_0(F)$ and $F'$ is equivalent to $F$;
2. there is an injection $i : F' \to F$ such that $\forall C \in F' : C \subseteq i(C)$;
3. $\text{whd}(F') \leq k$;
4. $F'$ is a transversal of $T_k(F)$.

**Proof:** Obtain $F'$ from $F$ by choosing for every $C \in F$ some $C' \in \text{prc}_0(F)$ with $C' \subseteq C$. Then the first two properties are obvious, while Property 3 follows from Part 1 of Lemma 6.1 in [45]. Assume that $F'$ is not a transversal of $T_k(F)$, that is, there is $C \in \text{prc}_0(F)$ with $F' \cap E_k^C = \emptyset$. Then $\varphi_C * F' \in \text{USAT}$, but every clause has length strictly greater than $k$, and thus $k$-resolution does not derive $\bot$ from $\varphi_C * F'$, contradicting $\text{whd}(F') \leq k$. □

Our lower bound method is now captured by the following theorem, which directly follows from Lemma 6.3:

**Theorem 6.4** For $k \in \mathbb{N}_0$ and $F \in \mathcal{WC}_k$ we have $c(F) \geq \tau(T_k(F))$.

Instead of lower-bounding the transversal number of $T_k(F)$, we use that every transversal has to have at least as many elements as there are disjoint hyperedges. So let $\nu(G)$ be the matching number of hypergraph $G$, the maximum number of pairwise disjoint hyperedges; we have $\tau(G) \geq \nu(G)$ for all hypergraphs $G$. So we have to show that there is a set $S \subseteq \text{prc}_0(F_k^h)$ of exponential size, such that the hyperedges $E_k^C$ for $C \in S$ are pairwise disjoint. For $F_k^h$ we use the doped clause-set $D(F^1(T))$ as considered in Subsection 5.3, where the special trees $T$ are constructed in the subsequent subsection.

### 6.2 Extremal trees

For a given hardness $k \geq 1$ we need to construct (full binary) trees which are as large as possible; this is achieved by specifying the height, and using trees which are “filled up” completely for the given parameter values:

**Definition 6.5** A pair $(k, h) \in \mathbb{N}_0^2$ with $h \geq k$ and $k = 0 \Rightarrow h = 0$ is called an **allowed parameter pair**. For an allowed parameter pair $(k, h)$ a full binary tree $T$ is called an **extremal tree of Horton-Strahler number $k$ and height $h$** if

1. $\text{hs}(T) = k$, $\text{ht}(T) = h$;
2. for all $T'$ with $\text{hs}(T') \leq k$ and $\text{ht}(T') \leq h$ we have $\text{nds}(T') \leq \text{nds}(T)$.

We denote the set of all extremal trees with Horton-Strahler number $k$ and height $h$ by $\text{HS}(k, h)$.

Note that for allowed parameter pairs $(k, h)$ we have $k = 0 \Leftrightarrow h = 0$. Extremal trees are easily characterised and constructed as follows:

1. $\text{HS}(0, 0)$ contains only the trivial tree (with one node).
2. HS(1, h) for \( h \in \mathbb{N} \) consists exactly of the full binary trees \( T \) with \( \text{hs}(T) = 1 \) and \( \text{ht}(T) = h \), which can also be characterised as those full binary trees \( T \) with \( \text{ht}(T) = h \) such that every node has at least one child which is a leaf.

3. For \( k \geq 2 \) and \( h \geq k \) we have \( T \in \text{HS}(k, h) \) iff \( T \) has the left subtree \( T_0 \) and the right subtree \( T_1 \), and there is \( \varepsilon \in \{0, 1\} \) with \( T_\varepsilon \in \text{HS}(k-1, h-1) \) and \( T_{1-\varepsilon} \in \text{HS}(\min(k, h-1), h-1) \).

**Lemma 6.6** For all allowed parameter pair \((k, h)\) we have \( \text{HS}(k, h) \neq \emptyset \).

The unique elements of \( \text{HS}(k, k) \) for \( k \in \mathbb{N}_0 \) are the perfect binary trees of Horton-Strahler number \( k \).

**Lemma 6.7** For an allowed parameter pair \((k, h)\) and for \( T \in \text{HS}(k, h) \) we have \( \#\text{lv}(T) = \alpha(k, h) := \sum_{i=0}^{k} \binom{h}{i} \). We have \( \alpha(k, h) = \Theta(h^k) \) for fixed \( k \).

**Proof:** For \( k \leq 1 \) we have \( \alpha(0, 0) = 1 \) and \( \alpha(1, h) = 1 + h \), which are obviously correct. Now consider \( k \geq 2 \). By induction hypothesis we get

\[
\#\text{nds}(T) = \alpha(k-1, h-1) + \alpha(\min(k, h-1), h-1).
\]

If \( h = k \), then \( \alpha(k, h) = 2^k \) (for all \( k \)), and we get \( \#\text{nds}(T) = \alpha(k-1, k-1) + \alpha(k-1, k-1) = 2 \cdot 2^{k-1} = 2^k = \alpha(k, k) \). Otherwise we have

\[
\#\text{nds}(T) = \alpha(k-1, h-1) + \alpha(k, h-1) = 
\sum_{i=0}^{k-1} \binom{h-1}{i} + \sum_{i=0}^{k} \binom{h-1}{i} = \binom{h-1}{0} + \sum_{i=1}^{k} \binom{h-1}{i-1} + \binom{h-1}{i} = 
\sum_{i=1}^{k} \binom{h}{i} = \alpha(k, h).
\]

**Example 6.8** Consider the following labelled binary tree \( T \):

![Diagram of a labelled binary tree]

Applying the recursive construction/characterisation we see \( T \in \text{HS}(2, 3) \). By simple counting we see that \( T \) has 7 leaves, in agreement with Lemma 6.6, i.e., \( \sum_{j=0}^{2} \binom{3}{j} = \binom{3}{0} + \binom{3}{1} + \binom{3}{2} = 1 + 3 + 3 = 7 \). Assuming that of the two subtrees at an inner node, the left subtree has Horton-Strahler numbers as least as big as the right subtree, the idea is that the sum runs over the number \( j \) of right turns in a path from the root to the leaves. In the above tree \( T \), the number of right turns is indicated as an index to the leaf-name. If the Horton-Strahler number is \( k \), with at most \( k \) right-turns we must be able to reach every leaf.
We summarise the additional knowledge over Theorem 5.22 (using additionally that most leaves of \( T \in HS(k, h) \) have depth precisely \( h \)):

**Lemma 6.9** Consider an allowed parameter pair \((k, h)\) and \( T \in HS(k, h) \), and let \( F := F^1(T) \).

1. \( n(D(F)) = 2 \cdot \alpha(k, h) - 1 \) (= \( \Theta(h^k) \) for fixed \( k \)).
2. \( c(D(F)) = \alpha(k, h) \) (= \( \Theta(h^k) \) for fixed \( k \)).
3. \( \ell(D(F)) \leq h \cdot \alpha(k, h) \) (= \( \Theta(h^{k+1}) \) for fixed \( k \)).
4. \( D(F) \in UC_k \setminus UC_{k-1} \) (for \( k \geq 1 \)).

In Theorem 6.14 we will see that these \( D(F) \) from Lemma 6.9 do not have short equivalent clause-sets of hardness \( k-1 \). A simple example demonstrates the separation between \( UC_0 \) and \( UC_1 \) (similar to [21], Example 2, which uses Example 6.1 from [77]):

**Example 6.10** The strongest separation is obtained by using \( F_k := D(F^1(T)) \) for \( T \in HS(1, h) \) and \( h \in \mathbb{N} \):

1. \( F^1(T) \), when considering all possible \( T \), covers precisely the saturated minimally unsatisfiable renamable Horn clause-set with \( h \) variables, which is up to isomorphism equal to \( \{\{v_1\}, \{v_1, v_2\}, \ldots, \{v_1, \ldots, v_h\}, \{v_1, \ldots, v_h\}\} \). By Lemma 5.4 these are precisely those \( F \in SMU_{h=1} \) with \( n(F) \geq 1 \) which contain a full clause.
2. \( n(F_h) = 2h + 1 \), \( c(F_h) = h + 1 \), and \( \text{hd}(F_h) = 1 \).
3. \( |\text{prc}_0(F_h)| = 2^{h+1} - 1 \).

Considering \( G_n := \{\{v_1\}, \ldots, \{v_n\}, \{v_1, \ldots, v_n\}\} \) for \( n \geq 2 \) and \( F_n := D(G_n) \) we obtain an example similar (but simpler) to Example 6.1 from [77]:

1. \( n(G_n) = n \) and \( c(G_n) = n + 1 \).
2. \( G_n \in MU_{h=1} \setminus SMU_{h=1} \). The above clause-sets \( F^1(T) \) are obtained precisely as saturations of the \( G_n \) (due to Lemma 5.4; a saturation adds literal occurrences until we obtain a saturated minimally unsatisfiable clause-set).
3. \( \text{mps}(G_n) \) consists precisely of the subsets of \( G_n \) containing the negative clause, plus the singleton-subsets given by the unit-clauses.
4. Thus \( |\text{mps}(G_n)| = 2^n + n \).
5. \( n(F_n) = 2n + 1 \), \( c(F_n) = n + 1 \), and \( \text{hd}(F_n) = 1 \).
6. \( |\text{prc}_0(F_n)| = 2^n + n \).

### 6.3 The exponential lower bound

The task is to find many disjoint hyperedges in \( T_k(F^k_h) \), where \( F^k_h := D(F^1(T)) \) for \( T \in HS(k+1, h) \). Our method for this is to show that there are many “incomparable” subsets of leaves in \( T \) in the following sense. The \textit{depth} of a node \( w \) in a rooted tree \( T \), denoted by \( d_T(w) \in \mathbb{N}_0 \), is the length of the path from the root of \( T \) to \( w \). Recall that two sets \( A, B \) are \textit{incomparable} iff \( A \nsubseteq B \) and \( B \nsubseteq A \). Furthermore we call two sets \( A, B \) \textit{incomparable on a set} \( C \) if the sets \( A \cap C \) and \( B \cap C \) are incomparable.
\textbf{Definition 6.11} Consider a full binary tree $T$, where every leaf has depth at least $k + 1$. Consider furthermore $\emptyset \subset V, V' \subseteq \text{lvs}(T)$. Then $V$ and $V'$ are \textit{depth-$k$-incomparable for $T$} if $V$ and $V'$ are incomparable on $\text{lvs}(T_w)$ for all $w \in \text{nds}(T)$ with $d_T(w) = k$.

Note that for all allowed parameter pairs $(k, h)$ and $T \in \text{HS}(k, h)$ every leaf has depth at least $k$.

\textbf{Lemma 6.12} Consider $k \in \mathbb{N}_0$, $T \in \mathcal{T}_1$, and $\emptyset \neq V_0, V_1 \subseteq \text{lvs}(T)$ which are depth-$k$-incomparable for $T$. Let $F := F^1(T)$ and consider $T_k(F)$ (recall Definition 6.4). Then the hyperedges $E^k_{C_{V_0}}, E^k_{C_{V_1}}$ are disjoint (recall Definition 7.4).

\textbf{Proof:} Assume that $E^k_{C_{V_0}}, E^k_{C_{V_1}}$ are not disjoint; thus there is $\emptyset \neq V \subseteq \text{lvs}(T)$ with $C_V \in E^k_{C_{V_0}} \cap E^k_{C_{V_1}}$. We will show that there is $\varepsilon \in \{0, 1\}$ with $|C_V \setminus C_{V_\varepsilon}| \geq k + 1$, which contradicts the definition of $T_k(F)$.

Since $V \neq \emptyset$, there is $w \in V$. Consider the first $k + 1$ nodes $w_1, \ldots, w_{k+1}$ on the path from the root to $w$. Let $w'_i$ be the child of $w_{i-1}$ different from $w_i$ for $i \in \{2, \ldots, k + 1\}$, and let $T_i := T_{w_{i+1}}$ for $i \in \{1, \ldots, k\}$, while $T_{k+1} := T_{w_{k+1}}$; see Figure 5. We show that each of $T_1, \ldots, T_{k+1}$ contributes at least two unique literals to $|C_V \setminus C_{V_0}| + |C_V \setminus C_{V_1}|$, so that we get $|C_V \setminus C_{V_0}| + |C_V \setminus C_{V_1}| \geq (k + 1) \cdot 2$, from which follows that there is $\varepsilon \in \{0, 1\}$ with $|C_V \setminus C_{V_\varepsilon}| \geq k + 1$ as claimed.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure5.png}
\caption{Illustration of sub-trees $T_1, \ldots, T_{k+1}$.}
\end{figure}

Due to the depth-$k$-incomparability of $V, V'$, for each $i \in \{1, \ldots, k + 1\}$ and each $\varepsilon \in \{0, 1\}$ there are nodes $v^*_i$ with $v^*_i \in (\text{lvs}(T_i) \cap V_\varepsilon) \setminus V_\varepsilon$. We have two cases now:

I If $v^*_i \in V$, then $u_{v^*_i} \subseteq C_V \setminus C_{V_\varepsilon}$.

II If $v^*_i \notin V$, then consider the first node $v$ on the path from $v^*_i$ to the root such that for the other child $v'$ of $v$, not on that path to the root, holds $\text{lvs}(T_{v'}) \cap V \neq \emptyset$: now for the literal $x$ labelling the edge from $v$ to $v'$ we have $x \in C_V \setminus C_{V_\varepsilon}$. Note that $v$ is below or equal to $w_i$ (due to $w \in V$).
For each \( \varepsilon \in \{0, 1\} \), the literals collected in \( C_V \setminus C_{V_\varepsilon} \) from these \( k+1 \) sources do not coincide, due to the pairwise node-disjointness of the trees \( T_1, \ldots, T_{k+1} \).

\[\square\]

**Theorem 6.13** Consider \( k \in \mathbb{N}_0 \), \( h \geq k+1 \), and \( T \in \text{HS}(k+1, h) \); let \( F := D(F^1(T)) \) and \( m := \alpha(1, h-k) = 1 + h-k \). We have

\[\nu(T_k(F)) \geq \left( \frac{m}{h} \right) \geq \frac{1}{\sqrt{2}} \frac{2^m}{\sqrt{m}} = \Theta\left( \frac{2^h}{\sqrt{h}} \right),\]

where the second inequality assumes \( h \geq k+5 \), while the \( \Theta \)-estimation assumes fixed \( k \).

**Proof:** For every \( S \subseteq \mathbb{P}(\text{lvs}(T)) \) with \( \emptyset \notin S \), such that every two different elements of \( S \) are depth-\( k \)-incomparable for \( T \), we have \( \nu(T_k(F)) \geq |S| \) by Lemma \ref{lem:nu_upper_bound}. We can actually determine the maximal size of such an \( S \), which is \( M := \binom{m}{m'} \), where \( m' := \lfloor \frac{m}{k+1} \rfloor \), as follows. Let \( T := \{ w : w \in \text{nds}(T) \land d_T(w) = k \} \); note that for \( T', T'' \in T \) with \( T' \neq T'' \) we have \( \text{lvs}(T') \cap \text{lvs}(T'') = \emptyset \). Choose \( T_0 \in T \) with minimal \( \#\text{lvs}(T_0) \); by Lemma \ref{lem:nu_lower_bound} we have \( \#\text{lvs}(T_0) = m \). Let \( S_0 := \{ V \in \text{lvs}(T_0) : V \in S \} \). Then \( S_0 \) is an antichain (i.e., the elements of \( S_0 \) are pairwise incomparable) and \( |S_0| = |S| \). By Sperner’s Theorem (\ref{thm:sperner}) holds \( |S_0| \leq M \), and this upper bound \( M \) is realised, just observing the antichain-condition, by choosing for \( S_0 \) the set \( \text{lvs}(T_0) \) of subsets of \( \text{lvs}(T_0) \) of size \( m' \). This construction of \( S_0 \) can be extended to a construction of \( S \) (of the same size) by choosing for each \( T' \in T \) an injection \( j_{T'} : S_0 \to \binom{\text{lvs}(T')}{m'} \) and defining \( S := \{ j_{T'}(V) \} \forall \in S_0 \). The given estimation of \( M \) follows from Stirling’s approximation. \(\square\)

We are now able to state the main result of this article, proving Conjecture 1.1 from \ref{con:unc} that \( \mathcal{UC}_k \), and indeed also \( \mathcal{WC}_k \), is a proper hierarchy of boolean functions regarding poly-size representations without auxiliary variables:

**Theorem 6.14** Consider \( k \in \mathbb{N}_0 \). We have \( \mathcal{WC}_k \rightarrow \mathcal{UC}_{k+1} \). The details are as follows.

For \( h \geq k+1 \) choose one \( T_h \in \text{HS}(k+1, h) \) (note there is up to left-right swaps exactly one element in \( \text{HS}(k+1, h) \)), and let \( F_h := D(F^1(T_h)) \). Consider the sequence \( (F_h)_{h \geq k+1} \).

1. By Lemma \ref{lem:nu_upper_bound} we have \( n(F_h) = \Theta(h^{k+1}) \) as well as \( c(F_h) = \Theta(h^{k+1}) \), and \( F_h \in \mathcal{UC}_{k+1} \).

2. Consider a sequence \( (F'_h)_{h \geq k+1} \) of clause-sets with \( F'_h \) equivalent to \( F_h \), such that \( F'_h \in \mathcal{WC}_k \). By Theorems \ref{thm:nu_upper_bound}, \ref{thm:nu_lower_bound} we have \( c(F'_h) = \Omega\left( \frac{2^h}{\sqrt{h}} \right) \).

We conjecture that Theorem \ref{thm:wc_uk} can be strengthened by including the PC-hierarchy in the following way:

**Conjecture 6.15** For every \( k \in \mathbb{N}_0 \) we have \( \mathcal{WC}_k \rightarrow \mathcal{PC}_{k+1} \).

A step towards Conjecture \ref{con:pc_uk} is Theorem \ref{thm:pc_uk}.

**7 Knowledge compilation properties**

In view of the above separation result for the hierarchies \( \mathcal{UC}_k, \mathcal{PC}_k \) and \( \mathcal{WC}_k \), we now place these hierarchies in their context in the knowledge compilation (KC) literature. First we need to review the basic setting. A boolean function \( f \) is
considered here as a map $f : \mathcal{TASS}(V) \to \{0, 1\}$, where $V \subset \mathcal{V}A$ is a finite set of variables, while $\mathcal{TASS}(V) := \{\varphi \in \mathcal{PASS} : \text{var}(\varphi) = V\}$ is the set of all total assignments on $V$. For every $\varphi \in \mathcal{PASS}$ with $\text{var}(\varphi) \supseteq V$ the value $f(\varphi) \in \{0, 1\}$ is defined via restriction of $\varphi$ to $V$. Knowledge compilation is about “representations” $F$ (in a general sense) of boolean functions $f$ such that basic queries can be answered efficiently:

- **Consistency checking (CO):** input $F$, whether $f$ is constant 0 or not.
- **Clausal entailment checking (CE):** input $F$ and clause $C$, whether $\varphi \ast \{C\} = \top$ for all $\varphi \in \mathcal{TASS}(V)$ with $f(\varphi) = 1$.
- **Validity checking (VA):** input $F$, whether $f$ is constant 1 or not.
- **Implicant checking (IM):** input $F$ and partial assignment $\varphi$ with $\text{var}(\varphi) \subseteq V$, whether $f(\varphi') = 1$ for all $\varphi' \in \mathcal{TASS}(V)$ with $\varphi' \supseteq \varphi$.
- **Semantic entailment (SE):** input $F, F'$, whether $f \models f'$ (i.e., whether for all $\varphi$ with $\text{var}(f) \cup \text{var}(f') \subseteq \text{var}(\varphi)$ holds $f(\varphi) = 1 \Rightarrow f'(\varphi) = 1$).
- **Equivalence checking (EQ):** input $F, F'$; whether $f \models f'$ and $f' \models f$.
- **Model enumeration (ME):** input $F$, enumerate $\varphi \in \mathcal{TASS}(V)$ with $f(\varphi) = 1$.
- **Model counting (MC):** input $F$, count $\varphi \in \mathcal{TASS}(V)$ with $f(\varphi) = 1$.

The motivation of [20] for defining $\mathcal{UC}$ was to introduce a class of clause-sets for knowledge compilation, such that these basic queries have the same query complexity as the PI class (where PI is the same as $\mathcal{UC}_0$). Theorem 7.1 now shows that $\mathcal{UC}_k, \mathcal{PC}_k$ and $\mathcal{WC}_k$ all fulfil the same criteria. This result along with Theorem 6.14 means, that $\mathcal{UC}_k, \mathcal{PC}_k$ and $\mathcal{WC}_k$ offer intermediate target classes for knowledge compilation inbetween the CNF and PI classes, where the parameter $k$ allows query time to be traded for size. Every fixed level of each hierarchy (except of $\mathcal{PC}_0$) is a complete class with respect to representation of boolean functions, unlike classes such as $2-C\ell S$ (CNF clause-sets with clauses of size at most two) or $\mathcal{HO}$ (Horn clause-sets), or other hierarchies for polynomial time satisfiability like $\mathcal{RHOC}_k$ (generalised renamable Horn clause-sets), each of which is included (as classes) at some fixed level of $\mathcal{UC}_k$.

**Theorem 7.1** For all fixed $k \in \mathbb{N}_0$ and all $F, F' \in \mathcal{WC}_k \supseteq \mathcal{UC}_k \supseteq \mathcal{PC}_k$, the queries $CO, CE, VA, IM, EQ, SE$ (as specified above) are decidable in polynomial time (in $\ell(F)$). Furthermore, ME, i.e., enumerating all satisfying assignments, is possible in time $p(\ell(F), m)$ for some fixed polynomial $p$, where $m$ is the number of satisfying total assignments for $F$.

**Proof:** That clausal entailment is decidable in poly-time for $\mathcal{WC}_k$ is shown in Subsection 6.5 of [40]. Since we are dealing with clause-sets (conjunctions of clauses), this implies that the other five query-decisions can be done in polynomial time (where $F$ is valid (a tautology) iff $F = \top$, while $\varphi$ corresponds to an implicant iff $\varphi \ast \top \Rightarrow F = \top$). Finally, that all models can be enumerated in poly-time in $\ell(F)$ and $m$ follows from the fact that we can build a decision tree with at most $m$ true-leaves and at most $n(F) \cdot m$ false-leaves (compare Lemma A.3 in [33]).

We finish with an overview on the status of polytime queries for our classes and well-known KC classes in Figure 3:

- **NNF** means “negation normal form”, which are circuits with AND’s and OR’s or unbounded fan-in and where the inputs are literals.
• DNNF means “decomposable NNF”, that is, different children of any AND do not have common variables.

• d-DNNF means “deterministic DNNF”, where additionally any two different children of any OR must be logically contradictory.

• DNF’s (disjunctive normal forms) are special DNNF’s.
  – DNF’s are just clause-sets, but now interpreted as disjunction of conjunction (not as CNF’s, which is the default for clause-sets, that is, conjunctions of disjunctions).
  – The intersection of d-DNNF and DNF is the class of orthogonal DNF’s (as clause-sets the hitting clause-sets, that is, each two different clauses have at least one clash).

• MODS (like “models”) are the DNF’s where each DNF-clause contains all variables (so these are special orthogonal DNF’s).

• IP means prime implicants, which as clause-sets is the same is PI (prime implicates), that is, $\mathcal{UC}_0 = \mathcal{WC}_0$ after removal of subsumed clauses, but interpreted as DNF’s.

• BDD means “binary decision diagrams”, OBDD means “ordered (reduced) BDD”, while for OBDD$_\leq$ one global order on the variables is used.

| L   | CO | VA | CE | IM | EQ | SE | CT | ME |
|-----|----|----|----|----|----|----|----|----|
| NNF | o  | o  | o  | o  | o  | o  | o  | o  |
| DNNF| ✓  | o  | ✓  | o  | o  | o  | o  | ✓  |
| d-DNNF| ✓  | ✓  | ✓  | ✓  | ✓  | ?  | o  | ✓  |
| BDD | o  | o  | o  | o  | o  | o  | o  | o  |
| OBDD| ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| OBDD$_\leq$| ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| DNF | ✓  | o  | ✓  | o  | o  | o  | o  | ✓  |
| IP  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| MODS| ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  |
| CNF | o  | ✓  | o  | ✓  | o  | o  | o  | o  |
| $\mathcal{WC}_k$| ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | o  | ✓  |
| $\mathcal{UC}_k$| ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | o  | ✓  |

Figure 6: Subsets of the NNF language and their corresponding polytime queries. ✓ means “query possible in polytime”, o means “not possible in polytime (in general) unless P = NP”, and ? means that there is no known result either way. Results for $\mathcal{WC}_k, \mathcal{UC}_k$ are from Theorem 7.1, and all other results are from [18].

8 Separating $\mathcal{UC}$ from $\mathcal{PC}$

Using [2], we show now that there is a polysize sequence in $\mathcal{UC}$ such that no equivalent polysize sequence exists in $\mathcal{PC}$ (where the separation in fact is exponential).

**Theorem 8.1** We have $\mathcal{PC} \not\rightarrow \mathcal{UC}$. This separation is achieved by the polysize family $(M_q)_{q \in \mathbb{N}}$ from Theorem 5.8 in [2].

• In Definition 5.2 in [2] for $q \in \mathbb{N}$, disjoint sets $X, Y, Z$ of size $q$ and $W \subseteq X \times Y \times Z$ the clause-set $\varphi_W$ is defined.
Now \( M_q := \varphi_{X \times Y \times Z} \) for Theorem 5.8 in [3].

It is shown in [2] that \( (M_q)_q \in \mathbb{N} \) has no equivalent polysize sequence in \( \mathcal{PC} \), while we have that \( (M_q)_q \in \mathbb{N} \) is in \( \mathcal{UC} \).

**Proof:** It remains to show that \( M_q \in \mathcal{UC} \). The proof sketch is as follows. The "hidden clause-set" in \( M_q \) is

\[
F_q := \{ \{a_1, \overline{a_2}\}, \{a_2, \overline{a_3}\}, \ldots, \{a_{q-1}, \overline{a_q}\}, \{a_q, a_1\} \} \in 2-\text{CLS} \cap \text{SAT}.
\]

We have \( \text{hd}(F_q) = 1 \) (by Lemma 6.6 in [27]), and thus also for all partial assignments \( \varphi \) holds \( \text{hd}(\varphi \ast F_q) \leq 1 \). Now the clauses of \( \varphi \ast F_q \) are the only clauses potentially usable in resolution refutations of \( \varphi \ast M_q \), using the fundamental insight from [43] (or see [39]), that clauses satisfiable by some autarky can not participate in any resolution refutation, while the definition of \( M_q \) precisely makes all clauses in \( \varphi \ast M_q \) satisfiable by some autarky, which still contain one of the other variables \( b^i_i, c^j_i \).

\( \square \)

Generalising Theorem 8.1 to a separation of \( \mathcal{UC}_k \) from \( \mathcal{PC}_k \) for all \( k \geq 1 \) requires more work:

**Conjecture 8.2** For all \( k \in \mathbb{N}_0 \) we have \( \mathcal{PC}_k \not\rightarrow \mathcal{UC}_k \).

With Theorem 8.1 we know Conjecture 8.2 for \( k \leq 1 \).

**9 Conclusion and open problems**

We conclude by directions for future research.

**9.1 A complete picture**

Conjecture 1.2 (recall the discussion in Subsection 1.2) paints a complete picture regarding the relations of the classes \( \mathcal{PC}_k, \mathcal{UC}_k, \mathcal{WC}_k \) w.r.t. polysize representation of equivalent boolean functions. By Theorem 6.14 we have \( \mathcal{WC}_k \not\rightarrow \mathcal{UC}_{k+1} \) for \( k \geq 0 \), and by Theorem 8.1 we have \( \mathcal{PC}_1 \not\rightarrow \mathcal{UC}_1 \). We now discuss what remains to be shown.

Conjecture 6.13 claims \( \mathcal{WC}_k \not\rightarrow \mathcal{PC}_{k+1} \). This would also imply \( \mathcal{PC}_k \not\rightarrow \mathcal{PC}_{k+1} \) (we already know \( \mathcal{UC}_k \not\rightarrow \mathcal{UC}_{k+1} \) and \( \mathcal{WC}_k \not\rightarrow \mathcal{WC}_{k+1} \), while currently we only know \( \mathcal{PC}_k \not\rightarrow \mathcal{PC}_{k+2} \)).

Conjecture 8.2 claims \( \mathcal{PC}_k \not\rightarrow \mathcal{UC}_k \), and additionally we have

**Conjecture 9.1** For all \( k \geq 2 \) holds \( \mathcal{UC}_k \not\rightarrow \mathcal{WC}_k \).

Since \( \mathcal{WC}_3 \not\subseteq \mathcal{UC}_k \), we get

**Conjecture 9.2** For all \( k \geq 0 \) holds \( \mathcal{UC}_k \not\rightarrow \mathcal{WC}_3 \).

And from \( \mathcal{WC}_2 \not\subseteq \mathcal{UC}_3 \) we get

**Conjecture 9.3** \( \mathcal{UC}_3 \not\rightarrow \mathcal{WC}_2 \).

Conjectures 1.2, 1.3 together imply Conjecture 9.1. If Conjecture 3.1 is true, then we get the stronger form (which implies Conjectures 9.1, 9.1, 9.3):

**Conjecture 9.4** For all \( k \geq 0 \) holds \( \mathcal{UC}_k \not\rightarrow \mathcal{WC}_2 \).

\(^9\)Furthermore \( \text{phd}(F_q) = 2 \) (since the literal \( a_1 \) is forced), which causes \( M_q \not\in \mathcal{PC} \), but this is not our concern here.
9.2 Alternative hierarchies for representations

In a sense, the $\text{UC}_k = \text{SLUR}_k$ hierarchy unified the three predecessor hierarchies $\text{SLUR}(k)$ introduced in \[31\], $\text{SLUR}^*(k)$ introduced in \[14\], and $\text{CANON}(k)$ introduced in \[3\]. In \[20\, 23\, 27\] we compared them directly (as sets of clause-sets) to the UC-hierarchy, and showed that they were properly included. An interesting question is now whether these more “shallow” hierarchies (for each level the CE-queries can be answered in linear time) also form strict hierarchies w.r.t. polysize representations without new variables. It is rather easy to see that the hierarchy $\text{CANON}(k)$ collapses to $\text{CANON}(0) = \text{UC}_0$:

Lemma 9.5 For $F \in \text{CLS}$ let $k(F)$ be the minimal $k \in \mathbb{N}_0$ such that $F \in \text{CANON}(k)$. Then the function $\text{prc}_0 : \text{CLS} \rightarrow \text{CANON}(0) = \text{UC}_0$ can be computed in time $O(c(F)^{3 \cdot 2^k} \cdot \ell(F))$, when the input is $F$ together with $k := k(F)$.

Proof: Let $K := 2^k$. So for every $C \in \text{prc}_0(F)$ there exists $F' \subseteq F$ with $F' \models C$ and $c(F') \leq K$, since a resolution tree of height $k$ has at most $K$ leaves. Now we compute $\text{prc}_0(F)$ as follows:

1. Set $P := \emptyset$.
2. Run through all $F' \subseteq F$ with $c(F') \leq K$; their number is $O(c(F)^K)$.
3. For each $F'$ determine whether $F' \models \text{puc}(F')$ holds, in which case clause $\text{puc}(F')$ is added to $P$; note that the test can be performed in time $O(2^k \cdot K)$.
4. The final $P$ obtained has $O(c(F)^K)$ many elements. After performing subsumption elimination (in cubic time) we obtain $\text{prc}_0(F)$ (by Lemma \[17\]). \qed

It is an interesting question whether also the hierarchies $\text{SLUR}(k)$, $\text{SLUR}^*(k)$ collapse or not, and whether they can be reduced to some $\text{UC}_k$ (for some fixed $k$).

9.3 Compilation procedures

For a given boolean function $f$ and $k \in \mathbb{N}_0$, how do we find algorithmically a “small” equivalent $F \in \text{UC}_k$? In \[20\], Section 8, the notion of a “$k$-base for $f$” is introduced, which is an $F \in \text{UC}_k$ equivalent to $f$, with $F \subseteq \text{prc}_0(f)$ and where no clause can be removed without increasing the hardness or destroying equivalence. It is shown that if $f$ is given as a 2-CNF, then a smallest $k$-base is computable in polynomial time, but even for $f$ with given $\text{prc}_0(f)$, where $\text{prc}_0(f)$ is a Horn clause-set, deciding whether a $k$-base of a described size for a fixed $k \geq 1$ exists is NP-complete.

There are interesting applications where $\text{prc}_0(f)$ is given (or can be computed), and where then some small equivalent $F \in \text{UC}_k$ is sought. The most basic approach filters out unneeded prime implicates; see \[23\, 24\] for some initial applications to cryptanalysis. A simple filtering heuristic, used in \[23\, 24\], is to favour (keeping) short-clauses. In a first phase, starting with the necessary elements of $\text{prc}_0(f)$, further elements are added (when needed) in ascending order of size for building up the initial $F \in \text{UC}_k$ (which in general is not a base). In the second phase, clauses from $F$ are removed in descending order of size when reducing to a $k$-base. The intuition behind this heuristic is that small clauses cover more total assignments (so fewer are needed), and they are also more likely to trigger $r_k$, making them more useful in producing small, powerful representations. Essentially the same heuristic is considered in \[1\] (called “length-increasing iterative empowerment”) when generating representations in $\text{PC}$.

For the case that $f$ is given by a CNF $F_0$, in \[20\, 55\] one finds refinements of the resolution procedure applied to $F_0$, which would normally compute $\text{prc}_0(f)$, i.e., the 0-base in $\text{UC}_0$, and where by some form of “compression” now an equivalent $F \in \text{UC}_1$ is computed. This approach needed to be generalised to arbitrary $\text{UC}_k$. 

34
9.4 Allowing auxiliary variables

In this report we considered representations of boolean functions by equivalent CNF-clause-sets, to be used in constructing “good” SAT translations or in knowledge compilation (KC). An important advantage of this approach is the ability to systematically search for good representations, as discussed in the previous Subsection 9.3. However it is also well-known that without the use of auxiliary variables, many relevant boolean functions do only have very large equivalent CNF-clause-sets at all. So we consider now the extension of the picture, as developed in the previous Subsection 9.1, by allowing auxiliary variables.

9.4.1 The notion of “CNF-representation”

First a framework for the meaning of auxiliary variables is needed. In general it is understood that existential quantification of the auxiliary variables is the right condition. In the SAT-context, it seems best to keep the quantification implicit, and we arrive at the following notion: A **CNF-representation** (possibly with auxiliary variables) of a boolean function \( f \) is a clause-set \( F \) with \( \text{var}(f) \subseteq \text{var}(F) \) such that the satisfying assignments of \( F \), projected to \( \text{var}(f) \), are precisely the satisfying assignments of \( f \), or, in other words, if for \( \varphi \in \text{TASS}(\text{var}(f)) \) holds \( f(\varphi) = 1 \Leftrightarrow \varphi \ast F \in \text{SAT} \).\(^{10}\)

Note that if for a CNF-representation \( F \) of \( f \) holds \( \text{var}(F) = \text{var}(f) \), then \( F \) is logically equivalent to \( f \). A sequence \( (F_n)_{n \in \mathbb{N}} \) is called a **CNF-representation of** \((F_n)_{n \in \mathbb{N}}\) if for all \( n \in \mathbb{N} \) the clause-set \( F_n \) is a CNF-representation of \( F_n \).

**Lemma 9.6** A clause-set \( F \in \text{CLS} \) is a CNF-representation of a boolean function \( f \) with \( \text{var}(f) \subseteq \text{var}(F) \) if and only if \( \text{prc}_0(f) = \{C \in \text{prc}_0(F) : \text{var}(C) \subseteq \text{var}(F)\} \).

**Proof:** Let \( V := \text{var}(f) \). First assume that \( F \) is a CNF-representation of \( f \). If \( C \in \text{prc}_0(f) \), then due to \( F \models f \) we have \( F \models C \), and if \( C \notin \text{prc}_0(F) \), then there would be \( C' \in \text{prc}_0(F) \) with \( C' \subseteq C \), and then for \( \varphi := \varphi_{C'} \) there would be an extension \( \varphi' \in \text{TASS}(V) \) with \( f(\varphi') = 1 \), but \( \varphi \ast F \in \text{USAT} \). For the other inclusion consider \( C \in \text{prc}_0(F) \) with \( \text{var}(C) \subseteq V \). If there would be an assignment \( \varphi \in \text{TASS}(V) \) with \( f(\varphi) = 1 \) but \( \varphi \ast \{C\} = \{\bot\} \), then we had \( \varphi \ast F \in \text{USAT} \) contradicting \( f(\varphi) = 1 \). So \( f \models C \), and due to \( F \models f \) it follows \( C \in \text{prc}_0(f) \).

Now assume \( \text{prc}_0(f) = \{C \in \text{prc}_0(F) : \text{var}(C) \subseteq V\} \), and we have to show that \( F \) is a CNF-representation of \( f \). So first consider \( \varphi \in \text{TASS}(V) \) with \( \varphi \ast F = \top \). If \( f(\varphi | V) = 0 \), then there would be \( C \in \text{prc}_0(f) \) with \( \varphi_{C} \subseteq \varphi \), but then \( C \in \text{prc}_0(F) \), and thus \( \varphi \ast F \in \text{USAT} \); so by contradiction \( f(\varphi | V) = 1 \). And if for \( \varphi \in \text{TASS}(V) \) holds \( f(\varphi) = 1 \) but \( \varphi \ast F \in \text{USAT} \), then there is \( C \in \text{prc}_0(F) \) with \( \varphi_{C} \subseteq \varphi \), and we had \( C \in \text{prc}_0(f) \). \( \square \)

As a KC “formalism” the CNF-representation of boolean functions is known as “∃CNF”, which we write as \( \exists \text{CLS} \), defined as the set of pairs \( (V, F) \) with \( F \in \text{CLS} \) and \( V \subseteq \text{var}(F) \); the variables of \( V \) are existentially quantified, and the boolean function represented by \( (V, F) \) is given by the QBF \( \exists v_1, \ldots, \exists v_m : F \), where \( V = \{v_1, \ldots, v_m\} \). Evaluation of the underlying boolean function is an NP-complete task, and so restrictions are needed to obtain efficient representations. A natural restriction is to demand that evaluation can be done by unit-clause propagation, and it is well-known that via the Tseitin-translation this corresponds, modulo linear-time transformations, to the circuit-representation of boolean functions (see \[12\] for closely related results). We call that class \( \exists \text{UP} \), the class of \( (V, F) \) with \( F \in \text{CLS} \),

---

\(^{10}\)To be completely precise, we needed to use “formal clause-sets” here, which can have variables actually not occurring in the clauses.
9.4.2 Absolute and relative condition

The question then is how to treat the classes \( \mathcal{PC}_k, \mathcal{UC}_k, \mathcal{WC}_k \) in this context. There are two possibilities, namely that the conditions constituting these classes also concern the auxiliary variables or not. The first case, that for a CNF representation \( F \) of \( f \) we simply require \( F \in \mathcal{PC}_k, \mathcal{UC}_k, \mathcal{WC}_k \) resp., we call the \textit{absolute condition}, while the second case we call the \textit{relative condition}. In the context of KC, the class \( \mathcal{UC} \) under the absolute condition is denoted by \( \mathcal{UC}[\exists] \) ([18]), while \( \exists \mathcal{UC} \) denotes the usage of the relative condition ([21]).

\( \mathcal{UC}[\exists] \) can be defined as pairs \((V, F)\), where \( F \in \mathcal{UC} \) while \( V \subseteq \text{var}(F) \) are the variables which are existentially quantified (the auxiliary variables). Similarly we get the “formalisms” \( \mathcal{UC}_k[\exists], \mathcal{PC}_k[\exists], \mathcal{WC}_k[\exists] \) for KC; the boolean functions represented by \((V, F)\) are obtained from the boolean functions given by the CNF \( F \) by projecting the satisfying assignments to \( V \). However \( \exists \mathcal{UC} \) can not be defined just from the class \( \mathcal{UC} \), but the underlying condition needs to be generalised: the elements are pairs \((V, F)\) (again \( F \in \mathcal{CLS} \) and \( V \subseteq \text{var}(F) \)), such that for all partial assignments \( \varphi \) with \( \text{var}(\varphi) \subseteq \text{var}(F) \setminus V \) holds \( \varphi * F \in \text{USAT} \Rightarrow r_1(\varphi * F) = \{\bot\} \). That is, the partial assignments considered are restricted to variables not using \( V \); in the same way we obtain \( \exists \mathcal{UC}_k, \exists \mathcal{PC}_k, \exists \mathcal{WC}_k \). Obviously we have for all these cases \( C[\exists] \subseteq C \subseteq \exists \mathcal{UP} \).

CNF-Representations of boolean functions in \( \mathcal{PC} \) under the relative condition, i.e., representations via \( \exists \mathcal{PC} \), are also known as “arc-consistent”; see [21] for more on this notion. There relativised hardness measurements, generalising the (p/w-)hardness as defined in Section 3, are considered to capture the relative condition.

That \( \mathcal{PC}_k, \mathcal{UC}_k \) and \( \mathcal{WC}_k \) for the absolute condition and without new variables do not collapse, shows that a rich structure was hidden under the carpet of the relative condition aka arc consistency. A basic difference between relative and absolute condition is that under the relative condition the new variables can be used to perform certain “computations”, since there are no conditions on the new variable other than not to distort the satisfying assignments. This is used to show the collapse to arc-consistency, as discussed in the following subsection, by encoding the stronger condition into the clause-sets in such a way that unit-clause propagation can perform the “computations”.

9.4.3 Separations under the relative condition

By definition it is clear that \( \mathcal{PC}_0 \) under the absolute and under the relative conditions still just represents only the constant 0/1 functions, and so \( \mathcal{PC}_0[\exists] \) as well as \( \exists \mathcal{PC}_0 \) have modulo simple transformations just the same power as \( \mathcal{PC}_0 \). And by Lemma [3] for \( \mathcal{UC}_0 = \mathcal{WC}_0 \) under the absolute and under the relative condition we just get the same power as \( \mathcal{UC}_0 = \mathcal{WC}_0 \) via equivalence, and so \( \exists \mathcal{UC}_0 \) and \( \mathcal{UC}_0[\exists] \) have modulo simple transformations the same power as \( \mathcal{UC}_0 \). In [29], we show that for the relative condition we have a collapse of \( \mathcal{WC}_k \) for \( k \geq 1 \) to \( \mathcal{PC}_1 \) by poltime transformation (for fixed \( k \)), that is, all classes \( \exists \mathcal{PC}_k, \exists \mathcal{UC}_k, \exists \mathcal{WC}_k \) for \( k \geq 1 \) can be translated in polynomial time to \( \exists \mathcal{PC}_1 \). Thus we can summarise: under the relative
condition all $PC_k, UC_k, WC_k$ collapse to one of $PC_0 \subset UC_0 \subset PC_1$ (w.r.t. polysize representation of boolean functions).

Motivated by [5], in [30] (Theorem 6.1) we show a close connection between representations in $∃UC_1$ and monotone circuits. This leads in [47] to the following separation between $CLS$ and $∃UC_1$ (and thus, by [29], between $CLS$ and $∃WC_k$ for any $k$):

**Lemma 9.7** For $m \in \mathbb{N}$ consider the satisfiable Pigeonhole clause-sets $PHP_m^m$ (“$m$ pigeons into $m$ holes”). Every sequence in $∃UC_1$ equivalent to $(PHP_m^m)_m \in \mathbb{N}$ is of superpolynomial size (in $m$ or $ℓ(HP_m^m)$).

Thus every representation under the relative condition of $(PHP_m^m)_m \in \mathbb{N}$ in $WC_k$ is super-polynomial, for every fixed $k$. Another example for this separation was shown in [30], namely that systems of linear equations over the two-element field, in other words, systems of XOR-constraints, have obvious and short CNF-representations (in $∃UP$, as usual), but have no polysize arc-consistent representations. In the other direction we will see a separation in Lemma 9.9.

The above collapse of the hierarchies under the relative condition is due to the free use of the auxiliary variables; on the contrary, under the absolute condition apparently we are very restricted with the free variables, and thus we conjecture, that there is a sequence of boolean functions which has polysize arc-consistent representations, but no polysize representations of bounded hardness, even for the w-hardness:

**Conjecture 9.8** There exists a polysize $(F_n^m)_n \in \mathbb{N}$ in $CLS$, with a polysize representation in $∃PC$, while for no $k \in \mathbb{N}_0$ there is a polysize CNF-representation $(F_n^m)_n \in \mathbb{N}$ of $(F_n^m)_n \in \mathbb{N}$ in $WC_k$ under the absolute condition (i.e., in $WC_k[∃]$).

Despite of the collapse to the first level, it might be interesting to consider the classes $∃PC_k, ∃UC_k, ∃WC_k$ for higher $k$, since the transformations to $∃PC$ are rather costly. And perhaps a more detailed picture is revealed when considering relations more fine-grained than just using “polysize”. We now consider the absolute condition, where we expect that all hierarchies are strict.

### 9.4.4 Separations under the absolute condition

First, to demonstrate the power of new variables is easy. We have already seen that level zero, i.e., $PC_0, UC_0, WC_0$, does not profit from new variables. But already with the smallest class of level 1 we can represent boolean functions which have no short CNF or DNF representations at all without new variables (as the default from now on, using the most stringent condition, the absolute condition; in KC-terminology we speak about $PC[∃]$):

**Lemma 9.9** The boolean function $f_n$ given by $v_1 \oplus \cdots \oplus v_n = 0$, $n \in \mathbb{N}$, has precisely one equivalent DNF and one equivalent CNF, each containing $2^{n-1}$ clauses of length $n$. While via splitting into sums containing only 2 variables each, we obtain the well-known $F := X_1(v_1 \oplus \cdots \oplus v_n = 0)$, as defined in [30], where it is shown that $F$ is a CNF-representation of $f_n$ with $F \in PC$ (in other words, we got a short representation of $f_n$ in $PC_1[∃]$).

We now strengthen the relation $C' \not\rightarrow C$ between classes of clause-sets to use auxiliary variables under the strong condition (only) on the left side (note that this yields a stronger non-simulation condition than when allowing auxiliary variables on both sides):
Definition 9.10 For $C, C' \subseteq \mathcal{CLS}$ the relation $C' \rightarrow_n C$ holds if there is a sequence $(F_n)_{n \in \mathbb{N}}$ in $C$ such that $n(F_n) = n$ and $F_n$ is computable in time $n^{O(1)}$, and such that there is no CNF-representation $(F'_n)_{n \in \mathbb{N}}$ of $(F_n)_{n \in \mathbb{N}}$ in $C'$ with $\ell(F'_n) = n^{O(1)}$.

By definition $C' \rightarrow_n C$ implies $C' \rightarrow C$. We are now able to state our main conjecture in its strong form (implying Conjecture 9.2).

Conjecture 9.11 (Main Conjecture, strong form) The relation $C \rightarrow_n C'$ holds for classes $C, C' \in \{PC_k, UC_k, WC_k : k \in \mathbb{N}_0\}$ if and only if $C' \not\subseteq C$.

9.5 Knowledge compilation

We have three levels of clausal KC-formalisms:

1. $PC_k, UC_k$ and $WC_k$ for $k \geq 0$, representing boolean functions by equivalent clause-sets in these classes; the internal relationships between these classes concerning KC (and poly-size-representations) are completely covered by Conjecture 1.2 (see Subsection 9.3).

2. $PC_k[\exists], UC_k[\exists]$ and $WC_k[\exists]$ for $k \geq 0$, representing boolean functions by clause-sets in these classes with existentially quantified auxiliary variables, i.e., employing the absolute condition; the internal relationships between these classes concerning KC are completely covered by Conjecture 9.11.

3. $\exists PC_k, \exists UC_k$ and $\exists WC_k$ for $k \geq 0$, representing boolean functions by clause-sets with existentially quantified auxiliary variables, where the defining conditions for these classes are used for the variables of the boolean function (only), i.e., employing the relative condition; the internal relationships between these classes concerning KC have been completely determined in Subsection 9.4.3.

The existential closure $C[\exists]$, i.e., representing boolean functions $f$ by $F \in C$ with $\text{var}(f) \subseteq \text{var}(F)$, and thus employing the absolute condition, has been introduced in [24] for KC, and further studied in [51]. It has the advantage that it is easily defined for all $C$. In contrast, the construction $\exists UC$, defined in [10], apparently can not be defined as $\exists C$ for arbitrary $C$: a boolean function $f$ is represented by a clause-set $F$ such that for the variables of $f$ the “underlying property” of $C$ holds (thus employing the relative condition). Extending Figure 5, in Figure 7 the queries supported by the stronger classes $UC[\exists]$ and $\exists UC$ are shown. Recall that $\exists UC$ can be transformed to $\exists PC$, which was already shown in [20]. We see that the possibly smaller class $UC[\exists]$ does not offer advantages here (recall Conjecture 9.8). More research is needed to determine whether the absolute condition might offer some other definitive advantages for KC, for example w.r.t. compilation. However for SAT solving the absolute condition is superior to the relative condition, as argued in [21, 29], since with the absolute condition also assigning to the auxiliary variables does not lead to hard unsatisfiable problems.

Note that for the classes $\exists UC, UC[\exists]$, queries such as SE are no longer poly-time decidable. The reason for this in a nutshell is, that for CNF clause-set $F, F'$ we have $F \models F'$ if $\forall C \in F' : F \models C$, which is not the case for existentially quantified CNFs. The point is that we want implication only on the original (free) variables, not on all variables (which we could check). In particular, it is shown in [10] that $\exists UC$, as well as other query classes built by taking the closure of $UC$ under disjunction, do not allow poly-time VA, IM, EQ or SE queries (as shown in Figure 7), while $UC$, and now more generally $UC_k$, does.

11) In [14] the class $UC$ is called URC-C, and $PC$ is called UPC-C.
In this report we concentrated on classes inside CNF (CLS). In [10] it is claimed (Proposition 4), that $\exists UC$ simulates DNNF, citing [36], but there is a mistake in [36] in that it claims that the Tseitin translation of all DNNF’s maintains arc-consistency via UCP (that is, yields $\exists PC$), where in fact this is only shown for smooth DNNF’s as confirmed by George Katsirelos via e-mail in January 2012; so the relation of $\exists UC$ (or $\exists PC$, which is the same here) to DNNF seems still open. Regarding the absolute condition, in [29] we show that DNF can be translated in linear time to $PC[3]$.

References

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

[2] Martin Babka, Tomáš Balyo, Ondřej Čepek, Štefan Gurský, Petr Kučera, and Václav Vlček. Complexity issues related to propagation completeness. Artificial Intelligence, 203:19–34, 2013.

[3] Tomáš Balyo, Štefan Gurský, Petr Kučera, and Václav Vlček. On hierarchies over the SLUR class. In Twelfth International Symposium on Artificial Intelligence and Mathematics (ISAIM 2012), January 2012. Available at http://www.cs.uic.edu/bin/view/Isaim2012/AcceptedPapers.

[4] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow — resolution made simple. In Proceedings of the 31th Annual ACM Symposium on Theory of Computing (STOC’99), pages 517–526, May 1999.

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

[6] Armin Biere. Picosat essentials. Journal on Satisfiability, Boolean Modeling and Computation, 4:75–97, 2008.

[7] Armin Biere. Lingeling, Plingeling, PicoSAT and PrecoSAT at SAT Race 2010. Technical report, Institute for Formal Models and Verification, Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria, August 2010. http://fmv.jku.at/papers/Biere-FMV-TR-10-1.pdf.

[8] Armin Biere. Lingeling and friends entering the SAT Challenge 2012. In Adrian Balint, Anton Belov, Daniel Diepold, Simon Gerber, Matti Järvisalo, and Carsten Sinz, editors, Proceedings of SAT Challenge 2012: Solver and Benchmark Descriptions, volume B-2012-2 of Department of Computer Science Series of Publications B, pages 33–34. University of Helsinki, 2012. https://helda.helsinki.fi/bitstream/handle/10138/34218/sc2012_proceedings.pdf.
[9] Armin Biere, Marijn J.H. Heule, Hans van Maaren, and Toby Walsh, editors. Handbook of Satisfiability, volume 185 of Frontiers in Artificial Intelligence and Applications. IOS Press, February 2009.

[10] Lucas Bordeaux, Mikoláš Janota, Joao Marques-Silva, and Pierre Marquis. On unit-refutation complete formulae with existentially quantified variables. In Knowledge Representation 2012 (KR 2012). Association for the Advancement of Artificial Intelligence (AAAI Press), June 2012.

[11] Lucas Bordeaux and Joao Marques-Silva. Knowledge compilation with empowerment. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, SOFSEM 2012: Theory and Practice of Computer Science, volume 7147 of Lecture Notes in Computer Science, pages 612–624. Springer, 2012.

[12] Uwe Bubeck and Hans Kleine Büning. The power of auxiliary variables for propositional and quantified boolean formulas. Studies in Logic, 3(3):1–23, 2010.

[13] Michael Buro and Hans Kleine Büning. On resolution with short clauses. Annals of Mathematics and Artificial Intelligence, 18(2-4):243–260, 1996.

[14] Ondřej Čepek, Petr Kučera, and Václav Vlček. Properties of SLUR formulae. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, SOFSEM 2012: Theory and Practice of Computer Science, volume 7147 of LNCS Lecture Notes in Computer Science, pages 177–189. Springer, 2012.

[15] Stephen A. Cook. An exponential example for analytic tableaux. Manuscript (see [29], page 432), 1973.

[16] Yves Crama and Peter L. Hammer, editors. Boolean Models and Methods in Mathematics, Computer Science, and Engineering, volume 134 of Encyclopedia of Mathematics and Its Applications. Cambridge University Press, 2010. ISBN 978-0-521-84752-0.

[17] Yves Crama and Peter L. Hammer. Boolean Functions: Theory, Algorithms, and Applications, volume 142 of Encyclopedia of Mathematics and Its Applications. Cambridge University Press, 2011. ISBN 978-0-521-84751-3.

[18] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. Journal of Artificial Intelligence Research, 17:229–264, 2002.

[19] Ronald de Haan, Iyad Kanj, and Stefan Szeider. Local backbones. Technical Report arXiv:1304.5479 [cs.CC], arXiv, May 2013.

[20] Alvaro del Val. Tractable databases: How to make propositional unit resolution complete through compilation. In Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR’94), pages 551–561, 1994.

[21] Hélène Fargier and Pierre Marquis. Extending the knowledge compilation map: Closure principles. In Malik Ghallab, editor, ECAI 2008: 18th European Conference on Artificial Intelligence, pages 50–54, 2008.

[22] John Franco and Allen Van Gelder. A perspective on certain polynomial-time solvable classes of satisfiability. Discrete Applied Mathematics, 125:177–214, 2003.
[23] Jan Friso Groote and Joost P. Warners. The propositional formula checker HeerHugo. *Journal of Automated Reasoning*, 24:101–125, 2000.

[24] Matthew Gwynne and Oliver Kullmann. Towards a better understanding of hardness. In *The Seventeenth International Conference on Principles and Practice of Constraint Programming (CP 2011): Doctoral Program Proceedings*, pages 37–42, September 2011. Proceedings available at [http://www.dmi.unipg.it/cp2011/downloads/dp2011/DP_at_CP2011.pdf](http://www.dmi.unipg.it/cp2011/downloads/dp2011/DP_at_CP2011.pdf).

[25] Matthew Gwynne and Oliver Kullmann. Towards a better understanding of SAT translations. In Ulrich Berger and Denis Therien, editors, *Logic and Computational Complexity (LCC’11), as part of LICS 2011*, June 2011. 10 pages, available at [http://www.cs.swansea.ac.uk/lcc2011/](http://www.cs.swansea.ac.uk/lcc2011/).

[26] Matthew Gwynne and Oliver Kullmann. Generalising and unifying SLUR and unit-refutation completeness. In Peter van Emde Boas, Frans C. A. Groen, Giuseppe F. Italiano, Jerzy Nawrocki, and Harald Sack, editors, *SOFSEM 2013: Theory and Practice of Computer Science*, volume 7741 of *Lecture Notes in Computer Science (LNCS)*, pages 220–232. Springer, 2013.

[27] Matthew Gwynne and Oliver Kullmann. Generalising unit-refutation completeness and SLUR via nested input resolution. *Journal of Automated Reasoning*, 2013. To appear; published online 09 March 2013 [http://link.springer.com/article/10.1007/s10817-013-9275-8](http://link.springer.com/article/10.1007/s10817-013-9275-8).

[28] Matthew Gwynne and Oliver Kullmann. Generalising unit-refutation completeness and SLUR via nested input resolution. Technical Report arXiv:1204.6529v5 [cs.LO], arXiv, January 2013.

[29] Matthew Gwynne and Oliver Kullmann. Guiding SAT translations by hardness measures. Technical Report To appear, arXiv, December 2013.

[30] Matthew Gwynne and Oliver Kullmann. On SAT representations of XOR constraints. Technical Report arXiv:1309.3060v2 [cs.CC], arXiv, October 2013.

[31] Matthew Gwynne and Oliver Kullmann. Towards a theory of good SAT representations. Technical Report arXiv:1302.4421v4 [cs.AI], arXiv, May 2013.

[32] John Harrison. Stålmarck’s algorithm as a HOL derived rule. In *Theorem proving in higher order logics: 9th International Conference, TPHOLs’96*, Lecture Notes in Computer Science 1125, pages 221–234, 1996.

[33] Marijn Heule, Mark Dufour, Joris van Zwieten, and Hans van Maaren. March_eq: Implementing additional reasoning into an efficient look-ahead SAT solver. In Holger H. Hoos and David G. Mitchell, editors, *Theory and Applications of Satisfiability Testing 2004*, volume 3542 of *Lecture Notes in Computer Science*, pages 345–359, Berlin, 2005. Springer. ISBN 3-540-27829-X.

[34] Marijn J. H. Heule and Hans van Maaren. Look-ahead based SAT solvers. In Biere et al. [4], chapter 5, pages 155–184.

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

[36] Jean Christoph Jung, Pedro Barahona, George Katsirelos, and Toby Walsh. Two encodings of DNNF theories, July 2008. Presented at ECAI’08 Workshop on Inference methods based on Graphical Structures of Knowledge. Proceedings at [http://www.irit.fr/LC/](http://www.irit.fr/LC/).
[37] Alex Kean and George Tsiknis. An incremental method for generating prime implicants/implicates. *Journal of Symbolic Computation*, 9(2):185–206, February 1990.

[38] Hans Kleine Büning. On generalized Horn formulas and $k$-resolution. *Theoretical Computer Science*, 116:405–413, 1993.

[39] Hans Kleine Büning and Oliver Kullmann. Minimal unsatisfiability and autarkies. In Biere et al. [4], chapter 11, pages 339–401.

[40] Oliver Kullmann. Investigating a general hierarchy of polynomially decidable classes of CNF’s based on short tree-like resolution proofs. Technical Report TR99-041, Electronic Colloquium on Computational Complexity (ECCC), October 1999.

[41] Oliver Kullmann. An application of matroid theory to the SAT problem. In *Fifteenth Annual IEEE Conference on Computational Complexity (2000)*, pages 116–124. IEEE Computer Society, July 2000.

[42] Oliver Kullmann. Investigating the behaviour of a SAT solver on random formulas. Technical Report CSR 23-2002, Swansea University, Computer Science Report Series (available from http://www-compsci.swan.ac.uk/reports/2002.html), October 2002. 119 pages.

[43] Oliver Kullmann. Lean clause-sets: Generalizations of minimally unsatisfiable clause-sets. *Discrete Applied Mathematics*, 130:209–249, 2003.

[44] Oliver Kullmann. The combinatorics of conflicts between clauses. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing 2003*, volume 2919 of *Lecture Notes in Computer Science*, pages 426–440, Berlin, 2004. Springer. ISBN 3-540-20851-8.

[45] Oliver Kullmann. Upper and lower bounds on the complexity of generalised resolution and generalised constraint satisfaction problems. *Annals of Mathematics and Artificial Intelligence*, 40(3-4):303–352, March 2004.

[46] Oliver Kullmann. Constraint satisfaction problems in clausal form II: Minimal unsatisfiability and conflict structure. *Fundamenta Informaticae*, 109(1):83–119, 2011.

[47] Oliver Kullmann. Hardness measures and resolution lower bounds. Technical Report arXiv:1310.7627v1 [cs.CC], arXiv, October 2013.

[48] Oliver Kullmann and Xishun Zhao. On Davis-Putnam reductions for minimally unsatisfiable clause-sets. *Theoretical Computer Science*, 492:70–87, June 2013.

[49] Chu Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proceedings of 15th International Joint Conference on Artificial Intelligence (IJCAI’97)*, pages 366–371. Morgan Kaufmann Publishers, 1997.

[50] Joao Marques-Silva. Computing minimally unsatisfiable subformulas: State of the art and future directions. *Journal of Multiple-Valued Logic and Soft Computing*, 19(1-3):163–183, 2012.

[51] Pierre Marquis. Existential closures for knowledge compilation. In *IJCAI’11: Proceeding of the Twenty-Second International joint conference on Artificial Intelligence*, pages 996–1001, 2011.
[52] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, 2011.

[53] John S. Schlipf, Fred S. Annexstein, John V. Franco, and R.P. Swaminathan. On finding solutions for extended Horn formulas. *Information Processing Letters*, 54:133–137, 1995.

[54] Mary Sheeran and Gunnar Stålmarck. A tutorial on Stålmarck’s proof procedure for propositional logic. In *FMCAD’98*, volume 1522 of *Lecture Notes in Computer Science*, pages 82–99, 1998.

[55] Carsten Sinz. Knowledge compilation for product configuration. In Michel Aldanondo, editor, *Configuration: Papers from the Workshop at ECAI 2002*, pages 23–26, 2002. [http://www.carstensinz.de/papers/ECAI-Config-WS-2002.pdf](http://www.carstensinz.de/papers/ECAI-Config-WS-2002.pdf).

[56] Robert H. Sloan, Balázs Sörényi, and György Turán. On k-term DNF with the largest number of prime implicants. *SIAM Journal on Discrete Mathematics*, 21(4):987–998, 2007.

[57] Mate Soos. Cryptominisat 2.5.0. [http://baldur.iti.uka.de/sat-race-2010/descriptions/solver_13.pdf](http://baldur.iti.uka.de/sat-race-2010/descriptions/solver_13.pdf) 2010.

[58] Emanuel Sperner. Ein Satz über Untermengen einer endlichen Menge. *Mathematische Zeitschrift*, 27(1):544–548, 1928.

[59] Gunnar Stålmarck and M. Säflund. Modeling and verifying systems and software in propositional logic. In B.K. Daniels, editor, *Safety of Computer Control Systems (SAFECOMP’90)*, pages 31–36, 1990.

[60] Alasdair Urquhart. The complexity of propositional proofs. *The Bulletin of Symbolic Logic*, 1(4):425–467, 1995.

[61] V. Vlček. Classes of boolean formulae with effectively solvable SAT. In Jana Safrankova and Jiri Pavlu, editors, *Proceedings of the 19th Annual Conference of Doctoral Students - WDS 2010*, volume 1, pages 42–47. Matfyzpress, 2010.