Electronic Colloquium on Computational Complexity, Report No. 41 (1999)

Investigating a general hierarchy of polynomially decidable classes of CNF's based on short tree-like resolution proofs

Oliver Kullmann[*]

Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4

e-mail: kullmann@cs.toronto.edu

http://mi.informatik.uni-frankfurt.de/people/kullmann/kullmann.html

August 21, 1999

Abstract

We investigate a *hierarchy* $G_k(\mathcal{U}, \mathcal{S})$ of classes of conjunctive normal forms, recognizable and SAT-decidable in *polynomial time*, with special emphasize on the corresponding *hardness parameter* $h_{\mathcal{U},\mathcal{S}}(F)$ for clause-sets F (the first level of inclusion). At level 0 an (incomplete, poly-time) oracle $\mathcal{U}$ for unsatisfiability detection and an oracle $\mathcal{S}$ for satisfiability detection is used. The hierarchy from [Pretolani 96] is improved in this way with respect to strengthened satisfiability handling, simplified recognition and consistent relativization. Also a hierarchy of *canonical* poly-time *reductions* with Unit-clause propagation at the first level is obtained.

General methods for *upper* and *lower bounds* on $h_{\mathcal{U},\mathcal{S}}(F)$ are developed and applied to a number of well-known examples. $h_{\mathcal{U},\mathcal{S}}(F)$ admits several *different characterizations*, including the *space complexity* of tree-like resolution and the use of *pebble games* as in [Esteban, Torán 99].

Using for $\mathcal{S}$ the class of *linearly satisfiable clause-sets* (based on linear programming) q-Horn clause-sets [Boros, Cramer, Hammer 90] are contained at level 2, and for $k \geq 1$ the "k-times nested Horn clause-sets" from [Gallo, Scutellà 88] are contained at level k.

The unsatisfiable clause-sets in $G_k(\mathcal{U}, \mathcal{S})$ are exactly those refutable by *relativized k-times nested input resolution*, and the SAT decision algorithm

[*]Supported by the Natural Sciences and Engineering Research Council of Canada and by the Communications and Information Technology Ontario.

ISSN 1433-8092

searching through the levels from below *quasi-automatizes* relativized *tree-like resolution* (using oracle $\mathcal{U}$), while by means of $h_{\mathcal{U}}(F)$ *nearly precise general bounds* on the (relativized) complexity of tree-like resolution (with oracle $\mathcal{U}$) are obtained.

In order to cope also with full resolution, a (more comprehensive) *hierarchy* $W_k(\mathcal{U})$ of unsatisfiable clause-sets is introduced, based on a new form of *width-restricted resolution*, and relativized general *upper* and *lower bounds* for *full resolution* are derived, generalizing [Ben-Sasson, Wigderson 99] and also releasing the lower bound from its dependence on the maximal input clause length. Motivated by [Bonet, Galesi 99] we give a simplified example where the lower bound is tight.

Keywords: satisfiability, polynomial time, resolution with oracles, generalized input resolution, generalized Unit-clause propagation, width restricted resolution, lower bounds on resolution, generalized Horn clause-sets, q-Horn, linear autarkies, linear programming, pigeonhole formulas, pebble games, pebbling formulas

Contents

1 Introduction

In this article we study a (cumulative) *hierarchy* $(\boldsymbol{G_k(\mathcal{U},\mathcal{S})})_{k\in\mathbb{N}_0}$ of classes of conjunctive normal forms ("clause-sets") and the corresponding *hardness parameter* $\boldsymbol{h_{\mathcal{U},\mathcal{S}}(F)}$ for clause-sets F, the minimal k with $F \in G_k(\mathcal{U},\mathcal{S})$. Thereby $\mathcal{U}$ and $\mathcal{S}$ are sets of unsatisfiable resp. satisfiable clause-sets building the basis of the hierarchy: $G_0(\mathcal{U},\mathcal{S}) = \mathcal{U} \cup \mathcal{S}$.

Membership "$F \in G_k(\mathcal{U},\mathcal{S})$?" as well as SAT decision "$F \in \mathcal{SAT}$?" for $F \in G_k(\mathcal{U},\mathcal{S})$ is *polynomially decidable*, considering $\mathcal{U}$ and $\mathcal{S}$ as oracles. An accompanying hierarchy $(\boldsymbol{r_k^{\mathcal{U}}})_{k\in\mathbb{N}_0}$ of *uniquely determined reduction operators* yields *canonical normal forms* for clause-sets.

Based on general methods, we investigate *upper and lower bounds* on the hardness $h_{\mathcal{U},\mathcal{S}}(F)$ for various formula classes (including generalized Horn clause-sets, p-CNF, the Pebbling and the Pigeonhole formulas). *Alternative characterizations* of $h_{\mathcal{U},\mathcal{S}}(F)$ are given, using generalized search trees for *DPL-algorithms* and *pebble games*.

Regarding unsatisfiability, we introduce k-times *nested input resolution* $\overset{\mathcal{U},k}{\vdash}$ with oracle $\mathcal{U}$, providing refutations exactly for the unsatisfiable clause-sets in $G_k(\mathcal{U},\mathcal{S})$ (here $\mathcal{S}$ can be ignored).

The natural SAT decision algorithm searching through the hierarchy from below *quasi-automatizes* tree-like resolution *with oracle $\mathcal{U}$*, while we get *nearly precise general bounds* on these calculi by means of the hardness $h_{\mathcal{U}}(F)$.

In order to obtain also *general bounds* on *full* (dag-like) *resolution* we introduce a *new form of width-restricted resolution* (using oracle $\mathcal{U}$), which is polynomially decidable for fixed width and simulates nested input resolution (generalizing Unit-resolution).

1.1 Combining several lines of research

This article is not one of the short ones, due to the fact that quite a number of several lines of research are affected, and I want to give a coherent picture of ideas scattered around in a lot of several articles (and also all details are given).

Ideas are combined belonging on the one side to the field of *lower bounds for proof systems*, and on the other side to the research on *efficient SAT algorithms* (especially polynomially decidable sub-cases). At the present stage there seems to be a dividing line between these communities. I hope that this article will help a bit to close the gap.

The story started for me with

YD83 and GS88, K93, EKM95 ([91, 37, 14, 29]) develop and explore a hierarchy $(H_k)_{k \in \mathbb{N}_0}$ of *generalized Horn formulas* decidable in polynomial time[1];

realizing that the construction can be enhanced and generalized to a hierarchy

$$G_k(\mathcal{U}, \mathcal{S}), \ k \in \mathbb{N}_0$$

by *abstracting* from the *special syntactic properties* of Horn formulas and extracting the recursive structure responsible for the poly-time decidability of the single levels.[2] $\mathcal{U}$ and $\mathcal{S}$ here are *oracles*, responsible at the bottom level for unsatisfiability resp. satisfiability detection (incomplete (of course)).

This hierarchy naturally leads to a concept of *k-times nested input resolution* (denoted by $\overset{\mathcal{U},k}{\vdash}$ here) refuting exactly the unsatisfiable clause-sets in $G_k(\mathcal{U}, \mathcal{S})$.

From

Kl93 ([14]) introduces the concept of *k-resolution* to give resolution refutations of the levels H_k; unlike *bounded resolution* introduced in [36], Unit-resolution is generalized in this way, but for $k \geq 3$ it is not known whether derivability of the empty clause is poly-time decidable (see [16]);

I learned this concept of *width-restricted resolution*, which in fact *simulates nested input resolution*, generalizing the equivalence between Unit-resolution and input resolution (for refutation purposes) in one way.

My interest in these hierarchies was revitalized by

CEI96 ([21]) search for tree-like resolution refutations by means of Gröbner bases in time *quasi-polynomial* in the length of the *shortest tree-like resolution refutation*, and also a weaker simulation of full resolution is given;

due to the simple observations

1. their simulation by degree-restricted polynomial calculus can already be carried out by (some sort of width-restricted) *resolution itself*;
2. and in case of tree-like resolution they implicitly use the *simulation process* of nested input resolution by k-resolution.

[1] for information on general poly-time SAT decision see [42] (section 10) and [33]

[2] A similar generalization Γ_k one finds in [75]; see Subsection 1.2 for a discussion.

Indeed, *nested input resolution suffices* to do the simulation job for tree-like resolution. It follows that the *SAT algorithm* naturally linked to the hierarchy $(G_k(\mathcal{U}, \mathcal{S}))$ (searching through the levels from below) *quasi-automatizes* tree-like resolution, that is, its running time is quasi-polynomial in the length of the shortest tree-like resolution refutation.

Furthermore the minimal level $h(F)$ sufficient for nested input resolution to refute F (in other words, the minimal level k with $F \in G_k(\mathcal{U}_0, \mathcal{S}_0)$, where $\mathcal{U}_0$ and $\mathcal{S}_0$ are the trivial oracles) yields *nearly precise general bounds*

$$2^{h(F)} \leq \mathrm{Comp}_{\mathrm{tR}}(F) \leq 2^{\log_2(n(F)+1) \cdot h(F)}$$

for the complexity of tree-like resolution ($n(F)$ is the number of variables).[3)]

To exemplify the strength of this lower bound one may regard the *pigeonhole formulas*, where it is fairly easy to compute $h(\mathrm{PHP}_k^m) = k$ and thus to derive the lower bound from [17].

After reading [BW99] ([5]), exploiting and deepening [21] for resolution lower bounds[4)], I decided to make a paper out of the whole material, to round up the picture and point out the connections. Additionally to the above topics, the following has been included:

- A *variant of k-resolution* is introduced (in fact a (slight) strengthening), solving the problem of *polynomial decidability* and enabling *general bounds on resolution complexity* (with oracles) as in [5], but not depending on the maximal input clause-length:

  $$\frac{1}{8} \frac{\mathrm{width}(F)^2}{n(F)} < \ln \mathrm{Comp}_{\mathrm{R}}(F) < (\ln n(F)) \cdot (\mathrm{width}(F) + 1) + 2,$$

  where $\mathrm{width}(F)$ is the minimal k such that the new form of k-resolution suffices to derive the empty clause. Motivated by [Bonet, Galesi 99] a simplified example is given where the lower bound can not by improved, using Krishnamurthy's original formulas ([52]).

- From [30] I learned the notion of *space complexity of resolution* and its correspondence to *pebble games* — now $h(F)$ is just the *space complexity of tree-like resolution*; this whole topic is discussed in some detail, using as intermediate step *generalized DPL-trees* (extending the correspondence between tree-like resolution and decision trees from [63]).

[3)] From Jan Friso Groote I heart that similar bounds have already been mentioned in the work of Stålmarck. See [84, 44, 81] and also [41] — in fact his algorithm is essentially related to our approach, searching for a refutation through levels of increasing "hardness" by the nested "dilemma rule". Unfortunately no details are given in the sources.

[4)] [4] has been the first paper pointing out that direction

- Another nice application of pebble games are the *pebbling formulas* (introduced in [5], generalizing [8], a forerunner is [51]): here calculating $h(F)$ gives a very lucid lower bound.
- For an efficient handling of *satisfiable* clause-sets the oracle of *linearly satisfiable clause-sets* is used, introduced in [58] and essentially relying on linear programming; as I learned from [89], in this way also *q-Horn formulas* (a common generalization of Horn formulas and 2-CNF's, see [11, 12, 13]) are captured.
- Nearly all results (including the general lower bounds) are *relativized*, using the oracles $\mathcal{U}$ and $\mathcal{S}$.

Before itemizing our main results in Subsection 1.3, in the next subsection we discuss the algorithmic ideas underlying our approach.

1.2 A general polynomial time decision scheme

Instead on (accidental) syntactic properties of formulas, our approach for poly-time SAT decision is based on an *algorithmic structure* enforcing polynomial behaviour. The basic idea is essentially already expressed in [75].

Consider a language $L \subseteq \Sigma^*$ such that any question "$x \in L$?" can be split into

$$x \in L \Longleftrightarrow \bigl(x_1 \in L \quad \text{or} \quad x_2 \in L\bigr)$$

where for some "length"

$$\ell : \Sigma^* \to \mathbb{N}_0$$

we have $\ell(x_1), \ell(x_2) < \ell(x)$. Let the number of possible splittings to be considered be $O(\ell(x))$.

Using a *level* $k \in \mathbb{N}_0$ of "difficulty" or "hardness," by the following algorithmic scheme we get polynomial time decision ((necessarily) incomplete) of "$x \in L$?" for any fixed k.

1. At level 0 any oracle (for example some trivial decision criterion) is used which may return "YES" or "NO" or "*not belonging to level* 0".
2. At level $k \geq 1$ we search for a possibility to split into x_1 and x_2 such that x_1 can be decided at level $k-1$ and x_2 at level k, in which case we return "YES" if $x_1 \in L$ or $x_2 \in L$, and "NO" otherwise. If there is no such possibility then the outcome is "not belonging to level k".

The output is either the decision "$x \in L$" resp. "$x \notin L$" or "not successful." It is easy to see that the number of leaves in the search tree is $\ell(F)^{O(k)}$, *if* the search for a pair (x_1, x_2) in step 2 proceeds as follows:

First ignore the condition for x_2 but just run through all possible x_1. If no x_1 is decidable at level $k-1$, then x is not belonging to level k. If $x_1 \in L$ then also $x \in L$. And in case $x_1 \notin L$ take the corresponding x_2 and reduce x to x_2, *without regarding* other possibilities for x_2. (Repeat the whole procedure for this new x.)

The class of formulas decidable at level k is *well-defined* if reduction of x to x_2 in case $x_1 \notin L$ is confluent, which is the case for our hierarchy, due to the concept of *stability under enforced assignments* (see below).

"Satisfiability-driven" vs. "unsatisfiability-driven"

Typically SAT decision algorithms are either *"satisfiability-driven"* such as local search algorithms ([80]) or the new type of algorithm in [73], thus output a *proof of satisfiability* (typically the satisfying assignment) and do not look out for unsatisfiability, or are *"unsatisfiability-driven"* as all refutation search procedures (like [84] or [21]), and thus output a *proof of unsatisfiability* (in some proof system) and do not care much about satisfiability.

In our framework we try to balance these two approaches by using *two* "oracles"

$$\mathcal{U} \subseteq \mathcal{USAT}, \ \mathcal{S} \subseteq \mathcal{SAT}$$

at level 0, one for (certain) unsatisfiable instances, one for (certain) satisfiable instances (if $F \notin \mathcal{U} \cup \mathcal{S}$ then F does not belong to level 0). The above algorithmic scheme may be seen as an "amplification mechanism."

Throughout the paper the use of $\mathcal{U}$ and $\mathcal{S}$ indicates that any oracles for unsatisfiability and satisfiability stable under *enforced variable assignments* can be used. More precisely, arbitrary sets $\mathcal{U} \supseteq \mathcal{U}_0$ of unsatisfiable clause-set, where $\mathcal{U}_0$ is the set of trivially unsatisfiable clause-sets already containing the empty clause, and arbitrary sets $\mathcal{S} \supseteq \mathcal{S}_0$ of satisfiable clause-sets, where $\mathcal{S}_0$ just contains the empty clause-set, can be used, provided that for any $F \in \mathcal{U}$ resp. $F \in \mathcal{S}$, any variable v and any $\varepsilon \in \{0, 1\}$ from

$$\langle v \to \varepsilon \rangle * F \notin \mathcal{SAT}$$

we can conclude $\langle v \to \overline{\varepsilon} \rangle * F \in \mathcal{U}$ resp. $\langle v \to \overline{\varepsilon} \rangle * F \in \mathcal{S}$, where "$*$" denotes application of a partial assignment, and "$\overline{}$" complementation.

For $\mathcal{U}$ this is the same as stability under application of (all) partial assignments. However, for $\mathcal{S}$ it is a much weaker notion which only allows us to consider $\mathcal{S}$ *on its own* (otherwise we also had to consider the complexity of unsatisfiable formulas emerging from members of $\mathcal{S}$ by "wrong" applications of partial assignments — and such formulas may be *much harder* than the original (satisfiable) formulas).

Treat satisfiability and unsatisfiability separately!

The family of hierarchies $G_k(\mathcal{U}, \mathcal{S})$ improves the hierarchies Π_k from [75] in the following aspects:

1. our hierarchies include the cases of satisfiable clause-sets, where setting one variable already yields a satisfiable clause-set at a smaller level, ignoring what happens when setting the variable to the complementary value;
2. the concept of "being closed under fixing" from [75] is generalized to "stability under enforced variable assignments" (see above);
3. the recognition process conceptually is very simple now;
4. dividing the ground level $G_0(\mathcal{U}, \mathcal{S}) = \mathcal{U} \cup \mathcal{S}$ into $\mathcal{U}$ and $\mathcal{S}$ emphasizes the *different nature of satisfiability and unsatisfiability detection* (a point which often has been missed by prior research; see [27] for some heuristical discussions), and, due to 2, opens up the perspective of choosing $\mathcal{U}$ and $\mathcal{S}$ completely separately.

1.3 The main results

Due to the length of the present article, I hope the following listing of the main theorems, lemmas and definitions will help the reader.

Section 3: Introducing the hierarchies

1. In Definition 3.3 the classes $\boldsymbol{G_k(\mathcal{U}, \mathcal{S})} \subseteq \mathcal{CLS}$ are defined.
2. Lemma 3.5 proves their *universal property.*
3. And Lemma 3.7 gives *polynomial time* upper bounds on their (SAT) decision.
4. The *hardness* functions $\boldsymbol{h_{\mathcal{U},\mathcal{S}}} : \mathcal{CLS} \to \mathbb{N}_0$ are defined in Definition 3.9.
5. Lemma 3.10 gives the running time of the *general SAT decision* algorithm searching through the hierarchy from below (and thus computing also the hardness).
6. In Lemma 3.11 important *stability properties* of the hardness are formulated.

7. Definition 3.12 introduces the *reductions* $\xrightarrow{k,\mathcal{U}}$ for clause-sets (generalizing 1-clause-elimination), extracting the aspect of *reduction by enforced assignments* from the definition of the classes $G_k(\mathcal{U},\mathcal{S})$.

8. In Lemma 3.13 it is shown that the relations $\xrightarrow{k,\mathcal{U}}$ are terminating and *confluent.*

9. Generalizing 1-clause-propagation, the (*uniquely determined*) reduction operator $\boldsymbol{r}_k^{\mathcal{U}} : \mathcal{CLS} \rightarrow \mathcal{CLS}$ is defined in Definition 3.14.

10. Using these canonical reductions, the *universal property* from Lemma 3.5 is *generalized* in Lemma 3.15.

11. In Subsection 3.4.1 it is discussed how Lemma 3.15 gives a general *method for upper bounds* on the hardness $h_{\mathcal{U},\mathcal{S}}(F)$.

12. While the general *method for lower bounds* is given in Lemma 3.17 and Subsection 3.4.2.

13. Lemma 3.18 gives some *easy upper and lower bounds* on the hardness.

14. And in Lemmata 3.19 and 3.20 examples are given where these bounds are *attained.*

15. Lemma 3.21 proves the *strictness* of the hierarchies w.r.t. unsatisfiability.

16. Finally in Definition 3.22 one finds *strengthened reductions* $\xrightarrow[+]{k,\mathcal{U}}$ capturing Stålmarck's central concept of reduction.

Section 4: DPL-trees and pebble games

1. Definition 4.1 introduces the concept of $(\mathcal{U},\mathcal{S})$-*DPL-trees.*

2. In Definition 4.2 the *leveled height* $\boldsymbol{h_{\mathcal{U},\mathcal{S}}(T)}$ of DPL-trees is given.

3. Theorem 4.3 shows the *correspondence* between the *hardness* of clause-sets and the *leveled height* of DPL-trees.

4. In Subsection 4.2 *various characterizations* of the leveled height $\boldsymbol{h(T)}$ in case of "closed trees" are given (where no "open leaves" are allowed, and thus only the shape of T is of importance).

5. Theorem 4.6 gives a characterization of $h(T)$ in terms of *embeddable full binary trees*, and shows that the leveled height is the same as the *pebbling complexity* of T.

Section 5: Upper bounds on the hardness

1. Lemma 5.1 obtains an *upper bound* on the hardness $\boldsymbol{h(F)}$ (using only the trivial oracles) of *uniquely satisfiable clause-sets F.*
2. While Lemma 5.3 gives an *upper bound* on $h(F)$ for satisfiable F in terms of the *number of satisfying assignments.*
3. And Lemma 5.5 gives an *upper bound* on the hardness for *unsatisfiable* clause-sets in terms of the *maximal clause-length.*
4. Lemma 5.7 determines the hardness for *large random p-CNF.*
5. Upper bounds on unsatisfiable $h(F)$ for the hierarchy of *generalized Horn formulas* are given in Lemma 5.10.
6. Lemma 5.12 shows that unsatisfiable *q-Horn clause-sets* have hardness at most 2.
7. In Subsection 5.4 *linearly satisfiable clause-sets* are considered, used in Lemma 5.14 to handle also *satisfiable* generalized Horn and q-Horn clause-sets.

Section 6: Lower bounds on the hardness

1. Lemma 6.2 determines the hardness of the *(weak) pigeonhole formulas.*
2. And Lemma 6.7 gives a natural lower bound on the hardness for the *pebbling formulas.*

Section 7: Tree-like resolution with oracles

1. In Definition 7.1 *k-times nested input resolution* $\overset{\boldsymbol{\mathcal{U},k}}{\vdash}$ *using oracle $\mathcal{U}$* is introduced.
2. Theorem 7.5 shows the *correspondence* between the necessary level k of *nested input resolution* $\overset{\mathcal{U},k}{\vdash}$ and the *hardness* $h_{\mathcal{U}}(F)$ for unsatisfiable clause-sets.
3. In Subsection 7.2 also the connection to the *space complexity of resolution* is discussed, and the *various characterizations* of the hardness $h_{\mathcal{U},\mathcal{S}}(F)$ are assembled.
4. Theorem 7.8 gives *quasi-precise general upper and lower bounds* on tree-like resolution (with oracles).
5. And Theorem 7.10 proves the *quasi-automatizability* of tree-like resolution with oracles.

Section 8: Width-restricted resolution

1. Subsection 8.1 discusses the *two known variants* of width restrictions for resolution.

2. Definition 8.3 introduces a *new form of width restriction* with oracles (our "standard form"). $\mathbf{width}_{\mathcal{U}}(\boldsymbol{F})$ is the minimal width needed for deriving the empty clause.

3. Lemma 8.5 gives the *relations* between the three forms of width restrictions.

4. Lemma 8.6 shows that deciding derivability by the new form of width restriction is in *polynomial time* (for fixed width).

5. In Lemma 8.7 the inequality $\mathrm{width}_{\mathcal{U}}(F) \leq h_{\mathcal{U}}(F)$ is proven (and thus *nested input resolution* can be *simulated* by this form of *width restricted resolution*).

6. Lemma 8.8 gives an *example* for constant width but where the hardness nearly is the number of variables.

7. In Theorem 8.10 *general upper and lower bound for (full) resolution with oracles* in terms of $\mathrm{width}_{\mathcal{U}}(F)$ are proven.

8. And in Subsection 8.4, Lemma 8.12 an example is given, where the lower bound of Theorem 8.10 is (nearly) tight.

Section 9: Final remarks on the complexity of resolution

1. In Subsection 9.1 the *applicability of the general bounds* on tree-like and full resolution is discussed.

2. And in Subsection 9.2 additional information on the *weak pigeonhole formulas* is provided. Two alternative proofs for a lower bound on tree-like resolution are given, and strengthened in Corollaries 9.6 and 9.7 by including also both types of *dual clauses* (expressing "onto" and "functionality").

2 Notation

The notions about *clause-sets* and *partial assignments* in 2.1 are used throughout the paper, while the (basic) notions on *resolution* in 2.2 are used from Section 7 on. The *Pigeonhole formulas* defined in 2.3 are considered in Subsections 6.1 and 9.2, and the notions on (directed) *graphs* and *Pebble games* in 2.4 are used in Subsections 4.3, 6.2, 7.2 and Lemma 8.8.

2.1 Clause-sets and partial assignments

Let $\boldsymbol{\mathcal{VA}}$ be the (infinite) set of *variables*. A *literal* x is either a variable $x = v$ or a *complemented* variable $x = \overline{v}$ for $v \in \mathcal{VA}$. The *complement* $\overline{x}$ of a literal x fulfills $\overline{\overline{x}} = x$. The underlying variable of a literal x is $\mathbf{var}(\boldsymbol{x}) \in \mathcal{VA}$.

A *clause* C is a (finite) set of literals without clashes, that is $C \cap \overline{C} = \emptyset$, where $\overline{C}$ is the set of literals obtained from C by element-wise complementation. The set of all clauses is $\boldsymbol{\mathcal{CL}}$. A *clause-set* is a (finite) set of clauses and the set of all clause-sets is denoted by $\boldsymbol{\mathcal{CLS}}$, while $\boldsymbol{p}\text{–}\boldsymbol{\mathcal{CLS}}$ is the set of clause-sets containing only clauses of length at most p. *Horn clause-sets* F are characterized by the condition that every clause $C \in F$ contains at most one positive literal (that is, $|C \cap \mathcal{VA}| \leq 1$). $\boldsymbol{\mathcal{HO}}$ is the set of all Horn clause-sets.

A special clause is the *empty clause* $\bot := \emptyset \in \mathcal{CL}$, and a special clause-set is the *empty clause-set* $\top := \emptyset \in \mathcal{CLS}$. Two clauses C_1, C_2 *clash* on x if $x \in C_1$ and $\overline{x} \in C_2$.

For $\mathcal{C} \subseteq \mathcal{CLS}$ let $\boldsymbol{\mathcal{C}^{+}} := \{ F \in \mathcal{C} : \forall C \in F\, [|C| \geq 2] \}$ be the sub-class of clause-sets not containing the empty clause or a 1-clause.

For a clause C its set of variable $\mathbf{var}(\boldsymbol{C})$ is $\{ \operatorname{var}(x) : x \in C \}$, and for a clause-set F we set $\mathbf{var}(\boldsymbol{F}) := \bigcup_{C \in F} \operatorname{var}(C)$. We denote by $\boldsymbol{n}(\boldsymbol{F}) := |\operatorname{var}(F)|$ the number of variables in F, by $\boldsymbol{c}(\boldsymbol{F}) := |F|$ the number of clauses in F and by $\boldsymbol{\ell}(\boldsymbol{F}) := \sum_{C \in F} |C|$ the number of literal occurrences in F. $\boldsymbol{p}(\boldsymbol{F})$ is the minimal $p \in \mathbb{N}_0$ such that $F \in p\text{–}\mathcal{CLS}$.

A *partial assignment* φ is a map from $V \subseteq \mathcal{VA}$ into $\{0,1\}$. We use $\mathbf{var}(\boldsymbol{\varphi}) := V$ and $\boldsymbol{n}(\boldsymbol{\varphi}) := |\operatorname{var}(\varphi)|$. The set of all partial assignments is called $\boldsymbol{\mathcal{PASS}}$. We write $\boldsymbol{\langle x_1 \to \varepsilon_1, \ldots, x_m \to \varepsilon_m \rangle}$ for literals x_i and truth values ε_i to specify partial assignments.

We regard partial assignments as subsets of $\mathcal{VA} \times \{0,1\}$ not containing both $(v,0)$ and $(v,1)$ at the same time for any $v \in \mathcal{VA}$, and hence for two partial assignments $\varphi_1, \varphi_2 \in \mathcal{PASS}$ we have $\varphi_1 \cap \varphi_2 \in \mathcal{PASS}$, while $\varphi_1 \cup \varphi_2 \in \mathcal{PASS}$ iff φ_1 and φ_2 are compatible. Note that also $\emptyset \in \mathcal{PASS}$ (the empty assignment).

The following correspondence between partial assignments and clauses is used: For a clause C let $\boldsymbol{\varphi_C} := \langle x \to 0 : x \in C \rangle$ denote that partial assignment which maps all literals of C to 0, while for a partial assignment φ with finite $\operatorname{var}(\varphi)$ the clause $\boldsymbol{C_\varphi} := \{ x : \varphi(x) = 0 \}$ contains all literals x set to 0 by φ.

$\mathcal{PASS}$ *acts* on $\mathcal{CLS}$ via $\boldsymbol{\varphi * F}$ for a partial assignment φ and a clause-set F, where $\varphi * F \in \mathcal{CLS}$ is obtained from F by eliminating all clauses containing a literal x assigned truth-value 1 by φ and cancelling all literals assigned truth-value 0 by φ from the remaining clauses.

We say that a class $\mathcal{C} \subseteq \mathcal{CLS}$ is *stable under partial assignments* if $\mathcal{C}$ is stable under the action of $\mathcal{PASS}$, that is, for all $F \in \mathcal{C}$ and $\varphi \in \mathcal{PASS}$ we have $\varphi * F \in \mathcal{C}$.

$\mathbf{mod_p(\boldsymbol{F})} := \{ \varphi \in \mathcal{PASS} : \mathrm{var}(\varphi) \subseteq \mathrm{var}(F) \wedge \varphi * F = \top \}$ is the set of all partial assignments *satisfying* F (the set of all "partial models"), and $\mathbf{mod_t(\boldsymbol{F})} := \{ \varphi \in \mathcal{PASS} : \mathrm{var}(\varphi) = \mathrm{var}(F) \wedge \varphi * F = \top \}$ is the set of all "total models."

By $\boldsymbol{\mathcal{SAT}} := \{ F \in \mathcal{CLS} : \mathrm{mod_p}(F) \neq \emptyset \}$ the set of *satisfiable clause-sets* is denoted, and by $\boldsymbol{\mathcal{USAT}} := \mathcal{CLS} \setminus \mathcal{SAT}$ the set of *unsatisfiable clause-sets.* If a clause-set F has exactly one satisfying assignment, that is $|\,\mathrm{mod_t}(F)| = 1$ ($\Leftrightarrow |\,\mathrm{mod_p}(F)| = 1$), then F is called *uniquely satisfiable.* The set of all uniquely satisfiable clause-sets is denoted by $\boldsymbol{\mathcal{SAT}}$**-1**.

By $\boldsymbol{F \overset{\mathrm{sat}}{\equiv} F'}$ we denote *satisfiability-equivalence* of F and F' (that is, either both clause-sets are satisfiable or both are unsatisfiable). A *reduction* is a relation between satisfiability-equivalent clause-sets.

A partial assignment φ is called an *autarky* for F ([69]), if for all clauses $C \in F$ we either have $\mathrm{var}(\varphi) \cap \mathrm{var}(C) = \emptyset$ or $\varphi * \{C\} = \top$. More generally a partial assignment φ is called an *autarky modulo contraction* for F if $\varphi * F \subseteq F$ holds ($\Rightarrow \varphi * F \overset{\mathrm{sat}}{\equiv} F$).

A *renaming* is a bijection σ from the set of literals onto itself such that for all literals x we have $\sigma(\overline{x}) = \overline{\sigma(x)}$. Application of renamings to clauses and clauses-sets is defined in the natural way: $\sigma(C) = \{ \sigma(x) : x \in C \}$ and $\sigma(F) = \{ \sigma(C) : C \in F \}$ (yielding again clauses resp. clause-sets).

The closure of $\mathcal{HO}$ under renaming is called

$$\boldsymbol{\mathcal{RHO}} := \big\{ \sigma(F) : F \in \mathcal{HO} \text{ and } \sigma \text{ renaming} \big\}.$$

2.2 Resolution

A clause R is the *resolvent* of clauses C_1, C_2 (called the "parent clauses") iff C_1, C_2 clash on a literal x and $R = (C_1 \setminus \{x\}) \cup (C_2 \setminus \{\overline{x}\})$. The variable $\mathrm{var}(x)$ is called the *resolution variable* here. (Note that any two clauses have at most one resolvent, since otherwise the resolvent would be tautological, which is forbidden by our notion of clauses.)

A *resolution tree* T, deriving a clause C from a clause-set F, for short

$$\boldsymbol{T : F \vdash C},$$

is a binary tree (directed from the root to the leaves) labeled with clauses, such that the leaves are labeled with clauses from F, the root is labeled by C, and the clause labeling an inner node is the resolvent of the clauses labeling its two direct successors. We use $\boldsymbol{T : F \vdash C' \subseteq C}$ to express that $T : F \vdash C'$ for some sub-clause $C' \subseteq C$ of C.

As usual, $\boldsymbol{F \models C}$ denotes $\forall \varphi : \varphi * F = \top \Rightarrow \varphi * \{C\} = \top$. Since [7] it is known that $F \models C$ holds if and only if $F \vdash C' \subseteq C$.

A resolution tree T is called *regular* ([87]), if there is no subtree $T' : F \vdash C'$ of T containing a resolution variable v with $v \in \mathrm{var}(C')$. And T is called an *input resolution tree* if for each node the distance to a leaf is at most one.

By $\boldsymbol{\#\mathrm{lvs}(T)}$ we denote the *number of leaves* of T, by $\boldsymbol{\mathrm{cl}(T)}$ the set of clauses labeling the nodes of T, and by $\boldsymbol{\#\mathrm{cl}(T)} := |\mathrm{cl}(T)|$ the *number of (different) clauses* in T. $\boldsymbol{n(T)} := n(\mathrm{cl}(T))$ is the number of variables occurring in T.

The *complexity* $\boldsymbol{\mathrm{Comp}_{\mathrm{tR}}(F)}$ *of tree-like resolution* for refuting a clause-set F is defined as the infimum of $\#\mathrm{lvs}(T)$ for resolution trees $T : F \vdash \bot$, while the *complexity* $\boldsymbol{\mathrm{Comp}_{\mathrm{R}}(F)}$ *of (full) resolution* for refuting F is the infimum of $\#\mathrm{cl}(T)$ for $T : F \vdash \bot$.

2.3 The Pigeonhole formulas

The variables of $\mathrm{PHP}^m_k \in \mathcal{USAT}$ for $m > k \geq 0$ are $v_{i,j}$ for $1 \leq i \leq m$ and $1 \leq j \leq k$, expressing "pigeon i is in hole j", and the clauses are

$$\begin{aligned} \mathrm{PHP}^m_k &:= P^m_k \cup N^m_k \\ P^m_k &:= \{ \{v_{i,j}\}_{1 \leq j \leq k} \}_{1 \leq i \leq m} \\ N^m_k &:= \{ \{\overline{v_{i_1,j}}, \overline{v_{i_2,j}}\} \}_{1 \leq i_1 < i_2 \leq m,\, 1 \leq j \leq k}. \end{aligned}$$

We have $n(\mathrm{PHP}^m_k) = m \cdot k$, and $c(\mathrm{PHP}^m_k) = m + \frac{1}{2}m(m+1) \cdot k$ for $k \geq 1$ ($\mathrm{PHP}^m_0 = \{\bot\}$). The m positive clauses of length k in P^m_k express "each pigeon i is in at least one hole j,", and the negative 2-clauses in N^m_k express "no two pigeons i_1, i_2 are in the same hole j."

2.4 Directed graphs and pebble games

Notions for directed graphs

Consider a directed graph G. The set of nodes (or "vertices") we denote by $\boldsymbol{V(G)}$. For a node $v \in V(G)$ let $\boldsymbol{\mathrm{ds}(v)}$ resp. $\boldsymbol{\mathrm{dp}(v)}$ be the set of direct successors resp. direct predecessors of v in G. v is called an **input** or **output** if $\mathrm{dp}(v) = \emptyset$ resp. $\mathrm{ds}(v) = \emptyset$ holds. G is called a dag ("directed acyclic graph") if G does not contain a non-trivial cycle.

A directed graph G' can be **embedded** into G if there is an injective map $f : V(G') \to V(G)$, such that for all nodes $v, w \in V(G')$ the existence of a path from v to w in G' implies the existence of a path from $f(v)$ to $f(w)$ in G.

And G' is called a **minor** of G, if an isomorphic copy of G' can be obtained from G by a series of deletions or contractions of edges or deletions of isolated vertices.

It is not hard to see that a directed graph G' can be embedded into a directed graph G if and only if G' is a minor of G, in which case we write $\boldsymbol{G' \leq G}$. The relation $G' \leq G$ always ignores possible labelings of the graphs. To denote that G and G' are *isomorphic* ($\Leftrightarrow G' \leq G \wedge G \leq G'$) we use $\boldsymbol{G \cong G'}$.

A **tree** T in the context of *pebbling* is a dag without non-trivial undirected cycles and with an unique output, the root of T — thus, here the direction of the edges of a tree is *opposite* to the convention we use throughout the rest of the paper:

In the context of DPL-like algorithms (and also for resolution trees) we regard a tree as a *splitting process*, thus evolving it from the root, while here we take the point of view of, say, a proof in an *axiomatic calculus*, which starts from the leaves and combines already proven formulas to new ones. When using the relation $T' \leq T$ for trees in fact the direction does not matter (if only both trees share the same orientation).

Notions for the pebble game (see [74])

For a dag G we denote by $\mathbf{peb}(\boldsymbol{G})$ the minimal number of pebbles needed to pebble G, where shifting of pebbles is also allowed. More precisely:

A "pebbling" of G is a sequence $(g_1, \ldots, g_m)$ of mappings $g_i : V(G) \to \mathbb{N}_0$ ($g_i(v) = 0$ means that v is unpebbled) such that for each $1 \leq i \leq m$ we have:

- $g_i(v) \neq 0$, $g_i(w) \neq 0$, $v \neq w \Longrightarrow g_i(v) \neq g_i(w)$ (at the same time two different nodes can not be pebbled by the same pebble);
- $g_i(v) \neq 0 \Longrightarrow \mathrm{dp}(v) = \emptyset$ or $i > 1$ and either $g_{i-1}(v) \neq 0$ or $\forall\, w \in \mathrm{dp}(v) : g_{i-1}(w) \neq 0$ (or both) (that is, a new node can be pebbled only if all direct predecessors are already pebbled).

New nodes other than inputs must not be pebbled in parallel, that is, for each i there is at most one node w with $g_i(w) \neq 0$, $\mathrm{dp}(v) \neq \emptyset$ and $g_{i-1}(w) = 0$. Furthermore all outputs v of G (an thus actually all nodes) must have been pebbled, that is there is $1 \leq i \leq m$ with $g_i(v) \neq 0$. The number of pebbles used by $(g_1, \ldots, g_m)$ is the maximum of $\bigcup_{1 \leq i \leq m} g_i(V(G))$. Now $\mathbf{peb}(\boldsymbol{G})$ is the minimum of the number of used pebbles over all pebblings of G.

3 A scheme for hierarchies of polynomially decidable (and recognizable) classes of CNF's

In this section two main concepts of this paper are introduced:

- the hierarchies $(G_k(\mathcal{U}, \mathcal{S}))_{k \in \mathbb{N}_0}$ of poly-time recognizable and SAT decidable classes of clause-sets, "amplifying" the (incomplete) oracles $\mathcal{U}, \mathcal{S}$ for unsatisfiability resp. satisfiability detection;

- the corresponding hardness functions $h_{\mathcal{U},\mathcal{S}} : \mathcal{CLS} \to \mathbb{N}_0$.

The basis for these hierarchies is the following "leveled" algorithm for poly-time (incomplete) SAT decision (solving the recognition problem at the same time).

The algorithmic idea

One very basic method to decide the question "$F \in \mathcal{SAT}$?," the core of DPL-like algorithms, is to look whether the question can be decided trivially, that is, whether $F = \top$ or $\bot \in F$ holds, and otherwise to choose a variable $v \in \operatorname{var}(F)$ and split the problem into two subproblems as shown in Figure 1.

$$\boxed{\begin{tikzcd} & F \arrow[dl] \arrow[dr] & \\ \langle v \to 0 \rangle * F & & \langle v \to 1 \rangle * F \end{tikzcd}}$$

Figure 1: Splitting into two subproblems

Now, as discussed in general in Subsection 1.2, a scheme to enforce *polynomial running time* is to introduce a *level* $k \in \mathbb{N}_0$ of "difficulty" or "hardness" and to require that for a suitable chosen variable at least for one branch the level must *go down*, while for the other branch the level must *not go up*. More specifically, we proceed as follows for $F \in \mathcal{CLS}$:

1. We search for a variable $v \in \operatorname{var}(F)$ and $\varepsilon \in \{0, 1\}$ such that $\langle v \to \varepsilon \rangle * F$ can be decided at level $k - 1$. If no such pair (v, ε) exists, the output of our procedure is "level $> k$." Otherwise we fix the first such pair (v, ε) we encountered.

2. If $\langle v \to \varepsilon \rangle * F$ has been found satisfiable, then we are already done with the output "satisfiable with level $\leq k$."

3. Otherwise reduce F to $F' := \langle v \to \overline{\varepsilon} \rangle * F$, and see whether F' can be decided at level k (recursively).

4. If F' has been found satisfiable resp. unsatisfiable at level k, then the same holds for F.

5. In case F' could not be decided at level k, output "level $> k$."

Using $f(k,n)$ for the number of leaves of the corresponding search tree for level k and n variables we get

$$f(k,0) = f(0,n) = 1$$
$$f(k,n) \leq 2n \cdot f(k-1,n-1) + f(k,n-1) \text{ for } k,n \geq 1$$

and thus $f(k,n) \leq (n+1)^{2k}$.

Two points of the above algorithms need special attention:

(i) In step 2 we equip our approach with some *guessing ability* (namely to guess the right truth value for up to k variables) — without that (as in [75]) basically we get a hierarchy for **#SAT**, as discussed at the end of Subsection 3.1.

(ii) The abort of the search in step 5 is essential for the polynomial upper bound. We show that in our setting this abort is justified, since no other choice of (v, ε) could have done better.

It remains the problem what to do at level 0: We use two (incomplete) *oracles* for satisfiability and unsatisfiability detection at level 0, assuming that they contain the most basic satisfiable and unsatisfiable instances and that they are *stable under enforced assignments*, an important (new) condition which justifies step 5 from above.

The hardness

Since the above family of hierarchies covers $\mathcal{CLS}$, by starting with level 0 and successively increasing the level until the input can be decided, we obtain a complete SAT algorithm. The first successful level we call the *hardness* of the input.

The thorough study of this "hardness" may be seen as the main contribution of the present article. Later we will see that this concept stands in intimate relation to the complexity of tree-like resolution, and allows much simplified lower bound proofs. Furthermore, for the first time we also give tools for lower bounds on *satisfiable* clause-sets, while prior work in this direction considered only unsatisfiable ones.

The organization of this section is as follows.

1. In 3.1 the *classes* $G_k(\mathcal{U}, \mathcal{S})$ are introduced, which are the k-levels as discussed before, while $\mathcal{U}$, $\mathcal{S}$ are the oracles for unsatisfiability resp. satisfiability. The *universal property* and *closure properties* of these classes as well as a *decision algorithm* are given.

2. The *general SAT decision* algorithm obtained by searching through the levels from below is the subject of Subsection 3.2. For a given clause-set F the first successful level is called the *hardness* of F, and basic properties are discussed.

3. Parameterizing *reduction by enforced assignments*, in 3.3 the concept of *k-reductions*, naturally related to the classes $G_k(\mathcal{U}, \mathcal{S})$ and generalizing Unit-clause propagation in a *canonical* way, is investigated.

4. *The general method to prove upper and lower bounds* on the hardness is discussed in 3.4. Some applications are given, including the strictness of the hierarchies w.r.t. unsatisfiability.

5. The first instance of a general SAT algorithm searching through levels of increasing hardness has been "Stålmarcks algorithm." In 3.5 we discuss his special form of reduction rule (additionally exploiting the relation between the branches $(v, 0)$ and $(v, 1)$ when running through the loop in step 1 of the above algorithm).

3.1 The classes $G_k(\mathcal{U}, \mathcal{S})$

If a clause-set F contains a 1-clause $\{x_1\} \in F$, then we can reduce F to $F_1 := \langle x_1 \rightarrow 1\rangle * F$. If again F_1 contains a 1-clause $\{x_2\} \in F_1$, we can reduce F_1 to $F_2 := \langle x_2 \rightarrow 1\rangle * F_1$. Now the partial assignment $\langle x_1 \rightarrow 1, x_2 \rightarrow 1\rangle$ is a basic example for an *enforced assignment*:

Definition 3.1 *A partial assignment $\varphi \in \mathcal{PASS}$ is called* ***enforced*** *for a clause-set F, if for all $v \in \mathrm{var}(\varphi)$ flipping the value of φ on v yields an unsatisfiable clause-set:*

$$\langle v \rightarrow \overline{\varphi(v)}\rangle * F \in \mathcal{USAT}.$$

A class $\mathcal{C} \subseteq \mathcal{CLS}$ is ***stable under enforced assignments*** *if for any $F \in \mathcal{C}$ and any enforced assignment φ for F we have $\varphi * F \in \mathcal{C}$.* ∎

Some easy remarks (for $\varphi, \varphi' \in \mathcal{PASS}$ and $F \in \mathcal{CLS}$):

1. The following assertions are equivalent:
   (a) φ enforced for F,
   (b) $\forall\, \varphi' \in \mathrm{mod}_{\mathrm{p}}(F) : \varphi \subseteq \varphi'$,
   (c) $\forall\, \varphi' \in \mathrm{mod}_{\mathrm{t}}(F) : \varphi \subseteq \varphi'$.

2. φ enforced for $F \Longrightarrow \varphi * F \overset{\mathrm{sat}}{\equiv} F$.

3. φ enforced for F, $\varphi' \subseteq \varphi \Longrightarrow \varphi'$ enforced for F.

4. For $F \in \mathcal{USAT}$ every partial assignment φ is enforced, and thus $\mathcal{C} \subseteq \mathcal{USAT}$ is stable under enforced assignments iff $\mathcal{C}$ is stable under partial assignments (at all).

5. For $F \in \mathcal{SAT}$ we have:

   (a) φ enforced for $F \Longrightarrow \mathrm{var}(\varphi) \subseteq \mathrm{var}(F)$;

   (b) φ, φ' enforced for $F \Longrightarrow \varphi, \varphi'$ are compatible;

   (c) φ enforced for F and φ' enforced for $\varphi * F \Longrightarrow \varphi \cup \varphi'$ enforced for F;

   (d) φ, φ' enforced for F and $\varphi \cap \varphi' = \emptyset \Longrightarrow \varphi$ enforced for $\varphi' * F$.

6. For any family $(\mathcal{C}_i)_{i \in I}$ of classes $\mathcal{C}_i \subseteq \mathcal{CLS}$ such that all $\mathcal{C}_i$ are stable under enforced assignments, $\bigcup_{i \in I} \mathcal{C}_i$ as well as $\bigcap_{i \in I} \mathcal{C}_i$ is also stable under enforced assignments.

Definition 3.2 *A class $\mathcal{C} \subseteq \mathcal{CLS}$ **allows substitution** if for every $F \in \mathcal{C}$ with* $\mathrm{var}(F) \neq \emptyset$ *there is $(v, \varepsilon) \in \mathrm{var}(F) \times \{0, 1\}$ with $\langle v \to \varepsilon \rangle * F \in \mathcal{C}$.* ∎

If $\mathcal{C} \subseteq \mathcal{USAT}$ is stable under partial assignments then $\mathcal{C}$ trivially allows substitution. But a satisfiable clause-set need not to have any enforced assignments, and thus $\mathcal{C} \subseteq \mathcal{SAT}$ which is stable under enforced assignments may not allow substitution.

For any family $(\mathcal{C}_i)_{i \in I}$ of classes $\mathcal{C}_i \subseteq \mathcal{CLS}$ such that all $\mathcal{C}_i$ allow substitution also $\bigcup_{i \in I} \mathcal{C}_i$ allows substitution.

Definition 3.3 *Consider $\mathcal{U} \subseteq \mathcal{USAT}$ and $\mathcal{S} \subseteq \mathcal{SAT}$ fulfilling*

(i) $\mathcal{U} \supseteq \boldsymbol{\mathcal{U}_0} := \{ F \in \mathcal{CLS} : \bot \in F \}$, $\mathcal{S} \supseteq \boldsymbol{\mathcal{S}_0} := \{\top\}$

(ii) $\mathcal{U}, \mathcal{S}$ are stable under enforced assignments and allow substitution.

Let

$$\boldsymbol{G_0^0(\mathcal{U}, \mathcal{S})} := \mathcal{U}, \quad \boldsymbol{G_0^1(\mathcal{U}, \mathcal{S})} := \mathcal{S}.$$

For $k \geq 0$ we define the classes $G_{k+1}^0(\mathcal{U}, \mathcal{S}), G_{k+1}^1(\mathcal{U}, \mathcal{S})$ as follows:

$F \in \boldsymbol{G_{k+1}^0(\mathcal{U}, \mathcal{S})}$ *iff either $F = \{\bot\}$ or there is $(v, \varepsilon) \in \mathrm{var}(F) \times \{0, 1\}$ with*

$$\langle v \to \varepsilon \rangle * F \in G_k^0(\mathcal{U}, \mathcal{S}) \text{ and } \langle v \to \overline{\varepsilon} \rangle * F \in G_{k+1}^0(\mathcal{U}, \mathcal{S}).$$

$F \in \boldsymbol{G_{k+1}^1(\mathcal{U}, \mathcal{S})}$ *iff either $F = \top$ or there is $(v, \varepsilon) \in \mathrm{var}(F) \times \{0, 1\}$ with*

$$\langle v \to \varepsilon \rangle * F \in G_k^1(\mathcal{U}, \mathcal{S}) \text{ or}$$

$$\big(\langle v \to \varepsilon \rangle * F \in G_k^0(\mathcal{U}, \mathcal{S}) \text{ and } \langle v \to \overline{\varepsilon} \rangle * F \in G_{k+1}^1(\mathcal{U}, \mathcal{S})\big).$$

Finally $\boldsymbol{G_k(\mathcal{U}, \mathcal{S})} := G_k^0(\mathcal{U}, \mathcal{S}) \cup G_k^1(\mathcal{U}, \mathcal{S})$. ∎

If not specified otherwise, in the sequel $\mathcal{U}, \mathcal{S}$ denote arbitrary sets according to Definition 3.3.

Since trivially $F \in G_{n(F)}(\mathcal{U}, \mathcal{S})$ for all $F \in \mathcal{CLS}$, we have

$$\bigcup_{k \in \mathbb{N}_0} G_k^0(\mathcal{U}, \mathcal{S}) = \mathcal{USAT}, \quad \bigcup_{k \in \mathbb{N}_0} G_k^1(\mathcal{U}, \mathcal{S}) = \mathcal{SAT}.$$

Obviously the classes $G_k^\varepsilon(\mathcal{U}, \mathcal{S})$ are monotonic increasing in $\mathcal{U}$ as well as in $\mathcal{S}$, that is, for $\varepsilon \in \{0, 1\}$ we have

$$\mathcal{U}' \supseteq \mathcal{U}, \mathcal{S}' \supseteq \mathcal{S} \Longrightarrow G_k^\varepsilon(\mathcal{U}', \mathcal{S}') \supseteq G_k^\varepsilon(\mathcal{U}, \mathcal{S}).$$

Since $G_k^0(\mathcal{U}, \mathcal{S})$ does not depend on $\mathcal{S}$, we often use $G_k^0(\mathcal{U})$ instead. If $\mathcal{U}$ and $\mathcal{S}$ are clear from the context, we simply write $G_k^{(\varepsilon)}$.

Compared with the hierarchies Π_k from [75], the main improvement is the now well-defined *realm of possible oracles*:

- the condition “stability under enforced assignments” ensures in the natural way that SAT decision as well as recognition is poly-time, since no backtracking at step 5 of the algorithm in the introduction to this section is necessary (see Lemma 3.7, part 1);
- “stability under enforced assignments” also allows to handle the oracles $\mathcal{U}$ and $\mathcal{S}$ separately (not so the condition “stability under partial assignments” used in [75]);
- considering $\mathcal{U}_0$ and $\mathcal{S}_0$ reveals the basic capacities of all hierarchies;
- with respect to the new “guessing capacity” for satisfiability see the discussion at the end of this subsection.

Lemma 3.4 *The $G_k(\mathcal{U}, \mathcal{S})$ form a cumulative hierarchy, that is, for all $k \geq 0$ we have $G_k(\mathcal{U}, \mathcal{S}) \subseteq G_{k+1}(\mathcal{U}, \mathcal{S})$.*

Proof: By induction on $n(F)$ we prove

$$\forall\, F \in \mathcal{CLS}\ \forall\, 0 \leq k < k' : F \in G_k \Rightarrow F \in G_{k'}.$$

For $n(F) = 0$ we have $F \in G_{k'}$ by definition. So assume $n(F) > 0$.

$\boldsymbol{k = 0}$**:** In case of $F \in \mathcal{S}$ there is $(v, \varepsilon) \in \mathrm{var}(F) \times \{0, 1\}$ with $\langle v \to \varepsilon \rangle * F \in \mathcal{S}$ since $\mathcal{S}$ allows substitution. By induction hypotheses we get $\langle v \to \varepsilon \rangle * F \in G_{k'-1}$ and thus, since $\langle v \to \varepsilon \rangle * F \in \mathcal{SAT}$, by Definition 3.3 it follows $F \in G_{k'}$. If $F \in \mathcal{U}$ then choose any $v \in \mathrm{var}(F)$. Since $\mathcal{U}$ is stable under partial assignments we have $\langle v \to 0 \rangle * F, \langle v \to 1 \rangle * F \in \mathcal{U}$, and thus by induction hypothesis and Definition 3.3 we get $F \in G_{k'}$ as well.

***k* > 0:** Here the assertion follows immediately by induction hypothesis and Definition 3.3 applied to G_k and $G_{k'}$. ∎

Next we show the *universal property* of these classes, basic for all proofs of containment of another hierarchy.

Lemma 3.5 *Consider any family $(S_k)_{k \in \mathbb{N}_0}$ of classes $S_k \subseteq \mathcal{CLS}$ of clause-sets. Assume*

1. *$S_m \subseteq G_m(\mathcal{U}, \mathcal{S})$ for some $m \geq 0$;*
2. *for $k > m$ and $F \in S_k \setminus S_{k-1}$ with $\mathrm{var}(F) \neq \emptyset$ we have $F \in G_k(\mathcal{U}, \mathcal{S})$ or there is $(v, \varepsilon) \in \mathrm{var}(F) \times \{0, 1\}$ with*

$$\langle v \to \varepsilon \rangle * F \in S_{k-1}$$

and

$$\langle v \to \varepsilon \rangle * F \notin \mathcal{SAT} \Longrightarrow \langle v \to \overline{\varepsilon} \rangle * F \in S_k.$$

Then $\forall\, k \in \mathbb{N}_0 : k \geq m \Rightarrow S_k \subseteq G_k(\mathcal{U}, \mathcal{S})$.

Proof: By induction on $n(F)$ we prove

$$\forall\, F \in \mathcal{CLS}\ \forall\, k \geq m : F \in S_k \Rightarrow F \in G_k(\mathcal{U}, \mathcal{S}).$$

If $n(F) = 0$ then $F \in G_k$. Thus consider $n(F) > 0$. Let $k \geq m$ be minimal with $F \in S_k$. If $k = m$, then $F \in G_k$. So assume $k > m$. If $F \in G_k$, then we are done. Thus there is $(v, \varepsilon) \in \mathrm{var}(F) \times \{0, 1\}$ according to the above condition, and by immediate applications of the induction hypothesis $F \in G_k$ follows. ∎

Note that the "or" in condition 2 of Lemma 3.5 is not exclusive — it simply says, that when under certain circumstances we already know $F \in G_k$ for a certain F, then we do not have to provide (v, ε) for that F.

The following corollary gives the basic closure properties of our hierarchies. For a class $\mathcal{C} \subseteq \mathcal{USAT}$ the condition of "being stable under formation of super-clause-sets" is used, which means that for $F \in \mathcal{C}$ and $F' \in \mathcal{CLS}$ with $F' \supseteq F$ also $F' \in \mathcal{C}$ holds, which for example is fulfilled for $\mathcal{U}_0$.

Corollary 3.6 *Closure properties of the classes $G_k(\mathcal{U}, \mathcal{S})$*

1. *All $G_k(\mathcal{U}, \mathcal{S})$ are stable under enforced assignments and allow substitution.*
2. *If $\mathcal{U}$ and $\mathcal{S}$ are stable under renaming, then so are all $G_k(\mathcal{U}, \mathcal{S})$.*
3. *If $\mathcal{U}$ is stable under formation of super-clause-sets, then so are all $G_k^0(\mathcal{U})$.*

Proof: For part 1 let S_k be the closure of G_k under enforced assignments and use commutativity of the application of compatible assignments (that substitution is allowed follows by definition). For part 2 let S_k be the closure of G_k under renaming, and for part 3 let S_k be the closure of G^0_k under super-clause-set formation. ∎

Using part 1, we easily see that for arbitrary $k_1, k_2 \in \mathbb{N}_0$ the equality

$$G_{k_1}(G^0_{k_2}(\mathcal{U}), G^1_{k_2}(\mathcal{U}, \mathcal{S})) = G_{k_1+k_2}(\mathcal{U}, \mathcal{S})$$

holds, and thus the hierarchies can be obtained inductively for $k \geq 0$ by

$$G_{k+1}(\mathcal{U}, \mathcal{S}) = G_1(G^0_k(\mathcal{U}), G^1_k(\mathcal{U}, \mathcal{S})).$$

In Figure 2 algorithm

$$\mathcal{D} : \mathcal{CLS} \times \mathbb{N}_0 \to \{0, 1, \notin\}$$

is given computing

$$\begin{array}{ll} D(F,k) = \varepsilon \in \{0,1\} & \text{iff } F \in G^\varepsilon_k(\mathcal{U}, \mathcal{S}) \\ D(F,k) = \ \notin & \text{iff } F \notin G_k(\mathcal{U}, \mathcal{S}). \end{array}$$

Compared to [75] the recognition process need not to be handled separately and is conceptually much simpler.

Lemma 3.7 *1. Algorithm $\mathcal{D}$ terminates and gives the correct answer.*

2. Consider the search tree reflecting the (recursive) calls of $\mathcal{D}$. This tree has at most $(n(F)+1)^{2k}$ leaves. The work at each inner node can be performed in time $O(\ell(F))$ (using the random access model).

3. If the decisions "$F \in \mathcal{U}$?" and "$F \in \mathcal{S}$?" each requires time $O(\ell(F)^q)$ for some $q \geq 1$, then $\mathcal{D}$ has total running time $O(\ell(F)^q * n(F)^{2k})$.

Proof: *Part 1:* Termination is obvious since any step eliminates at least one variable. To see that for F with $\mathcal{D}(F,k) = \varepsilon \in \{0,1\}$ we have $F \in G^\varepsilon_k(\mathcal{U}, \mathcal{S})$, note that ε gives obviously the right SAT status and use Lemma 3.5. And for $\mathcal{D}(F,k) = \notin \ \Rightarrow F \notin G_k(\mathcal{U}, \mathcal{S})$ assume there is F with $\mathcal{D}(F,k) = \notin$ but $F \in G_k(\mathcal{U}, \mathcal{S})$ and consider such F with minimal $n(F)$. Now a contradiction is obtained using Corollary 3.6, part 1.

Part 2: Consider the function $f : {\mathbb{N}_0}^2 \to \mathbb{N}_0$ given in the introduction to this section

$$f(k, 0) = f(0, n) = 1$$
$$f(k, n) = 2n \cdot f(k-1, n-1) + f(k, n-1).$$

```
    PROCEDURE $\mathcal{D}(F \in \mathcal{CLS}, k \in \mathbb{N}_0) : \{0, 1, \notin\}$;
    BEGIN
       IF $k = 0$ THEN
          IF $F \in \mathcal{U}$ THEN RETURN 0 ELSE
          IF $F \in \mathcal{S}$ THEN RETURN 1 ELSE
             RETURN $\notin$;
       ELSE IF $F = \{\bot\}$ THEN RETURN 0
       ELSE IF $F = \top$ THEN RETURN 1
       ELSE
          FOR $(v, \varepsilon) \in \mathrm{var}(F) \times \{0, 1\}$ DO
(*)          IF $\mathcal{D}(\langle v \to \varepsilon \rangle * F, k - 1) = 1$ THEN
                RETURN 1
             ELSE IF $\mathcal{D}(\langle v \to \varepsilon \rangle * F, k - 1) = 0$ THEN
                RETURN $\mathcal{D}(\langle v \to \overline{\varepsilon} \rangle * F, k)$;
          END FOR;
          RETURN $\notin$;
    END $\mathcal{D}$.
```

Figure 2: Algorithm $\mathcal{D}$ for $G_k^{\varepsilon}(\mathcal{U}, \mathcal{S})$ decision

$f(k, n)$ is an upper bound on the number of leaves and fulfills $f(k, n) \leq (n+1)^{2k}$ proved by induction on n as follows. For $n = 0$ or $k = 0$ we have $(n + 1)^{2k} = 1$, while for $n, k \geq 1$:

$$\begin{aligned}
f(k, n + 1) &= 2(n + 1) \cdot f(k - 1, n) + f(k, n) \\
&\leq 2(n + 1) \cdot (n + 1)^{2(k-1)} + (n + 1)^{2k} = (n + 1)^{2k-1} \cdot (n + 3) \\
&\leq (n + 2)^{2k-1} \cdot (n + 2) = ((n + 1) + 1)^{2k}.
\end{aligned}$$

For the amount of work needed at each inner node observe that, given sufficient "pointer support," the loop over the (v, ε) just visits each literal occurrence only twice.

Part 3 now follows immediately. ∎

The upper bound on the running time of $\mathcal{D}$ in Lemma 3.7 is quite rough, since our concern here is neither to discuss tuned versions of $\mathcal{D}$ nor implementation details. So for example in case of $k = 1$ the above bound gives for the worst case just a cubic upper bound, while linear time is sufficient ([25, 23, 93]).

Corollary 3.8 *If the decisions "$F \in \mathcal{U}$?" and "$F \in \mathcal{S}$?" can be done in polynomial time then membership as well as SAT decision for each class $G_k(\mathcal{U}, \mathcal{S})$*

is also polynomial time. ∎

3.1.1 Cancelling "guessing" yields a poly-time hierarchy for #SAT

If for $F \in \mathcal{CLS}$ we find $\langle v \to \varepsilon \rangle * F \in G^1_{k-1}(\mathcal{U}, \mathcal{S})$ then we are "satisfied" and output the result — since why should we bother investigating also the other branch?

In fact, if we cancel step $(*)$ in $\mathcal{D}$ (corresponding to step 2 in the algorithm from the introduction to this section), that is, if we define for $F \in \mathcal{CLS}$ with $\mathrm{var}(F) \neq \emptyset$ the alternative *counting hierarchy*

$$F \in \#G_{k+1}(\mathcal{U}, \mathcal{S}) \iff$$
$$\exists\,(v, \varepsilon) : \langle v \to \varepsilon \rangle * F \in \#G_k(\mathcal{U}, \mathcal{S}) \;\wedge\; \langle v \to \overline{\varepsilon} \rangle * F \in \#G_{k+1}(\mathcal{U}, \mathcal{S})$$

without differentiating between satisfiability and unsatisfiability as in Definition 3.3, then it is easy to see that $\mathcal{D}$ can be easily tuned (without compromising the time bounds) to output $|\mathrm{mod_t}(F)|$, the *number of satisfying assignments*, provided $\mathcal{S}$ is $\mathcal{S}_0$ or any class of satisfiable clause-sets for which $|\mathrm{mod_t}(F)|$ can be "reasonably computed" (depending on our goals for running time).

Thus, without the guessing possibility instead of investigating SAT decision we would investigate #SAT solving (a much harder problem (see e.g. [72])), which is interesting in its own right, but of subordinate interest for this article.

Due to its symmetrical definition, actually the counting hierarchy behaves often nicer (on satisfiable instances — on unsatisfiable instances there is no difference), so for example from the fact that $\#G_0(\mathcal{U}, \mathcal{S}) = \mathcal{U} \cup \mathcal{S}$ is stable under partial assignments it follows that all $\#G_k(\mathcal{U}, \mathcal{S})$ are stable under partial assignments. Since this assumption is fulfilled for $\mathcal{U}_0 \cup \mathcal{S}_0$, we get for the basic choice of oracles that w.r.t. counting application of partial assignments makes the problem only easier, which is a distinct difference to the problem of SAT decision — here fixing a variable to the false value can render an *easy satisfiable* instance into a *hard unsatisfiable* one.

3.2 The hardness $h_{\mathcal{U},\mathcal{S}}(F)$ of clause-sets

Definition 3.9 *For $F \in \mathcal{CLS}$:*

$$\begin{aligned} \boldsymbol{h_{\mathcal{U},\mathcal{S}}(F)} &:= \min\{\, k \in \mathbb{N}_0 : F \in G_k(\mathcal{U}, \mathcal{S}) \,\} \\ \boldsymbol{h(F)} &:= h_{\mathcal{U}_0,\mathcal{S}_0}(F). \end{aligned}$$ ∎

The *hardness* functions $h_{\mathcal{U},\mathcal{S}}(F)$ are monotonic decreasing in $\mathcal{U}$ and $\mathcal{S}$, that is

$$\mathcal{U}' \supseteq \mathcal{U}, \mathcal{S}' \supseteq \mathcal{S} \Longrightarrow \forall\, F \in \mathcal{CLS} : h_{\mathcal{U}',\mathcal{S}'}(F) \leq h_{\mathcal{U},\mathcal{S}}(F).$$

For $F \in \mathcal{USAT}$ the hardness $h_{\mathcal{U},\mathcal{S}}(F)$ does not depend on $\mathcal{S}$, thus we write often $h_{\mathcal{U}}(F)$ instead.

In Figure 3 the SAT decision algorithm

$$\mathcal{D}^* : \mathcal{CLS} \to \{0,1\} \times \mathbb{N}_0$$

is given, computing $\mathcal{D}^*(F) = (\varepsilon, k)$ such that $F \in G_k^\varepsilon(\mathcal{U}, \mathcal{S})$ and $h_{\mathcal{U},\mathcal{S}}(F) = k$.

```
PROCEDURE $\mathcal{D}^*(F \in \mathcal{CLS}) : \{0,1\} \times \mathbb{N}_0$;
BEGIN
    FOR $k = 0$ to $n(F)$ DO
        IF $\mathcal{D}(F,k) \neq \notin$ THEN
            RETURN $(\mathcal{D}(F,k), k)$
    END FOR
END $\mathcal{D}^*$.
```

Figure 3: The SAT decision algorithm $\mathcal{D}^*$

Lemma 3.7 immediately yields

Lemma 3.10 *Algorithm $\mathcal{D}^*$ terminates and computes the correct answer. The decisions "$F \in \mathcal{U}$?" and "$F \in \mathcal{S}$?" are called each at most $O(n(F)^{2h_{\mathcal{U},\mathcal{S}}(F)})$ times, and if their running time is bounded by $O(\ell(F)^q)$ for some $q \geq 1$, then $\mathcal{D}^*$ has total running time $O(\ell(F)^q * n(F)^{2h_{\mathcal{U},\mathcal{S}}(F)})$.* ▮

As a consequence we get in Subsection 7.4 that on unsatisfiable inputs the running time of algorithm $\mathcal{D}^*$ is quasi-polynomial in the length of the shortest tree-like resolution refutation. For upper and lower bound on the hardness see Subsection 3.4 and Sections 5, 6.

Due to their importance, the next lemma reformulates the stability properties from Lemma 3.6.

Lemma 3.11 *For $F \in \mathcal{CLS}$ and an enforced assignment φ for F we have $h_{\mathcal{U},\mathcal{S}}(\varphi * F) \leq h_{\mathcal{U},\mathcal{S}}(F)$.*

If $\mathcal{U}$ is stable under formation of super-clause-sets, then for $F \in \mathcal{USAT}$ and $F' \supseteq F$ we have $h_{\mathcal{U},\mathcal{S}}(F') \leq h_{\mathcal{U},\mathcal{S}}(F)$.

Thus for $F \in \mathcal{USAT}$ and an autarky (modulo contraction) φ for F under the same assumption $h_{\mathcal{U},\mathcal{S}}(\varphi * F) = h_{\mathcal{U},\mathcal{S}}(F)$ holds.

If $\mathcal{U}, \mathcal{S}$ are stable under renaming then $h_{\mathcal{U},\mathcal{S}}$ is invariant under renaming. ▮

Application of autarkies may decrease as well as increase the hardness on *satisfiable instances*, since for example on the one hand every satisfying assignment is an autarky, while on the other hand for example guessing the value for a pure literal x in F as 0 (opposed to the autark assignment $x \to 1$) may substantially enlarge the possibilities for enforced assignments.

Without guessing, in the counting hierarchies, application of autarkies can not increase the hardness, provided $\mathcal{S}$ is stable under application of autarkies (or under formation of sub-clause-sets). Furthermore, if $G_0(\mathcal{U}, \mathcal{S}) = \mathcal{U} \cup \mathcal{S}$ is stable under partial assignments, then w.r.t. the counting hierarchy also for $F \in \mathcal{SAT}$ we have $h_{\mathcal{U},\mathcal{S}}(\varphi * F) \le h(F)$ for every $\varphi \in \mathcal{PASS}$.

3.3 Canonical reductions

If a clause-set F contains a 1-clause $\{x\} \in F$, then F and $\langle x \to 1 \rangle * F$ are satisfiability-equivalent. Iterated elimination of 1-clauses, called *Unit-clause propagation*, is the most elementary and also the most important reduction for SAT decision.

In case of $\{x\} \in F$ the partial assignment $\langle x \to 1 \rangle$ is enforced for F. *Approximating reduction by general enforced assignments* we introduce in this subsection the concept of ***k*-reduction w.r.t. $\boldsymbol{\mathcal{U}}$**. The practical importance of this concept has been demonstrated in [57], where 2-reduction w.r.t. $\mathcal{U}_0$ has been applied successfully (yielding the fastest SAT solver (at this time) for random k-CNF's at the threshold).

The *unique* result of reducing a clause-set F by k-reductions w.r.t. $\mathcal{U}$ is called $\boldsymbol{r_k^{\mathcal{U}}(F)}$ (the special case of $k = 1$ and $\mathcal{U} = \mathcal{U}_0$ corresponds to Unit-clause propagation). Using this *normal form* we obtain alternative characterizations of the levels $G_k(\mathcal{U}, \mathcal{S})$.

Definition 3.12 *The relation* $\boldsymbol{F \xrightarrow{k,\mathcal{U}} F'}$ *between clause-sets* $F, F' \in \mathcal{CLS}$ *is defined for* $k \in \mathbb{N}_0$ *as follows:*

1. $F \xrightarrow{0,\mathcal{U}} \{\bot\}$ *iff* $F \in \mathcal{U}$ *and* $F \neq \{\bot\}$.
2. $F \xrightarrow{k,\mathcal{U}} \langle v \to \overline{\varepsilon} \rangle * F$ *for* $k \ge 1$ *iff there is* $(v, \varepsilon) \in \mathrm{var}(F) \times \{0,1\}$ *with*
$$\langle v \to \varepsilon \rangle * F \in G^0_{k-1}(\mathcal{U}).$$

By $\xrightarrow{k,\mathcal{U}}*$ *the reflexive-transitive closure of* $\xrightarrow{k,\mathcal{U}}$ *is denoted.* ∎

For $F \xrightarrow{k,\mathcal{U}} F'$ there is an enforced assignment φ for F with $F' = \varphi * F$ and thus $F \overset{\mathrm{sat}}{\equiv} F'$. The opposite direction holds if using $\mathcal{U} = \mathcal{USAT}$, that is

$$F \xrightarrow{1,\mathcal{USAT}} F' \iff \exists\, \varphi \text{ enforced for } F \text{ with } \varphi * F = F'.$$

Note that the reduction concept is cumulative:

$$0 \le k \le k' \text{ and } F \xrightarrow{k,\mathcal{U}} F' \Longrightarrow F \xrightarrow{k',\mathcal{U}} F'.$$

Lemma 3.13 *The relations $\xrightarrow{k,\mathcal{U}}$ are terminating (well-founded) and confluent.*

Proof: Since any reduction step eliminates at least one variable, termination is obvious.

To show confluence, by the diamond lemma ([71, 47]) we only have to show "local confluence," that is, for

$$F \xrightarrow{k,\mathcal{U}} F_1 \text{ and } F \xrightarrow{k,\mathcal{U}} F_2$$

there is F^* with

$$F_1 \xrightarrow{k,\mathcal{U}}\!\!* F^* \text{ and } F_2 \xrightarrow{k,\mathcal{U}}\!\!* F^*.$$

So assume $F_i = \langle v_i \to \overline{\varepsilon_i}\rangle * F$ and $\langle v_i \to \varepsilon_i\rangle * F \in G_{k-1}(\mathcal{U})$ for $i = 1, 2$.

In case $v_1 = v_2$ and $\varepsilon_1 = \overline{\varepsilon_2}$ we have $F_1, F_2 \in G^0_{k-1}(\mathcal{U})$, and thus

$$F_1, F_2 \xrightarrow{k-1,\mathcal{U}}\!\!* \{\bot\}.$$

Otherwise, if $v_1 \notin \mathrm{var}(F_2)$ or $v_2 \notin \mathrm{var}(F_1)$ then $F_1 = F_2$.

So assume $v_1 \in \mathrm{var}(F_2)$ and $v_2 \in \mathrm{var}(F_1)$. Due to Lemma 3.6, part 1

$$\langle v_2 \to \varepsilon_2\rangle * F_1 = \langle v_1 \to \overline{\varepsilon_1}\rangle * (\langle v_2 \to \varepsilon_2\rangle * F) \in G_{k-1}(\mathcal{U})$$
$$\langle v_1 \to \varepsilon_1\rangle * F_2 = \langle v_2 \to \overline{\varepsilon_2}\rangle * (\langle v_1 \to \varepsilon_1\rangle * F) \in G_{k-1}(\mathcal{U})$$

holds, and thus

$$F_1 \xrightarrow{k,\mathcal{U}} \langle v_2 \to \overline{\varepsilon_2}\rangle * F_1 = \langle v_1 \to \overline{\varepsilon_1}\rangle * F_2 \xleftarrow{k,\mathcal{U}} F_2. \quad \blacksquare$$

Definition 3.14 *For $F \in \mathcal{CLS}$ and $k \in \mathbb{N}_0$ let* $\boldsymbol{r^{\mathcal{U}}_k(F)}$ *be the (unique) $F' \in \mathcal{CLS}$ with $F \xrightarrow{k,\mathcal{U}}\!\!* F'$ such that there is no F'' with $F' \xrightarrow{k,\mathcal{U}} F''$. We use* $\boldsymbol{r_k}$ *instead of $r^{\mathcal{U}_0}_k$.*

For $F \in \mathcal{CLS}$ and $\mathcal{C} \subseteq \mathcal{CLS}$ we use $\boldsymbol{F \in \mathcal{C}}$ **mod** $\boldsymbol{r^{\mathcal{U}}_k}$ *for the assertion that there is $F' \in \mathcal{C}$ with $r^{\mathcal{U}}_k(F) = r^{\mathcal{U}}_k(F')$.* $\blacksquare$

r_1 is just unit-clause propagation (additionally with $r_1(F) = \{\bot\}$ in case $\bot$ is created). First (full) use of r_2 in a "DPLL-like" algorithm I'm aware of one finds in "OKsolver" ([57]).

Note that $r^{\mathcal{U}}_k(F_1) = r^{\mathcal{U}}_k(F_2)$ is equivalent to the existence of F' with

$$F_1 \xrightarrow{k,\mathcal{U}}\!\!* F' \text{ and } F_2 \xrightarrow{k,\mathcal{U}}\!\!* F'.$$

Some easy properties of $r^{\mathcal{U}}_k$ for $F \in \mathcal{CLS}$ are

1. $r_k^{\mathcal{U}}(F) \overset{\text{sat}}{\equiv} F$
2. $r_p^{\mathcal{U}}(r_q^{\mathcal{U}}(F)) = r_{\max(p,q)}^{\mathcal{U}}(F)$
3. (a) $h_{\mathcal{U},\mathcal{S}}(r_k^{\mathcal{U}}(F)) \leq h_{\mathcal{U},\mathcal{S}}(F)$
   (b) $h_{\mathcal{U},\mathcal{S}}(F) > k \Rightarrow h_{\mathcal{U},\mathcal{S}}(F) = h_{\mathcal{U},\mathcal{S}}(r_k^{\mathcal{U}}(F))$
4. $F \in G_k(\mathcal{U}, \mathcal{S}) \bmod r_k^{\mathcal{U}} \Rightarrow F \in G_k(\mathcal{U}, \mathcal{S})$.

Using the concept of k-reduction, the universal property from Lemma 3.5 can be rendered more easily applicable:

Lemma 3.15 *Consider a family $(S_k)_{k \in \mathbb{N}_0}$ of classes $S_k \subseteq \mathcal{CLS}$ and assume*

1. $S_m \subseteq G_m(\mathcal{U}, \mathcal{S})$ for some $m \geq 0$;

2. for $k > m$ and any clause-set $F \in S_k \setminus S_{k-1}$ with $\text{var}(F) \neq \emptyset$ we have $F \in G_k(\mathcal{U}, \mathcal{S})$ or there exists $(v, \varepsilon) \in \text{var}(F) \times \{0, 1\}$ fulfilling:

   (a) there is a clause-set $F_\varepsilon \in \mathcal{CLS}$ with $h_{\mathcal{U},\mathcal{S}}(F_\varepsilon) \geq h_{\mathcal{U},\mathcal{S}}(\langle v \to \varepsilon \rangle * F)$ and $F_\varepsilon \in S_{k-1} \bmod r_{k-1}^{\mathcal{U}}$, and

   (b) if $\langle v \to \varepsilon \rangle * F \notin \mathcal{SAT}$ then there is a clause-set $F_{\overline{\varepsilon}} \in \mathcal{CLS}$ with $h_{\mathcal{U},\mathcal{S}}(F_{\overline{\varepsilon}}) \geq h_{\mathcal{U},\mathcal{S}}(\langle v \to \overline{\varepsilon} \rangle * F)$ and $F_{\overline{\varepsilon}} \in S_k \bmod r_k^{\mathcal{U}}$.

Then $S_k \subseteq G_k(\mathcal{U}, \mathcal{S})$ for all $k \geq m$.

Proof: We prove $S_k \subseteq G_k$ by induction on k.

The case $k = m$ is covered by assumption. So assume $k > m$. Consider $F \in S_k$ and assume $F \notin G_k$. Thus $n(F) > 0$ and $F \notin S_{k-1}$ (since otherwise by induction hypothesis $F \in G_{k-1} \subseteq G_k$).

Now there are $(v, \varepsilon) \in \text{var}(F) \times \{0, 1\}$ and clause-sets $F_\varepsilon, F_{\overline{\varepsilon}}$ with

$$h_{\mathcal{U},\mathcal{S}}(F_\varepsilon) \geq h_{\mathcal{U},\mathcal{S}}(\langle v \to \varepsilon \rangle * F), \quad F_\varepsilon \in S_{k-1} \bmod r_{k-1}^{\mathcal{U}}$$
$$h_{\mathcal{U},\mathcal{S}}(F_{\overline{\varepsilon}}) \geq h_{\mathcal{U},\mathcal{S}}(\langle v \to \overline{\varepsilon} \rangle * F), \quad \langle v \to \varepsilon \rangle * F \notin \mathcal{SAT} \Rightarrow F_{\overline{\varepsilon}} \in S_k \bmod r_k^{\mathcal{U}}.$$

By definition there is $F'_\varepsilon \in S_{k-1}$, $r_{k-1}^{\mathcal{U}}(F_\varepsilon) = r_{k-1}^{\mathcal{U}}(F'_\varepsilon)$. By induction hypothesis we get $F'_\varepsilon \in G_{k-1}$, thus $F_\varepsilon \in G_{k-1}$, and we conclude $\langle v \to \varepsilon \rangle * F \in G_{k-1}$.

If $\langle v \to \varepsilon \rangle * F \in G_{k-1} \in \mathcal{SAT}$, then by definition $F \in G_k$. Otherwise we have $F_{\overline{\varepsilon}} \in S_k \bmod r_k^{\mathcal{U}}$, and hence there is $F'_{\overline{\varepsilon}} \in S_k$, $r_k^{\mathcal{U}}(F_{\overline{\varepsilon}}) = r_k^{\mathcal{U}}(F'_{\overline{\varepsilon}})$. Again by induction hypothesis $F'_{\overline{\varepsilon}} \in G_k$, $\Rightarrow F_{\overline{\varepsilon}} \in G_k$, $\Rightarrow \langle v \to \overline{\varepsilon} \rangle * F \in G_k$.

Altogether $F \in G_k$. ∎

We conclude this subsection with alternative characterizations of the hierarchies $G_k(\mathcal{U}, \mathcal{S})$ in terms of k-reductions.

For $F \in \mathcal{USAT}$ we have $h_{\mathcal{U}}(F) \leq k$ if and only if $r_k^{\mathcal{U}}(F) = \{\bot\}$. For $F \in \mathcal{SAT}$ k-reduction alone is not sufficient to determine satisfiability (the strongest possible reduction yields $r_1^{\mathcal{USAT}}(F) = \top \Leftrightarrow F \in \mathcal{SAT}\text{-}1$), but we have to alternate it with guessing. More precisely, $h_{\mathcal{U},\mathcal{S}}(F) \leq k$ holds if and only if the following non-deterministic algorithm is successful:

```
WHILE $k > 0$ reduce $F := r_k^{\mathcal{U}}(F)$:
    IF $F = \top$ then "success",
    ELSE *guess* a variable $v \in \mathrm{var}(F)$ and $\varepsilon \in \{0, 1\}$,
        SET $F := \langle v \to \varepsilon \rangle * F$ and $k := k - 1$.
IF (finally) $k = 0$, then we have "success" iff $F \in \mathcal{S}$.
```

Lemma 3.16 *For $F \in \mathcal{CLS}$ and $k \geq 0$ we have*

$$F \in G_k^1(\mathcal{U}, \mathcal{S})$$

iff there is $\varphi \in \mathcal{PASS}$ with $n(\varphi) \leq k$ and an ordering

$$\mathrm{var}(\varphi) = \{v_1, \ldots, v_{n(\varphi)}\}$$

such that

$$F_{n(\varphi)} \in \mathcal{S}$$

holds, where

$$\begin{aligned}
F_0 &:= r_k^{\mathcal{U}}(F) \\
F_1 &:= r_{k-1}^{\mathcal{U}}(\langle v_1 \to \varphi(v_1) \rangle * F_0) \\
F_2 &:= r_{k-2}^{\mathcal{U}}(\langle v_2 \to \varphi(v_1) \rangle * F_1) \\
&\vdots \\
F_{n(\varphi)} &:= r_{k-n(\varphi)}^{\mathcal{U}}(\langle v_{n(\varphi)} \to \varphi(v_{n(\varphi)}) \rangle * F_{n(\varphi)-1}). \quad \blacksquare
\end{aligned}$$

3.4 How to prove upper and lower bounds on the hardness

Throughout this article, the following considerations are basic for proving any upper or lower bounds on the hardness $h_{\mathcal{U},\mathcal{S}}(F)$.

3.4.1 Upper bounds

Suppose we want to show the upper bound

$$\forall\, F \in \mathcal{C} : h_{\mathcal{U},\mathcal{S}}(F) \leq u(F)$$

for $\emptyset \neq \mathcal{C} \subseteq \mathcal{CLS}$ and $u : \mathcal{C} \to \mathbb{N}_0$. For that purpose consider

$$S_k := \{\, F \in \mathcal{C} : u(F) \leq k \,\}$$

for $k \in \mathbb{N}_0$ and let m be the minimal k with $S_k \neq \emptyset$. Now due to Lemma 3.15 it suffices to show that $S_m \subseteq G_m(\mathcal{U}, \mathcal{S})$ holds, and that for $k > m$ and $F \in S_k \setminus S_{k-1}$ with $\operatorname{var}(F) \neq \emptyset$ for which $h_{\mathcal{U},\mathcal{S}}(F) \leq k$ is not already known there is a variable $v \in \operatorname{var}(F)$ and $\varepsilon \in \{0, 1\}$ fulfilling the following two conditions:

- without decreasing the hardness, $\langle v \to \varepsilon \rangle * F$ can be transformed into F_ε such that $F_\varepsilon \in S_{k-1} \bmod r^{\mathcal{U}}_{k-1}$;
- if $\langle v \to \varepsilon \rangle * F \notin \mathcal{SAT}$, then $\langle v \to \overline{\varepsilon} \rangle * F$ can be transformed into $F_{\overline{\varepsilon}}$ without decreasing the hardness such that $F_{\overline{\varepsilon}} \in S_k \bmod r^{\mathcal{U}}_k$.

To transform a clause-set F into F' with $h_{\mathcal{U},\mathcal{S}}(F') \geq h_{\mathcal{U},\mathcal{S}}(F)$[5], besides choosing $F' = F$ we know the following general means (see Lemma 3.11):

- application of *renamings* (supposed $\mathcal{U}$ (and $\mathcal{S}$) are stable under renaming);
- *elimination of clauses* in case of $F \in \mathcal{USAT}$ (supposed $\mathcal{U}$ is stable under formation of super-clause-sets).

3.4.2 Lower bounds

Lemma 3.17 *Consider a family $(S_k)_{k \in \mathbb{N}_0}$ of classes $S_k \subseteq \mathcal{CLS}$ and assume*

1. *$S_m \cap G_{m-1}(\mathcal{U}, \mathcal{S}) = \emptyset$ for some $m \geq 1$;*
2. *for $k > m$ and any $F \in S_k$ we have $F \notin G_0(\mathcal{U}, \mathcal{S})$, and furthermore $F \notin G_{k-1}(\mathcal{U}, \mathcal{S})$ holds true, or for all $v \in \operatorname{var}(F)$ and $\varepsilon \in \{0, 1\}$ there exist clause-sets $F_\varepsilon, F_{\overline{\varepsilon}}$ fulfilling the following conditions:*

$$h_{\mathcal{U},\mathcal{S}}(F_\varepsilon) \leq h_{\mathcal{U},\mathcal{S}}(\langle v \to \varepsilon \rangle * F), \quad h_{\mathcal{U},\mathcal{S}}(F_{\overline{\varepsilon}}) \leq h_{\mathcal{U},\mathcal{S}}(\langle v \to \overline{\varepsilon} \rangle * F);$$

   - *if $\langle v \to \varepsilon \rangle * F \in \mathcal{SAT}$ then $F_\varepsilon \in S_{k-1}$;*
   - *if $\langle v \to \varepsilon \rangle * F \in \mathcal{USAT}$ then $F_\varepsilon \in S_{k-1}$ or $F_{\overline{\varepsilon}} \in S_k$.*

Then for all $k \geq m$ we have $S_k \cap G_{k-1}(\mathcal{U}, \mathcal{S}) = \emptyset$.

Proof: Assume that there is $k > m$ with $S_k \cap G_{k-1} \neq \emptyset$ and consider the minimal such k: A contradiction to the minimality of k is immediately obtained from Definition 3.3. ∎

Now suppose we want to show the lower bound

$$\forall\, F \in \mathcal{C} : h_{\mathcal{U},\mathcal{S}}(F) \geq s(F)$$

[5] it is not required that the transformed clause-set is satisfiability-equivalent to the original clause-set; furthermore, we use "transformed" here only for better intuitive understanding, while in fact apart from the hardness condition there need not to be any relation between F and F'

for $\emptyset \neq \mathcal{C} \subseteq \mathcal{CLS}$ and $s : \mathcal{C} \to \mathbb{N}_0$. For that purpose consider

$$S_k := \{ F \in \mathcal{C} : s(F) \geq k \}$$

for $k \in \mathbb{N}_0$ and let m be the minimal $k \geq 1$ with $S_k \neq S_{k-1}$ ($m = 1 + \min s(\mathcal{C})$). If now conditions 1 (using this m) and 2 of Lemma 3.17 are fulfilled, then that lower bound in fact holds true.

To transform a clause-set F into F' with $h_{\mathcal{U},\mathcal{S}}(F') \leq h_{\mathcal{U},\mathcal{S}}(F)$, besides choosing $F' = F$ the following general means are provided by Lemma 3.11:

- application of *renamings* (supposed $\mathcal{U}$ (and $\mathcal{S}$) are stable under renaming);
- *addition of clauses* in case of $F \in \mathcal{USAT}$ (supposed $\mathcal{U}$ is stable under formation of super-clause-sets);
- application of *enforced assignments*.

3.4.3 Some easy applications

The next three lemmas are very easy applications of the methods from Subsections 3.4.1 and 3.4.2, so for the proofs we just state the needed facts on splitting, that is, how the formulas under considerations behave under splitting as in Figure 1.

Lemma 3.18 *1. For $F \in \mathcal{CLS}$ we have*

(a) $h(F) \leq n(F)$

(b) $h(F) \leq c(F)$.

2. For $F \in \mathcal{USAT}$ we have

(a) $h(F) < c(F)$

(b) $h(F) \geq \min_{C \in F} |C|$.

Proof: For Parts 1a, 1b split on any variable: In both branches at least one variable disappears, and in at least one branch at least one clause disappears.

For Part 2a additionally note, that $c(F) \geq 1$ and $c(F) = 1 \Leftrightarrow F = \{\bot\}$ holds. And for Part 2b observe that by splitting on a variable the minimal clause length can decrease at most by one. ∎

The upper bound $h(F) \leq n(F)$ is tight on unsatisfiable instances (see the next Lemma 3.19), while $h(F) \leq c(F)$ is tight on satisfiable instances (see Lemma 3.20). Using the parameter $p(F)$ (the maximal input clause length) the first upper bound will be refined on unsatisfiable instances in Lemma 5.5.

Both upper bounds actually hold for hardness w.r.t. the counting hierarchies $\#G_k(\mathcal{U}, \mathcal{S})$ (recall Subsection 3.1.1).

By applying Lemma 3.11 the lower bound can be strengthened somewhat ($F \in \mathcal{USAT}$):

$$h(F) \geq \max_{\varphi \in \mathcal{PASS}} \min_{C \in \varphi * F} |C|.$$

Lemma 3.19 *For finite $V \subseteq \mathcal{VA}$ let*

$$F_V := \{ C \in \mathcal{CL} : \operatorname{var}(C) = V \} \ (\in \mathcal{USAT}).$$

Then $h(F_V) = n(F_V) = |V|$.

More generally we consider $\mathcal{U}$ stable under renaming. Then either for all finite $V \subseteq \mathcal{VA}$ we have $F_V \in \mathcal{U}$ (and thus $h_{\mathcal{U}}(F_V) = 0$), or there is $m \in \mathbb{N}_0$ such that for all finite $V \subseteq \mathcal{VA}$

$$F_V \in \mathcal{U} \iff |V| \leq m$$

holds. Now we have $h_{\mathcal{U}}(F_V) = \max(0, n(F_V) - m)$.

Proof: $F_\emptyset = \{\bot\}$, and for any $v \in \operatorname{var}(F_V)$ and $\varepsilon \in \{0, 1\}$ we have

$$\langle v \to \varepsilon \rangle * F_V = F_{V \setminus \{v\}}. \ \blacksquare$$

Lemma 3.20 *Let $\mathcal{N}_2$ be the class of clause-sets $F \in \mathcal{CLS}$ containing only disjoint negative 2-clauses, that is $|C| = 2$, $C \subseteq \overline{\mathcal{VA}}$ and $C \neq C' \Rightarrow C \cap C' = \emptyset$ holds for all $C, C' \in F$. (Thus $\mathcal{N}_2 \subseteq \mathcal{SAT} \cap 2\text{-}\mathcal{CLS} \cap \mathcal{HO}$.)*

Then for $F \in \mathcal{N}_2$ we have $h_{\mathcal{USAT}, \mathcal{S}_0}(F) = c(F)$.

Proof: For $F \in \mathcal{N}_2$ and $v \in \operatorname{var}(F)$ the following holds:

- $\langle v \to 0 \rangle * F \in \mathcal{N}_2$, $c(\langle v \to 0 \rangle * F) = c(F) - 1$;
- $\langle v \to 1 \rangle * F \in \mathcal{SAT}$, and for $\{\overline{v}, \overline{w}\} \in F$ we have $\{\overline{w}\} \in \langle v \to 1 \rangle * F$ and $\langle \overline{w} \to 1 \rangle * (\langle v \to 1 \rangle * F) = \langle w \to 0 \rangle * F \in \mathcal{N}_2$. $\blacksquare$

We conclude this subsection by discussing the strictness of the hierarchies $(G_k(\mathcal{U}, \mathcal{S}))_{k \in \mathbb{N}_0}$.

Lemma 3.21 *If $\mathcal{U} \neq \mathcal{USAT}$ and $\mathcal{U}$ is stable under renaming, then for all $k \geq 1$ there is $F_k \in \mathcal{USAT}$ with $h_{\mathcal{U}}(F_k) = k$.*

Proof: Consider $F_1 \in \mathcal{USAT} \setminus \mathcal{U}$ with minimal $n(F)$. For $k \geq 2$ let σ_k be a renaming with

$$\operatorname{var}(\sigma_k(F_{k-1})) \cap \operatorname{var}(F_{k-1}) = \emptyset,$$

choose a variable $v_k \notin \operatorname{var}(\sigma_k(F_{k-1})) \cup \operatorname{var}(F_{k-1})$ and define

$$F_k := \{\, C \cup \{v_k\} : C \in F_{k-1} \,\} \cup \{\, C \cup \{\overline{v_k}\} : C \in \sigma_k(F_{k-1}) \,\}.$$

Since for any $v \in \operatorname{var}(F_1)$ we have $\langle v \to 0 \rangle * F_1, \langle v \to 1 \rangle * F_1 \in \mathcal{U}$, it follows $h_{\mathcal{U}}(F_1) = 1$. For $k \geq 2$ on the one side we have

$$\langle v_k \to 0 \rangle * F_k = F_{k-1},\ \langle v_k \to 1 \rangle * F_k = F_{k-1} \text{ mod renaming}.$$

On the other side for any $v \in \operatorname{var}(F_k) \setminus \{v_k\}$ and $\varepsilon \in \{0,1\}$, let $\varepsilon' := 1$ in case of $v \in \operatorname{var}(F_{k-1})$ and $\varepsilon' := 0$ otherwise and note

$$\langle v_k \to \varepsilon' \rangle * (\langle v \to \varepsilon \rangle * F_k) = F_{k-1} \text{ mod renaming}.$$

By Subsections 3.4.1 and 3.4.2 we obtain the assertion. ∎

Without $\mathcal{U}$ being stable under renaming, the hierarchies may collapse, as the following example shows.

Choose $v_0 \in \operatorname{var}(F)$ and let $F_0 := \{\, \{v_0\}, \{\overline{v_0}\} \,\}$. Define

$$\mathcal{U} := \{\, F \in \mathcal{USAT} : \neg\exists\, \varphi \in \mathcal{PASS}\, [\, \varphi * F = F_0 \,] \,\}.$$

Obviously $\mathcal{U}$ is stable under partial assignments and contains $\mathcal{U}_0$. Now

$$G_0^0(\mathcal{U}) = \mathcal{U}, \quad G_1^0(\mathcal{U}) = \mathcal{USAT},$$

since for $F \in \mathcal{USAT}$ in case of $v_0 \notin \operatorname{var}(F)$ we have $F \in \mathcal{U}$, and otherwise we have $\langle v_0 \to 0 \rangle * F, \langle v_0 \to 1 \rangle * F \in \mathcal{U}$.

3.5 Extracting more information (Stålmarck's algorithm)

In this final subsection of Section 3 we show how the core concept of Stålmarck's algorithm (see [84, 44, 81]) can be motivated within our approach as an attempt for more efficient use of the information gained in the search for reductions.

Recall (Subsection 3.3) that for $k \geq 1$ we can apply the reduction

$$F \xrightarrow{k,\mathcal{U}} \langle v \to \overline{\varepsilon} \rangle * F$$

for $F \in \mathcal{CLS}$ and $(v, \varepsilon) \in \operatorname{var}(F) \times \{0,1\}$ if and only if

$$r_{k-1}^{\mathcal{U}}(\langle v \to \varepsilon \rangle * F) = \{\bot\} \tag{1}$$

holds. The obvious way to search for such a reduction is to run through all possible pairs (v, ε) (independent of each other) and to try whether condition (1) holds true. Is there is a more efficient way, using for the current pair (v, ε) some information on previously regarded pairs (v', ε') ?!

We treat here another path: Given the (unsuccessful) computations on all pairs (v, ε), is there a chance to get nevertheless something out of it? A first step is to link the branches $(v, 0)$ and $(v, 1)$, as it is (in effect) the case for the Stålmarck algorithm.

Assume

$$r^{\mathcal{U}}_{k-1}(\langle v \to \varepsilon\rangle * F) = F_\varepsilon \neq \{\bot\}$$

for $\varepsilon = 0, 1$. Naturally in the course of computing F_ε the corresponding enforced assignment φ_ε for $\langle v \to \varepsilon\rangle * F$ with $\varphi_\varepsilon * (\langle v \to \varepsilon\rangle * F) = F_\varepsilon$ can be computed without additional effort. Now by the simple fact

> If the partial assignments φ_0, φ_1 are both enforced for $\langle v \to \varepsilon\rangle * F$, then $\varphi_0 \cap \varphi_1$ is enforced for F.

in case of $\varphi_0 \cap \varphi_1 \neq \emptyset$ in fact we found an additional reduction for F:

$$F \longrightarrow (\varphi_0 \cap \varphi_1) * F.$$

To put this into a simpler form, note that for $\langle w \to \overline{\delta}\rangle \subseteq \varphi_0 \cap \varphi_1$ we have

$$\forall\, \varepsilon \in \{0,1\} : \langle v \to \varepsilon\rangle * (\langle w \to \delta\rangle * F) \xrightarrow{k-1,\mathcal{U}}\!\!* \{\bot\},$$

and thus the following definition adequately expresses the above reduction process (applied recursively).

For $k = 1$ there is no additional reduction, that is

$$r^{\mathcal{U}}_0(\langle v \to \varepsilon\rangle * F) \neq \{\bot\} \Longrightarrow r^{\mathcal{U}}_0(\langle v \to \varepsilon\rangle * F) = \langle v \to \varepsilon\rangle * F$$

and thus $\varphi_\varepsilon = \emptyset$. Therefore at level 0 of the new reduction hierarchy we use already $\xrightarrow{1,\mathcal{U}}$ instead of $\xrightarrow{0,\mathcal{U}}$.

Definition 3.22 *The reduction* $\xrightarrow[+]{0,\mathcal{U}}$ *is the same as* $\xrightarrow{1,\mathcal{U}}$. *Consider* $k \geq 1$. *For* $F \in \mathcal{CLS}$ *and* $(w, \delta) \in \mathrm{var}(F) \times \{0,1\}$ *we have*

$$F \xrightarrow[+]{k,\mathcal{U}} \langle w \to \overline{\delta}\rangle * F$$

if there is $v \in \mathrm{var}(F)$ *with*

$$\forall\, \varepsilon \in \{0,1\} : \langle v \to \varepsilon\rangle * (\langle w \to \delta\rangle * F) \xrightarrow[+]{k-1,\mathcal{U}}\!\!* \{\bot\},$$

where "$\longrightarrow\!\!*$" denotes the reflexive-transitive closure of "$\longrightarrow$".

For $F \in \mathcal{USAT}$ *let* $\boldsymbol{h^+_{\mathcal{U}}(F)}$ *be the minimal* $k \geq 0$ *with* $F \xrightarrow[+]{k,\mathcal{U}}\!\!* \{\bot\}$. ∎

The relations $\xrightarrow[+]{k,\mathcal{U}}$ (again) are all terminating and confluent, and we have

$$\frac{1}{2}(h_{\mathcal{U}}(F) - 1) \leq h^{+}_{\mathcal{U}}(F) \leq h_{\mathcal{U}}(F) - 1$$

for $F \in \mathcal{USAT}$. The reductions $\xrightarrow[+]{k,\mathcal{U}_0}$ build the core of Stålmarck's algorithm (besides other obvious rules to handle general formulas; to handle satisfiability no special means are given).

In [41] an extended version of Stålmarck's algorithm is presented, using additionally certain resolution-based reductions (see [60] for an in-depth discussion of such reductions).

Consistent use of all branches

Consider any $I \subseteq \mathrm{var}(F) \times \{0, 1\}$ such that for all $(v, \varepsilon) \in I$ enforced assignments $\varphi_{v,\varepsilon}$ for $\langle v \to \varepsilon\rangle * F$ have been computed (w.l.o.g. $v \notin \mathrm{var}(\varphi_{v,\varepsilon})$).

Using the general facts (for $\varphi, \varphi' \in \mathcal{PASS}$ and $F \in \mathcal{CLS}$)

- φ is enforced for F iff for all literals x with $\langle x \to 1\rangle \subseteq \varphi$ the 1-clause $\{x\}$ is implied by F;
- more general, φ is enforced for $\varphi' * F$ if for all x with $\langle x \to 1\rangle \subseteq \varphi$ one has $F \models C_{\varphi'} \cup \{x\}$;

we get

$$F \models G_I := \{\, C_{\langle v \to \varepsilon\rangle} \cup \{x\} : (v, \varepsilon) \in I \wedge \langle x \to 1\rangle \subseteq \varphi_{v,\varepsilon} \,\}.$$

Since $G_I \in 2\text{–}\mathcal{CLS}$, we can decide in linear time whether $G_I \in \mathcal{USAT}$ holds, in which case we infer $F \in \mathcal{USAT}$. And if

$$G_I \models \{x\}$$

for some literal x, then $\langle x \to 1\rangle$ is enforced for F, and thus we have found a new reduction for F.

4 Generalized DPL-trees and pebble games

In this section the concept of *generalized DPL-trees* and its relation to the hierarchies $(G_k(\mathcal{U}, \mathcal{S}))_{k \in \mathbb{N}_0}$ is investigated.

DPL-trees are binary trees where every branching is a splitting on a variable as in Figure 1. Thus DPL-trees are the search trees for the most basic

SAT algorithms (with leaves labeled by elements of $\mathcal{U}_0$ or $\mathcal{S}_0$). DPL-trees for unsatisfiable clause-sets F immediately correspond to *decision trees* solving the "search problem" for F as defined in [63][6], and are also known as *semantic trees* (see [88]).

We consider a generalization, called "$(\mathcal{U}, \mathcal{S})$-DPL-trees," where leaves can be labeled by elements of $\mathcal{U}$ or $\mathcal{S}$, or can be left open in case there is some leave labeled with an element of $\mathcal{S}$. The immediate correspondence to (generalized) tree-like resolution refutations with "oracles" (in case of "$\mathcal{U}$-DPL-trees") is studied in Section 7.

Introducing the *leveled height* $\boldsymbol{h_{\mathcal{U},\mathcal{S}}(T)}$ of $(\mathcal{U}, \mathcal{S})$-DPL-trees we are able to characterize clause-sets F with hardness $h_{\mathcal{U},\mathcal{S}}(F) \leq k$ as those clause-sets having a $(\mathcal{U}, \mathcal{S})$-DPL-tree T with $h_{\mathcal{U},\mathcal{S}}(T) \leq k$.

For *closed* DPL-trees (no open leaves are allowed — always fulfilled for unsatisfiable clause-sets or for the counting hierarchies) the leveled height $h_{\mathcal{U},\mathcal{S}}(T)$ is a well studied quantity, equal to the *pebbling complexity* of T minus 1.

4.1 DPL-trees and their leveled height

Definition 4.1 *A* ***DPL-tree*** *for $F \in \mathcal{CLS}$ is a binary tree, the nodes labeled with clause-sets by*

$$F_T : V(T) \to \mathcal{CLS},$$

such that $F_T(\mathrm{root}(T)) = F$ holds, and for each inner node w of T with direct successors w_0, w_1 there is a variable $v \in \mathrm{var}(F_T(w))$ with

$$F_T(w_\varepsilon) = \langle v \to \varepsilon \rangle * F_T(w)$$

for $\varepsilon \in \{0, 1\}$. v is called the ***branching variable*** *for w.*

Now consider $\mathcal{U}, \mathcal{S}$ according to Definition 3.3.

A node w of a DPL-tree T is called an ***$\mathcal{U}$-node*** *if all leaves below w are labeled with elements of $\mathcal{U}$, while w is called a* ***$\mathcal{S}$-node*** *if there is a leaf below w labeled by an element of $\mathcal{S}$. If w either is an $\mathcal{U}$- or a $\mathcal{S}$-node, then w is called a* ***$(\mathcal{U}, \mathcal{S})$-node****. The same terminology as for nodes is used for the corresponding subtrees of T. (Thus we call a DPL-tree an $(\mathcal{U}, \mathcal{S})$-tree for example, if the root of T is an $(\mathcal{U}, \mathcal{S})$-node.)* ∎

Clearly every $F \in \mathcal{CLS}$ has a $(\mathcal{U}, \mathcal{S})$-DPL-tree T, and any such tree T for F is an $\mathcal{U}$-tree if $F \in \mathcal{USAT}$ and a $\mathcal{S}$-tree if $F \in \mathcal{SAT}$.

[6]given any assignment φ with $\mathrm{var}(\varphi) = \mathrm{var}(F)$, find a clause $C \in F$ falsified by φ

Definition 4.2 *The* ***leveled height*** $\boldsymbol{h_{\mathcal{U},\mathcal{S}}(T)}$ *for a DPL-tree T is defined by induction on the composition of T as follows.*

If T is trivial, then $h_{\mathcal{U},\mathcal{S}}(T) := 0$.

Otherwise let T consist of subtrees T_1 and T_2:

- *if one of T_1, T_2 is not an $(\mathcal{U},\mathcal{S})$-tree, or $h_{\mathcal{U},\mathcal{S}}(T_1) = h_{\mathcal{U},\mathcal{S}}(T_2)$, then*

$$h_{\mathcal{U},\mathcal{S}}(T) := \max\left(h_{\mathcal{U},\mathcal{S}}(T_1), h_{\mathcal{U},\mathcal{S}}(T_2)\right) + 1;$$

- *otherwise (that is, both T_1, T_2 are $(\mathcal{U},\mathcal{S})$-trees, and $h_{\mathcal{U},\mathcal{S}}(T_1) \neq h_{\mathcal{U},\mathcal{S}}(T_2)$)*

$$h_{\mathcal{U},\mathcal{S}}(T) := \max\left(h_{\mathcal{U},\mathcal{S}}(T_1), h_{\mathcal{U},\mathcal{S}}(T_2)\right). \blacksquare$$

Theorem 4.3 *The hardness $h_{\mathcal{U},\mathcal{S}}(F)$ for $F \in \mathcal{CLS}$ is equal to the minimal leveled height $h_{\mathcal{U},\mathcal{S}}(T)$ of $(\mathcal{U},\mathcal{S})$-DPL-trees for F.*

Proof: The proof is an easy application of the considerations in Subsections 3.4.1 and 3.4.2.

Let $\mu_{\mathcal{U},\mathcal{S}}(F)$ be the minimal leveled height $h_{\mathcal{U},\mathcal{S}}(T)$ for $(\mathcal{U},\mathcal{S})$-DPL-trees T for F. Obviously we have

$$h_{\mathcal{U},\mathcal{S}}(F) = 0 \Leftrightarrow \mu_{\mathcal{U},\mathcal{S}}(F) = 0.$$

To show $h_{\mathcal{U},\mathcal{S}}(F) \leq \mu_{\mathcal{U},\mathcal{S}}(F)$ let

$$S_k := \{\, F \in \mathcal{CLS} : \mu_{\mathcal{U},\mathcal{S}}(F) \leq k \,\}$$

for $k \geq 0$. Consider $k \geq 1$ and $F \in S_k$ with $\mathrm{var}(F) \neq \emptyset$. Let T be an $(\mathcal{U},\mathcal{S})$-DPL-tree for F with $h_{\mathcal{U},\mathcal{S}}(T) = \mu_{\mathcal{U},\mathcal{S}}(F)$, and let v be the branching variable at the root of T with associated subtrees T_0, T_1.

By definition of $h_{\mathcal{U},\mathcal{S}}(T)$ there is $\varepsilon \in \{0,1\}$ such that T_ε is an $(\mathcal{U},\mathcal{S})$-tree with

$$h_{\mathcal{U},\mathcal{S}}(T_\varepsilon) < h_{\mathcal{U},\mathcal{S}}(T).$$

Thus $\langle v \rightarrow \varepsilon \rangle * F \in S_{k-1}$.

If T_ε is an $\mathcal{U}$-tree, then $T_{\overline{\varepsilon}}$ must be an $(\mathcal{U},\mathcal{S})$-tree, and we get

$$\langle v \rightarrow \overline{\varepsilon} \rangle * F \in S_k.$$

For the opposite direction $h_{\mathcal{U},\mathcal{S}}(F) \geq \mu_{\mathcal{U},\mathcal{S}}(F)$ define

$$S_k := \{\, F \in \mathcal{CLS} : \mu_{\mathcal{U},\mathcal{S}}(F) \geq k \}$$

for $k \geq 0$. Consider $F \in S_k$ for $k \geq 1$ and $v \in \mathrm{var}(F)$, $\varepsilon \in \{0,1\}$.

First suppose $\langle v \to \varepsilon \rangle * F \in \mathcal{SAT}$. If now

$$\langle v \to \varepsilon \rangle * F \notin S_{k-1},$$

then there is a $\mathcal{S}$-DPL-tree T_ε for $\langle v \to \varepsilon \rangle * F$ with $h_{\mathcal{U},\mathcal{S}}(T_\varepsilon) \leq k-2$, and thus $\mu_{\mathcal{U},\mathcal{S}}(F) \leq k-1$ by combining T_ε with the trivial tree labeled $\langle v \to \overline{\varepsilon} \rangle * F$.

Now assume $\langle v \to \varepsilon \rangle * F \notin \mathcal{SAT}$. If

$$\langle v \to \varepsilon \rangle * F \notin S_{k-1} \text{ and } \langle v \to \overline{\varepsilon} \rangle * F \notin S_k,$$

then by combining the corresponding $(\mathcal{U},\mathcal{S})$-DPL-trees we get $\mu_{\mathcal{U},\mathcal{S}}(F) \leq k-1$ as well. ∎

4.2 Characterizations of the leveled height for closed trees

Definition 4.4 *A* ***closed*** $\boldsymbol{(\mathcal{U},\mathcal{S})}$***-DPL-tree*** *for $F \in \mathcal{CLS}$ is an $(\mathcal{U},\mathcal{S})$-DPL-tree for F where all nodes are $(\mathcal{U},\mathcal{S})$-nodes, that is, where all leaves are labeled with elements from $\mathcal{U} \cup \mathcal{S}$. Since for closed $(\mathcal{U},\mathcal{S})$-DPL-trees T the leveled height $h_{\mathcal{U},\mathcal{S}}(T)$ does not depend on the labeling, we use* $\boldsymbol{h(T)}$ *instead. We say that T has* ***level*** $\boldsymbol{k}$ *if $h(T) \leq k$.* ∎

Every $\mathcal{U}$-DPL-tree is closed by definition. Thus for *unsatisfiable* clause-sets considering only closed DPL-trees is no restriction.

Since the *counting hierarchy* (recall Subsection 3.1.1) does not use "guessing," also here only closed DPL-trees are relevant. Thus the hardness of $F \in \mathcal{CLS}$ for the counting hierarchy w.r.t. $\mathcal{U},\mathcal{S}$ by virtue of Theorem 4.3 is equal to the minimal level of a *closed* $(\mathcal{U},\mathcal{S})$-DPL-tree for F.

In the remainder of this section we give alternative characterizations of the leveled height $h(T)$ for (arbitrary) binary trees (regarding them as closed). The definition of $h(T)$ is simplified as follows:

$h(T) = 0$ if T is trivial, while in case T consists of two subtrees T_1, T_2

$$h(T) = \begin{cases} \max(h(T_1), h(T_2)) & \text{if } h(T_1) \neq h(T_2) \\ \max(h(T_1), h(T_2)) + 1 & \text{if } h(T_1) = h(T_2) \end{cases} \tag{2}$$

holds. Thus T is of level k iff either T is trivial or one of its two subtrees is of level $k-1$ and the other is of level k.

Another characterization is that T is of level k iff for any node in T there is a *path to some leaf of length at most k*.

Trees of level 1 are exactly those trees only allowed for *input resolution*. In Section 7 we will obtain a very natural form of (generalized) *nested input resolution* by translating DPL-trees into resolution trees.

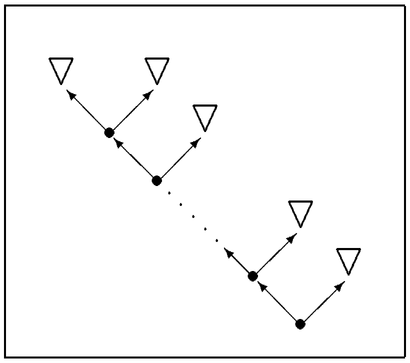
Figure 4: k-level trees: ∇'s are $(k-1)$-level trees

Any tree of level k contains at most k nested level-1-trees (see Figure 4):

A tree T is of level $k \geq 1$ iff there is a level-1-tree T_0 such that T can be obtained from T_0 by replacing the leaves of T_0 by suitable trees of level $k-1$.

Historical remarks

Equation (2) was considered the first time in [70]: There binary trees T coding arithmetic expressions have been considered, and $h(T)$ was used for the "number of accumulators necessary to calculate the expression."

In [32], and independently in [49], the average leveled height A_m for trees with m internal nodes (that is, with $m+1$ leaves), where all trees of the same size are considered to be equally likely, has been computed as $A_m \approx \log_4 m$.

[77] gives an alternative definition for the quantity $h(T)$, applicable to trees of arbitrary degree (and in fact to dag's):

Consider a (rooted) tree T, directed from the leaves to the root. Let

$$(v_1, \dots, v_m)$$

be a topological sorting of the nodes of T, that is, for nodes v_i, v_j with $v_i \neq v_j$ such that there is a (directed) path from v_i to v_j we have $i < j$. For $i \leq i \leq m$ let w_i be the number of edges (v_j, v_k) in T with $j < i < k$ (corresponding to auxiliary results which have to be stored). Now the "width" of $(v_1, \dots, v_m)$ is the maximal w_i for $1 \leq i \leq m$, and the width of T is the minimal width of a topological sorting of T. For binary trees T we have

$$h(T) = \text{ width of } T,$$

which can be easily proved by induction on the composition of T.

Apparently this above notion of "width," applied to dag's G, has been the first appearance of what now is called the *pebbling complexity* $\mathrm{peb}(G)$ *of* G (that is, $\mathrm{peb}(G) = 1 + (\text{width of } G)$ for any dag G).

4.3 The pebbling complexity of trees

All the following characterizations of the level height are "virtually well-known" (see [74] for an overview on pebbling), but we hope that our compilation is of use nevertheless, due to the somewhat scattered character of this knowledge. The relation to the "space complexity of resolution" (see [30, 86]) will be discussed in 7.2.

Let $\boldsymbol{B_k}$ the complete binary tree of height $k \in \mathbb{N}_0$ (thus $\#\mathrm{lvs}(T) = 2^k$). The following lemma states some auxiliary results on binary trees.

Lemma 4.5 1. *Consider two binary trees T, T' with $T' \leq T$.*

   (a) *$h(T') \leq h(T)$.*

   (b) *$\mathrm{peb}(T') \leq \mathrm{peb}(T)$.*

2. *$\mathrm{peb}(B_k) = h(B_k) = k + 1$.*

Proof: Part 1a is obvious, using for example the characterization of the leveled height according to Figure 4.

For part 1b note that since T' is binary, and contractions and eliminations of edges are permutable, for the process of transforming T into T' (as a *minor*) we only have to consider contractions of "singular" edges (u, v) where (at the given stage) u is the *only* predecessor of v. Now by an easy induction on the number of elimination steps one proves $\mathrm{peb}(G') \leq \mathrm{peb}(G)$ for any dag's G and G', where G' is obtained from G by eliminating edges or isolated nodes, or contracting singular edges.

Part 2 follows by an easy induction on the leveled height k. ∎

Theorem 4.6 *Consider a binary tree T.*

1. *For any $k \in \mathbb{N}_0$ we have:*

   (a) *$h(T) \leq k \Leftrightarrow B_{k+1} \not\leq T$;*

   (b) *$h(T) \geq k \Leftrightarrow B_k \leq T$.*

2. *$h(T) = \mathrm{peb}(T) - 1$.*

Proof: Part 1a, direction "$\Rightarrow$" follows by Lemma 4.5. Part 1b, direction "$\Rightarrow$" follows by induction on the composition of T:

If T is trivial, then we have $h(T) = 0$. So let T consist of two subtrees T_1, T_2 with $h(T_1) \leq h(T)-1$ and $h(T_2) \leq h(T)$. In case of $h(T_2) \leq h(T)-1$ actually we have $h(T_1) = h(T_2) = h(T)-1$, and thus we can apply the induction hypothesis to both T_1 and T_2 and get the assertion. Otherwise we have $h(T_2) = h(T)$, and now we can apply the induction hypothesis to T_2.

Part 2: Assume $h(T) > \text{peb}(T) - 1$. Thus $B_{\text{peb}(T)}$ is minor of T by part 1b, and furthermore by Lemma 4.5 we have $\text{peb}(T) \geq \text{peb}(B_{\text{peb}(T)}) = \text{peb}(T) + 1$. Thus in fact $h(T) \leq \text{peb}(T) - 1$ must hold.

Finally we prove $h(T) \geq \text{peb}(T) - 1$ by induction on the composition of T. If T is trivial, then we have $h(T) = 0$ and $\text{peb}(T) = 1$. So let T consist of two subtrees T_1, T_2 with $h(T_1) \leq h(T) - 1$ and $h(T_2) \leq h(T)$. By induction hypothesis we get

$$\begin{aligned} \text{peb}(T_1) &\leq h(T) \\ \text{peb}(T_2) &\leq h(T) + 1. \end{aligned}$$

Now by pebbling first T_2 within T using $h(T) + 1$ pebbles at most, keeping one pebble at the root of T_2, then pebbling T_1 within T using $h(T)$ pebbles at most (altogether at most $h(T) + 1$ pebbles are needed), and finally pebbling the root of T we obtain $\text{peb}(T) \leq h(T) + 1$. ∎

Immediately from part 1b of Theorem 4.6 we get the following lower bound on the number of leaves $\#\text{lvs}(T)$ for a binary tree T. The first appearance of this bound in the literature I'm aware of one finds in [32]. To point out the simplicity of the proof (which is underlying the lower bound for tree-like resolution in [21]) we give here additionally a direct proof.

Corollary 4.7 *For any binary tree T we have $h(T) \leq \log_2 \#\text{lvs}(T)$, that is $\#\text{lvs}(T) \geq 2^{h(T)}$.*

Proof: Induction on $\#\text{lvs}(T)$

In case of $\#\text{lvs}(T) = 1$ we have $h(T) = 0$. So let's assume that T consists of two subtrees T_1, T_2. W.l.o.g.: $\#\text{lvs}(T_1) \leq \#\text{lvs}(T_2)$. Now induction hypothesis applied to T_1 and T_2 yields

$$\begin{aligned} h(T_1) &\leq \log_2 \#\text{lvs}(T_1) \leq \log_2 \#\text{lvs}(T) - 1 \\ h(T_2) &\leq \log_2 \#\text{lvs}(T_2) \leq \log_2 \#\text{lvs}(T) \end{aligned}$$

and hence, by the definition of leveled height, we have $h(T) \leq \log_2 \#\text{lvs}(T)$. ∎

5 Upper bounds

For the following classes of clause-sets we give upper bounds on the hardness in this section:

1. General upper bounds for *uniquely satisfiable* clause-sets as well as for clause-sets with *very many satisfying assignments*, and also for unsatisfiable p–$\mathcal{CLS}$ are given in Subsection 5.1.
2. 2–$\mathcal{CLS} \cap \mathcal{USAT}$ and *large random* p–$\mathcal{CLS}$ one finds in Subsection 5.2.
3. In Subsection 5.3 two different forms of generalizations of Horn formulas, the *hierarchy of generalized Horn formulas* and *q-Horn formulas* are considered (first only w.r.t. unsatisfiability).
4. Finally, for the handling of *satisfiable* instances, in Subsection 5.4 the oracle $\mathcal{S}_1 = \mathcal{LSAT}$ of *linearly satisfiable clause-sets* in proposed. In this way also satisfiable q-Horn formulas (and hence 2-clause-sets and (generalized) Horn formulas) can be captured.

5.1 General upper bounds

In this subsection we derive some general upper bounds on the hardness $h(F)$ from known upper bounds on SAT decision. The focus is to point out the principal relations, while the numerical values likely are not of practical interest.

For *satisfiable* clause-sets, according to Lemma 3.16 the following cases are the extreme cases for $F \in G^1_k(\mathcal{U}, \mathcal{S})$:

(i) $r^{\mathcal{U}}_k(F) \in \mathcal{S}$ ("no guessing")

(ii) $\exists\, \varphi \in \mathcal{PASS} : n(\varphi) \leq k \wedge \varphi * F \in \mathcal{S}$ ("pure guessing").

For the basic choices $\mathcal{U} = \mathcal{U}_0$, $\mathcal{S} = \mathcal{S}_0$ this means

(i) all $F \in \mathcal{SAT}\text{-}1$ fulfilling $r_k(F) = \top$ are contained in $G^1_k(\mathcal{U}_0, \mathcal{S}_0)$;

(ii) all $F \in \mathcal{SAT}$ are in $G^1_k(\mathcal{U}_0, \mathcal{S}_0)$ which have $\varphi \in \mathrm{mod_p}(F)$ with $n(\varphi) \leq k$.

From the "satisfiability coding lemma" ([73]) we get (first) information about the level where $F \in \mathcal{SAT}\text{-}1$ will be detected (eventually):

Lemma 5.1 *For $F \in \mathcal{SAT}\text{-}1$, $F \neq \top$ there is $\varphi \in \mathcal{PASS}$ with*

$$n(\varphi) \leq (1 - \frac{1}{p(F)}) \cdot n(F) \quad \textit{and} \quad r_1(\varphi * F) = \top,$$

and thus

$$h(F) \leq (1 - \frac{1}{p(F)}) \cdot n(F) + 1. \ \blacksquare$$

Any F belonging to case (ii) has very many satisfying assignments, namely

$$|\operatorname{mod}_{\mathrm{t}}(F)| \geq 2^{n(F)-k}.$$

Now, as shown by Edward Hirsch ([46]), also the converse is true if we restrict ourselves to p–$\mathcal{CLS}$, that is, there is an upper bound on the hardness $h(F)$ depending only on the maximal clause-length $p(F)$ and the proportion δ of satisfying assignments. We make use of the following constants.

Definition 5.2 *Consider $p \in \mathbb{N}_0$. For $p \leq 1$ let $\boldsymbol{\tau_p} := 1$, and for $p \geq 2$ let $\boldsymbol{\tau_p}$ be the unique positive solution of $\sum_{i=1}^{p} x^{-i} = 1$. Furthermore $\boldsymbol{\alpha_p} := \log_2 \tau_p$.* $\blacksquare$

For example $\alpha_2 = 0$, $\alpha_3 = 0.694..$, $\alpha_4 = 0.879..$, and $\alpha_p \to 1$ for $p \to \infty$.

Lemma 5.3 *Define $k(p, \delta) := \dfrac{1 - \log_2 \delta}{\alpha_p} + p - 1$ for $0 < \delta \leq 1$ and $p \geq 2$.*

Consider $F \in \mathcal{SAT}$ with $p(F) \geq 2$. Let $\delta(F) := \dfrac{|\operatorname{mod}_{\mathrm{t}}(F)|}{2^{n(F)}}$. Now there is $\varphi \in \operatorname{mod}_{\mathrm{p}}(F)$ with $n(\varphi) \leq k(p(F), \delta(F))$, and thus

$$h(F) \leq k(p(F), \delta(F)). \ \blacksquare$$

From the well known upper bound on p–$\mathcal{CLS}$ decision, obtained independently in [69] and [64], we now immediately obtain a general upper bound on $h(F)$ for *unsatisfiable* $F \in \mathcal{USAT}$.

To capture the use of 1-clause-elimination in the algorithms from [69, 64], we call a binary tree T a ***k*-$\boldsymbol{\mathcal{U}}$-tree for $\boldsymbol{F}$** if T can be obtained from an $\mathcal{U}$-DPL-tree T_0 for F by applications of the following transitions:

- replacing node labels G by G' with $G \xrightarrow{k,\mathcal{U}}\!\!* \, G'$;
- cutting off subtrees when their root labels become an element of $\mathcal{U}$.

The subsequent lemma follows obviously from Theorem 4.3 and Corollary 4.7.

Lemma 5.4 *For any $F \in \mathcal{USAT}$ and any k-$\mathcal{U}$-DPL tree T for F we have*

$$h_{\mathcal{U}}(F) \leq h(T) + k \leq \log_2 \#\mathrm{lvs}(T) + k. \ \blacksquare$$

Using the clarified upper bound methods presented in [59, 60, 56], from [69, 64] on gets in fact the existence of a 1-$\mathcal{U}_0$-DPL-tree T for $F \in \mathcal{CLS}$ with

$$\#\mathrm{lvs}(T) \leq \frac{2}{\tau_{p(F)-1}} \cdot \tau_{p(F)-1}^{n(F)}.$$

Lemma 5.5 *For $F \in \mathcal{USAT}$ we have $h(F) \leq 2 + \alpha_{p(F)-1} \cdot (n(F) - 1)$.* ∎

For $F \in 2\text{–}\mathcal{CLS} \cap \mathcal{USAT}$ we get $h(F) \leq 2$. A direct proof for this one finds in Lemma 5.6.

Whether the improved 3-SAT algorithms in [78, 94, 56, 79, 59] can be simulated by tree-like resolution (and thus also yield upper bounds on the hardness) is not known yet.

5.2 2-CNF and large random k-CNF

Lemma 5.6 *For $F \in 2\text{–}\mathcal{CLS} \cap \mathcal{USAT}$ we have $h(F) \leq 2$. More precisely, in case of $F \neq \{\bot\}$ there is $v \in \mathrm{var}(F)$ with $r_1(\langle v \to \varepsilon\rangle * F) = \{\bot\}$ for $\varepsilon \in \{0, 1\}$.*

Proof: If for $\varphi \in \mathcal{PASS}$ and $F \in 2\text{–}\mathcal{CLS}$ we have $\varphi * F \in \mathcal{CLS}^+$, then φ is an autarky for F. Thus for any variable $v \in \mathrm{var}(F)$ either there is $\varepsilon \in \{0, 1\}$ with $r_1(\langle v \to \varepsilon\rangle * F) \subseteq F$ and hence we found a reduction $F \overset{\text{sat}}{\equiv} r_1(\langle v \to \varepsilon\rangle * F)$, or v is as required (which eventually must be the case). ∎

Now let's have a look at random $p\text{–}\mathcal{CLS}$. For $n, c \geq 0$ let

$$p\text{–}\mathcal{CLS}(n, c) := \{\, F \in \mathcal{CLS} : n(F) = n \,\wedge\, c(F) = c \,\wedge\, \forall C \in F\, [\,|C| = p\,]\,\}$$

and consider the uniform distribution (all members of $p\text{–}\mathcal{CLS}(n, c)$ are equally likely). From Theorem 2 in [3] it follows that for $n \to \infty$ and arbitrary $c \geq 8n^2$ for any $F \in 3\text{–}\mathcal{CLS}(n, c)$ and any variable $v \in \mathrm{var}(F)$ the (random) clause-sets

$$F_\varepsilon := (\langle v \to \varepsilon\rangle * F) \,\cap\, 2\text{–}\mathcal{CLS}$$

fulfill $c(F_\varepsilon) \geq 2n(F_\varepsilon)$ and (thus) are unsatisfiable for both $\varepsilon \in \{0, 1\}$ with probability tending to 1. We conclude $h(F) \leq 3$, and by Lemma 3.18 in fact $h(F) = 3$. More generally (without going into details), it is not hard to generalize this result to the following lemma.

Lemma 5.7 *For all $p \geq 3$ there are constants $K_p > 0$ such that for all $\varepsilon > 0$ there is n_0 such that for all $n \geq n_0$ and all $K_p \cdot n^{p-1} \leq c \leq 2^p \cdot \binom{n}{p}$ with probability greater than $1 - \varepsilon$ any $F \in p\text{–}\mathcal{CLS}(n, c)$ is unsatisfiable and fulfills $h(F) = p$.* ∎

5.3 Generalizations of Horn formulas

We recall the result of [45], that Unit-clause propagation finds a contradiction for F iff F contains an unsatisfiable renamable Horn formula:

Lemma 5.8 *For $F \in \mathcal{USAT}$ we have $h(F) \leq 1$ iff there is $F' \subseteq F$ with $F' \in \mathcal{RHO} \cap \mathcal{USAT}$.* ∎

[37] introduced a hierarchy of generalized Horn clause-sets, which, following the notation in [14][7)], is given as follows:

Definition 5.9 $\boldsymbol{H_1} := \mathcal{HO}$ *is the set of all Horn clause-sets. For* $k > 1$ *we have* $\boldsymbol{F} \in \boldsymbol{H_k}$ *if and only if there is an ordering*

$$F = \{ C_1, \dots, C_m \}$$

of the clauses, and there are sets

$$\emptyset \subseteq V_1 \subseteq \cdots \subseteq V_m \subseteq \mathcal{VA}$$

of variables, such that

$$\{ C_i \setminus V_i : 1 \leq i \leq m \} \in H_{k-1}$$

holds. ∎

Lemma 5.10 *Consider* $F \in H_k \setminus H_{k-1}$ *for* $k > 1$ *and* $n(F) \neq 0$ *together with variable sets* V_i *according to Definition 5.9. Choosing* $v \in V_i$ *for the minimal index* i *with* $V_i \neq \emptyset$ *we have*

$$\langle v \rightarrow 1 \rangle * F \in H_{k-1} \quad \textit{and} \quad \langle v \rightarrow 0 \rangle * F \in H_k.$$

By Lemmas 3.5 and 5.8 it follows that for all $k \geq 1$ *and* $F \in H_k \cap \mathcal{USAT}$ *we have* $h(F) \leq k$.

If $\mathcal{HO}^+ \subseteq \mathcal{S}$, *then* $\mathcal{HO} \subseteq G_1(\mathcal{U}_0, \mathcal{S})$ *and thus Lemma 3.5 yields that now for all* $F \in H_k$ *we have* $h_{\mathcal{U}_0, \mathcal{S}}(F) \leq k$.

(Actually then $H_k \subseteq \# G_k(\mathcal{U}_0, \mathcal{S})$ *holds.)* ∎

Some remarks:

1. Trivially we get $\mathcal{HO}^+ \subseteq \mathcal{S}$ by choosing for $\mathcal{S}$ the set of clause-sets F satisfiable by $\langle v \rightarrow 0 : v \in \mathcal{VA} \rangle$. However in this way we do not get stability under renaming of the classes $G_k(\mathcal{U}_0, \mathcal{S})$.

2. Choosing $\mathcal{S} = \mathcal{RHO} \cap \mathcal{SAT}$ as (essentially) in [75] ensures stability under renaming. However, to us the larger class $\mathcal{LSAT}$ introduced in the next subsection seems to be a more natural choice, since $\mathcal{LSAT}$ does not depend on accidental syntactical properties.

[7)] [14] also extended the levels of the hierarchy in the natural way to cover $\mathcal{CLS}$ completely while [37] considered only satisfiable clause-sets

3. In order to cover fast 2-SAT decision, [24] uses reduction by autarkies if one is encountered to build up a generalization of the hierarchy (H_k).[8)] We did not use this sort of "intermediate autarky testing" here because of the lack of canonicity (at this time).

In [11] the class of "q-Horn formulas" has been introduce, further explored in [12, 13, 33, 35, 89].

Definition 5.11 *The class* $\boldsymbol{\mathcal{QHO}(C_0)}$ *of "q-Horn clause-sets w.r.t. pattern* C_0*" for a clause* C_0 *is the class of all clause-sets* $F \in \mathcal{CLS}$*, such that for each clause* $C \in F$ *either* $|C \setminus C_0| \leq 1$ *holds, or* $|C \setminus C_0| = 2$ *is the case, and then* $C \cap \overline{C_0} = \emptyset$ *must hold.*

The class of ***q-Horn clause-sets*** *is defined as*

$$\boldsymbol{\mathcal{QHO}} := \bigcup_{C_0 \in \mathcal{CL}} \mathcal{QHO}(C_0). \blacksquare$$

Immediately

$$\mathcal{QHO}(\bot) = 2\text{–}\mathcal{CLS}$$

follows, and for a Horn clause-sets we have

$$F \in \mathcal{HO} \Rightarrow F \in \mathcal{QHO}(\overline{\mathrm{var}(F)}).$$

Since $\mathcal{QHO}$ is stable under renaming, also $\mathcal{RHO} \subset \mathcal{QHO}$ holds. As shown in [11, 12], recognition of $\mathcal{QHO}$ as well as SAT decision can be done in linear time.

From the results of [89] it follows, that for $F \in \mathcal{QHO}$ in case of $F \in \mathcal{SAT}$ we have $h_{\mathcal{U}_0,\mathcal{S}_1}(F) \leq 1$, where $\mathcal{S}_1$ is the class of "linearly satisfiable clause-sets" introduced in the next subsection, and in case of $F \in \mathcal{USAT}$ one gets $h(F) \leq 2$. We give an alternative proof of these facts, based on the above characterization of $\mathcal{QHO}$, while [89] uses the "complexity index" from [12].

Lemma 5.12 *1.* $\mathcal{QHO}$ *is stable under partial assignments.*

2. $F \in \mathcal{QHO}(C_0)^+ \Rightarrow \varphi_{\overline{C_0}}$ *is an autarky for* F *with* $\varphi_{\overline{C_0}} * F \in 2\text{–}\mathcal{CLS}$.

3. $F \in \mathcal{QHO} \cap \mathcal{USAT} \Longrightarrow h(F) \leq 2$.

4. Consider $\mathcal{S}$ *containing all* $F \in \mathcal{SAT}$ *with the property, that there is a satisfying assignment* $\varphi \in \mathrm{mod_p}(F)$ *such that for all clauses* $C \in F$ *there is at most one literal* $x \in C$ *with* $\varphi(x) = 0$.

Then $\mathcal{QHO}^+ \cap \mathcal{SAT} \subseteq \mathcal{S}$ *and thus* $h_{\mathcal{U}_0,\mathcal{S}}(F) \leq 1$ *for* $F \in \mathcal{QHO} \cap \mathcal{SAT}$.

[8)]However going back to [31], where fast 2-SAT decision by the autarky argument is presented, not referring to [69] (their algorithm in fact has exponential running time on $2\text{–}\mathcal{CLS}$ as shown in [61]).

Proof: Parts 1 and 2 follow immediately from the definitions.

Part 3: By part 1 we have $r_1(F) \in \mathcal{QHO}$. By part 2 there is $F' \subseteq r_1(F)$ with $F' \in 2\text{–}\mathcal{CLS} \cap \mathcal{USAT}$. Lemma 5.6 gives $h(F') \leq 2$, and thus we conclude $h(r_1(F)) \leq h(F') \leq 2 \Rightarrow h(F) \leq 2$.

Part 4: Note that the autarky $\varphi_{\overline{C_0}}$ from part 2 has the property that in each affected clause there is at most one falsified literal, and that the same property also holds for any autarky of a 2-clause-set. ∎

In the subsequent subsection we introduce a polynomially decidable class of satisfiable clause-sets fulfilling the condition from Part 4 of Lemma 5.12.

5.4 Linearly satisfiable clause-sets for enhanced satisfiability detection

Lemma 3.20 shows drastically the weakness of algorithm $\mathcal{D}^*$ for satisfiability detection if using $\mathcal{S} = \mathcal{S}_0$. So it seems to me, while the choice of $\mathcal{U}_0$ is a very natural candidate for the basic unsatisfiable cases, *for satisfiability additional algorithmic resources* are necessary.

Motivated by the following "duality result" in [58] (in its turn motivated by [38]), I propose the use of *autarkies* as the fundamental mechanism for completing the above approach:

> For each clause-set F and each clause $C \in F$ either there is a *resolution refutation* of F using C or there is an *autarky* φ satisfying C (but not both).

Now as an interesting (polynomial time) oracle $\mathcal{S}_1$ for satisfiability detection at level 0 we regard

$$\boldsymbol{\mathcal{S}_1} := \mathcal{LSAT}.$$

Definition 5.13 *[58] A partial assignment φ is called a* ***linear autarky*** *for $F \in \mathcal{CLS}$ if there is a "weight function"*

$$w : \mathcal{VA} \to \mathbb{Q}_{>0}$$

such that for all $C \in F$ we have

$$\sum_{x \in C, \varphi(x)=1} w(\operatorname{var}(x)) \geq \sum_{x \in C, \varphi(x)=0} w(\operatorname{var}(x)).$$

The set $\boldsymbol{\mathcal{LSAT}}$ *of* ***linearly satisfiable clause-sets*** *is the set of all clause-sets satisfiable by a linear autarky.* ∎

Note that a linear autarky in fact is an autarky. In [58] it is shown that reduction of $F \in \mathcal{CLS}$ by linear autarkies can be performed in polynomial time and results in an unique linear-autarky-free sub-clause-set $F_{\mathrm{la}} \subseteq F$. We have $F \in \mathcal{LSAT}$ iff $F_{\mathrm{la}} = \top$, and thus $\mathcal{LSAT}$ is polynomially decidable.

It is obvious from the definition of $\mathcal{LSAT}$ that

- $\mathcal{LSAT}$ is stable under enforced assignments as well as under renamings, and allows substitution;
- $\mathcal{LSAT}$ contains all satisfiable clause-sets having a satisfying (partial) assignment falsifying at most one literal per clause, and thus by Lemma 5.12 we have $\mathcal{QHO}^+ \cap \mathcal{SAT} \subseteq \mathcal{LSAT}$ (and also $2\text{–}\mathcal{CLS} \cap \mathcal{SAT} \subseteq \mathcal{LSAT}$).

Furthermore it is shown in [58] that all "matched clause-sets" introduced in [35] are contained in $\mathcal{LSAT}$. Lemmata 5.10 and 5.12 immediately yield the following assertions (using also Lemma 3.6, part 2).

Lemma 5.14 *1. $G_k(\mathcal{U}_0, \mathcal{S}_1)$ is stable under renaming for all $k \geq 0$.*

2. $F \in H_k$ for $k \geq 1 \Rightarrow h_{\mathcal{U}_0,\mathcal{S}_1}(F) \leq k$.

3. $F \in \mathcal{QHO} \cap \mathcal{SAT} \Rightarrow h_{\mathcal{U}_0,\mathcal{S}_1}(F) \leq 1$. ∎

The problem whether for $F \in \mathcal{CLS}$ there is a renaming σ with $\sigma(F) \in H_k$ has been shown NP-complete in [29] for $k \geq 2$ (while for $k = 1$ the problem is poly-time ([2])). In contrast, the classes $G_k(\mathcal{U}_0, \mathcal{S}_1)$ are stable under renaming, and thus contain all $\sigma(F)$ for any $F \in H_k$ and any renaming σ. So it seems to me that the classes H_k simply are too small, due to the focus on (simple) syntactic structures defining these classes.

6 Lower bounds

Using the (easy) general lower bound method from Subsection 3.4.2, we obtain lower bounds on the hardness $h(F)$ for the *pigeonhole formulas* (here actually we can determine the hardness exactly) and the *"pebbling formulas"* in this section.[9] The relation to the complexity of resolution will be discussed in some depth in Section 9.

[9] It seems to me also worthwhile to give a nice lower bound on the hardness for Tseitin's graph formulas ([87]): To describe the effect of splitting here is (again) easy. See [67] for a treatment of tree-like resolution.

6.1 The (weak) pigeonhole formulas

The determination of hardness for the pigeonhole formulas (recall the definition in Subsection 2.3) is fairly easy, due to their simple recursive structure:

Lemma 6.1 *Consider $m > k \geq 1$ and a variable $v \in \mathrm{var}(\mathrm{PHP}^m_k)$.*

1. *There are partial assignments $\varphi_0, \varphi_1 \in \mathcal{PASS}$ such that for $\varepsilon \in \{0,1\}$ we have*
$$\varphi_\varepsilon * \big(\langle v \to \varepsilon\rangle * \mathrm{PHP}^m_k\big) = \mathrm{PHP}^{m-1}_{k-1} \bmod \textit{renaming}.$$
2. *φ_1 can be chosen as $\varphi_1 = \varphi_1^1 \cup \varphi_1^2$ where φ_1^1 corresponds to 1-clause-eliminations (and thus gives a $\xrightarrow{1,\mathcal{U}_0}$-reduction) and φ_1^2 corresponds to eliminations of pure literals (and thus is an autarky).*

Proof: For $\varepsilon = 1$ the variables in the same column as v are eliminated by 1-clause-eliminations, while for the variables v' in the same row as v the literals $\overline{v'}$ become pure.

For $\varepsilon = 0$ note that the case $k = 1$ is trivial, while for $k \geq 2$ just choose a variable v' in the same row as v and apply the case $\varepsilon = 1$ to v'. ∎

Lemma 6.2 *For all $m > k \geq 0$ we have $h(\mathrm{PHP}^m_k) = k$.*

Proof: The lower bound $h(\mathrm{PHP}^m_k) \geq k$ follows immediately from Lemma 6.1, part 1 and Lemma 3.17.

For the upper bound $h(\mathrm{PHP}^m_k) \leq k$ consider the classes
$$S_k := \{\, \varphi * \mathrm{PHP}^m_k : m > k \geq 0 \wedge \varphi \in \mathcal{PASS} \,\}$$
and recall the discussion from Subsection 3.4.1. For $F \in S_k$ and $v \in \mathrm{var}(F)$ by definition of S_k we have $\langle v \to 0\rangle * F \in S_k$. So it remains to consider $\langle v \to 1\rangle * F$. Assume $F = \varphi * \mathrm{PHP}^m_k$ and $v \in \mathrm{var}(F) \subseteq \mathrm{var}(\mathrm{PHP}^m_k)$ and let φ_1 be as in part 2 of Lemma 6.1.

If φ and φ_1^1 are incompatible, then $\bot \in \langle v \to 1\rangle * F$. Otherwise one can cancel those (left) clauses in $\langle v \to 1\rangle * F$ which would be canceled in PHP^m_k by φ_1^2 and obtain $F' \subseteq \langle v \to 1\rangle * F$ with
$$F' \xrightarrow{1,\mathcal{U}_0}\!{}^* \ \varphi * \big(\varphi_1 * (\langle v \to 1\rangle * \mathrm{PHP}^m_k)\big) = \varphi * \mathrm{PHP}^{m-1}_{k-1} \bmod \mathrm{renaming}.$$
Hence $\langle v \to 1\rangle * F$ can be transformed by renaming and elimination of clauses into an element of S_{k-1} mod r_1. ∎

6.2 The pebbling formulas

In [5] the following class of formulas, called "pebbling formulas" here, has been introduced, generalizing a construction of [8], and in fact adapting a construction from [51] (Theorem 3.1.13). The reader might recall Subsection 2.4.

Definition 6.3 *Consider a dag G with unique output $o(G) \in V(G)$. Assume for each node $w \in V(G)$ distinct variables $w_0, w_1 \in \mathcal{VA}$:*

$$\mathbf{PF}(\boldsymbol{G}) := \langle o(G)_0 \to 0, o(G)_1 \to 0 \rangle * \\ \big\{ \{\overline{v_{\varepsilon(v)}}\}_{v \in \mathrm{dp}(w)} \cup \{w_0, w_1\} : w \in V(G) \wedge \varepsilon \in \{0,1\}^{\mathrm{dp}(w)} \big\}. \blacksquare$$

The following properties of $\mathrm{PF}(G)$ are obvious from the definition (using $d(G)$ for the maximal *in-degree* of G):

- $\mathrm{PF}(G) \in \mathcal{USAT}$ (due to the assignment $\langle o(G)_0 \to 0, o(G)_1 \to 0 \rangle$),
- $n(\mathrm{PF}(G)) = 2|V(G)| - 2$,
- $c(\mathrm{PF}(G)) \leq |V(G)| \cdot 2^{d(G)}$,
- $p(\mathrm{PF}(G)) \leq d(G) + 2$.

To describe the effect of splitting on a variable for $\mathrm{PF}(G)$ we need the following definition.

Definition 6.4 *For $W \subseteq V(G)$ let $\boldsymbol{b(W)}$ be the set of nodes $v \in V(G)$ "below W", that is, for which there is a (directed) path (possibly trivial) from v to some member of W.*

For $w \in V(G)$, $w \neq o(G)$ let

- $\boldsymbol{G_{w \to 0}}$ *be the subgraph of G induced by $b(\{w\})$;*
- $\boldsymbol{G_{w \to 1}}$ *be the subgraph of G induced by $b\big(V(G) \setminus b(\{w\})\big) \setminus \{w\}$.* $\blacksquare$

Lemma 6.5 *Consider $w \in V(G)$, $w \neq o(G)$ and $\varepsilon \in \{0,1\}$.*

1. *There is $\varphi \in \mathcal{PASS}$ with*

$$\varphi * (\langle w_\varepsilon \to 1 \rangle * \mathrm{PF}(G)) = \mathrm{PF}(G_{w \to 1}).$$

2. *There are $\varphi, \varphi' \in \mathcal{PASS}$ with*

$$\varphi * (\langle w_\varepsilon \to 0 \rangle * \mathrm{PF}(G)) = \mathrm{PF}(G_{w \to 0}),$$
$$\varphi' * (\langle w_\varepsilon \to 0 \rangle * \mathrm{PF}(G)) = \mathrm{PF}(G_{w \to 1}).$$

Proof: For Part 1 we can choose φ in an obvious way as an autarky for $\langle w_\varepsilon \to 1\rangle * \mathrm{PF}(G)$ (with $\langle w_{\overline{\varepsilon}} \to 0\rangle \subseteq \varphi$), and for Part 2 we can choose φ as an autarky for $\langle w_\varepsilon \to 0\rangle * \mathrm{PF}(G)$ (with $\langle w_{\overline{\varepsilon}} \to 0\rangle \subseteq \varphi$), while for φ' we use part 1 with $\langle w_{\overline{\varepsilon}} \to 1\rangle$. ∎

To ensure that pebblings of $G_{w\to 0}$ and $G_{w\to 1}$ can be combined to a pebbling of G we need the following (obvious) auxiliary lemma.

Lemma 6.6 *Consider $w \in V(G)$, $w \neq o(G)$ and $\varepsilon \in \{0,1\}$.*

1. *For $v \in V(G_{w\to\varepsilon})$ all direct predecessors of v in G are also contained in $G_{w\to\varepsilon}$ with the only exception that for $\varepsilon = 1$ and $v \in \mathrm{ds_G}(w)$ the direct predecessor w of v is missing in $G_{w\to 1}$.*
2. *$V(G_{w\to 0}) \cup V(G_{w\to 1}) = V(G)$.* ∎

Lemma 6.7 *$h(\mathrm{PF}(G)) \geq \mathrm{peb}(G)$.*

Proof: Consider $w \in V(G)$, $w \neq o(G)$. By Lemma 3.17 and Lemma 6.5 we have to show

$$\mathrm{peb}(G_{w\to 1}) \geq \mathrm{peb}(G) - 1 \quad \text{or} \quad \mathrm{peb}(G_{w\to 0}) \geq \mathrm{peb}(G).$$

So assume $\mathrm{peb}(G_{w\to 1}) \leq \mathrm{peb}(G) - 2$ and $\mathrm{peb}(G_{w\to 0}) \leq \mathrm{peb}(G) - 1$. Consider the following pebbling of G (possible due to Lemma 6.6) which uses at most $\mathrm{peb}(G) - 1$ pebbles and thus yields a contradiction:

First pebble the nodes $V(G_{w\to 0})$ by an optimal pebbling for $G_{w\to 0}$ using at most $\mathrm{peb}(G) - 1$ pebbles, and keep one pebble on w (but delete the rest). Now pebble the nodes $V(G_{w\to 1})$ using at most $\mathrm{peb}(G) - 2 + 1$ pebbles by an optimal pebbling for $G_{w\to 1}$. ∎

7 Tree-like resolution with oracles

The close relation between the hierarchies $(G_k^0(\mathcal{U}))_{k\in\mathbb{N}_0}$ of unsatisfiable clause-sets and tree-like resolution is the subject of this section. Leveled (and generalized) resolution calculi $\overset{\mathcal{U},k}{\vdash}$ are introduced allowing exactly falsification of the formulas in $G_k^0(\mathcal{U})$. The (minimal) level k ($= h_{\mathcal{U}}(F)$) yields "quasi-tight" bounds on the size of tree-like resolution proofs.

1. In Subsection 7.1 the hierarchies $\overset{\boldsymbol{\mathcal{U},k}}{\vdash}$ of **$\boldsymbol{k}$-times nested input resolution**, using $\mathcal{U}$ for the axioms, are introduced and their basic properties are given. Ignoring the levels, $\overset{\mathcal{U}}{\vdash}$ is just tree-like resolution using oracle $\mathcal{U}$.

2. In 7.2 the **correspondence** between $\overset{\mathcal{U}}{\vdash}$-proofs and $\mathcal{U}$-DPL-trees as well as the equivalence

$$F \overset{\mathcal{U},k}{\vdash} \bot \Longleftrightarrow h_{\mathcal{U}}(F) \leq k$$

are shown. The *different characterizations* of the hardness $h_{\mathcal{U}}(F)$ obtained in this paper are discussed.

3. In 7.3 the "quasi-tight" bounds

$$2^{h_{\mathcal{U}}(F)} \leq \mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(F) \leq 2^{\log_2(n(F)+1)\cdot h_{\mathcal{U}}(F)}$$

are derived, where $\mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(F)$ is the minimal size of a tree-like resolution refutation using oracle $\mathcal{U}$.

4. Finally one finds in Subsection 7.4 the **"quasi-automatization"** of the calculi $\overset{\mathcal{U}}{\vdash}$ (measuring the tree-complexity) by algorithm $\mathcal{D}^*$ from Subsection 3.2. (In this sense the hierarchies $(G_k^0(\mathcal{U}))_{k\in\mathbb{N}_0}$ contain all clause-sets refutable by "short" tree-like resolution proofs (and for $\mathcal{U} = \mathcal{U}_0$ exactly those are included).)

7.1 Nested input resolution with oracles

Definition 7.1 *For $F \in \mathcal{CLS}$ and a clause C:*

$$\boldsymbol{F} \overset{\mathcal{U},0}{\vdash} \boldsymbol{C} :\Leftrightarrow \varphi_C * F \in \mathcal{U}.$$

Generally we have $\boldsymbol{F} \overset{\mathcal{U},k}{\vdash} \boldsymbol{C}$ for $k \in \mathbb{N}_0$ if there is a resolution tree T with

$$\boldsymbol{T : F} \overset{\mathcal{U},k}{\vdash} \boldsymbol{C},$$

which in turn is fulfilled if $h(T) \leq k$ holds (recall equation 2 in Subsection 4.2) and there is a clause-set $F_0 \in \mathcal{CLS}$ with $T : F_0 \vdash C$ such that for all $C \in F_0$ we have $F \overset{\mathcal{U},0}{\vdash} C$. We use $\overset{\mathcal{U}}{\vdash}$ instead of $\overset{\mathcal{U},k}{\vdash}$ in case the parameter k does not matter.

Finally for $F \in \mathcal{USAT}$:

$$\mathbf{Comp}_{\mathbf{tR}(\mathcal{U})}(\boldsymbol{F}) := \min\{\#\mathrm{lvs}(T) \mid T : F \overset{\mathcal{U}}{\vdash} \bot\}. \blacksquare$$

In order to understand the transformations used in the subsequent subsection *in detail*, the following three easy auxiliary lemmata are needed.

The first lemma states that shortening the axioms or applying a partial assignment to the set of axioms as well as "undoing" such an assignment can only simplify resolution proofs. (With a little bit more care one shows that this holds not only for the "exterior" structure of resolution proofs as trees, but also with respect to the labeling by clauses, that is, the number of different clauses can also be preserved.)

Lemma 7.2 *For any clause-sets $F, F' \in \mathcal{CLS}$, clauses C, resolution trees T and partial assignments $\varphi \in \mathcal{PASS}$ we have*

1. *If $T : F \vdash C$, and for each $C \in F$ there is $C' \in F'$ with $C' \subseteq C$, then there is $T' \leq T$ with $T' : F' \vdash C' \subseteq C$.*
2. *If $T : F \vdash C$ and $\varphi * \{C\} \neq \top$ ($\Leftrightarrow C_\varphi \cap \overline{C} = \emptyset$) then there is $T' \leq T$ with $T' : \varphi * F \vdash C' \subseteq C \setminus C_\varphi$.*
3. *If $T : \varphi * F \vdash C$, then there is $T' \cong T$ with $T' : F \vdash C' \subseteq C \cup C_\varphi$.* ∎

Next the basic properties of the relation $\overset{\mathcal{U},0}{\vdash}$ are given. (For part 4 stability of $\mathcal{U}$ under partial assignments is used.)

Lemma 7.3 *For clause-sets $F \in \mathcal{CLS}$, partial assignments $\varphi \in \mathcal{PASS}$ and clauses C, C' we have*

1. (a) *$F \overset{\mathcal{U}_0,0}{\vdash} C \Leftrightarrow \exists C_0 \in F : C_0 \subseteq C$,*

   (b) *$F \overset{\mathcal{USAT},0}{\vdash} C \Leftrightarrow F \models C$;*
2. *for $C_\varphi \cap C = \emptyset$: $\varphi * F \overset{\mathcal{U},0}{\vdash} C \Leftrightarrow F \overset{\mathcal{U},0}{\vdash} C_\varphi \cup C$;*
3. *$F \overset{\mathcal{U},0}{\vdash} C \Leftrightarrow \varphi_C * F \overset{\mathcal{U},0}{\vdash} \bot$;*
4. *$F \overset{\mathcal{U},0}{\vdash} C$ and $C \subseteq C' \Rightarrow F \overset{\mathcal{U},0}{\vdash} C'$.* ∎

Finally the basic properties of $\overset{\mathcal{U},k}{\vdash}$ are stated. For the proofs use inductions on the composition of resolution trees and the two previous lemmas.

Lemma 7.4 *For clause-sets $F \in \mathcal{CLS}$, clauses C, resolution trees T, $k \in \mathbb{N}_0$ and partial assignments $\varphi \in \mathcal{PASS}$ we have*

1. (a) *If $T : F \overset{\mathcal{U}_0}{\vdash} C$ then there is $T' \leq T$ with $T' : F \vdash C' \subseteq C$.*

   (b) *And if $T : F \vdash C$, then also $T : F \overset{\mathcal{U}_0}{\vdash} C$.*
2. *If $T : F \overset{\mathcal{U}}{\vdash} C$, then for any clause $C' \supseteq C$ there is $T' \leq T$ with $T' : F \overset{\mathcal{U}}{\vdash} C'$.*
3. *$F \overset{\mathcal{U}}{\vdash} C \Leftrightarrow F \models C$.*
4. *If $T : F \overset{\mathcal{U}}{\vdash} C$ and $\varphi * F \neq \top$, then there is $T' \leq T$ with $T' : \varphi * F \overset{\mathcal{U}}{\vdash} C \setminus C_\varphi$.*

5. *If* $T : \varphi * F \overset{\mathcal{U}}{\vdash} C$ *and* $C \cap C_\varphi = \emptyset$*, then there is* $T' \leq T$ *with* $T' : F \overset{\mathcal{U}}{\vdash} C_\varphi \cup C$*.*

6. $F \overset{\mathcal{U},k}{\vdash} C \Leftrightarrow \varphi_C * F \overset{\mathcal{U},k}{\vdash} \bot$. ∎

Using $\mathcal{U}_0$, the first level of generalized tree-like resolution is exactly *input resolution*, that is we have

$$F \overset{\mathcal{U}_0,1}{\vdash} C \iff F \vdash C' \subseteq C \text{ by input resolution.}$$

And regarding Figure 4 (Subsection 4.2) we see that generally $F \overset{\mathcal{U},k}{\vdash} C$ holds iff there is a *k-times nested input resolution tree* deriving C where the clauses at the leaves are derived from F using $\overset{\mathcal{U},0}{\vdash}$.

Ignoring the levels k, the calculi $\overset{\mathcal{U}}{\vdash}$ are just tree-like resolution using oracle $\mathcal{U}$ for the axioms. Especially we have

$$\mathrm{Comp}_{\mathrm{tR}(\mathcal{U}_0)} = \mathrm{Comp}_{\mathrm{tR}}\,.$$

7.2 The correspondence to our hierarchies

Theorem 7.5 *Consider* $F \in \mathcal{USAT}$*.*

1. *Every* $\mathcal{U}$*-DPL-tree* T *for* F *can be transformed into a (regular) resolution tree* $T' : F \overset{\mathcal{U}}{\vdash} \bot$ *with* $T' \leq T$*. If* T *is of minimal size, then* $T' \cong T$*.*

2. *Vice versa, each resolution tree* $T : F \overset{\mathcal{U}}{\vdash} \bot$ *can be transformed into an* $\mathcal{U}$*-DPL-tree* T' *for* F *with* $T' \leq T$*. If* T *is of minimal size, then* $T' \cong T$*.*

3. $\mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(F) = \min\{\#\mathrm{lvs}(T) : T$ *is* $\mathcal{U}$*-DPL-tree for* $F\}$.

4. $h_{\mathcal{U}}(F) = \min\{h(T) : T : F \overset{\mathcal{U}}{\vdash} \bot\} = \min\{k : F \overset{\mathcal{U},k}{\vdash} \bot\}$.

5. *Finally for any clause* C *we have* $F \overset{\mathcal{U},k}{\vdash} C$ *iff* $h_{\mathcal{U}}(\varphi_C * F) \leq k$ *holds.*

Proof: Part 1: Induction on the composition of T

If T is trivial, then $F \in \mathcal{U}$, and thus $F \overset{\mathcal{U},0}{\vdash} \bot$. So assume T consists of two subtrees T_0, T_1 w.r.t. splitting variable v.

Applying the induction hypothesis to T_0 and T_1 we obtain resolution trees $T'_\varepsilon \leq T_\varepsilon$ with

$$T'_\varepsilon : \langle v \to \varepsilon \rangle * F \overset{\mathcal{U}}{\vdash} \bot$$

for $\varepsilon \in \{0,1\}$. By Lemma 7.4, part 5 we obtain $T''_\varepsilon \leq T'_\varepsilon$ with

$$T''_0 : F \overset{\mathcal{U}}{\vdash} \{v\}, \quad T''_1 : F \overset{\mathcal{U}}{\vdash} \{\overline{v}\}$$

and combining them we get T' as required.

Remark: If for the constructions in Lemma 7.4 one does not use weakening (part 2), then here it is also possible that already T''_0 or T''_1 derives $\bot$ — as we will discuss in Remark 2 below this situation corresponds exactly to what is known as "intelligent backtracking" (for ordinary DPL- and resolution trees).

Part 2: Induction on the composition of T

If T is trivial, then $F \overset{\mathcal{U},0}{\vdash} \bot$, and thus $F \in \mathcal{U}$. So assume T consists of two subtrees T_0, T_1 with resolution variable v.

By Lemma 7.4, part 4 we obtain $T'_\varepsilon \leq T_\varepsilon$ with

$$T'_\varepsilon : \langle v \to \varepsilon \rangle * F \overset{\mathcal{U}}{\vdash} \bot$$

for $\varepsilon \in \{0,1\}$. Induction hypothesis yields $\mathcal{U}$-DPL-trees $T''_\varepsilon \leq T'_\varepsilon$ for $\langle v \to \varepsilon \rangle * F$. In case of $v \notin \mathrm{var}(F)$ (which may happen due to weakening) choose one of them, while otherwise combining them gives T' as required.

Part 3 follows directly from parts 1, 2. For Part 4 use Theorem 4.3, while Part 5 follows from part 4 and Lemma 7.4, part 6. ∎

Remarks:

1. Parts 1 to 3 generalize the equivalence between *semantic trees*, *tree-like resolution* and *branching trees for the search problem* (see [88, 63]).

2. The process of cutting off unnecessary branches in the proof for Part 1 (without using weakening as mentioned there) can already by applied to the computation of T itself, when traversing the tree in depth-first order:

   For each node w of the $\mathcal{U}$-DPL-tree its corresponding clause $C(w)$ can be computed without much additional work (only the working path has to be considered, and thus also space remains polynomial), and in case we get $v \notin \mathrm{var}(C(w'))$ for one direct successor w' of w where v is the splitting variable at w, then the other branch for the second direct successor w'' of w does not need to be processed.

   In case of (pure) tree-like resolution this technique is known as *"intelligent backtracking"* (see [92]) or *"conflict directed backtracking"* (see [82]), while the enhancement of DPL-algorithms by adding the derived clauses $C(w)$ to the input formula in the process of evolving the DPL-tree (as done in [82, 92] up to a certain clause-length) actually enables the DPL-solver to gain (in principal) the power of *regular resolution* (here with oracles $\mathcal{U}$).

3. The easiest (non-trivial) case of a tree of level two is a tree consisting just of two input resolution trees (thus having the form of a V). According to Lemma 5.6 this special case suffices for unsatisfiable 2-clause-sets, while by Lemma 5.12 for unsatisfiable q-Horn clause-sets “Y-proofs” are required, that is, additionally an initial segment of input resolution is needed.

   “V-proofs” are responsible for the strengthened reductions from Subsection 3.5. For example the reduction $F \xrightarrow{2,\mathcal{U}} \langle v \to \overline{\varepsilon}\rangle * F$ is applicable iff $\langle v \to \varepsilon\rangle * F \overset{\mathcal{U},1}{\vdash} \bot$, while $F \xrightarrow[+]{2,\mathcal{U}} \langle v \to \overline{\varepsilon}\rangle * F$ is applicable iff $\langle v \to \varepsilon\rangle * F \vdash \bot$ by such a “V-proof” (using $\mathcal{U}$).

4. From Lemma 5.10 we get that for unsatisfiable generalized Horn formulas $F \in H_k$ we have $F \overset{\mathcal{U}_0,k}{\vdash} \bot$, strengthening [14] who used a form of width-restricted resolution (see Subsection 8.1). In [91] “LLRI resolution” (“linear layered resolution”), a restriction of linear resolution[10)], has been introduced to proof unsatisfiability for H_2. If $F \vdash \bot$ by LLRI resolution then in fact $h(F) \leq 2$.

7.2.1 The space complexity of (tree-like) resolution

In [30] the concept of *space complexity for resolution* has been introduced (refining [15]). The minimal space $\mathbf{space_R(\boldsymbol{F})}$ needed for a resolution refutation of F is the minimum of $\max_{1 \leq i \leq m} c(F_i)$ over all sequences $(F_1, \dots, F_m)$ of clause-sets fulfilling

- $F_1 \subseteq F$, $\bot \in F_m$;
- for each $2 \leq i \leq m$ there is exactly one $C \in F_i \setminus (F_{i-1} \cup F)$, for which again there must be clauses $D, E \in F_{i-1}$ such that C is the resolvent of D, E.

$\mathrm{space_R}(F)$ is equal to the minimum of $\mathrm{peb}(G)$ for resolution *dag's* G deriving $\bot$ from F [11)]. Allowing only resolution *trees*, they arrive at $\mathbf{space_{tR}(\boldsymbol{F})}$ which is the minimum of $\mathrm{peb}(T)$ for $T : F \vdash \bot$. By Theorems 7.5, 4.6 and 4.3 we get

$$\mathrm{space_{tR}}(F) = h(F) + 1$$

10) Let $\mathcal{P} = (C_1, \dots, C_m)$ be a linear resolution proof of $\bot$ from F, that is, for $1 < i \leq m$ the clause C_i is either resolvent of C_{i-1} with an input clause from F or with an ancestor clause C_j for $j < i - 1$. Consider the list $L = (j_1, \dots, j_k)$ of indices j of ancestor clauses C_j used in ancestor resolution steps (in order of appearance). Now $\mathcal{P}$ is LLRI iff we have $j_r \leq j_{r+1}$ for $1 \leq r < k$, and furthermore the sequence $(C_{j_1}, \dots, C_{j_k})$ of ancestor clauses fulfills $C_{j_r} \supseteq C_{j_{r+1}}$.

11) if using resolution *trees* to define $\mathrm{space_R}(F)$, one pebble on a node pebbles at the same time all other nodes with the same (clause) label

for all $F \in \mathcal{USAT}$.

How to characterize $\mathrm{space}_{\mathrm{R}}(F)$ and how to relate it to $\mathrm{space}_{\mathrm{tR}}(F)$ and $\mathrm{Comp}_{\mathrm{R}}(F)$ is not known yet[12)]. Lower bounds for Tseitin's formulas and the Pigeonhole formulas on $\mathrm{space}_{\mathrm{R}}$ are given in [86].

7.2.2 Compilation of the various interpretation for the hardness

Altogether we gained the following characterizations of the hardness $h_{\mathcal{U},\mathcal{S}}(F)$ for clause-sets $F \in \mathcal{CLS}$ (the first two hold also for satisfiable clause-sets, the last three only for unsatisfiable clause-sets):

1. $h_{\mathcal{U},\mathcal{S}}(F)$ is the first level of success of algorithm $\mathcal{D}^*$ from Subsection 3.2.
2. $h_{\mathcal{U},\mathcal{S}}(F)$ is the minimal leveled height $h_{\mathcal{U},\mathcal{S}}(T)$ of an $(\mathcal{U},\mathcal{S})$-DPL-tree T (Theorem 4.3).
3. $h_{\mathcal{U}}(F) + 1$ is the minimal pebbling complexity of an $\mathcal{U}$-DPL-tree T (Theorem 4.6).
4. $h_{\mathcal{U}}(F)$ is the minimal level such that nested input resolution $\overset{\mathcal{U},k}{\vdash}$ derives $\bot$ (Theorem 7.5).
5. $h_{\mathcal{U}}(F)+1$ is the minimal pebbling complexity of a resolution tree $T : F \overset{\mathcal{U}}{\vdash} \bot$ or the minimal space complexity of such a derivation.

7.3 Nearly precise general upper and lower bounds for tree-like resolution (with oracles)

Since *regular* resolution trees T have depth at most $n(T)$, the number of leaves of T can be bound by $h(T)$.

Lemma 7.6 *For every regular resolution tree $\#\mathrm{lvs}(T) \le (n(T) + 1)^{h(T)}$ holds.*

Proof: Induction on $h(T)$

In case of $h(T) = 0$ we have $\#\mathrm{lvs}(T) = 1$. So assume $h(T) > 0$. Consider T_0 with $h(T) \le 1$ and $\#\mathrm{lvs}(T_0) =: m$, and consider subtrees $T_1, \dots, T_m$ of T with $h(T_i) \le h(T) - 1$ such that replacing the leaves of T_0 by the T_i yields T.

Since T is regular, we have $m \le n(T) + 1$. Now by induction hypothesis

$$\begin{aligned} \#\mathrm{lvs}(T) &= \sum_{i=1}^{m} \#\mathrm{lvs}(T_i) \le \sum_{i=1}^{m} (n(T_i) + 1)^{h(T_i)} \le \sum_{i=1}^{n(T)+1} (n(T) + 1)^{h(T)-1} \\ &= (n(T) + 1)^{h(T)}. \blacksquare \end{aligned}$$

12) according to [85], Theorem 6 in [30] is false

Any resolution tree T can be transformed into a regular one ([87]). This can be seen by Theorem 7.5, parts 2 and 1 (transforming T into a DPL-tree and back — the regularization is hidden in Lemma 7.4, part 4), or directly by the following procedure:

Find a minimal irregular subtree $T' : F \vdash C'$ of T. Consider a resolution step in T' with resolution variable $v \in \mathrm{var}(C')$, and cut off that branch contributing the sign of v opposite to the sign of v in C. Simplify the remainder of T (by cutting off further branches) to obtain a resolution proof T_1. Repeat the process with T_1 until a regular tree T_m is obtained.

Since this procedure only prunes the resolution tree, the following lemma is obvious.

Lemma 7.7 *For every resolution tree $T : F \overset{\mathcal{U}}{\vdash} C$ there is a regular resolution tree $T' : F \overset{\mathcal{U}}{\vdash} C$ with $T' \leq T$.* ▮

(However, the number of (different) clauses in T is not preserved by this transformation. In [40] in fact a super-polynomial separation between regular resolution and (full) resolution is shown.)

Using Corollary 4.7, Theorem 7.5 and Lemmata 7.6, 7.7 (together with Lemma 4.5, part 1a) we get

Theorem 7.8 *For any clause-set $F \in \mathcal{USAT}$ the following relations between the hardness $h_{\mathcal{U}}(F)$ and the complexity of tree-like resolution (using $\mathcal{U}$) hold:*

$$2^{h_{\mathcal{U}}(F)} \leq \mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(F) \leq 2^{\log_2(n(F)+1) \cdot h_{\mathcal{U}}(F)}$$
$$\frac{\log_2 \mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(F)}{\log_2(n(F)+1)} \leq h_{\mathcal{U}}(F) \leq \log_2 \mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(F).$$ ▮

Since the case of tree-like resolution is of special importance, we spend a corollary on it:

Corollary 7.9 $F \in \mathcal{USAT} \Rightarrow 2^{h(F)} \leq \mathrm{Comp}_{\mathrm{tR}}(F) \leq 2^{\log_2(n(F)+1) \cdot h(F)}$. ▮

Similar bounds are mentioned (without proofs) in [81]. The lower bound $2^{h(F)}$ on the complexity of tree-like resolution can not be improved without using further data of F (see Lemma 3.19 and use the general upper bound $\mathrm{Comp}_{\mathrm{tR}}(F) \leq 2^{n(F)}$ for $F \in \mathcal{USAT}$).

In Section 9 we will give a compilation of our results on the complexity of tree-like resolution, and there also the relation to the lower bound on tree-like resolution from [5] (using width-restricted resolution) is discussed.

7.4 Quasi-automatizing tree-like resolution (with oracles)

In [10] (motivated by [21]) the notion of *automatizable proof systems* S has been introduced, which is a proof system (here for unsatisfiable clause-sets) admitting to find a S-refutation in deterministic *polynomial time* in the size of the *shortest* S-refutation, and it is shown that unless factoring is feasible Frege systems are not automatizable (for further information in that direction see [1]).

In [5] the question is raised whether (at least) tree-like resolution is *quasi-automatizable*, that is, whether a tree-like resolution refutation can be found in time quasi-polynomial in the size of the shortest tree-like resolution refutation. In fact if using $\mathcal{U}$ indeed as *oracle* (counting a request "$F \in \mathcal{U}$?" as one step), then tree-like resolution with arbitrary oracle $\mathcal{U}$ is quasi-automatizable:

Theorem 7.10 *If counting a request to oracle $\mathcal{U}$ as one step, the running time of SAT decision algorithm $D^*_{\mathcal{U}}$ (see Subsection 3.2) is bounded by*

$$O\big(\ell(F) \cdot 2^{\log_2(n(F)+1) \cdot 2h_{\mathcal{U}}(F)}\big) \leq O\big(\operatorname{Comp}_{\mathrm{tR}(\mathcal{U})}(F)^{2\log_2(n(F)+1)}\big).$$

Only impairing the running time of $D^*_{\mathcal{U}}$ by a constant factor, in fact a resolution tree $T : F \overset{\mathcal{U}}{\vdash} \bot$ can be computed with $h(T) = h_{\mathcal{U}}(T)$ and

$$\#\mathrm{lvs}(T) \leq 2^{\log_2(n(F)+1) \cdot h_{\mathcal{U}}(F)} \leq \operatorname{Comp}_{\mathrm{tR}(\mathcal{U})}(F)^{\log_2(n(F)+1)}.$$

Thus tree-like resolution with oracles $\mathcal{U}$ is quasi-automatizable. If the class $\mathcal{U}$ is polynomially decidable, then in fact tree-like resolution with oracle $\mathcal{U}$ is quasi-automatizable in the non-relativized sense (counting each step), where for $\mathcal{U}$ other than $\mathcal{U}_0$ we just consider the polynomial decision algorithm for $\mathcal{U}$ as proof system for the axioms of the resolution proofs.

As a special case we obtain quasi-automation of tree-like resolution by algorithm $D^*_{\mathcal{U}_0}$.

Proof: The bounds on running time follow by Lemma 3.10 and Theorem 7.8, From a successful computation of $D^*_{\mathcal{U}}$ on input F we easily extract a $\mathcal{U}$-DPL-tree for F, which is transformed into $T : F \overset{\mathcal{U}}{\vdash} \bot$ by Theorem 7.5, part 1. ∎

8 Width-restricted resolution

The subject of this section are resolution restrictions regarding the *length of clauses*. It is well known that *input resolution and 1-clause resolution* (at least one parent clause is always a 1-clause) have the same power w.r.t. refutation ([20]). Is there a *generalization for nested input resolution* (with oracles) as defined in Section 7.1 ?!

In Subsection 8.1 the two known variants of width-restricted resolution are discussed: The well-known "bounded resolution" introduced in [36] (*all* occurring clauses must not surmount the size limit), and the less known form introduced in [14] (for each resolution step *at least one* parent clause obeys the size limit). Since both forms have their drawbacks (the first can not handle Horn formulas, while for the stronger form it is not known, whether derivability of the empty clause for fixed size limit is poly-time decidable), a new variant (with oracles) is introduced in Subsection 8.2 (which in fact is a (slight) *strengthening* of the second form).

We show that the corresponding hierarchies $(W_k(\mathcal{U}))_{k \in \mathbb{N}_0}$ of unsatisfiable clause-sets subsume the hierarchies $(G_k^0(\mathcal{U}))_{k \in \mathbb{N}_0}$, while the reversal is essentially false.

In Subsection 8.3, Theorem 8.10 an analogue to the *bounds* on tree-like resolution in Theorem 7.8, this time for *full resolution*, is proved, generalizing the lower bound from [5] to the use of oracles (and removing the dependency on the maximal input clause length).

Motivated by [9], in Subsection 8.4 a width-lower bound for Krishnamurthy's formulas ([52]) is given, showing that the general lower bound for full resolution is *tight* for this example. The new form of width here pays off in allowing us to use the *original* formulas with their long clauses and applying just some elementary graph theoretic observations (while [9] had to translate them into 3-CNF).

8.1 The known width restrictions

In the literature there exist two different forms of "k-clause resolution." The most popular version, introduced in [36] as "bounded resolution" and further elaborated for example in [39, 35, 5], allows only resolution steps where both parent clauses as well as the resolvent have length at most k. The corresponding hierarchy of unsatisfiable clause-sets is defined as follows.

Definition 8.1 *For $k \in \mathbb{N}_0$ let $\boldsymbol{W'_k}$ be the set of clause-sets refutable by resolution proofs $T : F \vdash \bot$ using only clauses of length $\leq k$, that is $\mathrm{cl}(T) \in k\text{-}\mathcal{CLS}$. The corresponding hardness parameter is called*

$$\mathbf{width}'(\boldsymbol{F}) := \inf\{\, k \in \mathbb{N}_0 : F \in W'_k \,\}. \quad \blacksquare$$

This form of restricted resolution has the following nice properties ($k \in \mathbb{N}_0$):

1. For constant k the classes W'_k are polynomially decidable (by computing the closure under resolutions where both parent clauses and the resolvent are of length at most k).

2. The classes W'_k are stable under partial assignments, renaming and formation of super-clause-sets. Furthermore the hierarchy is strict.

3. In [5] (translating [21] into resolution) the general lower bound for full resolution
$$\mathrm{Comp}_{\mathrm{R}}(F) \geq 2^{\Omega(\frac{(\mathrm{width}'(F)-p(F))^2}{n(F)})}$$
for $F \in \mathcal{USAT}$ has been shown, while for tree-like resolution one has
$$\mathrm{Comp}_{\mathrm{tR}}(F) \geq 2^{\mathrm{width}'(F)-p(F)}.$$

4. Already in [36] a sequence (F_n) of unsatisfiable clause-sets $F_n \in 3\text{–}\mathcal{CLS}$ with $n(F_n) = n$, $c(F_n) \leq 8n$ and
$$\mathrm{width}'(F_n) \geq \frac{n}{6(\log_2 n)^2}$$
has been given.

However, as a main drawback the fact has to be considered, that, due to
$$\mathrm{width}'(F) \geq p(F)$$
for minimal unsatisfiable F, *no level* W'_k contains $\mathcal{HO} \cap \mathcal{USAT}$. (It is the same (avoidable, as we will see) sensitivity on the input clause-length $p(F)$ which spoils somewhat the lower bounds under 3.)

Another form of "k-clause resolution" has been introduced in [14], directly generalizing 1-clause resolution by requiring (only) that for all resolution steps one parent clause has to be of length at most k:

Definition 8.2 *For $k \in \mathbb{N}_0$ the classes $\boldsymbol{W''_k}$ contain those clause-sets F having a resolution refutation $T : F \vdash \bot$ such that for each resolution in T (at least) one parent clause has length $\leq k$. By* **width$''(F)$** *the minimal $k \in \mathbb{N}_0$ with $F \in W''_k$ for $F \in \mathcal{USAT}$ is denoted.* ∎

Here we have the following properties:

1. W''_k is stable under partial assignments, renaming and formation of super-clause-sets. Furthermore the hierarchy is strict.

2. $W''_1 = G^0_1(\mathcal{U}_0)$.

3. W''_2 is polynomially decidable as shown in [16].

4. $H_k \subseteq W''_k$ for all $k \geq 0$ (recall Definition 5.9), which was he original motivation in [14] to introduce this resolution restriction.

5. Obviously we have $\mathrm{width}''(F) \leq \mathrm{width}'(F)$ for all $F \in \mathcal{USAT}$, while Lemma 8.5 in the next subsection shows $\mathrm{width}'(F) - p(F) \leq \mathrm{width}''(F)$.

However, it is not known whether the classes W''_k for $k \geq 3$ can be decided in polynomial time.

In the next subsection we propose a strengthened variant of width restricted resolution, combining all above (positive) properties, and including furthermore our hierarchies $G^0_k(\mathcal{U})$.

8.2 A new variant of width restricted resolution

The (obvious) decision procedure for "$F \in W'_k$?" extends F in a series of steps $F = F_0, F_1, \dots, F_m$ by adding a clause C_i of length $|C_i| \leq k$ to F_i which is derivable from F_i by a *single resolution step*:

$$\exists\, D_1, D_2 \in F_i : C_i \text{ is resolvent of } D_1, D_2.$$

Now we are eliminating the limiting dependency on $p(F)$ and get at the same time the generalization for using oracles (in a poly-time manageable way) by admitting such clauses C_i with $|C_i| \leq k$ which are derivable by *generalized input resolution*:

$$F_i \overset{\mathcal{U},1}{\vdash} C_i.$$

Definition 8.3 *Suppose $\mathcal{U}$ is stable under formation of super-clause-sets. For $F \in \mathcal{CLS}$ and $k \in \mathbb{N}_0$ consider a sequence $F = F_0, F_1, \dots, F_m$ of clause-sets such that for $0 \leq i < m$ there are clauses $C_i \notin F_i$ with $F_{i+1} = F_i \cup \{C_i\}$ fulfilling*

$$|C_i| \leq k,\ \mathrm{var}(C_i) \subseteq \mathrm{var}(F),\ F_i \overset{\mathcal{U},\min(k,1)}{\vdash} C_i$$

while for F_m no such C_m exists (obviously $m \leq \sum_{i=0}^{k} 2^i \binom{n(F)}{i} = O(n^k)$). Due to the additional assumption on $\mathcal{U}$, $\boldsymbol{W^{\mathcal{U}}_k(F)} := F_m$ is uniquely determined. Now let $\boldsymbol{W_k(\mathcal{U})}$ be the set of clause-sets F with $\bot \in W^{\mathcal{U}}_k(F)$. The corresponding hardness parameter is called

$$\boldsymbol{\mathrm{width}_{\mathcal{U}}(F)} := \inf\{\, k \in \mathbb{N}_0 : F \in W_k(\mathcal{U}) \,\}.$$

We use $\boldsymbol{\mathrm{width}(F)} := \mathrm{width}_{\mathcal{U}_0}(F)$. ∎

In Lemma 8.7, part 6b we present a more elegant characterization of $W^{\mathcal{U}}_k(F)$ without the case distinction between $k = 0$ and $k \geq 1$ as in Definition 8.3.

Again all $W_k(\mathcal{U})$ are stable under partial assignments, renaming and formation of super-clause-sets. The first two levels of the hierarchy $(W_k(\mathcal{U}))_{k \in \mathbb{N}_0}$ are the same as for $(G^0_k(\mathcal{U}))_{k \in \mathbb{N}_0}$:

$$W_0(\mathcal{U}) = G^0_0(\mathcal{U}) = \mathcal{U},\ W_1(\mathcal{U}) = G^0_1(\mathcal{U}).$$

In order to emphasize the close relation especially to the second form of width-restricted resolution considered in the previous subsection, we also relativize this hierarchy:

Definition 8.4 *Suppose $\mathcal{U}$ is stable under formation of super-clause-sets. For $k \in \mathbb{N}_0$ let $\boldsymbol{W''_k(\mathcal{U})}$ be the set of clause-sets $F \in \mathcal{USAT}$ such that there is $T : F \overset{\mathcal{U}}{\vdash} \bot$ where for all resolution steps in T at least one parent clause is of length at most k. For $F \in \mathcal{USAT}$ let*

$$\mathbf{width}''_{\boldsymbol{\mathcal{U}}}\boldsymbol{(F)} := \inf\{\, k \in \mathbb{N}_0 : F \in W''_k(\mathcal{U})\,\}. \quad ■$$

Obviously we have $\mathrm{width}''_{\mathcal{U}_0}(F) = \mathrm{width}''(F)$.

Lemma 8.5 *For any $F \in \mathcal{USAT}$ and any $k \in \mathbb{N}_0$ we have*

1. (a) $W'_0 = \mathcal{U}_0$, $G^0_1(\mathcal{U}_0) \cap k\text{-}\mathcal{CLS} \subseteq W'_k$;
   (b) $\mathrm{width}'(F) - p(F) \le \mathrm{width}(F) \le \mathrm{width}'(F)$.
2. (a) $W''_0(\mathcal{U}) = G^0_0(\mathcal{U}) = \mathcal{U}$, $W''_1(\mathcal{U}) = G^0_1(\mathcal{U})$;
   (b) $W''_k(\mathcal{U}) \subseteq W_k(\mathcal{U}) \subseteq W''_{k+1}(\mathcal{U})$;
   (c) $\mathrm{width}_{\mathcal{U}}(F) \le \mathrm{width}''_{\mathcal{U}}(F) \le \mathrm{width}_{\mathcal{U}}(F) + 1$.

Proof: Part 1a: If $F \in G^0_1(\mathcal{U}_0)$ then F is refutable by 1-clause-elimination, yielding a resolution proof using only clauses of length $\le p(F)$.

Part 1b: $\mathrm{width}(F) \le \mathrm{width}'(F)$ is obvious from the definitions. To prove $\mathrm{width}'(F) \le \mathrm{width}(F) + p(F)$, let $k := \mathrm{width}(F)$ and assume $k \ge 1$. Consider the sequence of F_i and C_i from Definition 8.3: $F_i \overset{\mathcal{U}_0,1}{\vdash} C_i \Rightarrow \varphi_{C_i} * F_i \in G_1(\mathcal{U}_0)$ $\Rightarrow \varphi_{C_i} * F_i \in W'_{p(F)}$, and thus there is $T : F_i \vdash C' \subseteq C_i$ with $\mathrm{cl}(T) \in (p(F)+k)\text{-}\mathcal{CLS}$.

Part 2a is obvious from the definitions. Part 2c is just a reformulation of part 2b. For Part 2b first consider $F \in W''_k(\mathcal{U})$, and let $T : F \overset{\mathcal{U}}{\vdash} \bot$ be as in Definition 8.4. Assume there is a clause $C \in \mathrm{cl}(T)$ with $|C| \le k$ and $C \notin W^{\mathcal{U}}_k(F)$, and choose C such that the corresponding subtree T' of T with $T' : F \overset{\mathcal{U}}{\vdash} C$ is minimal, and thus

$$(*) \quad \forall\, D \in \mathrm{cl}(T') \setminus \{C\} : |D| \le k \Rightarrow D \in W^{\mathcal{U}}_k(F).$$

By induction on the composition of T' we prove that in fact for all $D \in \mathrm{cl}(T')$ we have $W^{\mathcal{U}}_k(F) \overset{\mathcal{U},1}{\vdash} D$, yielding the contradiction $C \in W^{\mathcal{U}}_k(F)$:

If D is a leaf of T', then $F \overset{\mathcal{U},0}{\vdash} D$. If D is the resolvent of clauses D_1, D_2, then w.l.o.g. $|D_1| \le k$, and thus by $(*)$ we have $D_1 \in W^{\mathcal{U}}_k(F)$, while induction hypothesis yields $W^{\mathcal{U}}_k(F) \overset{\mathcal{U},1}{\vdash} D_2$. We conclude $W^{\mathcal{U}}_k(F) \overset{\mathcal{U},1}{\vdash} D$. $\surd$

Finally consider $F \in W_k(\mathcal{U})$ for $k \geq 1$ and the sequences of F_i and C_i from Definition 8.3. Using part 2a we have $\varphi_{C_i} * F_i \in W''_1(\mathcal{U})$, and thus (using the construction from Lemma 7.4, part 5) there is $T_i : F_i \overset{\mathcal{U}}{\vdash} C_i$ such that for each resolution step in T_i one parent clause is of length at most $1 + |C_i| \leq 1 + k$. All T_i together yield $F \in W''_{1+k}(\mathcal{U})$. ∎

Lemma 8.6 *Using $\mathcal{U}$ as oracle, decision of "$F \in W_k(\mathcal{U})$?" can be performed in time $O\big(n^{3k+2} + \ell(F) \cdot n^{2k+2} + 1\big)$.*

Proof: Using the notations of Definition 8.3, we have $m \leq O(n^k)$, while for each F_i up to $O(n^k)$ many possible clauses C_i have to be tested, and the decision "$F_i \overset{\mathcal{U},1}{\vdash} C_i$?" can be performed in time $O(\ell(F_i) \cdot n(F)^2 + 1)$, where one can estimate $\ell(F_i) \leq \ell(F) + O(n(F)^k)$. ∎

Comparing this result to the unknown status of poly-time decidability of the classes $W''_k \subseteq W_k(\mathcal{U}_0)$ for $k \geq 3$ we see that the reason for the problems with deciding W''_k is not the strength of these classes but the lack of some refutation power to round up the refutation process.

The basic technical properties of the hierarchies $(W_k(\mathcal{U}))_{k \in \mathbb{N}_0}$ are given in the next lemma.

Lemma 8.7 *For clause-sets $F, F_1, F_2 \in \mathcal{CLS}$, partial assignments $\varphi \in \mathcal{PASS}$ with $\mathrm{var}(\varphi) \subseteq \mathrm{var}(F)$, literals x and numbers $k, k_1, k_2 \in \mathbb{N}_0$ we have:*

1. *$W_k^{\mathcal{U}} : \mathcal{CLS} \to \mathcal{CLS}$ is a closure operator:*
   - *(a) $F \subseteq W_k^{\mathcal{U}}(F)$,*
   - *(b) $F_1 \subseteq F_2 \Rightarrow W_k^{\mathcal{U}}(F_1) \subseteq W_k^{\mathcal{U}}(F_2)$,*
   - *(c) $W_k^{\mathcal{U}}(W_k^{\mathcal{U}}(F)) = W_k^{\mathcal{U}}(F)$.*
2. *(a) $\varphi * F \in W_k(\mathcal{U}) \Rightarrow C_\varphi \in W_{k+n(\varphi)}^{\mathcal{U}}(F)$,*
   - *(b) $C \in W_k^{\mathcal{U}}(F) \Rightarrow \varphi_C * F \in W_k(\mathcal{U})$.*
3. *If $k \geq 1$, $\{x\} \in W_k^{\mathcal{U}}(F)$ and $\langle x \to 1 \rangle * F \in W_k(\mathcal{U})$ then $F \in W_k(\mathcal{U})$.*
4. *$G_{k_1}^0(W_{k_2}(\mathcal{U})) \subseteq W_{k_1+k_2}(\mathcal{U})$.*
5. *(a) $G_k^0(\mathcal{U}) \subseteq W_k(\mathcal{U})$,*
   - *(b) $\mathrm{width}_{\mathcal{U}}(F) \leq h_{\mathcal{U}}(F)$.*
6. *(a) If $|C| \leq k$, $\mathrm{var}(C) \subseteq \mathrm{var}(F)$ and $W_k^{\mathcal{U}}(F) \overset{\mathcal{U},k-|C|}{\vdash} |C|$ hold, then also $C \in W_k^{\mathcal{U}}(F)$.*

(b) $W_k^{\mathcal{U}}(F)$ *is the smallest clause-set* $\hat{F} \in \mathcal{CLS}$ *with* $F \subseteq \hat{F}$ *and*

$$\forall\, C \in \mathcal{CL} : \mathrm{var}(C) \subseteq \mathrm{var}(F) \wedge h_{\mathcal{U}}(\varphi_C * \hat{F}) + |C| \le k \Longrightarrow C \in \hat{F}.$$

Proof: Part 1 follows by definition (for part 1b use stability of $\mathcal{U}$ under formation of super-clause-sets).

For Part 2a use Lemma 7.4, part 5, and for Part 2b use Lemma 7.4, part 4.

Part 3: We have $W_k^{\mathcal{U}}(\langle x \to 1\rangle * F) \subseteq W_k^{\mathcal{U}}(F)$ since $\langle x \to 0\rangle * W_k^{\mathcal{U}}(F) \in \mathcal{U}$ and thus $\langle x \to 1\rangle$ is enforced.

Part 4: First we prove by induction on $n(F)$ that

$$\forall\, k \in \mathbb{N}_0 : F \in G_1^0(W_k(\mathcal{U})) \Rightarrow F \in W_{k+1}(\mathcal{U}).$$

The case $n(F) = 0$ is trivial. According to the definition of G_1, for $n(F) > 0$ there is a literal x with $\mathrm{var}(x) \in \mathrm{var}(F)$ and

$$\langle x \to 0\rangle * F \in W_k(\mathcal{U}), \tag{1}$$
$$\langle x \to 1\rangle * F \in G_1^0(W_k(\mathcal{U})). \tag{2}$$

From (1) we get by part 2a

$$\{x\} \in W_{k+1}^{\mathcal{U}}(F),$$

while from (2) by induction hypothesis

$$\langle x \to 1\rangle * F \in W_{k+1}(\mathcal{U})$$

follows, and thus $F \in W_{k+1}(\mathcal{U})$ by part 3.

We conclude $G_1^0(W_k(\mathcal{U})) \subseteq W_{k+1}(\mathcal{U})$. Now the assertion of part 4 follows by induction on k_1: The case $k_1 = 0$ is trivial, while for $k_1 > 0$ we have

$$G_{k_1}^0(W_{k_2}(\mathcal{U})) = G_1^0(G_{k_1-1}^0(W_{k_2}(\mathcal{U}))) \subseteq G_1^0(W_{k_1-1+k_2}(\mathcal{U})) \subseteq W_{k_1+k_2}(\mathcal{U}).$$

Part 5a follows immediately from part 4 (with $k_1 := k$ and $k_2 := 0$), and Part 5b is just a reformulation of part 5a.

Part 6a: $W_k^{\mathcal{U}}(F) \overset{\mathcal{U},k-|C|}{\vdash} C$ is equivalent to $\varphi_C * W_k^{\mathcal{U}}(F) \in G_{k-|C|}^0(\mathcal{U})$. Now by parts 5a and 2a we obtain

$$\varphi_C * W_k^{\mathcal{U}}(F) \in W_{k-|C|}^0(\mathcal{U}), \Rightarrow C \in W_k^{\mathcal{U}}(W_k^{\mathcal{U}}(F)) = W_k^{\mathcal{U}}(F).$$

Finally Part 6b is a direct conclusion from part 6a. ■

The other direction of Lemma 8.7, part 5 does not hold, which can be concluded from Theorems 13, 14 in [8] or from Corollary 6.1 in [5].[13] We strengthen the argumentation by using the concept of hardness $h(F)$.

[13] In fact the reversal seems to be "essentially false" since width-restricted resolution does not pose structural restrictions on the shape of the derivation, and thus is inherently dag-like.

Lemma 8.8 *Consider the pebbling formulas* $\mathrm{PF}(G)$ *from Definition 6.3. Using* $d(G)$ *for the maximal in-degree of* G *we have*

$$\mathrm{width}'(\mathrm{PF}(G)) \leq 2 \cdot d(G) + 2$$

for any dag G.

According to [19], there is a sequence $(G_m)_{m \in \mathbb{N}}$ *of dag's with* $d(G_m) = 2$ *and* $\mathrm{peb}(G_m) \geq \Omega(m / \log m)$. *Thus by Lemma 6.7*

$$\mathrm{width}'(\mathrm{PF}(G_m)) \leq 6 \ll \kappa \cdot \frac{m}{\log m} \leq h(\mathrm{PF}(G_m))$$

for some constant $\kappa > 0$ *and sufficiently large* m.

(Furthermore $\mathrm{Comp}_{\mathrm{R}}(\mathrm{PF}(G_m)) \leq O(m)$ *holds.)*

Proof: A resolution refutation of $\mathrm{PF}(G)$ works as follows:

Running through the nodes of G in a "pebbling manner," consider for each node $w \neq o(G)$ the clause-set

$$\begin{aligned} F(w) \quad := \quad & \{\, \{\overline{v_{\varepsilon(v)}}\}_{v \in \mathrm{dp}(w)} \cup \{w_0, w_1\} : w \in V(G) \wedge \varepsilon \in \{0,1\}^{\mathrm{dp}(w)} \,\} \cup \\ & \{\, \{v_0, v_1\} : v \in \mathrm{dp}(w) \,\}. \end{aligned}$$

Since $F(w) \models \{w_0, w_1\}$, we have $F(w) \vdash \{w_0, w_1\}$, where any resolution proof uses trivially only clauses of length at most

$$n(F(w)) = 2 \cdot |\,\mathrm{dp}(w)| + 2.$$

For $w = o(G)$ one would obtain the clause $\{o(G)_0, o(G)_1\}$ which in fact is the empty clause here due to the assignment $\langle o(G)_0 \to 0, o(G)_1 \to 0\rangle$ used for the definition of $\mathrm{PF}(G)$. ∎

Thus we have $\mathrm{space}_{\mathrm{tR}}(\mathrm{PF}(G)) \geq \mathrm{peb}(G)$ (recall Subsection 7.2.1). Furthermore it seems to me that also $\mathrm{space}_{\mathrm{R}}(\mathrm{PF}(G))$ can not be less than $\mathrm{peb}(G)$ (at least if using the resolution refutation from the proof of Lemma 8.8), which would refute a speculation in [86] about a (close) relation between $\mathrm{space}_{\mathrm{R}}(F)$ and $\mathrm{width}(F)$.

8.3 Lower bounds for full resolution with oracles

Definition 8.9 *For* $\mathcal{U}$ *stable under formation of super-clause-sets and clause-sets* $F \in \mathcal{USAT}$ *let* $\boldsymbol{\mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(F)}$ *be the minimal number of clauses* $\#\mathrm{cl}(T)$ *for resolution trees* $T : F \overset{\mathcal{U}}{\vdash} \bot$. ∎

For all $F \in \mathcal{USAT}$ we have

$$\mathrm{Comp}_{\mathrm{R}(\mathcal{U}_0)}(F) = \mathrm{Comp}_{\mathrm{R}}(F)$$

by the standard transformation underlying Lemma 7.2, part 1.[14)]

Theorem 8.10 *For $F \in \mathcal{USAT}$ with $n(F) \neq 0$ we have*

$$\frac{1}{8}\frac{\mathrm{width}_{\mathcal{U}}(F)^2}{n(F)} < \ln \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(F) < (\ln n(F)) \cdot (\mathrm{width}_{\mathcal{U}}(F) + 1) + 2.$$

Proof: For the lower bound we have to generalize the proof from [21, 5].

Consider a fixed $d \in \mathbb{R}_{>0}$.

For a clause-set F let

$$L(F) := \{ C \in F : |C| > d \}$$

be the "large clauses" in F, and let

$$\mu(F) := \min \{ |L(\mathrm{cl}(T))| : T : F \overset{\mathcal{U}}{\vdash} \bot \}$$

be the minimal number of large clauses needed in a resolution refutation (using $\mathcal{U}$) of F. Furthermore we use

$$b(F) := \begin{cases} 0 & \text{if } 2 \cdot n(F) \leq d \\ 1 + \frac{d}{2 \cdot n(F) - d} & \text{if } 2 \cdot n(F) > d \end{cases}$$

and for $k \in \mathbb{N}_0$ let

$$\Gamma_k := \{ F \in \mathcal{USAT} : \mu(F) \leq b(F)^{k+1} \}$$

(note that $n(F) \leq d$ implies $\mu(F) = 0$ and thus $F \in \Gamma_0$).

For all $k \geq 0$ the inclusion

$$\Gamma_k \subseteq G_k^0(\Gamma_0) \tag{3}$$

holds, which is proven as follows (recall Lemma 3.5):

Consider $k \geq 1$ and $F \in \Gamma_k$. If $\mu(F) = 0$ then $F \in \Gamma_0$. So assume $\mu(F) \geq 1$.

Consider $T : F \overset{\mathcal{U}}{\vdash} \bot$ with $|L(\mathrm{cl}(T))| = \mu(F)$. Choose a literal x whose number $\#_x$ of occurrences in $L(\mathrm{cl}(T))$ is maximal and thus fulfills

$$\#_x \geq \frac{\ell(\mathrm{cl}(T))}{2 \cdot n(F)} > \frac{d \cdot \mu(F)}{2 \cdot n(F)}.$$

14) Consider $T : F_0 \vdash \bot$ such that $F \overset{\mathcal{U}_0, 0}{\vdash} D$ for all $D \in F_0$, that is, there is $C_D \in F$ with $C_D \subseteq D$. Replacing axioms $D \in F_0$ in T by C_D and simplifying the tree accordingly (in an uniform manner) we obtain $T' : F \vdash \bot$ with $\#\mathrm{cl}(T') \leq \#\mathrm{cl}(T)$.

Obviously we have $\langle x \to 0 \rangle * F \in \Gamma_k$. If $\mu(\langle x \to 1 \rangle * F) = 0$ holds true, then $\langle x \to 1 \rangle * F \in \Gamma_0$ and we are done already. So assume $\mu(\langle x \to 1 \rangle * F) \neq 0$.

Applying $\langle x \to 1 \rangle$ to T (compare Lemma 7.2, part 2) we obtain a resolution tree $T' : \langle x \to 1 \rangle * F \overset{\mathcal{U}}{\vdash} \perp$ where $\#_x$ many large clause have been eliminated. We conclude

$$\mu(\langle x \to 1 \rangle * F) \leq \mu(F) - \#_x \leq \mu(F) - \frac{d \cdot \mu(F)}{2 \cdot n(F)} =$$
$$= \mu(F) \cdot b(F)^{-1} \leq b(F)^k \leq b(\langle x \to 1 \rangle * F)^k$$

and hence (proving (3))

$$\langle x \to 1 \rangle * F \in \Gamma_{k-1}. \checkmark$$

For level 0 we have

$$\Gamma_0 \subseteq W_{\lfloor d \rfloor}(\mathcal{U}) \tag{4}$$

by the following argumentation: Consider $F \in \Gamma_0$. If $\mu(F) = 0$ then obviously $F \in W_{\lfloor d \rfloor}(\mathcal{U})$. So assume $\mu(F) \geq 1$. Thus $n(F) > d$ holds, implying $b(F) < 2$ and hence actually $\mu(F) = 1$ holds. We conclude $F \in W''_{\lfloor d \rfloor}(\mathcal{U})$ and thus by Lemma 8.5, part 2b $F \in W_{\lfloor d \rfloor}(\mathcal{U})$ is the case.

By (3), (4) and Lemma 8.7, part 4 we get for all $k \geq 0$

$$\Gamma_k \subseteq W_{\lfloor d \rfloor + k}(\mathcal{U}),$$

and thus $\mathrm{width}_{\mathcal{U}}(F) \leq \lfloor d \rfloor + \min\{\, k \in \mathbb{N}_0 : F \in \Gamma_k \}$ for all $F \in \mathcal{USAT}$. The definition of Γ_k in turn says

$$\min\{\, k \in \mathbb{N}_0 : F \in \Gamma_k \} = \begin{cases} 0 & \text{if } \mu(F) = 0 \\ \max(\lceil \log_{b(F)} \mu(F) \rceil - 1, 0) & \text{else} \end{cases},$$

and thus, using $\mu(F) \leq \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(F)$ we obtain for all $F \in \mathcal{USAT}$

$$\mathrm{width}_{\mathcal{U}}(F) \leq \inf_{0 < d < 2n(F)} \big(\, \lfloor d \rfloor + \max(\lceil \log_{b(F)} \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(F) \rceil - 1, 0)\, \big). \tag{5}$$

In order to estimate (5), consider the function

$$f(d) := d + \log_{\frac{s}{s-d}} c$$

for $0 < d < s$ and parameters $s, c \geq 2$ fulfilling $\ln c < s$ (if $\mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(F) = 1$ then $\mathrm{width}_{\mathcal{U}}(F) = 0$, while the general upper bound $\mathrm{Comp}_{\mathrm{R}}(F) \leq 2^{n(F)+1} - 1$ yields $\ln(\mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(F)) < 2 \cdot n(F)$).

We "guess" $d_0 := \sqrt{s \cdot \ln c}$.[15)]

For the second addend in $f(d)$ we get

$$\log_{\frac{s}{s-d_0}} c = \frac{\ln c}{-\ln(1-\sqrt{\frac{\ln c}{s}})} \leq \frac{\ln c}{\sqrt{\frac{\ln c}{s}}} = d_0, \tag{6}$$

using $\forall\, 0 \leq x < 1 : e^x \leq \frac{1}{1-x}$, which is equivalent to $-\ln(1-x) \geq x$.

Now finally by (5) and (6) (with $s := 2n(F)$, $c := \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(F)$ and excluding the trivial case $\mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(F) = 1$):

$$\begin{aligned} \mathrm{width}_{\mathcal{U}}(F) &\leq \lfloor d_0 \rfloor + \max(\lceil \log_{\frac{2n(F)}{2n(F)-d_0}} \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(F) \rceil - 1, 0) \\ &\leq \lfloor d_0 \rfloor + \lceil d_0 \rceil - 1 < 2 \cdot d_0 = 2\sqrt{2n(F) \cdot \ln \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(F)}, \end{aligned}$$

Easy transformations yield the lower bound of the assertion.

The upper bound is obtained as follows for $F \in \mathcal{USAT}$ (using $n := n(F)$):

$$\begin{aligned} \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(F) \leq |W^{\mathcal{U}}_{\mathrm{width}_{\mathcal{U}}(F)}(F)| \cdot (n+1) = \sum_{i=0}^{\mathrm{width}_{\mathcal{U}}(F)} 2^i \binom{n}{i} \cdot (n+1) \\ \leq (2 \cdot n^{\mathrm{width}_{\mathcal{U}}(F)} + 1) \cdot (n+1) \leq 6 \cdot n^{\mathrm{width}_{\mathcal{U}}(F)+1}. \quad \blacksquare \end{aligned}$$

8.4 The tightness of the lower bound

In [52] a sequence of unsatisfiable clause-sets, called $(\mathrm{GT}_m)_{m\in\mathbb{N}}$ in [9], has been introduced, coding

Every finite transitive directed graph without cycles of length two
must have an input node.

Definition 8.11 *For $m \in \mathbb{N}$ we define $\mathbf{GT}_m \in \mathcal{USAT}$ as follows:*

GT_m has $m \cdot (m-1)$ many variables

$$\mathrm{var}(\mathrm{GT}_m) = \{v_{i,j} : i,j \in \{1,\dots,m\} \text{ and } i \neq j\}.$$

There are three types of clauses in GT_m

$$\mathbf{GT}_m := \{\, \boldsymbol{A_{i,j,k}}, \boldsymbol{B_{i,j}}, \boldsymbol{C_j} : i,j,k \in \{1,\dots,m\} \text{ and } |\{i,j,k\}| = 3 \,\}$$

where the clauses

[15)] We have $f'(d) = 1 - \frac{\ln c}{s-d}\frac{1}{(\ln(1-\frac{d}{s}))^2}$. Doing the formulation $d \ll s$ we get $s - d \approx s$ and $-\ln(1-\frac{d}{s}) \approx \frac{d}{s}$, and thus the equation $f'(d) = 0$ becomes $1 - \frac{\ln c}{s} \cdot \frac{1}{(\frac{d}{s})^2} = 0$, $\Rightarrow d = \sqrt{s \cdot \ln c}$.

$$A_{i,j,k} := \{\overline{v_{i,j}}, \overline{v_{j,k}}, v_{i,k}\}$$

express transitivity, the clauses

$$B_{i,j} := \{\overline{v_{i,j}}, \overline{v_{j,i}}\}$$

express that there is no 2-cycle, and the clauses

$$C_j := \{v_{i,j} : i \in \{1, \dots, m\} \setminus \{j\}\}$$

state that every node j has at least one direct predecessor. ∎

Krishnamurthy's conjecture that these formula have no polynomial size resolution refutations was refuted in [83] by showing in fact

$$\mathrm{Comp}_{\mathrm{R}}(\mathrm{GT}_m) \leq O(\ell(GT_m)) = O(m^3) = O(n(\mathrm{GT}_m)^{3/2}).$$

Recently in [9] it has been proven that also several resolution restrictions admit polynomial length resolution refutations. Furthermore by splitting the long clauses $C_j \in \mathrm{GT}_m$ into 3-clauses in the standard way (using new variables) they obtain $\mathrm{MGT}_m \in 3\text{-}\mathcal{CLS}$ and they prove

$$\mathrm{Comp}_{\mathrm{R}}(\mathrm{MGT}_m) \leq O(n(\mathrm{MGT}_m)^{3/2}) \quad \text{and} \quad \mathrm{width}'(\mathrm{MGT}_m) \geq \Omega(\sqrt{n(\mathrm{MGT}_m)}).$$

Thus the general lower bound for (full) resolution from [5] (see Remark 3 in Subsection 8.1) can not be improved (much) without using further data from the formula F.

We prove the analogous result for our lower bound from Theorem 8.10, whereby now we do not have to split the long clauses (profiting from the fact that the dependency on $p(F)$ is eliminated) but can use GT_m itself, and thus the lower bound proof for the width becomes much simpler.

Lemma 8.12 *For $m \geq 3$ we have* $\mathrm{width}(\mathrm{GT}_m) \geq m - 2 \geq \frac{1}{1+o(m)}\sqrt{n(\mathrm{GT}_m)}$.

Proof: In the course of the proof only clauses C are considered with $\mathrm{var}(C) \subseteq \mathrm{var}(\mathrm{GT}_m)$. Regarding C as a set of *undirected* edges between nodes from $\{1, \dots, m\}$, let $\boldsymbol{s(C)}$ be the size of a maximal independent subset $C' \subseteq C$ ("independent" means, that C' does not contain an (undirected) circuit; in other words, $s(C)$ is the number of edges in a spanning forest for C).

Furthermore we say that C has property $\mathfrak{C}$ (written $\boldsymbol{\mathfrak{C}(C)}$) if C is a (non-disjoint) union of *signed (directed)* circuits:

Here a positive literal $v_{i,j} \in C$ corresponds to a "forward edge" from i to j, while $\overline{v_{i,j}} \in C$ corresponds to a "backward edge" from j to i.

Obviously for all possible indices i, j, k we have

$$\mathfrak{C}(A_{i,j,k}),\ \mathfrak{C}(B_{i,j}) \tag{1}$$
$$s(C_j) = m - 1. \tag{2}$$

Consider resolvable clauses D_1, D_2 with resolvent R where $D_1 \cap \overline{D_2} = \{x\}$. Trivially we have

$$s(R) \geq \max(s(C_1), s(C_2)) - 1. \tag{3}$$

The first observation is that resolution with clauses having property $\mathfrak{C}$ does not decrease s:

$$\mathfrak{C}(D_2) \Rightarrow s(R) \geq s(D_1), \tag{4}$$

since, using $\mathrm{var}(x) = v_{i_0,j_0}$, in case of $s(R) < s(D_1)$ there would be no undirected path from i_0 to j_0 by edges from R, contradicting $\mathfrak{C}(D_2)$ and $D_2 \setminus \{\overline{x}\} \subseteq R$.

The second observation is that property $\mathfrak{C}$ is hereditary:

$$\mathfrak{C}(D_1) \wedge \mathfrak{C}(D_2) \Rightarrow \mathfrak{C}(R) \tag{5}$$

which follows from the "strong elimination property of signed circuits" (see for example [6], Theorem 3.2.5 on pages 107, 108 together with pages 1 - 4) and the fact, that x is the only clashing literal of D_1, D_2.

Now consider a sequence $D_1, \ldots, D_t$ of clauses such that

$$T_i : \mathrm{GT}_m \cup \{D_1, \ldots, D_{i-1}\} \vdash D_i$$

for $1 \leq i \leq t$, where $D_t = \perp$ and the T_i are input resolution trees $(h(T_i) \leq 1)$.

Let q be the minimal index such that T_i uses at least one axiom $C_j \in \mathrm{GT}_m$. By (1) and (5) for all $1 \leq i < q$ we have $\mathfrak{C}(D_i)$. Hence from (2), (3) and (4) we get $|D_q| \geq s(D_q) \geq (m-1) - 1 = m - 2$. ∎

Thus Theorem 8.10 can not be improved by replacing the term $\mathrm{width}(F)^2$ by $\mathrm{width}(F)^{2+\varepsilon}$ for any $\varepsilon > 0$.

8.4.1 Linear resolution versus tree-like and full resolution

We conclude this section by pointing out some facts on linear resolution (which seem not to be widely known). Already from [50] it is clear that linear resolution can *linearly simulate* tree-like resolution. Explicitly proven is this fact in [90] (see also [62]). On the other side in [65] it has been "proven" that tree-like resolution can polynomially simulate linear resolution — that this proof holds only for a *restricted kind* of linear resolution has already been noticed by the author ([66]), and that in fact MGT_m have polynomial size linear resolution refutations but only (sub-)exponential tree-like refutations is shown in [9]. We here can use the simpler version, that GT_m have polynomial size linear resolution refutations ([9]) and that

$$\mathrm{Comp}_{\mathrm{tR}}(\mathrm{GT}_m) \geq 2^{m-2} \geq 2^{\Omega(\sqrt{n(\mathrm{GT}_m)})}$$

which follows from Lemma 8.12, Lemma 8.7, Part 5b and Theorem 7.8, to conclude that linear resolution is *strictly stronger* than tree-like resolution.

In [28] a (deterministic) linear time transformation $F \to F^*$ is introduced, adding one new variable and using a *tautological clause*, such that trivially $\mathrm{Comp}_{\mathrm{(t)R}}(F) \leq \mathrm{Comp}_{\mathrm{(t)R}}(F^*) \leq O(\mathrm{Comp}_{\mathrm{(t)R}}(F))$ holds and such that the complexity of linear resolution for F^* is *polynomially bounded* by $\mathrm{Comp}_{\mathrm{R}}(F^*)$. Whether the use of tautological clauses is necessary here is not known to me (as well as whether in general tautological clauses can be polynomially eliminated from linear resolution refutations).

Thus, when allowing tautological clauses, actually *any* sequence of clause-sets yielding a separation between full resolution and tree-like resolution gives rise to a separation between tree-like resolution and linear resolution. The relation between full resolution and linear resolution for the *original* clause-sets seems to be open.

9 Some final remarks on calculating the complexity of resolution

9.1 On the tightness of the general bounds for resolution

Suppose a family $(F_m)_{m \in \mathbb{N}}$ of unsatisfiable clause-sets is given, and the aim is either to prove a polynomial upper bound

$$\mathrm{Comp}(F_m) \leq \ell(F_m)^{O(1)}$$

or to prove a super-polynomial lower bound

$$\mathrm{Comp}(F_m) \geq m^{\omega(1)}$$

where $\mathrm{Comp}(F_m)$ is either tree- or dag-like resolution (with or without oracles). According to Theorems 7.8 and 8.10 (representing the state of the art w.r.t. general methods at this time) the following is known.

The tree-like case

If the hardness is bounded by a constant then the family (F_m) has polynomial tree-like resolution refutations

$$h(F_m) \leq O(1) \Rightarrow \mathrm{Comp}_{\mathrm{tR}}(F_m) \leq n(F_m)^{O(1)}.$$

If on the other side $h(F_m)$ is faster growing than $\log m$, then the family (F_m) has only super-polynomial tree-like resolution refutations

$$h(F_m) \geq \omega(\log m) \Rightarrow \mathrm{Comp}_{\mathrm{tR}}(F_m) \geq m^{\omega(1)}.$$

For the grey area in between where the hardness is not bounded but grows not faster than the logarithm we can not say anything but

$$\omega(1) \leq h(F_m) \leq O(\log m) \Rightarrow \omega(1) \leq \mathrm{Comp}_{\mathrm{tR}}(F) \leq n(F_m)^{O(\log m)}.$$

In order to estimate $h(F_m)$ the following means have been provided:

- several characterizations of the hardness have been given (see Subsection 7.2.2 for a compilation);
- in Subsection 3.4.2 a general method has been established allowing quite simple lower bounds as exemplified in Subsection 3.4.3 and Section 6,
- while a general method for upper bounds one finds in Subsection 3.4.1 with examples in Subsection 3.4.3 and Section 5.

A predecessor of the general bounds from Theorem 7.8 is the lower bound

$$2^{\mathrm{width}'(F)-p(F)} \leq \mathrm{Comp}_{\mathrm{tR}}(F)$$

for $F \in \mathcal{USAT}$ from [5], which can be improved to (see Section 8)

$$2^{\mathrm{width}(F)} \leq \mathrm{Comp}_{\mathrm{tR}}(F).$$

However in general this gives only poor bounds: See Lemma 8.8 for a family (F_m) of clause-sets with $\mathrm{width}(F_m) \leq 6$, but $h(F_m) \geq \Omega(\frac{m}{\log m})$ and thus

$$\mathrm{Comp}_{\mathrm{tR}}(F_m) \geq 2^{\Omega(\frac{m}{\log m})}.$$

The dag-like case

If the width is bounded by a constant then the family (F_m) has polynomial resolution refutations

$$\mathrm{width}(F_m) \leq O(1) \Rightarrow \mathrm{Comp}_{\mathrm{R}}(F_m) \leq n(F_m)^{O(1)}.$$

If the width is faster growing than the square root of the number of variables times $\log m$, then the family (F_m) has only super-polynomial tree-like resolution refutations

$$\mathrm{width}(F_m) \geq \omega(\sqrt{n(F_m) \cdot \log m}) \Rightarrow \mathrm{Comp}_{\mathrm{R}}(F_m) \geq m^{\omega(1)}.$$

For the grey area in between we can not say anything but

$$\omega(1) \le h(F_m) \le O(\sqrt{n(F_m) \cdot \log m}) \Rightarrow$$
$$\omega(1) \le \mathrm{Comp}_{\mathrm{tR}}(F) \le n(F_m)^{O(\sqrt{\log m} \cdot \sqrt{n(F_m)})}.$$

Compared to its predecessor from [5]

$$\Omega(\frac{(\mathrm{width}'(F) - p(F))^2}{n(F)}) \le \log_2 \mathrm{Comp}_{\mathrm{R}}(F)$$

the lower bound from Theorem 8.10 has the following advantages:

- the somewhat disturbing dependence on $p(F)$ has been eliminated (replacing "width$'(F) - p(F)$" by "width(F)");
- any oracle $\mathcal{U}$ (stable under formation of super-clause-sets) can be used.

Historical remark: After Galil in [36] it were Krishnamurthy/Moll in [53] who emphasized the importance of large clauses for the analysis of resolution proofs:

For their sequence of "Ramsey formulas" (R_m) they proved

$$\mathrm{width}'(R_m) \ge \Omega(\sqrt{n(R_m)})$$

(where $p(R_m) \le (2 + \log_2 n(R_m))^2$), and they formulated the general conjecture

$$\text{“ } \mathrm{Comp}_{\mathrm{R}}(F) \ge 2^{\mathrm{width}'(F)} \text{ ”}.$$

Unfortunately the width bound for the Ramsey formulas falls into the above grey area, so their resolution complexity is still open. The direction of the conjecture on large clauses has finally by shown correct (while the precise formulation was somewhat too optimistic).

9.2 The weak pigeonhole formulas

The pigeonhole formulas PHP_k^{k+1} have been used in [43] to show the first (sub)exponential lower bound on *full* resolution. Simplifications of his proof have been achieved in [22] and, using the proof search idea from [21], in [4]. The known results on the complexity of resolution are

1. $\mathrm{Comp}_{\mathrm{R}}(\mathrm{PHP}_k^{k+1}) \ge 2^{\Omega(k)}$ ([43]), more precisely, from [4] one can obtain $\mathrm{Comp}_{\mathrm{R}}(\mathrm{PHP}_k^{k+1}) \ge 2^{k/20}$ for $k \ge 1676$
2. $\mathrm{Comp}_{\mathrm{R}}(\mathrm{PHP}_k^{m}) \ge 2^{\Omega(k^2/m)}$ ([18])

3. $\mathrm{Comp}_{\mathrm{tR}}(\mathrm{PHP}_k^m) \geq 2^k$ ([17]).

The main remaining problem is to prove (or disprove) a super-polynomial lower bound on $\mathrm{Comp}_{\mathrm{R}}(\mathrm{PHP}_k^m)$ for $k^2/\log k < m < k^{O(1)}$. Approaches one finds in [17] and [76].

For the lower bound on tree-like resolution we have given an alternative (easy) proof in Lemma 6.2 (the lower bound) together with Corollary 7.9, just exploiting the simple recursive structure of the pigeonhole formulas. Another (simple) possibility is to use

$$2^{\mathrm{width}(F)} \leq 2^{h(F)} \leq \mathrm{Comp}_{\mathrm{tR}}(F)$$

and the following lemma.

Lemma 9.1 *For all $m, k \in \mathbb{N}_0$ we have* $\mathrm{width}(\mathrm{PHP}_k^m) \geq k - 1$.

Proof: Any clause C derived by (non-trivial) input resolution from PHP_k^m has length $|C| \geq k - 1$ since a resolvent with a "long clause" from PHP_k^m has at least $k - 1$ occupied columns, while resolution with a "short clause" does not alter the number of occupied columns. ∎

We conclude our considerations by strengthening $\mathrm{Comp}_{\mathrm{tR}}(\mathrm{PHP}_k^m) \geq 2^k$ to the case where also the "dual" clauses are present in the formulation of the pigeonhole principle.

Adding the "dual" clauses

The standard formulation of the pigeonhole principle (see Subsection 2.3) can be enhanced to include also the "dual" clauses

$$\begin{aligned} dP_k^m &:= \big\{\{v_{i,j}\}_{1 \leq i \leq m}\big\}_{1 \leq j \leq k} \\ dN_k^m &:= \big\{\{\overline{v_{i,j_1}}, \overline{v_{i,j_2}}\}\big\}_{1 \leq i \leq m,\, 1 \leq j_1 < j_2 \leq k} \end{aligned}$$

("d" for "dual") expressing that "every hole contains at least one pigeon" and "a pigeon can not be in more than one hole simultaneously."

In [55] the lower bound

$$\mathrm{Comp}_{\mathrm{R}}(\mathrm{PHP}_k^{k+1} \cup dP_k^{k+1} \cup dN_k^{k+1}) \geq 2^{k/20}$$

for $k \geq 1676$ is proven (the proof of [4] carries over), still leaving open the possibility of exponential speed-ups (that is, lowering the exponent) when adding the dual clauses. Adding only the positive dual clause in fact this is impossible, since in [17] it is shown that

$$\mathrm{Comp}_{\mathrm{R}}(\mathrm{PHP}_k^m \cup dP_k^m) \geq \mathrm{Comp}_{\mathrm{R}}(\mathrm{PHP}_k^m)/(2k^2 + 2k + 1)$$

holds for arbitrary $m > k$.

The argumentation from [17] is not transferable to the tree-like case. We will show now that in fact for tree-like resolution addition of the dual clauses does not change the complexity *at all*. One may start with the observation that no regular resolution tree inferring the empty clause from $\mathrm{PHP}^m_k \cup dP^m_k$ or $\mathrm{PHP}^m_k \cup dN^m_k$ can use the dual clauses, since no dual clause is resolvable with any original clause.

Lemma 9.2 *For every regular resolution tree $T : \mathrm{PHP}^m_k \cup dP^m_k \vdash \bot$ as well as for every regular $T : \mathrm{PHP}^m_k \cup dN^m_k \vdash \bot$ in fact we have $T : \mathrm{PHP}^m_k \vdash \bot$.* ∎

To be able to add both types of dual clauses at the same time, we need some tools for resolution trees. Fundamental is the following distributivity property.

Lemma 9.3 *Consider a resolution tree $T : F \vdash D_0$ and a clause D_1 such that D_0, D_1 are resolvable with resolvent D and resolution variable v. Let F' be F together with all resolvents on resolution variable v of C, D_1 for clauses $C \in F$.*

Then there is a resolution tree $T' : F' \vdash D' \subseteq D$ with $T' \le T$.

Proof: Induction on the composition of T. ∎

Recall that two clauses C_0, C_1 are *not resolvable* iff either they do not clash (that is, $C_0 \cap \overline{C_1} = \emptyset$), or they clash on at least two literals (that is, $|C_0 \cap \overline{C_1}| \ge 2$).

We call two *clause-sets F_0, F_1 not resolvable* if for all $C_0 \in F_0$ and $C_1 \in F_1$ the clauses C_0, C_1 are not resolvable.

Lemma 9.4 *Assume F_0, F_1 are clause-sets which are not resolvable.*

1. *Consider a resolution tree $T : F_0 \vdash D_0$ and a clause $D_1 \in F_1$ such that D_0, D_1 are resolvable with resolvent D.*

   Then there is a resolution tree $T' : F_0 \vdash D' \subseteq D$ with $T' \le T$.

2. *Consider resolutions trees $T_i : F_i \vdash D_i$ for $i = 1, 2$ such that D_0, D_1 are resolvable with resolvent D.*

   Then there is a resolution tree $T_0^+ : F_0 \vdash D' \subseteq D$ with $T_0^+ \le T_0$ or there is a resolution tree $T_1^* : F_1 \vdash D' \subseteq D$ (possibly larger than T_1).

Proof: Part 1 follows immediately from Lemma 9.3. For part 2 apply Lemma 9.3 with $T = T_0$, and obtain $T' : F_0' \vdash D' \subseteq D$ with $T_0' \le T_0$, where F_0' is F_0 together with all resolvents of C, D_1 for $C \in F_0$, such that the resolution variable is the same as for D_0, D_1.

If $T' : F_0 \vdash D' \subseteq D$ then we are done: $T^+ = T'$. Otherwise apply part 1 first to all the axioms of T', and then iteratively "bottom up" until a resolution tree $T^* : F_1 \vdash D'' \subseteq D' \subseteq D$ is obtained. ∎

Lemma 9.5 *Consider two clause-sets F_0, F_1 with $F_1 \in \mathcal{SAT}$, such that F_0 and F_1 are not resolvable. Then we have*

$$\mathrm{Comp}_{\mathrm{tR}}(F_0 \cup F_1) = \mathrm{Comp}_{\mathrm{tR}}(F_0).$$

Proof: Consider a resolution tree $T : F_0 \cup F_1 \vdash \bot$. Apply Lemma 9.4, part 2 iteratively "bottom up" to subtrees of T and replace them by the corresponding T_0^+ resp. T_1^* until a resolution tree T' is obtained with either $T : F_0 \vdash \bot$ or $T' : F_1 \vdash \bot$. Since $F_1 \in \mathcal{SAT}$, we have in fact $T : F_0 \vdash \bot$, and now Lemma 9.4, part 2 guarantees $\#\mathrm{lvs}(T') \leq \#\mathrm{lvs}(T)$. ∎

Corollary 9.6 *For any $m > k \geq 0$ adding the dual clauses to PHP_k^m does not alter the complexity with respect to tree-like resolution refutations:*

$$\mathrm{Comp}_{\mathrm{tR}}(\mathrm{PHP}_k^m \cup dP_k^m \cup dN_k^m) = \mathrm{Comp}_{\mathrm{tR}}(\mathrm{PHP}_k^m).$$

Proof: Apply Lemma 9.5 with $F_0 = \mathrm{PHP}_k^m$ and $F_1 = dP_k^m \cup dN_k^m$. ∎

Corollary 9.7 $\mathrm{Comp}_{\mathrm{tR}}(\mathrm{PHP}_k^m \cup dP_k^m \cup dN_k^m) \geq 2^k$.

Proof: By Corollary 9.6, Lemma 6.2 and Corollary 7.9. ∎

References

[1] Michael Alekhnovich, Sam Buss, Shlomo Moran, and Toniann Pitassi. Minimum propositional proof length is NP-hard to linearly approximate. Journal of Symbolic Logic, to appear, 1999.

[2] B. Aspvall. Recognizing disguised NR(1) instances of the satisfiability problem. *Journal of Algorithms*, 1:97–103, 1980.

[3] Paul Beame, Richard Karp, Toniann Pitassi, and Michael Saks. On the complexity of unsatisfiability proofs for random k-CNF formulas. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 1998.

[4] Paul Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *37th Symposium on Foundations of Computer Science (FOCS' 96)*, pages 274–282, 1996.

[5] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow — resolution made simple. Technical Report TR99-022, ECCC Electronic Colloquium on Computational Complexity, July 1999.

[6] Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M. Ziegler. *Oriented Matroids*, volume 46 of *Encyclopedia of Mathematics*. Cambridge University Press, 1993.

[7] Archie Blake. *Canonical expressions in Boolean algebra.* PhD thesis, Chicago, 1937. See [68].

[8] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. Exponential separation between restricted resolution and cutting planes proof systems. *Electronic Colloquium on Computational Complexity (ECCC)*, 035, 1998.

[9] Maria Luisa Bonet and Nicola Galesi. A study of proof search algorithms for resolution and polynomial calculus. Preprint.

[10] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. No feasible interpolation for TC^0-Frege proofs. In *38th Symposium on Foundations of Computer Science (FOCS' 97)*, pages 254–263, 1997.

[11] Endre Boros, Y. Crama, and Peter L. Hammer. Polynomial-time inference of all valid implications for Horn and related formulae. *Annals of Mathematics and Artificial Intelligence*, 1:21–32, 1990.

[12] Endre Boros, Y. Crama, Peter L. Hammer, and M. Saks. A complexity index for satisfiability problems. *SIAM Journal on Computing*, 23(1):45–49, February 1994.

[13] Endre Boros, Peter L. Hammer, and Xiaorong Sun. Recognition of q-Horn formulae in linear time. *Discrete Applied Mathematics*, 55:1–13, 1994.

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

[15] Hans Kleine Büning and Theodor Lettmann. *Aussagenlogik: Deduktion und Algorithmen.* B.G. Teubner, 1994.

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

[17] Sam Buss and Toniann Pitassi. Resolution and the weak pigeonhole principle. In Mogens Nielsen and Wolfgang Thomas, editors, *Computer Science Logic (CSL) 97*, volume 1414 of *Lecture Notes in Computer Science*, pages 149–156, 1998.

[18] Sam Buss and G. Turán. Resolution proof of generalized pigeonhole principles. *Theoretical Computer Science*, 62:311–317, 1988.

[19] J.R. Celoni, Wolfgang J. Paul, and Robert Endre Tarjan. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977.

[20] C. L. Chang. The unit proof and the input proof in theorem proving. *Journal of the Association for Computing Machinery*, 17(4):698–707, October 1970.

[21] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th ACM Symposium on Theory of Computation*, pages 174–183, 1996.

[22] Stephen A. Cook and Toniann Pitassi. A feasibly constructive lower bound for resolution proofs. *Information Processing Letters*, 34:81–85, March 1990.

[23] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81:31–57, 1996.

[24] Mukesh Dalal and David W. Etherington. A hierarchy of tractable satisfiability problems. *Information Processing Letters*, 44:173–180, 1992.

[25] W.F. Dowling and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 1:267–284, 1984.

[26] Dingzhu Du, Jun Gu, and Panos M. Pardalos, editors. *Satisfiability Problem: Theory and Applications (DIMACS Workshop March 11-13, 1996)*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1997.

[27] Olivier Dubois, P. Andre, Y. Boufkhad, and C. Carlier. SAT versus UNSAT. In Johnson and Trick [48], pages 415–436. The Second DIMACS Challenge.

[28] Ulf Dunker. *Zur Effizienz der Beweissuche in der Logikverarbeitung*. PhD thesis, Universität Paderborn, July 1997.

[29] Thomas Eiter, Pekka Kilpeläinen, and Heikki Mannila. Recognizing renamable generalized propositional Horn formulas is NP-complete. *Discrete Applied Mathematics*, 59:23–31, 1995.

[30] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. In C. Meinel and S. Tison, editors, *16th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *LNCS*. Springer, 1999.

[31] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal Computing*, 5(4):691–703, 1976.

[32] P. Flajolet, J.C. Raoult, and J. Vuillemin. On the average number of registers required for evaluating arithmetic expressions. In *18th Annual Symposium on Foundations of Computer Science*, pages 196–205, 1977.

[33] John Franco. Relative size of certain polynomial time solvable subclasses of satisfiability. In Du et al. [26], pages 211–223.

[34] John Franco, Giorgio Gallo, Hans Kleine Büning, Ewald Speckenmeyer, and Cosimo Spera, editors. *Workshop on the Satisfiability Problem.* Universität zu Köln, Report No. 96-230, 1996. Siena, April 29 - May 3 (1996).

[35] John Franco and Allen Van Gelder. A perspective on certain polynomial time solvable classes of satisfiability. Submitted to Discrete Applied Mathematics, 1998.

[36] Zvi Galil. On the validity and complexity of bounded resolution. In *Proceedings of seventh annual ACM Symposium on Theory of Computing*, pages 72–82, May 1975.

[37] Giorgio Gallo and Maria Grazia Scutellà. Polynomially solvable satisfiability problems. *Information Processing Letters*, 29:221–227, November 1988.

[38] Allen Van Gelder. Autarky pruning in propositional model elimination reduces failure redundancy. *Journal of Automated Reasoning*, 25(2), 2000. To appear.

[39] Allen Van Gelder and Yumi K. Tsuji. Satisfiability testing with more reasoning and less guessing. In Johnson and Trick [48], pages 559–586. The Second DIMACS Challenge.

[40] Andreas Goerdt. Regular resolution versus unrestricted resolution. *SIAM Journal on Computing*, 22(4):661–683, August 1993.

[41] Jan Friso Groote and Joost P. Warners. The propositional formula checker HeerHugo. Submitted to Journal of Automated Theorem Proving (SAT 2000). Report SEN-R9905, Centre for Mathematics and Computer Scienc (CWI), Amsterdam, 1999.

[42] Jun Gu, Paul W. Purdom, John Franco, and Benjamin W. Wah. Algorithms for the satisfiability (SAT) problem: A survey. In Du et al. [26], pages 19–151.

[43] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[44] 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.

[45] Lawrence J. Henschen and Lawrence Wos. Unit refutations and horn sets. *Journal of the Association for Computing Machinery*, 21(4):590–605, October 1974.

[46] Edward Hirsch. A fast deterministic algorithm for formulas that have many satisfying assignments. *Logic Journal of the IGPL*, 6(1):59–71, 1998.

[47] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. In *18th Annual Symposium on Foundations of Computer Science*, pages 30–45. IEEE, 1977.

[48] David S. Johnson and Michael A. Trick, editors. *Cliques, Coloring, and Satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996. The Second DIMACS Challenge.

[49] Rainer Kemp. The average number of registers needed to evaluate a binary tree optimally. Technical Report A 77/04, University of Saarbrücken, Germany, 1977.

[50] R. Kowalski and D. Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2:227–260, 1971.

[51] Dexter Kozen. Lower bounds for natural proof systems. In *18th Annual Symposium on Foundations of Computer Science*, pages 254–266, 1977.

[52] Balakrishnan Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22:253–275, 1985.

[53] Balakrishnan Krishnamurthy and Robert N. Moll. Examples of hard tautologies in the propositional calculus. In *13th ACM Symposium on Theory of Computing (STOC'81)*, pages 28–37, 1981.

[54] Oliver Kullmann. A note on a generalization of extended resolution. In Franco et al. [34], pages 73–95. Siena, April 29 - May 3 (1996).

[55] Oliver Kullmann. On a generalization of extended resolution. To appear in Discrete Applied Mathematics (special edition on the satisfiability problem), 25 pages; enhanced version of [54], 1996.

[56] Oliver Kullmann. Worst-case analysis, 3-SAT decision and lower bounds: Approaches for improved SAT algorithms. In Du et al. [26], pages 261–313.

[57] Oliver Kullmann. Heuristics for SAT algorithms: Searching for some foundations. Submitted to Discrete Applied Mathematics, 23 pages, September 1998.

[58] Oliver Kullmann. Investigations on autark assignments. Submitted to Discrete Applied Mathematics, 19 pages, October 1998.

[59] Oliver Kullmann. New methods for 3-SAT decision and worst-case analysis. *Theoretical Computer Science*, 223(1-2):1–72, July 1999.

[60] Oliver Kullmann and Horst Luckhardt. Algorithms for SAT/TAUT decision based on various measures. Submitted to Information and Computation; 71 pages, December 1998.

[61] Marc-André Lemburg. Methoden zur Lösung von Formeln der Aussagenlogik in reiner Implikations- und konjunktiver Normalform. Master's thesis, Heinrich Heine-Universität Düsseldorf, 1997.

[62] Reinhold Letz. *First-Order Calculi and Proof Procedures for Automated Deduction*. PhD thesis, Technische Universität München, July 1993. See http://sunjessen24.informatik.tu-muenchen.de/personen/letz.html.

[63] László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search problems in the decision tree model. In *32nd Symposium on Foundations of Computer Science (FOCS' 91)*, pages 576–585, 1991.

[64] Horst Luckhardt. Obere Komplexitätsschranken für TAUT-Entscheidungen. In *Frege Conference 1984, Schwerin*, pages 331–337. Akademie-Verlag Berlin, 1984.

[65] Klaus Mayr. Refinements and extensions of model elimination. In Andrei Voronkov, editor, *Logic Programming and Automated Reasoning (LPAR'93)*, volume 698 of *Lecture Notes in Artificial Intelligence*, pages 217–228. Springer, 1993.

[66] Klaus Mayr, May 1999. Personal communication.

[67] David J. McClurkin. A lower bound for tree resolution. *Discrete Applied Mathematics*, 54:37–53, 1994.

[68] J. C. C. McKinsey. Archie Blake: Canonical expressions in Boolean algebra. Review, The Journal of Symbolic Logic (3), 1938.

[69] B. Monien and Ewald Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, 10:287–295, 1985.

[70] I. Nakata. On compiling algorithms for arithmetic expressions. *Communications of the ACM*, 10:492–494, 1967.

[71] M.H.A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43:223–243, 1942.

[72] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, 1994.

[73] Ramamohan Paturi, Pavel Pudlak, and Francis Zane. Satisfiability coding lemma. In *Proc. 38th Annual Symp. Foundations of Computer Science (FOCS 97)*, pages 566–574, 1997.

[74] Nicholas Pippenger. Pebbling. Technical report, Mathematical Sciences Department, IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598, 1980.

[75] Daniele Pretolani. Hierarchies of polynomially solvable satisfiability problems. *Annals of Mathematics and Artificial Intelligence*, 17(3-4):339–357, 1996.

[76] Alexander Razborov, Avi Wigderson, and Andrew Yao. Read-once branching programs, rectangular proofs of the pigeonhole principle and the transversal calculus. In *29th A.C.M. Symposium on the Theory of Computing*, pages 739–748, 1997.

[77] R.R. Redziejowski. On arithmetic expressions and trees. *Communications of the ACM*, 12:81–84, 1969.

[78] Ingo Schiermeyer. Solving 3-satisfiability in less than 1.579^n steps. In *Selected papers from Computer Science Logic '92*, volume 702 of *Lecture Notes Computer Science*, pages 379–394, 1992.

[79] Ingo Schiermeyer. Pure literal look ahead: An $O(1,497^n)$ 3-satisfiability algorithm. In Franco et al. [34], pages 127–136. Extended abstract.

[80] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of AAAI'92*, pages 440–446, July 1992.

[81] 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.

[82] Joao P. Marques Silva and Karem A. Sakallah. GRASP—a new search algorithm for satisfiability. Technical Report CSE-TR-292-96, University of Michigan, Department of Electrical Engineering and Computer Science, 1996.

[83] Gunnar Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33:277–280, 1996.

[84] 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.

[85] Jacobo Torán, 1999. Private communication.

[86] Jacobo Torán. Lower bounds for space in resolution. In *Proceedings of CSL'99*. European Association for Computer Science Logic, Springer, 1999. To appear.

[87] G.S. Tseitin. On the complexity of derivation in propositional calculus. In *Seminars in Mathematics*, volume 8. V.A. Steklov Mathematical Institute, Leningrad, 1968. English translation: Studies in mathematics and mathematical logic, Part II (A.O. Slisenko, editor), 1970, pages 115-125.

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

[89] Hans van Maaren. A short note on linear autarkies, q-Horn formulas and the complexity index. Technical Report 99-26, DIMACS, May 1999. Can be obtained from http://dimacs.rutgers.edu/TechnicalReports/.

[90] André Vellino. *The complexity of automated reasoning.* PhD thesis, Department of Philosophy, University of Toronto, 1989.

[91] Susumu Yamasaki and Shuji Doshita. The satisfiability problem for a class consisting of Horn sentences and some non-Horn sentences in propositional logic. *Information and Control*, 59:1–12, 1983.

[92] Hantao Zhang. SATO: an efficient propositional prover. In *Proc. of International Conference on Automated Deduction (CADE-97)*, 1997.

[93] Hantao Zhang and Mark E. Stickel. An efficient algorithm for unit propagation. In *Proc. of the Fourth International Symposium on Artificial Intelligence and Mathematics. Ft. Lauderdale, Florida*, 1996.

[94] Wenhui Zhang. Number of models and satisfiability of sets of clauses. *Theoretical Computer Science*, 155:277–288, 1996.