Upper and lower bounds on the complexity of generalised resolution and generalised constraint satisfaction problems

Oliver Kullmann (o.kullmann@swansea.ac.uk) *

Computer Science Department
University of Wales Swansea
Swansea, SA2 8PP, UK
http://cs-svr1.swan.ac.uk/~csoliver/

Abstract. Capturing propositional logic, constraint satisfaction problems and systems of polynomial equations, we introduce the notion of *systems with finite instantiation by partial assignments*, *fipa-systems* for short, which are independent of special representations of problem instances, but which are based on an *axiomatic approach* with instantiation (or substitution) by partial assignments as the fundamental notion. Fipa-systems seem to constitute the most general framework allowing for a *theory of resolution with non-trivial upper and lower bounds*. For every fipa-system we generalise relativised hierarchies originating from *generalised Horn formulas* (Gallo and Scutellà, 1988; Büning, 1993; Pretolani, 1996; Kullmann, 1999b), and obtain hierarchies of problem instances with recognition and satisfiability decision in polynomial time and linear space, *quasi-automatising* relativised and generalised *tree resolution* and utilising a general "quasi-tight" lower bound for tree resolution. And generalising *width-restricted resolution* from (Galil, 1975; Büning, 1993; Ben-Sasson and Wigderson, 1999; Kullmann, 1999b), for every fipa-system a (stronger) family of hierarchies of unsatisfiable instances with polynomial time recognition is introduced, weakly automatising relativised and generalised *full resolution* and utilising a general lower bound for full resolution generalising (Galil, 1975; Clegg et al., 1996; Ben-Sasson and Wigderson, 1999; Kullmann, 1999b).

Key words: satisfiability problem (SAT), systems with instantiation, propositional logic, constraint satisfaction problems, polynomial time hierarchies, generalised resolution, lower bounds for resolution, upper bounds for SAT algorithms, automatisation of proof systems, generalised input resolution, generalised width restricted resolution, induced width of constraint satisfaction problems

1. Introduction

The most natural attempt to decide whether a *problem instance* P (of any kind) has a *satisfying assignment* φ, assigning *values* $\varphi(v) \in D_v$ to

* Partially supported by the Natural Sciences and Engineering Research Council of Canada and by the Communications and Information Technology Ontario.

(some) *variables* v with *domain* D_v, is to consider a *branching variable* v "occurring" in P and to split P into the $|D_v|$-many problem instances obtained from P by *substituting* all possible values $\varepsilon \in D_v$ of variable v into v. So, instead of a specific representation of problem instances (as propositional formulas, constraint satisfaction problems or graphs for example), we regard substitution of values into variables as the basic notion of an axiomatic approach, where only some very basic properties of the substitution process are required, while the specific nature of "problem instances" is left open. *Satisfiable problem instances* are those instances where there is a partial assignment such that substitution by this assignment (called "application of partial assignments") yields a specific instance $\top$, called the "trivial satisfiable instance."

It is common knowledge, that the process of splitting up propositional clause-sets as captured by the notion of "semantic trees" or "D(P)LL-trees" and the refutation system of tree resolution are essentially different points of view of the same thing (on unsatisfiable clause-sets). We take the underlying construction of this equivalence as starting point and introduce (generalised) *resolution proofs* for arbitrary systems of problem instances by *identifying clauses and partial assignments*. More precisely we proceed in two stages: From arbitrary problem instances P we go to *generalised clause-sets* F via the relation $P \xrightarrow{\mathcal{U}} F$ of *direct entailment*, that is, each generalised clause $C \in F$ (or "nogood" as they are called in the artificial intelligence literature) corresponds to a partial assignment such that applying this partial assignment to the problem instance P yields an instance recognised by an *oracle* $\mathcal{U}$ as unsatisfiable. And for generalised clause-sets there is a natural generalisation of resolution, as already used in (Baker, 1995), which enables us to define $F \vdash \bot$. Altogether we obtain $P \overset{\mathcal{U}}{\vdash} \bot$.

In this way we are able to generalise many of the results from (Kullmann, 1999b), where in turn the following lines of research have been brought together:

- bounded resolution and lower bounds for resolution (Galil, 1975; Clegg et al., 1996; Ben-Sasson and Wigderson, 1999);
- generalisations of Horn formulas (Yamasaki and Doshita, 1983; Gallo and Scutellà, 1988; Büning, 1993) and the satisfiability algorithm in (Pretolani, 1996) derived from these ideas;
- space complexity of resolution (Esteban and Torán, 1999; Torán, 1999);

– Stålmarck's proof search algorithm (Stålmarck and Säflund, 1990; Harrison, 1996; Sheeran and Stålmarck, 1998; Groote and Warners, 2000).

Different from (Mitchell, 1998), our (general) upper and lower bounds for resolution and tree resolution work directly, *without translation*, while we overcome the *resolution barrier* (that is, exponential lower bounds for resolution) by means of *oracles*, incorporating additional power for satisfiability decision of "elementary cases." (In this article we give only the general outline of this approach, with the aim of unifying "resolution based methods" with "special methods" — for some specific examples see (Kullmann, 1999b).)

We introduce two hierarchies $(\mathcal{G}_k(\mathcal{U},\mathcal{S}))_{k\in\mathbb{N}_0}$ and $(\mathcal{W}_k(\mathcal{U}))_{k\in\mathbb{N}_0}$ of problem instances, depending in the first case on oracles $\mathcal{U}$ for unsatisfiability and $\mathcal{S}$ for satisfiability and covering all problem instances, while the second hierarchy covers only unsatisfiable instances. For both hierarchies we give efficient algorithms deciding membership at level k and also satisfiability in the affirmative case for hierarchy $(\mathcal{G}_k(\mathcal{U},\mathcal{S}))_{k\in\mathbb{N}_0}$, which is (much) more space efficient than hierarchy $(\mathcal{W}_k(\mathcal{U}))_{k\in\mathbb{N}_0}$ and is not restricted to unsatisfiable instances. Hierarchy $(\mathcal{W}_k(\mathcal{U}))_{k\in\mathbb{N}_0}$ is stronger on unsatisfiable instances, but needs additional (good) upper bounds on the possible level of inclusion in the hierarchy to be able to handle also satisfiable instances (efficiently).

The algorithm associated with the hierarchy $(\mathcal{G}_k(\mathcal{U},\mathcal{S}))_{k\in\mathbb{N}_0}$ *quasi-automatises* tree resolution (on unsatisfiable instances), while the algorithm associated with the hierarchy $(\mathcal{W}_k(\mathcal{U}))_{k\in\mathbb{N}_0}$ accomplishes a weak form of automatisation of (full) resolution.

We conclude this introduction by an overview on the content of the five remaining sections of this article (the reader finds more specific introductions at the beginning of each section):

1. In Section 2 the basic notions for this article are developed: The *monoid of partial assignments* in Subsection 2.1, *systems with finite instantiation by partial assignments (fipa-systems)* in Subsection 2.2, and *generalised clause-sets* in Subsection 2.3, yielding the first example for fipa-systems, while further examples are given in Subsection 2.4.

2. *Branching trees* are our basic model for satisfiability decision, and are introduced in Section 3, especially in Subsection 3.1. In Subsection 3.2 we introduce a complexity measure $h(T)$ for branching

trees T, called *levelled height*, reflecting in some way how difficult it is to find T. Problem instances having a branching tree with levelled height at most k together form level k of our first hierarchy, denoted by $(\mathcal{G}_k(\mathcal{U},\mathcal{S}))_{k\in\mathbb{N}_0}$.

3. In Subsection 4.1 of Section 4 we show that each level of the hierarchy $(\mathcal{G}_k(\mathcal{U},\mathcal{S}))_{k\in\mathbb{N}_0}$ has an algorithm deciding membership at level k as well as deciding satisfiability in the affirmative case in *polynomial time*. Running through the levels from below we obtain a general *satisfiability decision algorithm* in Subsection 4.2.

4. The subject of Section 5 is *generalised (tree) resolution*. Basic notions and tools are developed in Subsection 5.1. In Subsection 5.2 the correspondence between branching trees and resolution trees for unsatisfiable instances is worked out. In Subsection 5.3 we recover the unsatisfiable instances in $\mathcal{G}_k(\mathcal{U},\mathcal{S})$ as the instances having a *k-times nested input refutation*. *General upper and lower bounds* on the complexity of tree resolution are proven in Subsection 5.4, yielding *quasi-automatisation* of tree resolution by the algorithm from Subsection 4.2.

5. Finally, Section 6 is devoted to *width restricted resolution* and (general) bounds for *full resolution*. First Unit resolution is generalised to *k-resolution* in Subsection 6.1, and in Subsection 6.2 it is shown that k-resolution *simulates* k-times nested input resolution. Then in Subsection 6.3 we consider the complexity of full (or dag-like) resolution and prove in Subsection 6.4 a *general lower bound* for full resolution in terms of *width* associated to k-resolution. Whether a slight strengthening of k-resolution is able to derive the empty clause is shown to be decidable in *polynomial time* in Subsection 6.5, yielding a *weak form of automatisation* of full resolution in Subsection 6.6 (again by running through the levels). In Subsection 6.7 we discuss the relation between k-resolution and *bounded resolution* (where *all* clauses must have length at most k, while for k-resolution for each resolution step one clause is allowed to have length greater than k). And finally in Subsection 6.8 we show that $(k+1)$-bounded resolution is sufficient to refute unsatisfiable constraint satisfaction problems with *induced width* of at most k (that is, the constraint graph has *treewidth* less than or equal to k).

2. Generalised constraint satisfaction problems

In this basic section we present a general framework for "satisfiability problems", including propositional clause-sets, generalised clause-sets for constraint satisfaction problem and constraint satisfaction problems in the usual sense, but also ordinary propositional logic (over any basis), and even arbitrary languages (recursive or not).

This generality is achieved by an *axiomatic approach* with the notion of a "partial assignment" as the starting point, where a partial assignment is a map from some "variables" to "values" of their respective "domains". More precisely, it is the *monoid* $\mathcal{PASS}$ of partial assignments which is considered as basic, where the composition $\varphi \circ \psi$ of partial assignments is the natural choice of combining (partial) functions which shall act from the left on some other system: Just take φ and ψ together, and in case of a conflict regard ψ as dominant. Now instead of isolated problem instances we consider any set $\mathcal{PI}$ of "problem instances," where we only demand that an operation $*$ of $\mathcal{PASS}$ on $\mathcal{PI}$ is given, called "application of partial assignments", and that a special problem instance $\top \in \mathcal{PI}$ is given, called the "trivial satisfiable instance", which is invariant under application of partial assignments. A problem instance $P \in \mathcal{PI}$ is defined as satisfiable if there is a partial assignment $\varphi \in \mathcal{PASS}$ with $\varphi * P = \top$. The triple $(\mathcal{PI}, \top, *)$ is called a *system with finite instantiation by partial assignments*, or *fipa-system* for short. See Subsections 2.1 and 2.2 for the details.

While this article has been in the refereeing process, the author has further developed the abstract theory of generalised satisfiability problems in (Kullmann, 2003), where instead of the monoid of partial assignments arbitrary monoids are allowed. The focus is on the abstract theory of autarkies, which requires further structure on the set of problem instances. Fixing the monoid of substitutions as $\mathcal{PASS}$, the most general case considered is called a "qo-system with finite instantiation by partial assignments", where "qo" stands for some quasi-order on the set of problem instances (compatible with the operation of $\mathcal{PASS}$). For the purposes of the present article we ignore this quasi-order, which is formally "legalised" by using the quasi-order $\leq_t$ on $\mathcal{PI}$ given by the following rules:

1. for all $P \in \mathcal{PI}$ we have $\top \leq P$;
2. for all $P, Q \in \mathcal{PI} \setminus \{\top\}$, such that both P and Q are invariant under the operation of $\mathcal{PASS}$, we have $P \leq Q$;

3. for all other $P \in \mathcal{PI}$ we just have $P \leq P$.

In this way, any fipa-system $(\mathcal{PI}, \top, *)$ yields in a canonical way a qo-system $((\mathcal{PASS}, \circ, \emptyset), (\mathcal{PI}, \leq_t, \top), *)$ with finite instantiation by partial assignments in the sense of (Kullmann, 2003) (here $\emptyset$ is the empty partial assignment, the identity element of the monoid $\mathcal{PASS}$), where the general equivalence relation $\equiv$ on $\mathcal{PI}$ defined in (Kullmann, 2003) becomes ordinary *satisfiability equivalence*.

The basic example for a fipa-system is given by the system $\mathcal{CLS}$ of (generalised) clause-sets, introduced in Subsection 2.3. Literals x are now pairs $x = (v, \varepsilon)$ of variables v and values $\varepsilon \in D_v$, where D_v is the "domain" of v, expressing that v must *not* take value ε. Clauses are "non-tautological" finite sets C of literals, that is, there are no literals $(v, \varepsilon_1), (v, \varepsilon_2) \in C$ with $\varepsilon_1 \neq \varepsilon_2$, while clause-sets are finite sets of clauses. ((Baker, 1995) uses the same notion of clauses, which are also referred to as "nogoods" in the AI literature.)

Finally some other examples for systems of problem instances are given in Subsection 2.4.

2.1. Partial assignments

Let $\boldsymbol{\mathcal{VA}}$ be the set of *variables* (which are just (arbitrary) names), and for each variable $v \in \mathcal{VA}$ let $\boldsymbol{D_v}$ be any non-empty set, the *domain* of v. A *partial assignment* is a map $\varphi : V \to \bigcup_{v \in \mathcal{VA}} D_v$, where $V \subseteq \mathcal{VA}$ is a finite set of variables, and for all $v \in V$ we have $\varphi(v) \in D_v$ (i.e., φ is a choice function for the family $(D_v)_{v \in V}$). The domain of φ we denote by $\mathbf{var}(\boldsymbol{\varphi}) := V$.

Let $\boldsymbol{\mathcal{PASS}}$ be the set of all partial assignments. On $\mathcal{PASS}$ a natural law of composition $\circ : \mathcal{PASS} \times \mathcal{PASS} \to \mathcal{PASS}$, called *composition of partial assignments*, is given as follows. For $\varphi, \psi \in \mathcal{PASS}$ we define $\boldsymbol{\psi \circ \varphi}$ as the partial assignment with domain $\mathrm{var}(\psi \circ \varphi) = \mathrm{var}(\varphi) \cup \mathrm{var}(\psi)$, defined for $v \in \mathrm{var}(\varphi)$ as $(\psi \circ \varphi)(v) := \varphi(v)$, while for $v \in \mathrm{var}(\psi) \setminus \mathrm{var}(\varphi)$ we set $(\psi \circ \varphi)(v) := \psi(v)$. In other words, for the evaluation of $(\psi \circ \varphi)(v)$ we first try to use φ, and only in case v is not in the domain of φ we use ψ. It is easy to check that $\mathcal{PASS}$ together with $\circ$ is an idempotent monoid with identity element the empty partial assignment $\emptyset \in \mathcal{PASS}$, where two partial assignments

$\varphi, \psi \in \Psi$ commute iff they are compatible:

$$\emptyset \circ \varphi = \varphi \circ \emptyset = \varphi;\ (\varphi_1 \circ \varphi_2) \circ \varphi_3 = \varphi_1 \circ (\varphi_2 \circ \varphi_3);\ \varphi \circ \varphi = \varphi;$$
$$\varphi \circ \psi = \psi \circ \varphi \Leftrightarrow \forall v \in \mathrm{var}(\varphi) \cap \mathrm{var}(\psi) : \varphi(v) = \psi(v).$$

We use $\langle \boldsymbol{v \to \varepsilon} \rangle \in \mathcal{PASS}$ for a variable $v \in \mathcal{VA}$ and $\varepsilon \in D_v$ to denote the "elementary" partial assignment with domain $\{v\}$ and mapping $v \mapsto \varepsilon$. The elementary partial assignments generate the monoid $\mathcal{PASS}$, moreover $\mathcal{PASS}$ is *the* monoid (up to isomorphism) generated by the set $\{\langle v \to \varepsilon \rangle\}_{v \in \mathcal{VA}, \varepsilon \in D_v}$ of elementary partial assignments with defining relations

$$\langle v \to \varepsilon' \rangle \circ \langle v \to \varepsilon \rangle = \langle v \to \varepsilon \rangle$$
$$\langle v \to \varepsilon \rangle \circ \langle v' \to \varepsilon' \rangle = \langle v' \to \varepsilon' \rangle \circ \langle v \to \varepsilon \rangle \text{ for } v \neq v'.$$

Considering partial assignments φ as sets of ordered pairs, that is assuming the equality $\varphi = \{(v, \varphi(v)) : v \in \mathrm{var}(F)\}$ (and thus $\langle v \to \varepsilon \rangle = \{(v, \varepsilon)\}$), we have the natural subsumption relation $\varphi \subseteq \psi$ between partial assignments, expressing that ψ is an extension of φ (or φ is a restriction of ψ). A characterisation of this order in terms of composition of partial assignments is given by $\varphi \subseteq \psi \Leftrightarrow \psi \circ \varphi = \psi$. For $\varphi_1 \subseteq \varphi_2$ we have $\varphi_1 \circ \psi \subseteq \varphi_2 \circ \psi$ and $\psi \circ \varphi_1 \subseteq \psi \circ \varphi_2$.

2.2. Systems with finite instantiation by partial assignments ("fipa-systems")

We recall a few elementary notions from algebra (see for example chapter I, paragraph 5 in (Bourbaki, 1989)). An *operation* of a monoid M (with $\circ$ as law of composition and e as identity element) on a set E is a map $* : M \times E \to E$ fulfilling

$$e * x = x, \quad \alpha * (\beta * x) = (\alpha \circ \beta) * x$$

for all $\alpha, \beta \in M$ and $x \in E$. The *stabiliser* of an element $a \in E$ is the set of all $\alpha \in M$ with $\alpha * a = a$. An *invariant element* of E is an element $a \in E$ with whole M as its stabiliser.

Now a *system with finite instantiation by partial assignments* (*fipa-system* for short) is a set $\boldsymbol{\mathcal{PI}}$ together with an operation $*$ of $\mathcal{PASS}$ on $\mathcal{PI}$ (called *application of partial assignments*) and an invariant element $\top$ of $\mathcal{PI}$, called the *trivial satisfiable problem instance*, such that for all instances $P \in \mathcal{PI}$ there are only finitely many variables $v \in \mathcal{VA}$ with the property that there is $\varepsilon \in D_v$ with $\langle v \to \varepsilon \rangle * P \neq P$.

Let's have a closer look at the stabiliser of an instance P. If for a variable $v \in \mathcal{VA}$ there is $\varepsilon \in D_v$ with $\langle v \to \varepsilon \rangle * P = P$, then in fact for all $\varepsilon' \in D_v$ the assignment $\langle v \to \varepsilon' \rangle$ is in the stabiliser of P, since

$$\langle v \to \varepsilon' \rangle * P = \langle v \to \varepsilon' \rangle * (\langle v \to \varepsilon \rangle * P) = (\langle v \to \varepsilon' \rangle \circ \langle v \to \varepsilon \rangle) * P$$
$$= \langle v \to \varepsilon \rangle * P = P.$$

So we may define the set $\mathbf{var(P)}$ of *relevant variables* as the set of variables $v \in \mathcal{VA}$ such that for one or for all $\varepsilon \in D_v$ the assignment $\langle v \to \varepsilon \rangle$ does not stabilise P, that is we have $\langle v \to \varepsilon \rangle * P \neq P$. By definition we have $\varphi * P = P$ for all partial assignments $\varphi \in \mathcal{PASS}$ with $\mathrm{var}(\varphi) \cap \mathrm{var}(P) = \emptyset$. The additional requirement on a fipa-system can be stated as the assumption that for all instances P the set $\mathrm{var}(P)$ is finite.

For any partial assignment $\varphi \in \mathcal{PASS}$ and any instance $P \in \mathcal{PI}$ we have

$$\mathrm{var}(\varphi) \cap \mathrm{var}(\varphi * P) = \emptyset$$

since for any variable $v \in \mathrm{var}(\varphi)$ and any $\varepsilon \in D_v$ we have the equality $\langle v \to \varepsilon \rangle * (\varphi * P) = (\langle v \to \varepsilon \rangle \circ \varphi) * P = \varphi * P$. And furthermore

$$\mathrm{var}(\varphi * P) \subseteq \mathrm{var}(P)$$

holds, since for $\langle v \to \varepsilon \rangle * P = P$ we have $\langle v \to \varepsilon \rangle * (\varphi * P) = (\langle v \to \varepsilon \rangle \circ \varphi) * P$, and now either $\langle v \to \varepsilon \rangle \circ \varphi = \varphi$ holds, or we have $\langle v \to \varepsilon \rangle \circ \varphi = \varphi \circ \langle v \to \varepsilon \rangle$, and in both cases we get $\langle v \to \varepsilon \rangle * (\varphi * P) = \varphi * P$. In other words, applying a partial assignment eliminates its variables from the instance and does not create new variables.

We use $\boldsymbol{n(P)} := |\mathrm{var}(P)|$ for the number of variables in an instance P. Note that P is invariant (under the operation of $\mathcal{PASS}$) iff $n(P) = 0$ holds. The set $\boldsymbol{\mathcal{UPI}_0}$ of *trivial unsatisfiable problem instances* is the set of invariant instances different from $\top$.

We call a partial assignment φ a *satisfying assignment* for P if $\varphi * P = \top$ holds. The set $\boldsymbol{\mathcal{SPI}}$ of *satisfiable problem instances* is defined as the set of $P \in \mathcal{PI}$ which have a satisfying (partial) assignment. The set of *unsatisfiable problem instances* is $\boldsymbol{\mathcal{UPI}} := \mathcal{PI} \setminus \mathcal{SPI}$. Be definition we have $\top \in \mathcal{SPI}$ and $\mathcal{UPI}_0 \subseteq \mathcal{UPI}$. For any partial assignment $\varphi \in \mathcal{PASS}$ and any instance $P \in \mathcal{PI}$ from $\varphi * P \in \mathcal{SPI}$ also $P \in \mathcal{SPI}$ follows, while from $P \in \mathcal{UPI}$ we get $\varphi * P \in \mathcal{UPI}$. We say that instance $P_1 \in \mathcal{PI}$ *entails* instance $P_2 \in \mathcal{PI}$, denoted by $\boldsymbol{P_1 \models P_2}$, if for all partial assignments $\varphi \in \mathcal{PASS}$ with $\varphi * P_1 = \top$ we also have $\varphi * P_2 = \top$. Thus from $P_1 \models P_2$ in case of P_1 being satisfiable

we can conclude that P_2 is also satisfiable, while in case of P_2 being unsatisfiable we conclude that also P_1 is unsatisfiable.

Given any family $(\mathcal{S}_i)_{i\in I}$ of fipa-systems $\mathcal{S}_i = (\mathcal{PI}_i, *_i, \top_i)$, the

$$\textit{direct sum} \bigoplus_{i\in I} \boldsymbol{\mathcal{S}_i}$$

is defined as the 3-tuple $(\mathcal{PI}, *, \top)$, where $\mathcal{PI}$ is the set of all families $(P_i)_{i\in I}$ with $\forall i \in I : P_i \in \mathcal{PI}_i$ such that for only finitely many indices i we have $P_i \neq \top_i$, the operation $*$ of $\mathcal{PASS}$ on $\mathcal{PI}$ is defined componentwise (i.e., $\varphi * (P_i)_{i\in I} := (\varphi *_i P_i)_{i\in I}$), and $\top := (\top_i)_{i\in I}$. Obviously $\bigoplus_{i\in I} \mathcal{S}_i$ is also a fipa-system.

2.3. Generalised clause-sets

A *literal* in our context is a pair (v, ε) with $v \in \mathcal{VA}$ and $\varepsilon \in D_v$, and the set of all literals is denoted by $\boldsymbol{\mathcal{LIT}}$. For a literal $x = (v, \varepsilon)$ we set $\mathbf{var}(\boldsymbol{x}) := v$. The intended interpretation of a literal (v, ε) is that "variable v shall *not* get value ε." A *clause* is a finite set C of literals such that there are no literals $x, y \in C$ with $x \neq y$ but $\mathrm{var}(x) = \mathrm{var}(y)$. The set of all clauses is denoted by $\boldsymbol{\mathcal{CL}}$. A *clause-set* is a finite set of clauses, and the set of all clause-sets is denoted by $\boldsymbol{\mathcal{CLS}}$.

For a partial assignment $\varphi \in \mathcal{PASS}$ and a clause-set $F \in \mathcal{CLS}$ we define $\boldsymbol{\varphi * F}$ as the set of clauses $C' = \{x \in C : \mathrm{var}(x) \notin \mathrm{var}(\varphi)\} \in \mathcal{CL}$ for such $C \in F$ where no literal $(v, \varepsilon) \in C$ exists with $v \in \mathrm{var}(\varphi)$ and $\varphi(v) \neq \varepsilon$. In other words, $\varphi * F$ results from F by deleting all clauses fulfilled by φ and removing literals falsified by φ from the remaining clauses.

In this way an operation of the monoid $\mathcal{PASS}$ on $\mathcal{CLS}$ is given, and the system $(\mathcal{CLS}, *, \top)$ is a canonical example for a fipa-system as defined in Subsection 2.2, using $\top := \emptyset \in \mathcal{CLS}$, the *empty clause-set*. The only other invariant clause-set is $\{\bot\} \in \mathcal{CLS}$ with $\bot := \emptyset \in \mathcal{CL}$, and the sets of satisfiable resp. unsatisfiable problem instances are the usual sets $\boldsymbol{\mathcal{SAT}} \subset \mathcal{CLS}$ and $\boldsymbol{\mathcal{USAT}} = \mathcal{CLS} \setminus \mathcal{SAT}$ of satisfiable resp. unsatisfiable clause-sets. Using $\mathbf{var}(\boldsymbol{C}) := \{\mathrm{var}(x) : x \in C\}$ for a clause C, we get $\mathrm{var}(F) = \bigcup_{C\in F} \mathrm{var}(C)$ for clause-sets $F \in \mathcal{CLS}$.

For clause-sets $F \in \mathcal{CLS}$ and clauses $C \in \mathcal{CL}$ we use $\boldsymbol{F \models C}$ as abbreviation for $F \models \{C\}$, and thus F is unsatisfiable iff $F \models \bot$ holds true.

Of fundamental importance is the following natural bijective *correspondence* between $\mathcal{PASS}$ and $\mathcal{CL}$: For a clause $C \in \mathcal{CL}$ let $\boldsymbol{\varphi_C} \in \mathcal{PASS}$ be the partial assignment with domain $\mathrm{var}(C)$ mapping $v \mapsto \varepsilon$ for literals $(v, \varepsilon) \in C$, and for a partial assignment $\varphi \in \mathcal{PASS}$ let $\boldsymbol{C_\varphi} \in \mathcal{CL}$ be the clause $\{(v, \varphi(v)) : v \in \mathrm{var}(\varphi)\}$. Considering the equation $\varphi * \{C\} = \{\bot\}$, the clause C_φ is the largest solution for C (when fixing φ), while φ_C is the smallest solution for φ (when fixing C). Using the representation of partial assignments as sets of ordered pairs in fact we have $\mathcal{PASS} = \mathcal{CL}$, and $\varphi_C = C$ as well as $C_\varphi = \varphi$ holds.

2.4. Examples of fipa-systems

As already mentioned, the set of (generalised) clause-sets $\mathcal{CLS}$ together with the natural action of $\mathcal{PASS}$ is a fipa-system in the sense of Subsection 2.2. In the Boolean case, that is in case for all $v \in \mathcal{VA}$ we have $D_v = \{0, 1\}$, our generalised notion of clause-sets becomes the ordinary notion of (propositional) clause-sets (conjunctive normal forms), where a literal $(v, 0)$ stands for v, while $(v, 1)$ stands for $\overline{v}$. But we could take for $\mathcal{PI}$ also *any set of propositional formulas* stable under substitutions of truth values, so for example all formulas over the basis $\{\wedge, \vee, \neg\}$ or over $\{\wedge, \vee, \neg, \to, \leftrightarrow, 0, 1\}$, or also over $\{\oplus, 0, 1\}$ for example.

Another important example for a fipa-system is given by the class of constraint satisfaction problems. A *constraint* is a pair $(\vec{x}, R)$, where $\vec{x} \in \mathcal{VA}^m$ for some $m \in \mathbb{N}_0$ and $R \subseteq \prod_{i=1}^m D_{\vec{x}_i}$, while a *constraint satisfaction problem* is a finite set of constraints. We obtain a fipa-system by allowing also substitution of constants, that is considering constraints $(\vec{x}, R, \psi)$, where $(\vec{x}, R)$ is as before, while $\psi \in \mathcal{PASS}$ with $\mathrm{var}(\psi) \subseteq \{\vec{x}_1, \ldots, \vec{x}_m\}$ stands for already fixed values of variables from $\vec{x}$; an original constraint (without substitutions) is represented by choosing $\psi = \emptyset$. For a partial assignment φ and a constraint satisfaction problem P we define $\varphi * P$ as the set of constraints $(\vec{x}, R, (\varphi|\{\vec{x}_1, \ldots, \vec{x}_m\}) \circ \psi)$ for $(\vec{x}, R, \psi) \in P$. To obtain $\top$, we identify all trivially satisfiable constraints satisfaction problems P, that is, where for all $(\vec{x}, R, \psi) \in P$ we have $\mathrm{var}(\psi) = \{\vec{x}_1, \ldots, \vec{x}_m\}$ and $(\psi(\vec{x}_1), \ldots, \psi(\vec{x}_m)) \in R$ (note that all such constraints are invariant under substitutions), and denote the result of this identification by $\top$. Obviously all axioms for fipa-systems are fulfilled, and the satisfiable instances are the satisfiable constraint satisfaction problems in the usual sense. A trivial unsatisfiable problem is a constraint satisfaction problem P where for all $(\vec{x}, R, \psi) \in P$ we have $\mathrm{var}(\psi) = \{\vec{x}_1, \ldots, \vec{x}_m\}$ and $(\psi(\vec{x}_1), \ldots, \psi(\vec{x}_m)) \notin R$.

Finally we show how to obtain a fipa-system from *any language* $\emptyset \neq L \subseteq \Sigma^*$, where Σ is some alphabet (possibly infinite). Consider a function $P : \Sigma^* \to \Sigma^*$ with image L, that is, $x \in L \Leftrightarrow \exists\, w \in \Sigma^* : P(w) = x$, where words w with $P(w) = x$ are regarded as "proofs" for the membership of x in L. In case we assume P to be computable, we obtain exactly the (non-empty) *recursively enumerable languages* L and without loss of generality we may assume P to be poly-time computable. Now let $\mathcal{VA} = \mathbb{N}$, while the universal domain is $D_n = \Sigma$. Let a "problem instance" be a pair (w, x) with $x \in \Sigma^*$ and $w \in (\Sigma \cup \{\text{"?"}\})^*$, where we assume "?" $\notin \Sigma$. For a partial assignment $\langle i \to a \rangle$ we define $\langle i \to a \rangle * (w, x)$ as (w, x) in case $i > |w|$ or in case we have $i \leq |w|$ but $w_i \neq$ "?" holds, and otherwise as (w', x) where w' is obtained from w by replacing the "?" at position i by a. Finally by identifying all instances (w, x) with $w \in \Sigma^*$ and $P(w) = x$, and defining $\top$ as the result of this identification, we obtain a system of problem instances where the satisfiable instances are those (w, x) such that there is a substitution into the open positions in w (marked with "?") yielding a proof for membership of x in L w.r.t. the "proof system" P.

3. Branching trees and their levelled height

Given a problem instance $P \in \mathcal{PI}$, the simplest way to decide satisfiability of P is to choose a "branching variable" $v \in \mathrm{var}(P)$ and to split P into the $|D_v|$ many problems obtained by assigning all possible values to variable v as shown in Figure 1.

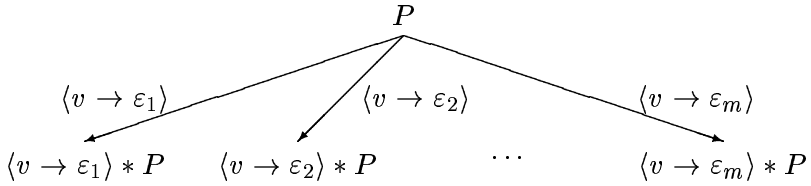

Figure 1. Splitting of problem instance P, using branching variable v with domain $D_v = \{\varepsilon_1, \ldots, \varepsilon_m\}$

Using sets $\mathcal{UPI}_0 \subseteq \boldsymbol{\mathcal{U}} \subseteq \mathcal{UPI}$ and $\top \in \boldsymbol{\mathcal{S}} \subseteq \mathcal{SPI}$ as *oracles* for unsatisfiability resp. satisfiability, completed decision of satisfiability of P by splitting is captured by the notion of an "$(\mathcal{U}, \mathcal{S})$-branching tree T for P" as defined in Subsection 3.1. Here by "completed decision" we mean that the instances labelling the leaves of T are either elements of $\mathcal{U}$ or of $\mathcal{S}$. The basic examples of oracles $\mathcal{U}$ for *unsatisfiability* (beyond $\mathcal{UPI}_0$) are the set $\mathcal{U}_0 \subseteq \mathcal{USAT}$ of clause-sets $F \in \mathcal{CLS}$ with

$\bot \in F$, and the set $\mathcal{U}_0$ of constraint satisfaction problems P (in the sense of Subsection 2.4) containing some constraint $(\vec{x}, R, \psi) \in P$ with $\vec{x} = (x_1, \ldots, x_m)$ such that ψ assigns to every variable of $\vec{x}$ a value (i.e., $\mathrm{var}(\psi) = \{x_1, \ldots, x_m\}$), and $(\psi(x_1), \ldots, \psi(x_m)) \notin R$ holds. As an interesting oracle for *satisfiability* in (Kullmann, 1999b) the class of "linearly satisfiable" clause-sets (introduced in (Kullmann, 2000b), based on Linear Programming) has been used for the case of propositional clause-sets. For propositional logic our notion of branching trees is very close to the notion of "semantics trees" (cf. (Reckhow, 1975)), but due to the use of oracles the question is emphasised, which paths of semantic trees should be regarded as "closed."

A natural complexity measure for an instance $P \in \mathcal{PI}$ is the minimal number of leaves in a $(\mathcal{U}, \mathcal{S})$-branching tree for P, which is denoted by $\mathbf{Comp_{\mathcal{U},\mathcal{S}}(P)}$. For constraint satisfaction problems P the quantity $\mathrm{Comp}_{\mathcal{UPI}_0,\{\top\}}(P)$ equals $\prod_{v \in \mathrm{var}(\varphi)} |D_v|$, the number of all total assignments for P (partial assignments $\varphi \in \mathcal{PASS}$ with $\mathrm{var}(\varphi) = \mathrm{var}(P)$), and in fact all branching trees have the same number of leaves (since in each branch all variables have to be assigned). More interestingly, the quantity $\mathrm{Comp}_{\mathcal{U}_0,\{\top\}}(P)$ for unsatisfiable P equals the minimal number of leaves in a "backtracking tree" as well as in a "backjumping tree" (see (Baker, 1995), chapter 1)[1)], while for unsatisfiable clause-sets this quantity equals the minimal number of leaves in a resolution tree (as we will also prove formally). Assuming a maximal domain size d, obviously we have the upper bound $\#\mathrm{lvs}(T) \leq d^{n(P)}$ on the number of leaves for any $(\mathcal{U}, \mathcal{S})$-branching tree T for any $P \in \mathcal{PI}$.

At this point the problem of finding an *optimal branching variable* arises, which is a variable $v \in \mathrm{var}(P)$ such that there is a $(\mathcal{U}, \mathcal{S})$-branching tree T for P with $\#\mathrm{lvs}(T) = \mathrm{Comp}_{\mathcal{U},\mathcal{S}}(P)$ and branching on v at the root. As we have already seen, using the trivial oracles $\mathcal{U} = \mathcal{UPI}_0$ and $\mathcal{S} = \{\top\}$ (that is, we detect satisfiability resp. unsatisfiability only in case all variables have been eliminated) leads in case of constraint satisfaction problems to a trivial optimal branching variable problem (every variable is optimal). However, using these trivial oracles with *clause-sets* seems to lead already to a non-trivial problem, since

[1)] In order to subsume the behaviour of the backtracking algorithm on clause-sets (where a clause can be satisfied by setting just *one* literal in it to true), one had to consider $\mathrm{Comp}_{\mathcal{U}_0,\mathcal{S}_0}(P)$, where $\mathcal{S}_0$ is the set of CSP's P such that for all partial assignments φ the instance $\varphi * P$ is satisfiable. However, the oracle $\mathcal{S}_0$ (like the oracle of unsatisfiable CSP's P containing a constraint $C \in P$ such that already $\{C\}$ is unsatisfiable) is not considered in the AI literature, since the relations R are assumed to be given only as "oracles" (black boxes).

by setting a variable whole clauses can vanish, and with them other variables. We will not investigate further this (important) subject in this article; see (Kullmann, 2002) (under preparation) for a general approach to heuristics. Transferring Theorem 1 in (Iwama, 1997) to tree resolution (it is claimed that this transfer can be done), it follows that if finding an optimal branching variable for tree resolution (that is in our context, finding an optimal branching variable for unsatisfiable boolean clause-sets using the oracle $\mathcal{U}_0$) can be done in polynomial (quasi-polynomial, sub-exponential) time, then SAT decision is possible in polynomial (quasi-polynomial, sub-exponential) time, while the claimed transfer of Theorem 2 of (Iwama, 1997) to tree resolution implies, that finding an optimal branching variable for tree resolution is NP-hard. Considering DPLL-algorithms with elimination of unit clauses and pure literals, in (Liberatore, 2000) it is shown explicitely that finding an optimal branching variable for unsatisfiable boolean clause-sets is NP-hard; since unit clause propagation and elimination of pure literals is confluent (see Lemma 3.13 in (Kullmann, 1999b)), these reductions can be modelled by changing the operation of $\mathcal{PASS}$ to automatically include these reductions, and in this modified fipa-system over $\mathcal{CLS}$, optimal branching variables in the sense of (Liberatore, 2000) on unsatisfiable clause-sets are the optimal branching variables for oracle $\mathcal{U}_0$.

In Section 5 we will show for unsatisfiable instances $P \in \mathcal{UPI}$, that $\mathrm{Comp}_{\mathcal{U},\mathcal{S}}(P)$ ($= \mathrm{Comp}_{\mathcal{U}}(P)$, since the oracle for satisfiability is not used on unsatisfiable instances) is equal to the minimal number of leaves in a (generalised and relativised) *resolution tree refutation* of P, generalising the well-known equivalence between decision trees, semantic trees and tree resolution for the case of propositional clause-sets (see (Reckhow, 1975; Lovász et al., 1991; Urquhart, 1995); in (Kullmann, 1999b) one finds the case of relativised propositional tree resolution).

In Subsection 3.2 we introduce the notion of the *levelled height* $\boldsymbol{h(T)}$ of branching trees, which will be shown in Section 5 to yield nearly precise general upper and lower bounds for the complexity of tree resolution. The definition of $h(T)$ (depending only on the underlying tree and the information, which leaf is in $\mathcal{U}$ and which is in $\mathcal{S}$) is motivated as follows:

Consider $P \in \mathcal{PI}$. If there is a variable $v \in \mathrm{var}(P)$ and $\varepsilon \in D_v$, such that for all other $\varepsilon' \in D_v \setminus \{\varepsilon\}$ the instance $\langle v \to \varepsilon' \rangle * P$ is detected unsatisfiable by the oracle (i.e., $\langle v \to \varepsilon' \rangle * P \in \mathcal{U}$ holds), then we can reduce P satisfiability equivalently to $\langle v \to \varepsilon \rangle * P$. While if $\langle v \to \varepsilon \rangle * P$

is detected satisfiable by the oracle (i.e., $\langle v \to \varepsilon \rangle * P \in \mathcal{S}$), then P is satisfiable and we can stop.

If by iterating this process finally we can decide whether P is satisfiable or not, then we have found a $(\mathcal{U}, \mathcal{S})$-branching tree T for P of the form shown in Figure 2 (for the binary case, where for all variables we have $|D_v| = 2$). These trees are known as "input trees" in case they represent resolution refutations. We now define that "branching input trees" T have levelled height $\boldsymbol{h(T) = 1}$, and we let $\boldsymbol{\mathcal{G}_1(\mathcal{U}, \mathcal{S})}$ be the class of instances P having a $(\mathcal{U}, \mathcal{S})$-branching tree T with $h(T) \leq 1$. (In case of propositional clause-sets and using the basic oracle $\mathcal{U}_0$, the unsatisfiable instances in $\mathcal{G}_1(\mathcal{U}_0, \mathcal{S})$ are exactly those clause-sets containing an unsatisfiable "hidden" Horn formula).

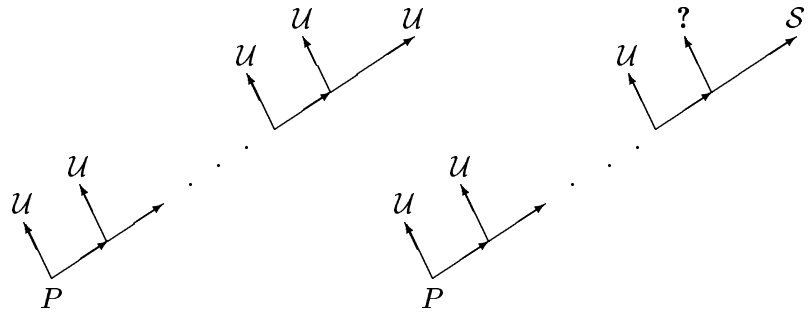

Figure 2. Branching tree T for unsatisfiable resp. satisfiable P with levelled height $h(T) \leq 1$. Symbols "$\mathcal{U}$" resp. "$\mathcal{S}$" show leaves labelled with elements of $\mathcal{U}$ resp. $\mathcal{S}$, while "?" marks a leaf labelled with an arbitrary problem instance P', that is, where the branching tree for P' is ignored.

Now let's have a look at the next level ("failed literals" (see e.g. (Li and Anbulagan, 1997)) or "generalised Unit resolution" (Kullmann, 1999a) in the case of propositional clause-sets):

If there is a variable $v \in \mathrm{var}(P)$ and $\varepsilon \in D_v$, such that for all other $\varepsilon' \in D_v \setminus \{\varepsilon\}$ the instance $\langle v \to \varepsilon' \rangle * P$ is detected unsatisfiable by the previous process, i.e., $\langle v \to \varepsilon' \rangle * P \in \mathcal{G}_1(\mathcal{U}, \mathcal{S}) \cap \mathcal{UPI}$ holds, then we can reduce P satisfiability equivalently to $\langle v \to \varepsilon \rangle * P$. While if $\langle v \to \varepsilon \rangle * P$ is detected satisfiable by the previous process, i.e., we have $\langle v \to \varepsilon' \rangle * P \in \mathcal{G}_1(\mathcal{U}, \mathcal{S}) \cap \mathcal{SPI}$, then P is satisfiable and we can stop. If by iterating this process finally we can decide whether P is satisfiable or not, then we have found a $(\mathcal{U}, \mathcal{S})$-branching tree T for P of levelled height $\boldsymbol{h(T) = 2}$. The class of all instances P having a $(\mathcal{U}, \mathcal{S})$-branching tree T with $h(T) \leq 2$ is called $\boldsymbol{\mathcal{G}_2(\mathcal{U}, \mathcal{S})}$. For the general definition of $h(T)$ and $\mathcal{G}_k(\mathcal{U}, \mathcal{S})$ see Subsection 3.2.

Before going into details, we add some references to the literature. The roots of the notion of levelled height of branching trees one finds in the hierarchy of generalised Horn clause-sets developed in (Yamasaki and Doshita, 1983; Gallo and Scutellà, 1988; Büning, 1993). In (Kullmann, 1999b) (with a forerunner in (Pretolani, 1996)) this hierarchy has been strengthened (by abstracting from syntactical properties) and relativised, arriving at the hierarchy $(\mathcal{G}_k(\mathcal{U},\mathcal{S}))_{k\in\mathbb{N}_0}$ and the notion of "levelled height" (or "hardness") for the case of propositional clause-sets. A somewhat similar attempt one finds in (Stålmarck and Säflund, 1990; Harrison, 1996; Sheeran and Stålmarck, 1998). For more information on the literature (and the close relation of levelled height to the pebble game and space complexity[2)]) the reader is referred to (Kullmann, 1999b).

3.1. The notion of a branching tree

All "trees" in this article are rooted directed trees (the edges directed towards the leaves) of finite height, but possibly having infinite branchings (in case of branching on a variable with infinite domain). The set of vertices of a tree T is denoted by $\boldsymbol{V(T)}$ and the set of edges by $\boldsymbol{E(T)} \subseteq V(T)^2$, while the root of T is denoted by $\mathbf{root(T)}$ and the set of leaves by $\mathbf{lvs(T)}$. Furthermore we use $\mathbf{ds_T(w)}$ for the set of direct successors of w in T (so that w is a leaf of T iff $\mathrm{ds_T}(w) = \emptyset$ holds), while by $\boldsymbol{T_w}$ the sub-tree of T rooted at w is denoted.

Consider $\mathcal{UPI}_0 \subseteq \boldsymbol{\mathcal{U}} \subseteq \mathcal{UPI}$ and $\top \in \boldsymbol{\mathcal{S}} \subseteq \mathcal{SPI}$ (so that $\mathcal{U}$ and $\mathcal{S}$ together cover all invariant problem instances). A **$(\boldsymbol{\mathcal{U},\mathcal{S}})$-branching tree for** $P_0 \in \mathcal{PI}$ is a tree T with labelings

$$P : V(T) \to \mathcal{PI}$$
$$v : V(T) \setminus \mathrm{lvs}(T) \to \mathcal{VA}$$
$$\varphi : E(T) \to \mathcal{PASS}$$

where $P(w)$ is called the *branching instance*, $v(w)$ the *branching variable* and $\varphi((w,w'))$ the *branching assignment*, such that the following conditions hold:

- $P(\mathrm{root}(T)) = P_0$ (the branching instance at the root is the original instance)

[2)] in case all branchings in T are binary and all leaves of T are unsatisfiable, $h(T)+1$ is the minimal number of pebbles needed to pebble T

- $w \in \mathrm{lvs}(T) \Rightarrow P(w) \in \mathcal{U} \cup \mathcal{S}$ (the branching instances at leaves are "recognised" by the oracles)
- $w \in V(T), w' \in \mathrm{ds}_{\mathrm{T}}(w) \Rightarrow P(w') = \varphi((w,w')) * P(w)$ (the branching instance at a direct successor w' of a node w is obtained by applying the branching assignment for edge (w,w') to the branching instance at w)
- if $w \in V(T) \setminus \mathrm{lvs}(T)$ then we have $v(w) \in \mathrm{var}(P(w))$ and there is a bijection τ from $\mathrm{ds}_{\mathrm{T}}(w)$ to $D_{v(w)}$ with $\varphi((w,w')) = \langle v(w) \to \tau(w') \rangle$ for all $w' \in \mathrm{ds}_{\mathrm{T}}(w)$ (for inner nodes w the branching variable $v(w)$ at w occurs in the branching instance at w, and the outgoing edges at w are in direct correspondence to the possible elementary partial assignments $\langle v \to \varepsilon \rangle$ for values $\varepsilon \in D_{v(w)}$ in the domain of $v(w)$).

Since a branching variable must occur in the associated branching instance, and then becomes eliminated along all branches, no branching variable can appear more than once on any path in a branching tree. For a vertex $w \in V(T)$ let $\boldsymbol{\varphi(w)}$ denote the composition of branching assignments along the path from the root to w (notice that the order of composition does not matter here), so that we have $P(w) = \varphi(w) * P_0$. Some basic properties of branching trees are given in the next lemma.

Lemma 3.1 *For every problem instance $P_0 \in \mathcal{PI}$ there exists a $(\mathcal{U},\mathcal{S})$-branching tree, where in case P_0 is not trivial the branching variable at the root may be any variable in* $\mathrm{var}(P_0)$. *If $P_0 \in \mathcal{SPI}$ then every $(\mathcal{U},\mathcal{S})$-branching tree contains a leaf w with $P(w) \in \mathcal{S}$, while in case of $P_0 \in \mathcal{UPI}$ for every $(\mathcal{U},\mathcal{S})$-branching tree and any leaf w we have $P(w) \in \mathcal{U}$. For any vertex w in a $(\mathcal{U},\mathcal{S})$-branching tree T the subtree T_w rooted at w is a $(\mathcal{U},\mathcal{S})$-branching tree for $P(w) = \varphi(w) * P_0$.*

For $P \in \mathcal{PI}$ we denote by $\mathbf{Comp}_{\mathcal{U},\mathcal{S}}(\boldsymbol{P})$ the minimal number $\#\mathrm{lvs}(T)$ of leaves in a $(\mathcal{U},\mathcal{S})$-branching tree T for P. To further investigate this basic complexity measure we study the "levelled height" of branching trees in the following subsection.

3.2. Levelled height of branching trees

Consider any $(\mathcal{U},\mathcal{S})$-branching tree. We give a recursive definition of the *levelled height* $\boldsymbol{h(T)}$ of T.

If T is trivial (consists only of one node) we set $h(T) := 0$. Otherwise let $w_0 := \mathrm{root}(T)$ be the root of T. In case $P(w_0)$ is satisfiable, let the set S_c of "critical successors" be the set of $w \in \mathrm{ds}_{\mathrm{T}}(w_0)$ where $P(w)$ is also satisfiable, while in case $P(w_0)$ is unsatisfiable let S_c be the set of $w \in \mathrm{ds}_{\mathrm{T}}(w_0)$ such that $h(T_w)$ is maximal, i.e., where $h(T_w) = \max_{w' \in \mathrm{ds}_{\mathrm{T}}(w_0)} h(T_{w'})$ holds.

If now $S_c = \{w^*\}$, that is S_c has only one element, and furthermore for all other successors $w \in \mathrm{ds}_{\mathrm{T}}(w_0) \setminus \{w^*\}$ we have $h(T_w) < h(T_{w^*})$, then we set

$$h(T) := h(T_{w^*})$$

while otherwise we set

$$h(T) := 1 + \min_{w \in S_c} h(T_w).$$

For any problem instance $P \in \mathcal{PI}$ we define the *hardness* (or *levelled height*) $\boldsymbol{h_{(\mathcal{U},\mathcal{S})}(P)}$ as the minimal levelled height $h(T)$ for a $(\mathcal{U},\mathcal{S})$-branching tree T for P, and for $k \in \mathbb{N}_0$ we let $\boldsymbol{\mathcal{G}_k(\mathcal{U},\mathcal{S})}$ be the set of instances P with $h_{(\mathcal{U},\mathcal{S})}(P) \leq k$. Before proceeding with some elementary properties of the levelled height, I would like to point out the "breadth-first nature" of the notion of levelled height: In case $P(w_0)$ is satisfiable (using the above notations) we ignore the unsatisfiable branches for the "critical successors", and in case there is more than one critical successors we consider only the minimum of the levelled height for these successors due to the parameterised "breadth-first" search underlying these definitions — the branches are explored not in depth-first manner (where the search can get "lost") but in a tentative way, "guarded" by the notion of levelled height, and "giving up" in case the levelled height is exceeded.

Lemma 3.2 *For any $P \in \mathcal{PI}$ we have $h_{(\mathcal{U},\mathcal{S})}(P) \leq n(P)$.*

Proof. We use induction on $n(P)$. If $n(P) = 0$, then $P \in \mathcal{U} \cup \mathcal{S}$ must hold, and thus one node, labelled with P, is a $(\mathcal{U},\mathcal{S})$-branching tree T for P. If $n(P) > 0$, choose $v \in \mathrm{var}(P)$, and construct T as follows: The root is labelled with P, and for every $\varepsilon \in D_v$ we attach a branching tree T_ε for $\langle v \to \varepsilon \rangle * P$ with $h(T_\varepsilon) \leq n(\langle v \to \varepsilon \rangle * P) \leq n(P) - 1$, which exists by induction hypothesis. For any branching tree T we have

$$h(T) \leq 1 + \max_{w \in \mathrm{ds}(\mathrm{root}(w))} h(T_w)$$

and the assertion follows. □

Thus for every instance $P \in \mathcal{PI}$ we have $P \in \mathcal{G}_{n(P)}(\mathcal{U}, \mathcal{S})$. Some basic properties of the hierarchy $(\mathcal{G}_k(\mathcal{U}, \mathcal{S}))_{k \in \mathbb{N}_0}$ are as follows:

1. $\mathcal{U} \cup \mathcal{S} \subseteq \mathcal{G}_0(\mathcal{U}, \mathcal{S})$
2. $k \leq k' \Rightarrow \mathcal{G}_k(\mathcal{U}, \mathcal{S}) \subseteq \mathcal{G}_{k'}(\mathcal{U}, \mathcal{S})$
3. $\mathcal{U} \subseteq \mathcal{U}', \mathcal{S} \subseteq \mathcal{S}' \Rightarrow \mathcal{G}_k(\mathcal{U}, \mathcal{S}) \subseteq \mathcal{G}_k(\mathcal{U}', \mathcal{S}')$.

In the following characterisation of the levels $\mathcal{G}_k(\mathcal{U}, \mathcal{S})$ we use the notion of a *trivially enforced assignment*, which is a partial assignment $\varphi \in \mathcal{PASS}$ such every $v \in \mathrm{var}(\varphi)$ is a *trivial variable*, that is $|D_v| = 1$ holds. If φ is a trivially enforced assignment, then $\varphi * P$ and P are satisfiability equivalent for all $P \in \mathcal{PI}$. Reducing P in this way as long as possible yields a satisfiability equivalent problem P' without trivial variables. Furthermore P' is uniquely determined, namely $P' = \varphi_0 * P$, where φ_0 is the maximal trivially enforced assignment φ with $\mathrm{var}(\varphi) \subseteq \mathrm{var}(P)$.

Theorem 3.3 *$\mathcal{G}_0(\mathcal{U}, \mathcal{S})$ is the set of problem instances $P \in \mathcal{PI}$ such that there is a trivially enforced assignment φ with $\varphi * P \in \mathcal{U} \cup \mathcal{S}$. For $k \geq 1$ a problem instance $P \in \mathcal{PI}$ is in $\mathcal{G}_k(\mathcal{U}, \mathcal{S})$ if and only if either $P \in \mathcal{G}_{k-1}(\mathcal{U}, \mathcal{S})$ holds, or we have $P \notin \mathcal{G}_{k-1}(\mathcal{U}, \mathcal{S})$ and there is a variable $v \in \mathrm{var}(P)$ and $\varepsilon \in D_v$ such that either*

$$\begin{gathered} \langle v \to \varepsilon \rangle * P \in \mathcal{G}_k(\mathcal{U}, \mathcal{S}) \text{ holds,} \\ \text{and for all } \varepsilon' \in D_v \setminus \{\varepsilon\} \text{ we have} \\ \langle v \to \varepsilon' \rangle * P \in \mathcal{G}_{k-1}(\mathcal{U}, \mathcal{S}) \cap \mathcal{UPI}, \end{gathered}$$

or we have

$$\langle v \to \varepsilon \rangle * P \in \mathcal{G}_{k-1}(\mathcal{U}, \mathcal{S}) \cap \mathcal{SPI}$$

(or both).

Proof. Since a branching tree T has levelled height 0 iff T is just one path, $\mathcal{G}_0(\mathcal{U}, \mathcal{S})$ is the set of all instances P which by elimination of some trivial variables become member of $\mathcal{U}$ or $\mathcal{S}$. Now let's consider the case $k \geq 1$.

For the "only if"-part of the assertion consider a $(\mathcal{U}, \mathcal{S})$-branching tree T for P with $h(T) = k \geq 1$. Let v be the branching variable at the root of T. Thus for $\varepsilon \in D_v$ there is a subtree T_ε attached at the root of T which is a $(\mathcal{U}, \mathcal{S})$-branching tree for $\langle v \to \varepsilon \rangle * P$.

First consider the case that P is satisfiable. Choose $\varepsilon \in D_v$ such that $\langle v \to \varepsilon \rangle * P$ is satisfiable and for all other $\varepsilon' \in D_v \setminus \{\varepsilon\}$ with

satisfiable $\langle v \to \varepsilon' \rangle * P$ we have $h(T_\varepsilon) \leq h(T_{\varepsilon'})$. If $h(T_\varepsilon) \leq k-1$ is the case, then we have $\langle v \to \varepsilon \rangle * P \in \mathcal{G}_{k-1}(\mathcal{U},\mathcal{S}) \cap \mathcal{SPI}$ as desired. So assume $h(T_\varepsilon) = k$ (and thus $\langle v \to \varepsilon \rangle * P \in \mathcal{G}_k(\mathcal{U},\mathcal{S})$). By definition of levelled height and the choice of ε, for all $\varepsilon' \in D_v \setminus \{\varepsilon\}$ the instance $\langle v \to \varepsilon' \rangle * P$ must be unsatisfiable and $h(T_{\varepsilon'}) < h(T_\varepsilon)$ must hold, that is $\langle v \to \varepsilon' \rangle * P \in \mathcal{G}_{k-1}(\mathcal{U},\mathcal{S}) \cap \mathcal{UPI}$, and the case of satisfiable P is completed.

Now assume that P is unsatisfiable. By definition of $h(T)$ there is at most one $\varepsilon \in D_v$ with $h(T_\varepsilon) = h(T) = k$. Thus there is $\varepsilon \in D_v$ such that for all $\varepsilon' \in D_v \setminus \{\varepsilon\}$ we have $\langle v \to \varepsilon' \rangle * P \in \mathcal{G}_{k-1}(\mathcal{U},\mathcal{S}) \cap \mathcal{UPI}$, while (trivially) $\langle v \to \varepsilon \rangle * P \in \mathcal{G}_k(\mathcal{U},\mathcal{S})$ is the case.

Altogether the "only if" part is proven. For the "if" part first consider P such that there is $v \in \mathrm{var}(P)$ and $\varepsilon \in D_v$ with $\langle v \to \varepsilon \rangle * P \in \mathcal{G}_k(\mathcal{U},\mathcal{S})$ and for all other $\varepsilon' \in D_v \setminus \{\varepsilon\}$ we have $\langle v \to \varepsilon' \rangle * P \in \mathcal{G}_{k-1}(\mathcal{U},\mathcal{S}) \cap \mathcal{UPI}$. By definition there is a $(\mathcal{U},\mathcal{S})$-branching tree T_ε for $\langle v \to \varepsilon \rangle * P$ with $h(T_\varepsilon) \leq k$ and there are $(\mathcal{U},\mathcal{S})$-branching trees $T_{\varepsilon'}$ ($\varepsilon' \in D_v \setminus \{\varepsilon\}$) for $\langle v \to \varepsilon' \rangle * P$ with $h(T_{\varepsilon'}) \leq k-1$. Let the branching tree T for P have v as branching variable at the root and T_ε and the $T_{\varepsilon'}$ attached as subtrees at the root. Now by definition $h(T) \leq k$ follows, and thus $P \in \mathcal{G}_k(\mathcal{U},\mathcal{S})$.

Finally consider the case that there is $v \in \mathrm{var}(P)$ and $\varepsilon \in D_v$ with $\langle v \to \varepsilon \rangle * P \in \mathcal{G}_{k-1}(\mathcal{U},\mathcal{S}) \cap \mathcal{SPI}$. There is a $(\mathcal{U},\mathcal{S})$-branching tree T_ε for $\langle v \to \varepsilon \rangle * P$ with $h(T_\varepsilon) \leq k-1$. Let the branching tree T for P have v as branching variable at the root and T_ε be the subtree associated with the branch $\langle v \to \varepsilon \rangle * P$, while all other branches are arbitrary. Then by definition we have $h(T) \leq h(T_\varepsilon) + 1 \leq k$, and thus $P \in \mathcal{G}_k(\mathcal{U},\mathcal{S})$. □

Corollary 3.4 *For all $p, q \in \mathbb{N}_0$ we have*

$$\mathcal{G}_{p+q}(\mathcal{U},\mathcal{S}) = \mathcal{G}_p\left(\mathcal{G}_q(\mathcal{U},\mathcal{S}) \cap \mathcal{UPI},\ \mathcal{G}_q(\mathcal{U},\mathcal{S}) \cap \mathcal{SPI}\right).$$

Proof. For abbreviation we use $\mathcal{G}_k := \mathcal{G}_k(\mathcal{U},\mathcal{S})$ and $\mathcal{G}_k^0 := \mathcal{G}_k \cap \mathcal{UPI}$, $\mathcal{G}_k^1 := \mathcal{G}_k \cap \mathcal{SPI}$ for $k \in \mathbb{N}_0$. We prove the claim by induction on p.

First we show $\mathcal{G}_{0+q} = \mathcal{G}_0(\mathcal{G}_q^0, \mathcal{G}_q^1)$. The inclusion $\mathcal{G}_q \subseteq \mathcal{G}_0(\mathcal{G}_q^0, \mathcal{G}_q^1)$ holds trivially. So consider $P \in \mathcal{G}_0(\mathcal{G}_q^0, \mathcal{G}_q^1)$. By Theorem 3.3 there is a trivially enforced assignment φ with $\varphi * P \in \mathcal{G}_q$. If $q = 0$, then by Theorem 3.3 also $P \in \mathcal{G}_q$. And if $P \in \mathcal{G}_{q-1}$, so also $P \in \mathcal{G}_q$. So assume $q \geq 1$ and $P \notin \mathcal{G}_{q-1}$, and we prove $P \in \mathcal{G}_q$ by induction on $n(\varphi)$ (for fixed $q \geq 1$, arbitrary $P \notin \mathcal{G}_{q-1}$ and any trivially enforced assignment φ with $\varphi * P \in \mathcal{G}_q$).

If $n(\varphi) = 0$, then $\varphi * P = P$. So assume $n(\varphi) \geq 1$, and thus $\varphi = \varphi' \circ \langle v \to \varepsilon \rangle$ for some partial assignment φ' with $n(\varphi') = n(\varphi) - 1$ and for some variable v and $\varepsilon \in D_v$. Let $P' := \langle v \to \varepsilon \rangle * P$. We know $\varphi' * P' \in \mathcal{G}_q$, and therefore by induction hypothesis $P' \in \mathcal{G}_q$. Now with the first main case of Theorem 3.3 (since there is no $\varepsilon' \in D_v \setminus \{\varepsilon\}$) we get $P \in \mathcal{G}_q$. Altogether we have proven $\mathcal{G}_q = \mathcal{G}_0(\mathcal{G}_q^0, \mathcal{G}_q^1)$.

We now prove the equality $\mathcal{G}_{q+1} = \mathcal{G}_1(\mathcal{G}_q^0, \mathcal{G}_q^1)$, that is $P \in \mathcal{G}_{q+1} \Leftrightarrow P \in \mathcal{G}_1(\mathcal{G}_q^0, \mathcal{G}_q^1)$, using induction on $n(P)$ for both directions. The induction hypothesis is trivial in both directions, so we assume $n(P) > 0$. First consider $P \in \mathcal{G}_{q+1}$, and we want to show $P \in \mathcal{G}_1(\mathcal{G}_q^0, \mathcal{G}_q^1)$. By Theorem 3.3 either we have $P \in \mathcal{G}_q$, and thus $P \in \mathcal{G}_0(\mathcal{G}_q^0, \mathcal{G}_q^1) \subseteq \mathcal{G}_1(\mathcal{G}_q^0, \mathcal{G}_q^1)$, or there is $v \in \mathrm{var}(P)$ and $\varepsilon \in D_v$, and we have to consider two cases:

(i) $\langle v \to \varepsilon \rangle * P \in \mathcal{G}_{q+1}$, and for $\varepsilon' \in D_v \setminus \{\varepsilon\}$ we have $\langle v \to \varepsilon' \rangle * P \in \mathcal{G}_q^0$;

(ii) $\langle v \to \varepsilon \rangle * P \in \mathcal{G}_q^1$.

In case (i) by induction hypothesis we know $\langle v \to \varepsilon \rangle * P \in \mathcal{G}_1(\mathcal{G}_q^0, \mathcal{G}_q^1)$, and then the first main case of Theorem 3.3 yields $P \in \mathcal{G}_1(\mathcal{G}_q^0, \mathcal{G}_q^1)$. And in case (ii) immediately $P \in \mathcal{G}_1(\mathcal{G}_q^0, \mathcal{G}_q^1)$ follows. Thus direction "$\Rightarrow$" is shown. The other direction follows in a similar fashion, using the other directions of the respective equivalences of Theorem 3.3.

Finally for arbitrary $p \geq 0$ by induction we have

$$\mathcal{G}_{p+1}(\mathcal{G}_q^0, \mathcal{G}_q^1) = \mathcal{G}_1(\mathcal{G}_p^0(\mathcal{G}_q^0, \mathcal{G}_q^1), \mathcal{G}_p^1(\mathcal{G}_q^0, \mathcal{G}_q^1)) = \mathcal{G}_1(\mathcal{G}_{p+q}^0, \mathcal{G}_{p+q}^1) = \mathcal{G}_{p+q+1},$$

using first the assertion in the special case just proven, then the induction hypothesis, and then the special case again. □

4. A general SAT decision scheme

In the preceeding Section 3 we have introduced a cumulative hierarchy $(\mathcal{G}_k(\mathcal{U}, \mathcal{S}))_{k \in \mathbb{N}_0}$ of classes of problem instances, covering altogether whole $\mathcal{PI}$. In this section now we show that each level $\mathcal{G}_k(\mathcal{U}, \mathcal{S})$ has a poly-time recognition algorithm $\mathfrak{G}(P, k)$, which as well solves in polynomial time the satisfiability problem for instances $P \in \mathcal{G}_k(\mathcal{U}, \mathcal{S})$ (see Theorem 4.3). The algorithm is as follows (see Figure 3 for a more detailed description), where we have to allow that the algorithm "gives

up" without an answer — in order to guarantee that in the "don't know"-case in fact the levelled height of the input P is (strictly) greater than k we need an additional assumption on the oracles $\mathcal{U}$ and $\mathcal{S}$ (until now besides $\mathcal{U}_0 \subseteq \mathcal{U}$ and $\top \in \mathcal{S}$ the oracles have been arbitrary).

$\mathfrak{G}(P \in \mathcal{PI}, k \in \mathbb{N}_0) : \{$"satisfiable", "unsatisfiable", "don't know"$\}$

1. a) If $P \in \mathcal{S}$, then $\mathfrak{G}(P,k) :=$ "satisfiable".
   b) If $P \in \mathcal{U}$, then $\mathfrak{G}(P,k) :=$ "unsatisfiable".
2. If there is a variable $v \in \mathrm{var}(P)$ and a value $\varepsilon \in D_v$ such that we get $\mathfrak{G}(\langle v \to \varepsilon \rangle * P, k-1) =$ "satisfiable", then $\mathfrak{G}(P,k) :=$ "satisfiable".
3. If there is a variable $v \in \mathrm{var}(P)$ and a value $\varepsilon \in D_v$, such that the assignment $\langle v \to \varepsilon \rangle$ has been detected to be "enforced" at level $k-1$, that is, for all other values $\varepsilon' \in D_v \setminus \{\varepsilon\}$ we have $\mathfrak{G}(\langle v \to \varepsilon' \rangle * P, k-1) =$ "unsatisfiable", then P and $\langle v \to \varepsilon \rangle * P$ are satisfiability equivalent (that is, either both P and $\langle v \to \varepsilon \rangle * P$ are satisfiable, or both P and $\langle v \to \varepsilon \rangle * P$ are unsatisfiable), and we set $\mathfrak{G}(P,k) := \mathfrak{G}(\langle v \to \varepsilon \rangle * P, k)$.
4. Otherwise $\mathfrak{G}(P,k) :=$ "don't know".

The crucial point here is, that when applying the enforced assignment $\langle v \to \varepsilon \rangle$ in step 3, we apply $\mathfrak{G}$ to $\langle v \to \varepsilon \rangle * P$ *at the same level* k. So to get a polynomial upper bound on the running time, it is essential that there is no backtracking in algorithm $\mathfrak{G}$, that is, the application of an enforced assignment is never made undone later.

How to interprete the outcome of algorithm $\mathfrak{G}$? Clearly, if $\mathfrak{G}(P,k)$ returns "satisfiable" or "unsatisfiable", then indeed P is satisfiable resp. unsatisfiable, and furthermore $P \in \mathcal{G}_k(\mathcal{U},\mathcal{S})$ holds (due to Theorem 3.3). Now to ensure that in case the output is "don't know" we can conclude $P \notin \mathcal{G}_k(\mathcal{U},\mathcal{S})$, we have to make sure that application of an enforced assignment in step 3 does not destroy membership in $\mathcal{G}_k(\mathcal{U},\mathcal{S})$, so that we never have to branch on different possibilities for enforced assignments. In other words, we have to assume that $\mathcal{G}_k(\mathcal{U},\mathcal{S})$ is *stable under enforced assignments*. Fortunately, as shown in Lemma 4.2, it suffices to demand that only $\mathcal{U}$ and $\mathcal{S}$ are stable under enforced assignments.

From the parameterised algorithm $\mathfrak{G}(P,k)$ we obtain a general satisfiability algorithm $\widehat{\mathfrak{G}}(P)$ by simply applying $\mathfrak{G}(P,0), \mathfrak{G}(P,1), \ldots$, until

the output is "satisfiable" or "unsatisfiable". Some general remarks on $\widehat{\mathfrak{G}}$ one finds in Subsection 4.2. In Section 5 we will show that $\widehat{\mathfrak{G}}$ accomplishes "quasi-automatisation" of tree resolution (for general systems of problem instances and arbitrary oracles $\mathcal{U}$ and $\mathcal{S}$ stable under enforced assignments), that is, the running time of $\widehat{\mathfrak{G}}(P)$ on unsatisfiable P is quasi-polynomial in the length of a shortest tree resolution refutation for P.

Finally some remarks on the literature (again, for an extensive treatment (in the case of propositional clause-sets) the reader is referred to (Kullmann, 1999b)): The algorithm $\mathfrak{G}(P,k)$ has been presented for the case of propositional clause-sets in (Kullmann, 1999b), with a forerunner in (Pretolani, 1996) and based on the ideas of (Yamasaki and Doshita, 1983; Gallo and Scutellà, 1988; Büning, 1993). A similar algorithm is Stålmarck's algorithm — see (Stålmarck and Säflund, 1990; Harrison, 1996; Sheeran and Stålmarck, 1998) and also (Groote and Warners, 2000). For literature on automatisability of (tree) resolution see (Alekhnovich and Razborov, 2001).

4.1. Decision of the levels of the hierarchy in poly-time

In Figure 3 (page 23) an algorithm $\mathfrak{G}(P,k)$ is given (using oracles $\mathcal{U}$ and $\mathcal{S}$), deciding satisfiability of P in case of $P \in \mathcal{G}_k(\mathcal{U},\mathcal{S})$ as well as whether P belongs to $\mathcal{G}_k(\mathcal{U},\mathcal{S})$ at all. Obviously algorithm $\mathfrak{G}$ terminates for any input, and in case the output of $\mathfrak{G}(P,k)$ is "satisfiable" or "unsatisfiable" we have $P \in \mathcal{G}_k(\mathcal{U},\mathcal{S}) \cap \mathcal{SPI}$ resp. $P \in \mathcal{G}_k(\mathcal{U},\mathcal{S}) \cap \mathcal{UPI}$. To ensure that in case the output is "higher level" indeed $P \notin \mathcal{G}_k(\mathcal{U},\mathcal{S})$ holds, we need the (natural) assumption on the "oracles" $\mathcal{S}$ and $\mathcal{U}$ to be "stable under application of enforced assignments":

Call an assignment $\langle v \to \varepsilon\rangle \in \mathcal{PASS}$ *enforced for* $P \in \mathcal{PI}$, if for all $\varepsilon' \in D_v \setminus \{\varepsilon\}$ the instances $\langle v \to \varepsilon'\rangle * P$ are unsatisfiable. Note that a trivially enforced assignment $\langle v \to \varepsilon\rangle$ is enforced for any $P \in \mathcal{PI}$, and that any assignment $\langle v \to \varepsilon\rangle$ is enforced for $P \in \mathcal{UPI}$. Now a class $\mathcal{C} \subseteq \mathcal{PI}$ of problem instances is called *stable under application of enforced assignments*, if for all instances $P \in \mathcal{C}$ and all enforced assignments $\langle v \to \varepsilon\rangle$ for P we have $\langle v \to \varepsilon\rangle * P \in \mathcal{C}$. In case of $\mathcal{C} \subseteq \mathcal{UPI}$, stability under application of *enforced* assignments is equivalent to stability under application of *all* assignments. If $\langle v \to \varepsilon\rangle$ is enforced for $P \in \mathcal{SPI}$ then also $\langle v \to \varepsilon\rangle * P$ is satisfiable.

$\mathfrak{G}(P \in \mathcal{PI};\ k \in \mathbb{Z}) : \{$"satisfiable", "unsatisfiable", "higher level"$\}$
BEGIN
  FOR $v \in \mathrm{var}(P)$ DO
    IF there is $\varepsilon_0 \in D_v$ with $D_v = \{\varepsilon_0\}$ THEN
(*) $P := \langle v \to \varepsilon_0 \rangle * P$;
  IF $P \in S$ THEN RETURN "satisfiable";
  F $P \in U$ THEN RETURN "unsatisfiable";
  IF $k = 0$ THEN RETURN "higher level";
  FOR $v \in \mathrm{var}(P)$ DO
    $nhl := 0$; { number of branches yielding only "higher level" }
    FOR $\varepsilon \in D_v$ DO
      IF $\mathfrak{G}(\langle v \to \varepsilon \rangle * P, k-1) =$ "satisfiable" THEN
        RETURN "satisfiable"
      ELSE IF $\mathfrak{G}(\langle v \to \varepsilon \rangle * P, k-1) =$ "higher level" THEN
        $nhl := nhl + 1$; $\varepsilon_0 := \varepsilon$;
    END (FOR ε);
    IF $nhl = 0$ THEN { all branches were "unsatisfiable" }
      RETURN "unsatisfiable"
    ELSE IF $nhl = 1$ THEN
(*)   RETURN $\mathfrak{G}(\langle v \to \varepsilon_0 \rangle * P, k)$;
  END (FOR v);
  RETURN "higher level";
END $\mathfrak{G}$.

Figure 3. Algorithm $\mathfrak{G}(P, k)$ for decision whether problem instance P lies in level k of the hierarchy $(\mathcal{G}_k)_k$ or not, and whether P in the affirmative case is satisfiable or not

Lemma 4.1 *Consider $P \in \mathcal{PI}$ and $\varphi \in \mathcal{PASS}$ and assume that the assignment $\langle v \to \varepsilon \rangle$ is enforced for P.*

1. *If $\varphi * P$ is satisfiable, then φ and $\langle v \to \varepsilon \rangle$ are compatible.*
2. *If $v \in \mathrm{var}(\varphi * P)$, then $\langle v \to \varepsilon \rangle$ is enforced also for $\varphi * P$.*

Proof. Part 1 is obvious. For part 2 consider $\varepsilon' \in D_v \setminus \{\varepsilon\}$. From the assumption $v \in \mathrm{var}(\varphi * P)$ we get $v \notin \mathrm{var}(\varphi)$, and thus

$$\langle v \to \varepsilon' \rangle * (\varphi * P) = \varphi * (\langle v \to \varepsilon' \rangle * P) \in \mathcal{UPI},$$

whence $\langle v \to \varepsilon \rangle$ is enforced for $\varphi * P$. □

In (Kullmann, 2003) the notion of an "always-safe" assignment φ for a problem instance P has been introduced, which is defined by the condition that for all partial assignments ψ we have $\psi * (\varphi * P) \equiv \psi * P$, where $\equiv$ denotes that two instances are "instantiationally equivalent". Using the canonical translation of fipa-systems into qo-systems with finite instantiation by partial assignments mentioned in the introduction to Section 2, the relation $\equiv$ just denotes satisfiability equivalence, and by Lemma 4.1, part 1 we see, that every enforced assignment φ for P is also always-safe for P.

Lemma 4.2 *If the oracles $\mathcal{S}$ and $\mathcal{U}$ are stable under application of enforced assignments, then so are all levels $\mathcal{G}_k(\mathcal{U},\mathcal{S})$ for $k \in \mathbb{N}_0$.*

Proof. By Corollary 3.4 we only have to show that $\mathcal{G}_0(\mathcal{U},\mathcal{S})$ and $\mathcal{G}_1(\mathcal{U},\mathcal{S})$ are stable under application of enforced assignment. So consider an enforced assignment $\langle w \to \eta\rangle$ for $P \in \mathcal{PI}$.

First consider the case $P \in \mathcal{G}_0(\mathcal{U},\mathcal{S})$. By Theorem 3.3 there is a trivially enforced assignment ψ with $\psi * P \in \mathcal{U} \cup \mathcal{S}$. By assumption and Lemma 4.1, part 2 we get

$$\langle w \to \eta\rangle * (\psi * P) \in \mathcal{U} \cup \mathcal{S}.$$

$\langle w \to \eta\rangle$ must be compatible with $\psi * P$, whence

$$\langle w \to \eta\rangle * (\psi * P) = \psi * (\langle w \to \eta\rangle * P) \in \mathcal{U} \cup \mathcal{S}$$

and so by Theorem 3.3 we get $\langle w \to \eta\rangle * P \in \mathcal{G}_0(\mathcal{U},\mathcal{S})$.

Now consider the case $P \in \mathcal{G}_1(\mathcal{U},\mathcal{S})$. By induction on $n(P)$ we will show $\langle w \to \eta\rangle * P \in \mathcal{G}_1(\mathcal{U},\mathcal{S})$. If $\langle w \to \eta\rangle * P \in \mathcal{G}_0(\mathcal{U},\mathcal{S})$ then we are done. So assume $\langle w \to \eta\rangle * P \notin \mathcal{G}_0(\mathcal{U},\mathcal{S})$. First consider the case that there is $v \in \mathrm{var}(P)$ and $\varepsilon \in D_v$ with $\langle v \to \varepsilon\rangle * P \in \mathcal{G}_0(\mathcal{U},\mathcal{S}) \cap \mathcal{SPI}$. By stability of $\mathcal{G}_0(\mathcal{U},\mathcal{S})$ under enforced assignments and Lemma 4.1, part 2 we get

$$\langle w \to \eta\rangle * (\langle v \to \varepsilon\rangle * P) \in \mathcal{G}_0(\mathcal{U},\mathcal{S}) \cap \mathcal{SPI}.$$

Since $\langle v \to \varepsilon\rangle * P$ is satisfiable, by Lemma 4.1, part 1 the assignments $\langle v \to \varepsilon\rangle$ and $\langle w \to \eta\rangle$ are compatible, and therefore

$$\langle w \to \eta\rangle * (\langle v \to \varphi\rangle * P) = \langle v \to \varepsilon\rangle * (\langle w \to \eta\rangle * P) \in \mathcal{G}_0(\mathcal{U},\mathcal{S}) \cap \mathcal{SPI}$$

whence by Theorem 3.3 we get $\langle w \to \eta\rangle * P \in \mathcal{G}_1(\mathcal{U},\mathcal{S})$.

Now consider the case that there is $v \in \mathrm{var}(P)$ and $\varepsilon \in D_v$ such that $\langle v \to \varepsilon\rangle * P \in \mathcal{G}_1(\mathcal{U},\mathcal{S})$ holds, and for all other $\varepsilon' \in D_v \setminus \{\varepsilon\}$

we have $\langle v \to \varepsilon' \rangle * P \in \mathcal{G}_0(\mathcal{U}, \mathcal{S}) \cap \mathcal{UPI}$. If the assignments $\langle w \to \eta \rangle$ and $\langle v \to \varepsilon \rangle$ are not compatible, then for some $\varepsilon' \in D_v \setminus \{\varepsilon\}$ we have $\langle w \to \eta \rangle = \langle v \to \varepsilon' \rangle$, and thus $\langle w \to \eta \rangle * P \in \mathcal{G}_0(\mathcal{U}, \mathcal{S}) \subseteq \mathcal{G}_1(\mathcal{U}, \mathcal{S})$, while in case of $\langle w \to \eta \rangle = \langle v \to \varepsilon \rangle$ we also have $\langle w \to \eta \rangle * P \in \mathcal{G}_1(\mathcal{U}, \mathcal{S})$. So assume $w \neq v$. Now by induction hypothesis (and using Lemma 4.1, part 2) we get

$$\langle v \to \varepsilon \rangle * (\langle w \to \eta \rangle * P) = \langle w \to \eta \rangle * (\langle v \to \varepsilon \rangle * P) \in \mathcal{G}_1(\mathcal{U}, \mathcal{S})$$
$$\langle v \to \varepsilon' \rangle * (\langle w \to \eta \rangle * P) = \langle w \to \eta \rangle * (\langle v \to \varepsilon' \rangle * P) \in \mathcal{G}_0(\mathcal{U}, \mathcal{S}) \cap \mathcal{UPI}$$

whence by Theorem 3.3 we get $\langle w \to \eta \rangle * P \in \mathcal{G}_1(\mathcal{U}, \mathcal{S})$. □

To ensure termination of algorithm $\mathfrak{G}$ we assume that the maximal size $\boldsymbol{d} := \sup_{v \in \mathcal{VA}} |D_v|$ of the domains of variables is a finite number.

Theorem 4.3 *Assume that $\mathcal{U}$ and $\mathcal{S}$ are stable under application of enforced assignments. Then algorithm $\mathfrak{G}$ is correct, that is, for all inputs $P \in \mathcal{PI}$ and $k \in \mathbb{Z}$ we have $\mathfrak{G}(P, k) =$ "higher level" if and only if $P \notin \mathcal{G}_k(\mathcal{U}, \mathcal{S})$ (i.e., if $h_{\mathcal{U},\mathcal{S}}(F) > k$ holds), while in case of $\mathfrak{G}(P, k) =$ "unsatisfiable" we have $P \in \mathcal{G}_k(\mathcal{U}, \mathcal{S}) \cap \mathcal{UPI}$, and in case of $\mathfrak{G}(P, k) =$ "satisfiable" we have $P \in \mathcal{G}_k(\mathcal{U}, \mathcal{S}) \cap \mathcal{SPI}$. The running time of $\mathfrak{G}$ in the oracle model, where computation of $\mathrm{var}(P)$ and $\langle v \to \varepsilon \rangle * P$ as well as the questions "$P \in \mathcal{U}$?" and "$P \in \mathcal{S}$?" count as one step, is bounded by $O((n(P)+1)^{d \cdot \max(k,1)})$, and thus for all constants $k \in \mathbb{N}_0$ membership in $\mathcal{G}_k(\mathcal{U}, \mathcal{S})$ as well as satisfiability for inputs $P \in \mathcal{G}_k(\mathcal{U}, \mathcal{S})$ is decidable in polynomial time.*

Proof. With respect to correctness the only critical points in algorithm $\mathfrak{G}$ are marked by $(*)$, where we apply enforced assignments $\langle v \to \varepsilon_0 \rangle$, which could increase the hardness of P. But here Lemma 4.2 guarantees that if $P \in \mathcal{G}_k(\mathcal{U}, \mathcal{S})$ is the case, then also $\langle v \to \varepsilon_0 \rangle * P \in \mathcal{G}_k(\mathcal{U}, \mathcal{S})$ holds. For the running time consider the number of recursive calls of procedure $\mathfrak{G}(P, k)$ (inclusive the initial call). Obviously the function $f(n, k)$, $n, k \in \mathbb{N}_0$ defined by

$$f(n, 0) = f(0, k) := 1$$
$$f(n+1, k+1) := d \cdot n \cdot f(n, k) + f(n, k+1)$$

gives an upper bound on the number of recursive calls when using $n = n(P)$. Now we have $f(n, k) \leq (n+1)^{d \cdot k}$, since for $n = 0$ or $k = 0$

we have $f(n,k) = 1 = (n+1)^{d \cdot k}$, while inductively

$$f(n{+}1,k{+}1) = d{\cdot}n{\cdot}f(n,k){+}f(n,k{+}1) \le d{\cdot}n{\cdot}(n{+}1)^{d\cdot k}{+}(n{+}1)^{d\cdot(k+1)}$$
$$\le d \cdot (n+1)^{d\cdot(k+1)-1} + (n+1)^{d\cdot(k+1)} < ((n+1)+1)^{d\cdot(k+1)}$$

follows. The "max"-part in the upper bound is due to the elimination of trivially enforced assignments, which needs to be done only once, the first time the procedure is called. □

4.2. Running through the levels

In Figure 4 (page 26) the general satisfiability decision algorithm $\widehat{\mathfrak{G}}(P)$ for inputs $P \in \mathcal{PI}$ is given, computing also the hardness $h_{\mathcal{U},\mathcal{S}}(P)$.

PROCEDURE $\widehat{\mathfrak{G}}(P \in \mathcal{PI}) : \{\text{"satisfiable"}, \text{"unsatisfiable"}\} \times \mathbb{N}_0$;
BEGIN
  FOR $k = 0$ to $n(F)$ DO
    IF $\mathfrak{G}(P,k) \neq$ "higher level" THEN
      RETURN $(\mathfrak{G}(P,k), k)$
  END (FOR k)
END $\widehat{\mathfrak{G}}$.

Figure 4. The SAT decision algorithm $\widehat{\mathfrak{G}}$, computing also $h_{\mathcal{U},\mathcal{S}}(P)$

Lemma 4.4 *If oracles $\mathcal{U}$ and $\mathcal{S}$ are stable under application of enforced assignments, then algorithm $\widehat{\mathfrak{G}}$ terminates and is correct, that is for all input instances $P \in \mathcal{PI}$ the output is equal to ("satisfiable", k) or ("unsatisfiable", k) if and only if $h_{\mathcal{U},\mathcal{S}}(P) = k$ holds and $P \in \mathcal{SPI}$ or $P \in \mathcal{UPI}$ is the case. The running time in the oracle model is bounded by*

$$O((n(P)+1)^{d\cdot\max(h_{\mathcal{U},\mathcal{S}}(P),1)})$$

Some final remarks:

1. For $P \in \mathcal{PI}$ let

$$\#(P) := |\{\varphi \in \mathcal{PASS} : \mathrm{var}(\varphi) = \mathrm{var}(P) \wedge \varphi * P = \top\}|$$

be the number of satisfiable assignments. We have $\#(\top) = 1$, while $\#(P) = 0 \Leftrightarrow P \in \mathcal{UPI}$. As shown in Subsection 3.1.1 of (Kullmann, 1999b) for the case of propositional clause-sets, if we do

not stop the computation of algorithm $\mathcal{G}(P_0, k)$ in case we found P_0 satisfiable, but we have to process all branches, then without compromising the bounds from Theorem 4.3 and Lemma 4.4 we can actually compute $\#(P_0)$ (in case the output is not "higher level") by adding the contributions

$$\#(P(w)) \cdot \prod_{v \in \mathrm{var}(P_0) \setminus (\mathrm{var}(\varphi(w)) \cup \mathrm{var}(P(w)))} |D_v|$$

of each leaf w of T.

2. For upper and lower bounds on $h_{\mathcal{U},\mathcal{S}}(P)$ in the case of propositional clause-sets see sections 3.4, 5 and 6 in (Kullmann, 1999b).

3. The process of applying enforced assignments by algorithm $\mathcal{G}(P, k)$ can be isolated and yields a canonical hierarchy of (relativised) reductions generalising Unit clause propagation. See Subsection 3.3 in (Kullmann, 1999b).

5. Generalised tree resolution and input resolution

In the case of propositional clause-sets it is well known that interpreting the *partial assignments* $\varphi(w)$ *as clauses* $C_{\varphi(w)}$ at the nodes w of branching trees for clause-sets F we (essentially) obtain exactly all tree resolution refutations of F; see (Lovász et al., 1991; Urquhart, 1995; Kullmann, 1999b). In the same spirit, in Subsection 5.1 from the notion of $\mathcal{U}$-branching trees for unsatisfiable problem instances we obtain the definition of the generalised resolution rule (already considered in (Baker, 1995), definition 5.11) and the notion of resolution trees $T : F \vdash C$, deriving (generalised) clauses $C \in \mathcal{CL}$ from (generalised) clause-sets $F \in \mathcal{CLS}$. And in a second step we obtain the general definition of a resolution tree $T : P \overset{\mathcal{U}}{\vdash} C$ for problem instances $P \in \mathcal{PI}$ and oracles $\mathcal{UPI}_0 \subseteq \mathcal{U} \subseteq \mathcal{UPI}$, using the relation $P \xrightarrow{\mathcal{U}} F$ of *direct entailment* between problem instances and clause-sets (the relation of direct entailment for constraint satisfaction problems in (Baker, 1995), definition 5.10, is the special case when using the basic oracle $\mathcal{U}_0$ for constraint satisfaction problems). The exact correspondence between branching trees and tree resolution is worked out in Subsection 5.2, showing in Theorem 5.10 that $\mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(P)$, the minimal number of leaves in a tree resolution refutation of an unsatisfiable instance P, is equal to the minimal number of leaves in a branching tree for P.

The definition of levelled height $h(T)$ from Subsection 3.2 is interpreted for resolution trees in Subsection 5.3, yielding the natural interpretation of $h(T) \le k$ as "k-times nested input resolution", generalising (Kullmann, 1999b), where this notion has been introduced for propositional clause-sets, and where one finds also the (tight) connection to pebble games and the space complexity of tree resolution[3)]. In Subsection 5.4 we show that $h_{\mathcal{U}}(P)$ yields good general bounds for tree resolution:

$$2^{h_{\mathcal{U}}(P)} \le \mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(P) \le (n(P) \cdot (d-1)+1)^{h_{\mathcal{U}}(P)},$$

where d is the maximal domain size. It follows immediately, that the running time of algorithm $\widehat{\mathfrak{G}}$ (see Subsection 4.2) on unsatisfiable inputs

[3)] Space complexity of generalised resolution for constraint satisfaction problems has also been considered in (Baker, 1995), subsection 5.6 — the "cache size" of a resolution proof as defined in definitions 5.37, 5.38 equals the pebbling complexity of the resolution graph as considered in (Esteban and Torán, 1999; Torán, 1999) (when using generalised resolution).

$P \in \mathcal{UPI}$ with $\mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(P) > 1$ is bounded by

$$O\big(\mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(P)^{d\cdot\log_2(n(P)+1)}\big),$$

and thus (generalised) tree resolution is "quasi-automatisable."

The notion of "automatisable proof systems" has been introduced in (Bonet et al., 1997) (one can find a proof in time polynomial in the length of a shortest proof) with the intention to prove that some proof systems are *not* automatisable, and further explored in this direction in (Bonet et al., 1999; Alekhnovich et al., 2001; Alekhnovich and Razborov, 2001). Quasi-automatisation of tree resolution for the case of propositional clause-sets is implicitly contained in (Clegg et al., 1996) together with (Beame and Pitassi, 1996), was asked as a question in (Ben-Sasson and Wigderson, 1999) and has been explicitely proven in (Kullmann, 1999b) for the relativised propositional case.

Finally I would like to mention an interesting connection between (generalised) resolution and autarkies for generalised clause-sets. In (Kullmann, 2000b) (Theorem 3.16) it has been shown that a propositional clause-set F has no non-trivial autarky if and only if every clause of F can be used in a tree resolution refutation of F. This kind of "duality" between resolution and autarkies has been extended to generalised clause-sets in (Kullmann, 2003).

5.1. Generalised tree resolution

Given clauses $C_\varepsilon \uplus \{(v,\varepsilon)\} \in \mathcal{CL}$ for each $\varepsilon \in D_v$ ("$\uplus$" stands for disjoint union), the *resolvent* of the set $\{C_\varepsilon \cup \{(v,\varepsilon)\}\}_{\varepsilon\in D_v}$ of *parent clauses* is the clause $R = \bigcup_{\varepsilon\in D_v} C_\varepsilon$, provided that R in fact is a clause ($R \in \mathcal{CL}$), that is, for all literals $x \in C_{\varepsilon_1}$ and $y \in C_{\varepsilon_2}$ in case of $\mathrm{var}(x) = \mathrm{var}(y)$ we also have $x = y$. Variable v is called the *resolution variable* and is uniquely determined by the set of parent clauses.

A *resolution tree* T is a tree together with a labelling of its vertices w by clauses $C(w)$ such each non-leaf node is labelled by the resolvent of the clauses labelling its direct successors (as for branching trees, the edges of resolution trees are directed towards the leaves). So each non-leaf node w of T is associated with a unique resolution variable $v(w)$, and we have $|\mathrm{ds}_{\mathrm{T}}(w)| = |D_{v(w)}|$. We write $\boldsymbol{T : F \vdash C}$ if T is a resolution tree such that the root of T is labelled by C, while each clause labelling a leaf of T is element of F. Furthermore we use $\boldsymbol{F \vdash C}$ for the assertion that there is a resolution tree T with $T : F \vdash C$.

Obviously resolution is correct, that is from $F \vdash C$ follows $F \models C$. Corollary 5.8 in Subsection 5.2 gives also completeness in the sense, that if $F \models C$ holds, then there is a clause $C' \subseteq C$ with $F \vdash C'$. In the next lemma we gather some standard properties of resolution derivations, using the following notions.

For trees T, T' the relation $\boldsymbol{T \leq T'}$ holds if (and only if) T can be *embedded* into T', that is there is an injective map $f : V(T) \to V(T')$ from the nodes of T into the nodes of T' such that for nodes $w_1, w_2 \in V(T)$ there is a directed path in T from w_1 to w_2 if and only if there is a directed path from $f(w_1)$ to $f(w_2)$ in T', where the edges in a tree are directed towards the leaves. The trees T, T' are *isomorphic*, denoted by $\boldsymbol{T \cong T'}$, iff $T \leq T'$ and $T' \leq T$ holds. Note that in case $T \leq T'$ we have $\#\mathrm{lvs}(T) \leq \#\mathrm{lvs}(T')$. We remark that the relation $T \leq T'$ between (rooted) trees is the same as for example used for the proof of Theorem 12.2.1 in (Diestel, 2000).

For a clause $C \in \mathcal{CL}$ and a partial assignment $\varphi \in \mathcal{PASS}$ with $\varphi * \{C\} \neq \top$ we define $\varphi * C$ as that clause $C' \in \mathcal{CL}$ with $\varphi * \{C\} = \{C'\}$, i.e., $\varphi * C = C \setminus C_\varphi$.

Lemma 5.1 *For any clause-sets $F, F' \in \mathcal{CLS}$, clauses $C_0 \in \mathcal{CL}$, resolution trees T and partial assignments $\varphi \in \mathcal{PASS}$ we have*

1. *If $T : F \vdash C_0$, and for each $C \in F$ there is $C' \in F'$ with $C' \subseteq C$, then there is a resolution tree $T' \leq T$ and a clause $C'_0 \subseteq C_0$ with $T' : F' \vdash C'_0$.*
2. *If $T : F \vdash C_0$ and $\varphi * \{C_0\} \neq \top$, then there is a resolution tree $T' \leq T$ and a clause $C'_0 \subseteq \varphi * C_0 = C_0 \setminus C_\varphi$ with $T' : \varphi * F \vdash C'_0$.*
3. *If $T : \varphi * F \vdash C_0$, then there is a resolution tree $T' \cong T$ and a clause C'_0 with $C_0 \subseteq C'_0 \subseteq C_0 \cup C_\varphi$ and $T' : F \vdash C'_0$.*

Proof. Part 1 is proven by induction on the height of T:

If T is trivial, then $C_0 \in F$ must hold, thus there is $C'_0 \in F'$ with $C'_0 \subseteq C_0$, and we set T' as the trivial tree labelled with C'_0.

Otherwise let v be the resolution variable at $w_0 := \mathrm{root}(T)$, and for each direct successor $w \in \mathrm{ds_T}(w_0)$ apply the induction hypothesis to the resolution tree $T_w : F \vdash C(w)$ and obtain $T'_w : F' \vdash C'(w) \subseteq C(w)$ with $T'_w \leq T_w$. If there is one direct successor $w \in \mathrm{ds_T}(w_0)$ with $v \notin \mathrm{var}(C'(w))$, then let $T' := T'_w$ and $C'_0 := C'(w) \subseteq C_0$, while otherwise

let C'_0 be the resolvent of $\{C'(w) : w \in \mathrm{ds_T}(w_0)\}$, and let T' be the tree with the T'_w as its subtrees attached at the root, and with the root labelled by $C'_0 \subseteq C_0$. $\surd$

For *part 2* let clause-set F' be defined as the union of $\varphi * F$ and the set of clauses $C \setminus C_\varphi$ for such $C \in F$ with $\varphi * \{C\} = \top$. By part 1 there is a resolution tree $T' \leq T$ and a clause $C'_0 \subseteq C_0$ with $T' : F' \vdash C'_0$. Assume that there is a clause $C \in F' \setminus (\varphi * F)$ used as an axiom in T'. There is a literal $(v, \varepsilon) \in C$ with $\varphi(v) \neq \varepsilon$, and no clause in F' contains the literal $(v, \varphi(v))$. Thus there is no resolution on v in T', and whence $(v, \varepsilon) \in C'_0 \subseteq C_0$ must be case, contradicting $\varphi * C_0 \neq \top$. So we have proven that in fact $T' : \varphi * F \vdash C'_0$ must hold, and therefore also $C'_0 \subseteq \varphi * C_0$ is the case, since we have $\mathrm{var}(\varphi * F) \cap \mathrm{var}(\varphi) = \emptyset$, and thus $\mathrm{var}(C'_0) \cap \mathrm{var}(\varphi) = \emptyset$ follows. $\surd$

Part 3 again is proven by induction on the height of T:

If T is trivial, then $C_0 \in \varphi * F$ must hold, so there is $C'_0 \in F$ with $\varphi * C'_0 = C_0$ — let T' be the trivial tree labelled with C'_0.

Otherwise let v be the resolution variable at $w_0 := \mathrm{root}(T)$, and for each direct successor $w \in \mathrm{ds_T}(w_0)$ apply the induction hypothesis to the resolution tree $T_w : \varphi * F \vdash C(w)$ and obtain $T'_w : F \vdash C'(w)$ with $T'_w \cong T_w$ and $C(w) \subseteq C'(w) \subseteq C(w) \cup C_\varphi$. Since $\mathrm{var}(\varphi) \cap \mathrm{var}(\varphi * F) = \emptyset$, we have $v \notin \mathrm{var}(C_\varphi)$, and thus the clauses $C'(w)$ are resolvable, yielding the resolvent C'_0 with $C_0 \subseteq C'_0 \subseteq C_0 \cup C_\varphi$. Let T' be the tree with the trees T'_w as its subtrees attached to the root, and with the root labelled by C'_0. $\square$

In what follows we will extend our language and speak of the general system $\mathcal{PI}$ of problem instances and the special system $\mathcal{CLS}$ of clause-sets *at the same time* by considering the direct sum $\mathcal{PI} \oplus \mathcal{CLS}$ together with the canonical embeddings of $\mathcal{PI}$ and $\mathcal{CLS}$ via $P \mapsto (P, \top)$ resp. $F \mapsto (\top, F)$, which allows us to identify instances P with the pair $(P, \top)$ and clause-sets F with the pair $(\top, F)$. So now $P \models C$ for $P \in \mathcal{PI}$ and $C \in \mathcal{CL}$ is equivalent to the assertion that for all $\varphi \in \mathcal{PASS}$ with $\varphi * P = \top$ also $\varphi * \{C\} = \top$ holds, which in turn is equivalent to $\varphi_C * P \in \mathcal{UPI}$. For every clause-set F the relation $P \models F$ is equivalent to $\forall\, C \in F : P \models C$.

Consider any set $\mathcal{U}$ of unsatisfiable problem instances containing all trivial unsatisfiable instances, that is $\mathcal{UPI}_0 \subseteq \mathcal{U} \subseteq \mathcal{UPI}$. We use $\mathcal{U}$ as

an "oracle" for unsatisfiability. Define the relation

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

expressing that T is a resolution tree deriving the clause C from problem instance P by means of oracle $\mathcal{U}$, as true if (and only if) T is a resolution tree $T : F \vdash C$ for some clause-set $F \in \mathcal{CLS}$ with $\forall\, C \in F : \varphi_C * P \in \mathcal{U}$ (and thus $P \models F \models C$ follows). Since in our proofs we often will use this special relation between an instance P and a clause-set F, we spent an extra notation on it, and use $\boldsymbol{P \xrightarrow{\mathcal{U}} F}$ (read "P directly entails F w.r.t. $\mathcal{U}$") if for all clauses $C \in F$ we have $\varphi_C * P \in \mathcal{U}$. Thus $T : P \overset{\mathcal{U}}{\vdash} C$ holds iff there is a clause-set F with $P \xrightarrow{\mathcal{U}} F$ and $T : F \vdash C$.

Consider the set $\boldsymbol{\mathcal{U}_0} := \{F \in \mathcal{CLS} : \bot \in F\}$ of all clause-sets containing the empty clause. For clause-sets $F, F' \in \mathcal{CLS}$ the relation $F \xrightarrow{\mathcal{U}_0} F'$ holds iff for all $C' \in F'$ there is a clause $C \in F$ with $C \subseteq C'$, and thus relativised resolution $\overset{\mathcal{U}_0}{\vdash}$ using oracle $\mathcal{U}_0$ is just ordinary resolution $\vdash$ with added weakening of axioms:

- if $T : F \vdash C$, then by definition also $T : F \overset{\mathcal{U}_0}{\vdash} C$;
- if $T : F \overset{\mathcal{U}_0}{\vdash} C$, then by Lemma 5.1, part 2 there is a resolution tree $T' \leq T$ with $T' : F \vdash C'$ for some clause $C' \subseteq C$.[4)]

Before giving the basic properties of relativised resolution in Lemma 5.3, we insert a little helpful lemma on the relation of "direct entailment" as defined above.

Lemma 5.2 *Consider $P \in \mathcal{PI}$, $F \in \mathcal{CLS}$ and $\varphi \in \mathcal{PASS}$.*

1. *If $\mathcal{U}$ is stable under application of partial assignments, and $P \xrightarrow{\mathcal{U}} F$ holds, then also $\varphi * P \xrightarrow{\mathcal{U}} \varphi * F$.*
2. *If $\varphi * P \xrightarrow{\mathcal{U}} \varphi * F$ holds, and for all $C \in F$ we have $C_\varphi \subseteq C$, then also $P \xrightarrow{\mathcal{U}} F$ is the case.*

[4)] The transformation from T into T' can be done in such a way, that not only the number of leaves is not increased by this transformation, but also the number of *different clauses* labelling the nodes.

Proof. For part 1 consider $C \in \varphi * F$: There is $C_0 \in F$ with $\varphi * C_0 = C$, and we have $\varphi_{C_0} * P \in \mathcal{U}$, thus also $\varphi * (\varphi_{C_0} * P) \in \mathcal{U}$, while $\varphi_C * (\varphi * P) = (\varphi_C \circ \varphi) * P = (\varphi \circ \varphi_{C_0}) * P = \varphi * (\varphi_{C_0} * P) \in \mathcal{U}$. For part 2 consider $C \in F$: We have $\varphi * \{C\} \neq \top$ and $\varphi_{\varphi * C} * (\varphi * P) \in \mathcal{U}$, while $\varphi_{\varphi * C} \circ \varphi = \varphi_C$. □

Lemma 5.3 *Assume that $\mathcal{U}$ is stable under application of partial assignments. Then for all instances $P \in \mathcal{PI}$, clauses $C, C' \in \mathcal{CL}$ and partial assignments $\varphi \in \mathcal{PASS}$ the following holds.*

1. *If $T : \varphi * P \overset{\mathcal{U}}{\vdash} C$ and $C' \supseteq C \cup C_\varphi$, then there is a resolution tree $T' \leq T$ with $T' : P \overset{\mathcal{U}}{\vdash} C'$.*
2. *If $T : P \overset{\mathcal{U}}{\vdash} C$ and $\varphi * \{C\} \neq \top$, then there is a resolution tree $T' \leq T$ with $T' : \varphi * P \overset{\mathcal{U}}{\vdash} \varphi * C$.*

Proof. For *part 1* consider a clause-set F with $\varphi * P \overset{\mathcal{U}}{\to} F$ and $T : F \vdash C$. By Lemma 5.1, part 2 there is a resolution tree $T_0 : \varphi_{C'} * F \vdash \bot$, while by Lemma 5.2, part 1 we have $\varphi_{C'} * (\varphi * P) = \varphi_{C'} * P \overset{\mathcal{U}}{\to} \varphi_{C'} * F$. Since $\mathrm{var}(\varphi_{C'} * F) \cap \mathrm{var}(C') = \emptyset$, we are allowed to replace all axioms D in T_0 by $D \cup C'$, and obtain a resolution tree $T' : F' \vdash C'$, where $\varphi_{C'} * F' = \varphi_{C'} * F$ and for all $D \in F'$ we have $C' \subseteq D$. Thus by Lemma 5.2, part 2 also $P \overset{\mathcal{U}}{\to} F'$ holds, and we conclude $T' : P \overset{\mathcal{U}}{\vdash} C'$.

Part 2 now follows immediately from Lemma 5.2, part 1, Lemma 5.1, part 2 and part 1 of the present lemma (with $\varphi := \emptyset$). □

For any unsatisfiable problem instance $P \in \mathcal{PI}$ let $\mathbf{Comp_{tR(\mathcal{U})}(P)}$ denote the minimal number $\#\mathrm{lvs}(T)$ of leaves in a resolution tree $T : P \overset{\mathcal{U}}{\vdash} \bot$. For $P = F \in \mathcal{CLS}$ and using $\mathcal{U} = \mathcal{U}_0$ as defined above, $\mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(P)$ becomes the ordinary complexity of tree resolution in the boolean case (where all domains have two elements).

5.2. The correspondence between tree resolution and branching trees

Now we will present the correspondence between resolution trees and branching trees in detail.

1. In Subsection 5.2.1 we first show in Lemma 5.4 that each resolution tree can be made "regular" without increasing the number of leaves. And regular resolution trees deriving the empty clause can be immediately interpreted as branching trees by interpreting the resolution variables as branching variables (but it may happen that some branching variables do not occur in their branching instances, since our form of resolution allows "weakening" of axioms).

2. And in Subsection 5.2.2 we show in Lemma 5.7 that each branching tree becomes a resolution tree (deriving the empty clause) by interpreting the branching variables as resolution variables.[5] The completeness of (generalised) resolution follows immediately (Corollary 5.8)

It may be interesting to note, that when we transform a branching tree T into a resolution tree T' and then transform T' into a branching tree T'', then T and T'' are identical, but when we transform a resolution tree refutation T into a branching tree T' and back into a resolution tree T'', then T and T'' are identical with respect to their "resolution structure", that is, they are isomorphic as trees and have also the same resolution variables, but the clauses in T'' are now the maximal clauses for the given resolution structure.

5.2.1. *From resolution trees to branching trees*

A resolution tree T is called *regular* if T does not contain two non-leaf nodes w, w' such that w is a successor of w' and for the resolution variable v at w we have $v \in \mathrm{var}(C(w'))$, where $C(w')$ is the clause labelling node w' in T. See Figure 5 for a typical non-regular resolution tree. The following lemma generalises the regularisation lemma in (Tseitin, 1968).

Lemma 5.4 *Every resolution tree $T : F \vdash C$ can be transformed into a regular resolution tree $T' \leq T$ with $T' : F \vdash C' \subseteq C$.*

Proof. Consider Figure 5. The only reason for resolution at node w is to cut off the resolution variable v, but since at node w' variable v again

[5] In our framework it is not necessary to cut off branches in the branching tree which do not contribute to the refutation (this process is sometimes called "intelligent backtracking" (e.g. (Zhang, 1997)) or "conflict directed backtracking" (e.g. (Silva and Sakallah, 1996)) or "backjumping" (c.f. (Baker, 1995))), since, as already mentioned, our form of resolution supports weakening of axioms.

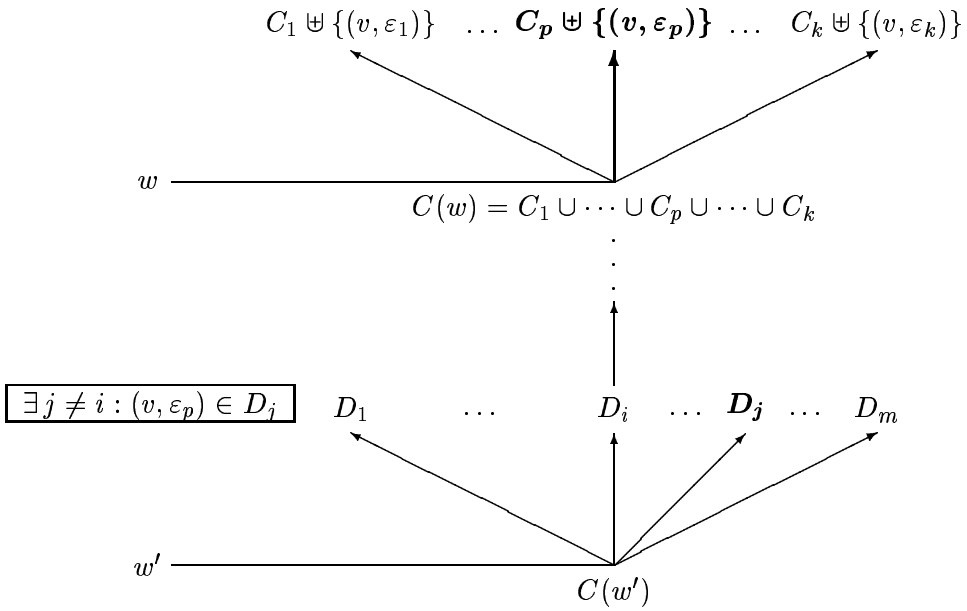

Figure 5. The part in a non-regular resolution tree T violating regularity

is added (by branch j), in fact the resolution at node w was pointless (at least for tree resolution), and thus for $(v,\varepsilon_p) \in D_j$ we simply cut off at node w all branches except of the branch yielding $C_p \uplus \{(v,\varepsilon_p)\}$, and simplify the resolution tree accordingly. Since in this way no new non-regularity is created, by iterating this process we finally obtain a regular resolution tree T' using only axioms also used by T and deriving a sub-clause $C' \subseteq C$ of clause C derived by the original resolution tree T. □

The number of *different clauses* in the regularised resolution tree T' can not be polynomially bounded in the number of different clauses in T, as shown in (Goerdt, 1993) (for the boolean case).

Lemma 5.5 *Consider a regular resolution tree T with $\bot$ at the root (i.e., $C(\mathrm{root}(T)) = \bot$). Assign to each edge (w,w') in T the partial assignment $\varphi((w,w')) := \langle v \to \varepsilon \rangle$, where v is the resolution variable at node w and $(v,\varepsilon) \in C(w')$ is the occurrence of the resolution variable cut off in the branch $w \to w'$. For any node w in T let $\varphi(w)$ be the composition of partial assignments labelling the edges along the path from the root to w. Since T is regular, no resolution variable appears*

twice on any path in T and thus the order of compositions to compute $\varphi(w)$ does not matter.

Now for any node w of T we have $\varphi(w) * C(w) = \bot$.

Proof. We use induction on the length of the path from the root to w. If w is the root, then $C(w) = \bot$ and thus $\varphi(w) * C(w) = \bot$. Now consider any edge (w, w') in T with $\varphi((w, w')) = \langle v \to \varepsilon \rangle$. By definition we have $C(w') \subseteq C(w) \uplus \{(v, \varepsilon)\} \in \mathcal{CL}$ and $\varphi(w') = \varphi(w) \circ \langle v \to \varepsilon \rangle$, and thus by induction hypothesis:

$$\varphi(w') * C(w') \subseteq \varphi(w') * (C_w \cup \{(v, \varepsilon)\}) =$$
$$\varphi(w) * (\langle v \to \varepsilon \rangle * (C_w \cup \{(v, \varepsilon)\})) = \varphi(w) * C(w) = \bot. \quad \square$$

In case of a tree $T : P \overset{\mathcal{U}}{\vdash} \bot$ using oracle $\mathcal{U}$, for any leaf w of T by definition we have $\varphi_{C(w)} * P \in \mathcal{U}$, while Lemma 5.5 yields $\varphi_{C(w)} \subseteq \varphi(w)$, establishing the main part of the correctness proof of the following transformation of a regular resolution tree into a branching tree. Since for *unsatisfiable* instances P the notion of a "$(\mathcal{U}, \mathcal{S})$-branching tree for P" does not depend on the oracle $\mathcal{S}$ for satisfiability, we use the notion of a "$\mathcal{U}$-branching tree for P" instead.

Lemma 5.6 *Consider a regular resolution tree $T : P \overset{\mathcal{U}}{\vdash} \bot$ and assume, that $\mathcal{U}$ is stable under application of partial assignments. Let the edge labelling $\varphi((w, w'))$ (which becomes the "branching assignment" in the branching tree to be constructed) and the node labelling $\varphi(w)$ be defined as in Lemma 5.5, and let the "branching variable" $v(w)$ be the resolution variable at w, while the "branching instance" $P(w)$ at w is defined as $P(w) := \varphi(w) * P$. By Lemma 5.5 and stability of $\mathcal{U}$ under the action of $\mathcal{PASS}$, for any leaf w of T we have $P(w) \in \mathcal{U}$, and thus T together with the above labelings is a "pre-$\mathcal{U}$-branching tree" for P, fulfilling all the conditions on a $\mathcal{U}$-branching tree for P except that for a non-leaf node w the branching variable $v(w)$ may not occur in the branching instance at w (i.e., $v(w) \notin \mathrm{var}(P(w))$ may happen). By cancelling all such superfluous "empty" branchings the tree T becomes only smaller, and we arrive at a $\mathcal{U}$-branching tree T' for P with $T' \leq T$.*

5.2.2. *From branching trees to resolution trees*

Lemma 5.7 *Consider any $\mathcal{U}$-branching tree T for $P \in \mathcal{UPI}$. For any non-leaf node $w \in V(T) \setminus \mathrm{lvs}(T)$ of T the clause $C_{\varphi(w)}$ is the resolvent of the clauses $\{C_{\varphi(w')} : w' \in \mathrm{ds_T}(w)\}$ with resolution variable $v(w)$ (the branching variable at w). And for any leaf w we have $\varphi_{C_{\varphi(w)}} * P = \varphi(w) * P \in \mathcal{U}$, while $C_{\varphi(\mathrm{root}(T))} = \bot$. Thus by setting $C(w) := C_{\varphi(w)}$ for nodes w in T the $\mathcal{U}$-branching tree T becomes a resolution tree $T : P \overset{\mathcal{U}}{\vdash} \bot$.*

Corollary 5.8 *For any instance $P \in \mathcal{PI}$ and any clause $C \in \mathcal{CL}$ we have $P \models C$ if and only if $P \overset{\mathcal{U}}{\vdash} C$ holds.*

Proof: Assume $P \models C$. Now $\varphi_C * P \in \mathcal{UPI}$ holds, and thus there is a $\mathcal{U}$-branching tree T for P, whence by Lemma 5.7 we obtain a resolution tree $T : \varphi_C * P \overset{\mathcal{U}}{\vdash} \bot$, yielding a resolution tree $T' : P \overset{\mathcal{U}}{\vdash} C$.

Corollary 5.9 *For $P \in \mathcal{UPI}$ we have $\mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(P) \leq d^{n(P)}$.*

Lemma 5.7 together with Lemmas 5.6 and 5.4 yields

Theorem 5.10 *Assume $\mathcal{U}$ is stable under application of partial assignments. Then for each instance $P \in \mathcal{UPI}$ we have*

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

that is, the minimal relativised tree resolution refutation complexity is equal to the minimal number $\#\mathrm{lvs}(T)$ of leaves in a $\mathcal{U}$-branching tree T for P.

5.3. Generalised input resolution

In this subsection we will reexamine the hierarchy $(\mathcal{G}_k(\mathcal{U},\mathcal{S}))_{k\in\mathbb{N}_0}$ for *unsatisfiable* instances, using $\boldsymbol{\mathcal{G}_k(\mathcal{U})} := \mathcal{G}_k(\mathcal{U},\mathcal{S}) \cap \mathcal{UPI}$ for short, since on unsatisfiable instances the oracles $\mathcal{S}$ never is used. The levelled height $\boldsymbol{h(T)}$ of branching trees (as defined in Subsection 3.2) now depends only on the tree itself, not on the labelling, and has the following simpler recursive definition:

- if T is trivial (has only one node) then $h(T) := 0$;

– if T is not trivial, then we distinct two cases:

  - if there is a node $w \in \mathrm{ds_T}(\mathrm{root}(T))$ such that for all other nodes $w' \in \mathrm{ds_T}(\mathrm{root}(T)) \setminus \{w\}$ we have $h(T_w) > h(T_{w'})$, then we set $h(T) := h(T_w)$,
  - while otherwise we let $h(T) := 1 + \max_{w \in \mathrm{ds_T}(\mathrm{root}(T))} h(T_w)$.

A tree of levelled height 1 is the analogue of what is called an *input tree*, only here we allow branchings of arbitrary breadth (not just binary branching as in the case for ordinary resolution), while a tree of levelled height k is a k-times nested input tree as picturised in Figure 6.

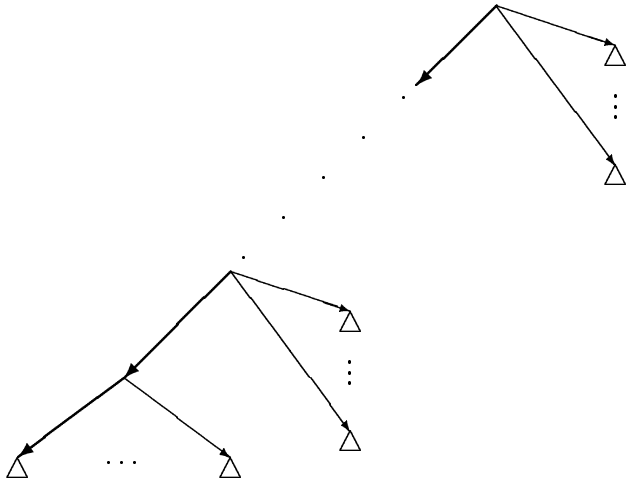

Figure 6. A tree T of levelled height at most k: The $\triangle$'s are trees of levelled height at most $k-1$; the "backbone" of T is shown in bold.

Some simple properties of the levelled height for arbitrary trees T, T' are:

1. If $T' \leq T$, then $h(T') \leq h(T)$ holds.
2. $h(T) \leq \mathrm{ht}(T)$, where $\mathbf{ht(}\boldsymbol{T}\mathbf{)}$ is the usual height of T (the length of a longest path in T).
3. For any node in T there is a path to a leaf of T of length at most $h(T)$.

To turn property 3 into a sufficient condition, not only T itself has to fulfil property 3, but also all trees T' obtained from T by interchanging two subtrees hanging at the "backbone" of T (see Figure 6 — the

triangles are meant) or at the backbone of some subtree of T (obviously these operations do not alter the levelled height).

From the correspondence between regular resolution refutations and branching trees (established in Lemmas 5.6 and 5.7) and by using regularisation (Lemma 5.4), the following characterisation of level $\mathcal{G}_k(\mathcal{U})$ as the set of instances refutable by k-times nested input resolution immediately follows.

Lemma 5.11 *Let $\mathcal{U}$ be stable under application of partial assignments. For any instance $P \in \mathcal{PI}$ we have $P \in \mathcal{G}_k(\mathcal{U})$ if and only if there is a resolution tree $T : P \overset{\mathcal{U}}{\vdash} \bot$ with $h(T) \leq k$, in other words, iff there is a relativised k-times nested input resolution refutation for P. It follows that $h_{\mathcal{U}}(P)$ for $P \in \mathcal{UPI}$ is equal to the minimal $h(T)$ for resolution trees $T : P \overset{\mathcal{U}}{\vdash} \bot$.*

Using Lemma 5.3 we get

Corollary 5.12 *Let $\mathcal{U}$ be stable under application of partial assignments. For any problem instance $P \in \mathcal{PI}$ a clause $C \in \mathcal{CL}$ can be derived by k-times nested input resolution ($k \in \mathbb{N}_0$), i.e., there is a resolution tree $T : P \overset{\mathcal{U}}{\vdash} C$ with $h(T) \leq k$, if and only if $\varphi_C * P \in \mathcal{G}_k(\mathcal{U})$ holds.*

5.4. Upper and lower bounds

Lemma 5.13 *For any tree T we have*

$$2^{h(T)} \leq \#\mathrm{lvs}(T) \leq (\mathrm{ht}(T) \cdot (d-1) + 1)^{h(T)},$$

where d is the maximal outdegree of a node in T.

Proof. We use induction on $\mathrm{ht}(T)$.

If $\mathrm{ht}(T) = 0$, then $1 = 2^0 = \#\mathrm{lvs}(T) = (0 \cdot (d-1)+1)^0 = 1$. Otherwise consider the subtrees $T_1, \ldots, T_m$ attached to the root of T. By induction hypothesis we know $2^{h(T_i)} \leq \#\mathrm{lvs}(T_i) \leq (\mathrm{ht}(T_i) \cdot (d-1) + 1)^{h(T_i)}$ for all $i \in \{1, \ldots, m\}$. If there is T_i with $h(T_i) = h(T)$, then for all indices

$j \neq i$ we know $h(T_j) < h(T_i)$ and thus

$$2^{h(T)} = 2^{h(T_i)} \leq \#\mathrm{lvs}(T_i) \leq \#\mathrm{lvs}(T) =$$
$$\sum_{j=1}^{m} \#\mathrm{lvs}(T_j) = \#\mathrm{lvs}(T_i) + \sum_{j \neq i} \#\mathrm{lvs}(T_j) \leq$$
$$(\mathrm{ht}(T_i) \cdot (d-1)+1)^{h(T_i)} + \sum_{j \neq i} (\mathrm{ht}(T_j) \cdot (d-1)+1)^{h(T_j)} \leq$$
$$((\mathrm{ht}(T)-1) \cdot (d-1)+1)^{h(T)} + (d-1) \cdot ((\mathrm{ht}(T)-1) \cdot (d-1)+1)^{h(T)-1} =$$
$$((\mathrm{ht}(T)-1) \cdot (d-1)+1)^{h(T)-1} \cdot ((\mathrm{ht}(T)-1) \cdot (d-1)+1+(d-1)) \leq$$
$$(\mathrm{ht}(T) \cdot (d-1)+1)^{h(T)-1} \cdot (\mathrm{ht}(T) \cdot (d-1)+1) = (\mathrm{ht}(T) \cdot (d-1)+1)^{h(T)}. \surd$$

So assume $h(T_i) < h(T)$ for all $i \in \{1, \ldots, m\}$. By definition of $h(T)$ there are $1 \leq i_1 < i_2 \leq m$ with $h(T_{i_1}) = h(T_{i_2}) = h(T) - 1$, and thus

$$2^{h(T)} = 2 \cdot 2^{h(T)-1} \leq \#\mathrm{lvs}(T_{i_1}) + \#\mathrm{lvs}(T_{i_2}) \leq \#\mathrm{lvs}(T) =$$
$$\sum_{i=1}^{m} \#\mathrm{lvs}(T_i) \leq \sum_{i=1}^{m} (\mathrm{ht}(T_i) \cdot (d-1)+1)^{h(T_i)} \leq$$
$$\sum_{i=1}^{m} (\mathrm{ht}(T) \cdot (d-1)+1)^{h(T)-1} \leq d \cdot (\mathrm{ht}(T) \cdot (d-1)+1)^{h(T)-1} \leq$$
$$(\mathrm{ht}(T) \cdot (d-1)+1) \cdot (\mathrm{ht}(T) \cdot (d-1)+1)^{h(T)-1} = (\mathrm{ht}(T) \cdot (d-1)+1)^{h(T)},$$

concluding the proof. □

A regular resolution tree $T : F \vdash C$ has height at most $n(F)$. Thus Lemma 5.13 together with Lemma 5.11 and regularisation (Lemma 5.4) yields

Theorem 5.14 *Assume $\mathcal{U}$ is stable under application of partial assignments. Then for any unsatisfiable problem instance $P \in \mathcal{UPI}$ we have the following bounds on the complexity of relativised tree resolution:*

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

Theorem 5.14 and Lemma 4.4 immediately yield

Theorem 5.15 *Assume $\mathcal{U}$ is stable under application of partial assignments. The running time of SAT decision algorithm $\widehat{\mathfrak{G}}$ in the oracle model on an unsatisfiable input $P \in \mathcal{UPI} \setminus \mathcal{G}_0(\mathcal{U})$ is bounded by*

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

(while the running time for $P \in \mathcal{G}_0(\mathcal{U})$ is $O(n(P))$). Only impairing the running time of $\widehat{\mathfrak{G}}$ by a constant factor, in fact a resolution tree $T : P \overset{\mathcal{U}}{\vdash} \bot$ can be computed with $h(T) = h_{\mathcal{U}}(P)$ and

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

Thus relativised tree resolution for fipa-systems is quasi-automatisable.

6. Width restricted and general resolution

Bounded resolution, which uses only clauses of bounded size, has been introduced in (Galil, 1975), and it has been shown that certain Tseitin-type formulas in 3-CNF require very long clauses in any (full) resolution refutation; the author expressed hope, that this result might be useful for a super-polynomial lower bound for (full) resolution. Later in 1981, Krishnamurthy/Moll made a precise general conjecture in (Krishnamurthy and Moll, 1981) about a strong relation between the necessary size of clauses in resolution refutations and the necessary length of such refutations. Finally in 1999, Ben-Sasson/Wigderson justified in (Ben-Sasson and Wigderson, 1999) these attempts[6] by proving that if "long clauses" are necessary in a resolution refutation, then only "large" refutations are possible. Their proof is an adaption of an analogous result of Clegg, Edmonds and Impagliazzo from 1996 ((Clegg et al., 1996)) for "degree restricted Groebner bases proof systems"[7].

The main disadvantage of bounded resolution is its inability of handling even Horn formulas, which is also reflected in the dependence of the general lower bound for resolution in (Ben-Sasson and Wigderson, 1999) on the maximal input clause length, preventing that bound from being used for clause-sets already containing "long clauses." In

[6] especially Galil's attempt, while the precise form of the conjecture of Krishnamurthy/Moll has been refuted by (Bonet and Galesi, 1999)

[7] that the argumentation in (Clegg et al., 1996) just needs the resolution calculus has first been noted in the literature in (Beame and Pitassi, 1996)

(Kullmann, 1999b) a new form of width restricted resolution has been introduced, called "CUBI resolution" in (Kullmann, 2000a) ("closure under bounded input resolution"), overcoming these drawbacks. In the present article, besides generalising all results to arbitrary fipa-systems, we now consider a third form of width restricted resolution, called *k-resolution* (introduced in (Büning, 1993) for the case of propositional clause-sets), as the "right form" of width restricted resolution. k-resolution is the natural generalisation of Unit resolution (it was introduced for the purpose of refuting generalised Horn clause-sets), and the general lower bounds for resolution have a (slightly) more natural environment in this form of resolution. On the other side, the main disadvantage of k-resolution is that it does not seem likely to be poly-time decidable for $k \geq 3$; in (Buro and Büning, 1996) only 2-resolution has been shown to be poly-time decidable (for the case of propositional clause-sets), exploiting the special fact, that in the propositional case the resolvent of binary clauses is again a binary clause. So here is now the place of CUBI resolution, which turns out to be just a slight strengthening of k-resolution.

The results of this section are as follows:

1. In Subsection 6.1 we introduce k-resolution and establish the main technical tools.
2. In Subsection 6.2, Lemma 6.9 we generalise the well-known equivalence between input resolution and Unit resolution (w.r.t. their refutational power), and more generally we show in Lemma 6.8 that k-resolution simulates k-times nested input resolution for all $k \geq 0$ (while the converse does not hold for $k \geq 2$ as shown in (Kullmann, 1999b), Lemma 8.9).
3. Now in Subsection 6.3 we turn to the case of full (or dag-like) resolution, that is, instead counting leaves as for tree resolution, we now count different clauses, and we present some tools for this more complicated measure.
4. One main result of Section 6 then is Theorem 6.12 in Subsection 6.4 with a general lower bound for (full) resolution, based on the notion of width provided by k-resolution (this bound is nearly sharp as proven in (Kullmann, 1999b), Subsection 8.4, motivated by (Bonet and Galesi, 1999)).
5. In Subsection 6.5 we show how to decide a slight strengthening of k-resolution (that is, whether k-resolution is able to derive the

empty clause) in polynomial time in Theorem 6.16, using "CUBI resolution", which tries to add clauses of length at most k derived by input resolution (where the length of clauses used within the derivation is not restricted) to the database until either the empty clause has been obtained or no new clauses can be derived anymore.

6. Combining Theorems 6.12 and 6.16 with an upper bound on the length of resolution refutations in terms of width in Lemma 6.17 (similar to the upper bound from Lemma 5.13), we show in Theorem 6.18 in Subsection 6.6 how to obtain satisfiability decision faster than the trivial upper bound $d^{n(P)}$ in case we know, that if P is unsatisfiable, then there must be resolution refutation of size $d^{O(n(P)^\alpha)}$ for some $\alpha < 1$.

7. In Subsection 6.7 we show how to simulate k-resolution by bounded resolution (using a maximal clause length depending on k and the maximal "input clause length").

8. And finally in Subsection 6.8 we discuss the notion of "induced width" for constraint satisfaction problems and its relation to bounded resolution.

6.1. Generalising Unit resolution

Consider some $k \in \mathbb{N}_0$ and the resolvent R of (resolvable) clauses $C_1, \ldots, C_m$. We say that R is a *k-resolvent* of $C_1, \ldots, C_m$ if all but one parent clauses have length at most k, that is if there is an index $i \in \{1, \ldots, m\}$ such that for all other $j \in \{1, \ldots, m\} \setminus \{i\}$ we have $|C_j| \leq k$. A resolution tree T is called a *k-resolution tree* if all resolvent in T are k-resolvents, and we write $\boldsymbol{T : F \overset{k}{\vdash} C}$ if $T : F \vdash C$ and T is a k-resolution tree. The following basic lemma can be proven exactly the same way as the corresponding Lemma 5.1, one just has to observe that the transformations used in the proofs of part 1 and 2 of Lemma 5.1 not only do not enlarge the clause at the root, but also no other clause.

Lemma 6.1 *For clause-sets $F, F' \in \mathcal{CLS}$, clauses $C \in \mathcal{CL}$, partial assignments $\varphi \in \mathcal{PASS}$ and numbers $k \in \mathbb{N}_0$ we have*

1. *If $F \overset{k}{\vdash} C$ and $F' \xrightarrow{\mathcal{U}_0} F$, then there is a clause $C' \subseteq C$ with $F' \overset{k}{\vdash} C'$.*

2. *If* $F \overset{k}{\vdash} C$ *and* $\varphi * \{C\} \neq \top$, *then there is a clause* $C' \subseteq \varphi * C = C \setminus C_\varphi$ *with* $\varphi * F \overset{k}{\vdash} C'$.

3. *If* $\varphi * F \overset{k}{\vdash} C$, *then there is a clause* C' *with* $C \subseteq C' \subseteq C \cup C_\varphi$ *and* $F \overset{k+n(\varphi)}{\vdash} C'$.

Using an oracle $\mathcal{UPI}_0 \subseteq \mathcal{U} \subseteq \mathcal{UPI}$ for unsatisfiability of problem instances, we write $\boldsymbol{T : P \overset{\mathcal{U},k}{\vdash} C}$ for a problem instance $P \in \mathcal{PI}$ and a clause $C \in \mathcal{CL}$ if $T : P \overset{\mathcal{U}}{\vdash} C$ holds and T is a k-resolution tree. The following standardisation is often useful in the sequel.

Lemma 6.2 *If* $P \overset{\mathcal{U},k}{\vdash} C$ *for some instance* $P \in \mathcal{PI}$ *and clause* $C \in \mathcal{CL}$, *then there is a clause-set* $F \in \mathcal{CLS}$ *and a clause* $C' \subseteq C$ *with* $P \overset{\mathcal{U}}{\to} F$, $F \overset{k}{\vdash} C'$ *and* $\mathrm{var}(F) \subseteq \mathrm{var}(P)$ *(that is, not using extra variables).*

Proof. By definition there is F_0 with $P \overset{\mathcal{U}}{\to} F_0$ and $F_0 \overset{k}{\vdash} C$. Obtain F from F_0 by crossing out all literals with variables not appearing in P, that is let $F := \big\{ \{x \in C : \mathrm{var}(x) \in \mathrm{var}(P)\} : C \in F' \big\}$. Obviously still $P \overset{\mathcal{U}}{\to} F$ holds, and by Lemma 6.1, part 1 there is $C' \subseteq C$ with $F \overset{k}{\vdash} C'$. □

Lemma 6.3 *For all* $P \in \mathcal{PI}$, $C \in \mathcal{CL}$, $\varphi \in \mathcal{PASS}$ *and* $k \in \mathbb{N}_0$ *we have*

1. *If* $\mathcal{U}$ *is stable under application of partial assignments and* $P \overset{\mathcal{U},k}{\vdash} C$ *holds with* $\varphi * \{C\} \neq \top$, *then there is a clause* $C' \subseteq C$ *with* $\varphi * P \overset{\mathcal{U},k}{\vdash} C'$.

2. *If* $\varphi * P \overset{\mathcal{U},k}{\vdash} \bot$, *then also* $P \overset{\mathcal{U},k+n(\varphi)}{\vdash} C_\varphi$.

Proof. For part 1 consider a clause-set F with $P \overset{\mathcal{U}}{\to} F$ and $F \overset{k}{\vdash} C$. By Lemma 6.1, part 2 there is $C' \subseteq \varphi * C$ with $\varphi * F \overset{k}{\vdash} C'$, while by Lemma 5.2, part 1 we have $\varphi * P \overset{\mathcal{U}}{\to} \varphi * F$, and thus $\varphi * P \overset{\mathcal{U},k}{\vdash} C'$ holds.

For part 2 consider a clause-set F with $\varphi * P \overset{\mathcal{U}}{\to} F$, $F \overset{\mathrm{k}}{\vdash} \bot$ and $\mathrm{var}(F) \subseteq \mathrm{var}(\varphi * P)$. Obtain F' from F by adding C_φ to all clauses, that is $F' := \{C \cup C_\varphi : C \in F\}$. By Lemma 5.2, part 2 we get $P \overset{\mathcal{U}}{\to} F'$, while obviously $F' \overset{k+|C_\varphi|}{\vdash}$ holds true. □

The next lemma (following immediately from Lemma 6.1, part 1) shows the form of "transitivity" satisfied by k-resolution which makes the difference to k-times nested input resolution:

Lemma 6.4 *If for some clause-set F and a clause C_0 we have $F \overset{\mathrm{k}}{\vdash} C_0$, while for some instance P and every clause $C \in F$ there is a clause $C' \subseteq C$ with $P \overset{\mathcal{U},\mathrm{k}}{\vdash} C'$, then there is a clause $C'_0 \subseteq C_0$ with $P \overset{\mathcal{U},\mathrm{k}}{\vdash} C'_0$.*

For an unsatisfiable instance $P \in \mathcal{PI}$ we denote by $\mathbf{wid_{\mathcal{U}}(P)}$ the minimal k such that $P \overset{\mathcal{U},\mathrm{k}}{\vdash} \bot$. And for $k \in \mathbb{N}_0$ by $\boldsymbol{\mathcal{W}_k(\mathcal{U})}$ the class of problem instances $P \in \mathcal{UPI}$ with $\mathrm{wid}_{\mathcal{U}}(P) \leq k$ is denoted. Lemma 6.3, part 1 yields immediately

Lemma 6.5 *If $\mathcal{U}$ is stable under application of partial assignments, then so are all levels $\mathcal{W}_k(\mathcal{U})$.*

6.2. Simulating nested input resolution by k-resolution

Lemma 6.6 $\mathcal{W}_0(\mathcal{U}) = \mathcal{G}_0(\mathcal{U})$

Proof. We have to find out what 0-resolvents are: If a clause C contains a literal x such that $\mathrm{var}(x)$ is a trivial variables (that is, $|D_{\mathrm{var}(x)}| = 1$), then $C \setminus \{x\}$ is a 0-resolvent of C. And if on the other hand clause C is a 0-resolvent of clauses $C_1, \ldots, C_m$ with resolution variable v, then $m = 1$ must hold and v must be a trivial variable. We conclude $\mathcal{W}_0(\mathcal{U}) = \mathcal{G}_0(\mathcal{U})$ by the characterisation of $\mathcal{G}_0(\mathcal{U})$ in Theorem 3.3. □

The next lemma is based on an application of Unit resolution, used in the induction step of the simulation of nested input resolution by k-resolution.

Lemma 6.7 *Consider $P \in \mathcal{PI}$ and $k \geq 1$. If there is a variable v and $\varepsilon \in D_v$ such that $\langle v \to \varepsilon \rangle * P \overset{\mathcal{U},\mathrm{k}}{\vdash} \bot$, while for all $\varepsilon' \in D_v \setminus \{\varepsilon\}$ we have $P \overset{\mathcal{U},\mathrm{k}}{\vdash} \{(v, \varepsilon')\}$, then $P \overset{\mathcal{U},\mathrm{k}}{\vdash} \bot$ holds.*

Proof. Consider $F_\varepsilon \in \mathcal{CLS}$ with $\langle v \to \varepsilon \rangle * P \overset{\mathcal{U}}{\to} F_\varepsilon$, $F_\varepsilon \overset{\mathrm{k}}{\vdash} \bot$ and $\mathrm{var}(F_\varepsilon) \subseteq \mathrm{var}(\langle v \to \varepsilon \rangle * P)$, and for $\varepsilon' \in D_v \setminus \{\varepsilon\}$ consider $F_{\varepsilon'} \in \mathcal{CLS}$ with $P \overset{\mathcal{U}}{\to} F_{\varepsilon'}$ and $F_{\varepsilon'} \overset{\mathrm{k}}{\vdash} \{(v, \varepsilon')\}$. Obtain F'_ε from F_ε by adding the literal (v, ε) to all clauses, that is $F'_\varepsilon := \{C \cup \{(v, \varepsilon)\} : C \in F_\varepsilon\}$. By Lemma 5.2, part 2 we have $P \overset{\mathcal{U}}{\to} F'_\varepsilon$, and thus for the clause-set F defined by $F := F'_\varepsilon \cup \bigcup_{\varepsilon' \in D_v \setminus \{\varepsilon\}} F_{\varepsilon'}$ also $P \overset{\mathcal{U}}{\to} F$ holds. Now for all $C \in F_\varepsilon$ by 1-resolution we have $F \overset{\mathrm{k}}{\vdash} C$, and thus by Lemma 6.4 finally $P \overset{\mathcal{U},\mathrm{k}}{\vdash} \bot$ follows. □

Lemma 6.8 *k-resolution simulates k-times nested input resolution for all $k \geq 0$. More specifically:*

1. *For all $p, q \in \mathbb{N}_0$ we have $\mathcal{G}_p(\mathcal{W}_q(\mathcal{U})) \subseteq \mathcal{W}_{p+q}(\mathcal{U})$.*
2. *For all $k \in \mathbb{N}_0$ we have $\mathcal{G}_k(\mathcal{U}) \subseteq \mathcal{W}_k(\mathcal{U})$, and thus for all unsatisfiable $P \in \mathcal{PI}$ we get $\mathrm{wid}_{\mathcal{U}}(P) \leq h_{\mathcal{U}}(P)$.*

Proof. We prove *part 1* by induction on p. First consider $p = 0$: The inclusion $\mathcal{G}_0(\mathcal{W}_q(\mathcal{U})) \subseteq \mathcal{W}_q(\mathcal{U})$ follows by the characterisation of $\mathcal{G}_0(\mathcal{U})$ in Theorem 3.3 (using the same argumentation as in Lemma 6.6).

Now we turn to the case $p = 1$ and prove $P \in \mathcal{G}_1(\mathcal{W}_q(\mathcal{U})) \Rightarrow P \in \mathcal{W}_{q+1}(\mathcal{U})$ by induction on $n(P)$. So consider $P \in \mathcal{G}_1(\mathcal{W}_q(\mathcal{U}))$. If $P \in \mathcal{G}_0(\mathcal{W}_q(\mathcal{U}))$ (including the case $n(P) = 0$), then $P \in \mathcal{W}_q(\mathcal{U}) \subseteq \mathcal{W}_{q+1}(\mathcal{U})$ by the case $p = 0$. So assume $P \notin \mathcal{G}_0(\mathcal{W}_q(\mathcal{U}))$. By Theorem 3.3 there is a variable $v \in \mathrm{var}(P)$ and $\varepsilon \in D_v$ such that $\langle v \to \varepsilon \rangle * P \in \mathcal{G}_1(\mathcal{W}_q(\mathcal{U}))$ holds, as well as $\langle v \to \varepsilon' \rangle * P \in \mathcal{G}_0(\mathcal{W}_q(\mathcal{U})) = \mathcal{W}_q(\mathcal{U})$ for all other $\varepsilon' \in D_v \setminus \{\varepsilon\}$. In other words, for $\varepsilon' \in D_v \setminus \{\varepsilon\}$ we have $\langle v \to \varepsilon' \rangle * P \overset{\mathcal{U},q}{\vdash} \bot$, and thus by Lemma 6.3, part 2 we get $P \overset{\mathcal{U},q+1}{\vdash} \{(v, \varepsilon')\}$.[8] On the other

[8] It is interesting to note, that in case of $q = 0$ in fact $P \overset{\mathcal{U},0}{\vdash} \{(v, \varepsilon')\}$ holds, since then the newly added literal occurs only in the "don't care" parent clauses of 0-resolutions step (simply because there are no other parent clauses). See Lemma 6.9.

hand, induction hypothesis yields $\langle v \to \varepsilon\rangle * P \in \mathcal{W}_{q+1}(\mathcal{U})$, i.e., $\langle v \to \varepsilon\rangle * P \overset{\mathcal{U},\,q+1}{\vdash} \bot$, and so altogether by Lemma 6.7 we obtain $P \in \mathcal{W}_{q+1}(\mathcal{U})$.

Finally, for arbitrary $p \geq 0$ by induction hypothesis and Corollary 3.4 we get

$$\mathcal{G}_{p+1}(\mathcal{W}_q(\mathcal{U})) = \mathcal{G}_1(\mathcal{G}_p(\mathcal{W}_q(\mathcal{U}))) \subseteq \mathcal{G}_1(\mathcal{W}_{p+q}(\mathcal{U})) \subseteq \mathcal{W}_{p+q+1}(\mathcal{U}).$$

Part 2 now is a simple consequence of part 1 (using Lemma 6.6 and Corollary 3.4):

$$\mathcal{G}_k(\mathcal{U}) = \mathcal{G}_k(\mathcal{G}_0(\mathcal{U})) = \mathcal{G}_k(\mathcal{W}_0(\mathcal{U})) \subseteq \mathcal{W}_k(\mathcal{U}). \quad \square$$

Regarding level one of our hierarchies, we generalise the equivalence between input resolution and unit resolution (see (Chang, 1970)).[9)]

Lemma 6.9 *If $\mathcal{U}$ is stable under application of partial assignments, then $\mathcal{G}_1(\mathcal{U}) = \mathcal{W}_1(\mathcal{U})$ holds.*

Proof. By Lemma 6.8, part 2, we only have to prove $\mathcal{W}_1(\mathcal{U}) \subseteq \mathcal{G}_1(\mathcal{U})$. So consider $P \in \mathcal{W}_1(\mathcal{U})$. We show $P \in \mathcal{G}_1(\mathcal{U})$ by induction on $n(P)$. If $P \in \mathcal{U}$, then $P \in \mathcal{G}_0(\mathcal{U}) \subseteq \mathcal{G}_1(\mathcal{U})$. So assume $P \notin \mathcal{U}$ (excluding also $n(P) = 0$), and consider a resolution tree $T : F \overset{1}{\vdash} \bot$ with $P \overset{\mathcal{U}}{\to} F$. Using a first resolution in T (using only parent clauses from F) we see that there is a literal (v, ε) such that for all $\varepsilon' \in D_v \setminus \{\varepsilon\}$ we must have $\{(v, \varepsilon')\} \in F$, that is $\langle v \to \varepsilon'\rangle * P \in \mathcal{U}$. On the other hand Lemma 6.5 gives us $\langle v \to \varepsilon\rangle * P \in \mathcal{W}_1(\mathcal{U})$, and thus by induction hypothesis we get $\langle v \to \varepsilon\rangle * P \in \mathcal{G}_1(\mathcal{U})$. Now Theorem 3.3 yields $P \in \mathcal{G}_1(\mathcal{U})$. $\square$

6.3. General resolution

For a clause-set $F \in \mathcal{CLS}$ let $\boldsymbol{c(F)} := |F| = \sum_{C \in F} 1$ denote the number of clauses in F, and for a resolution tree T let $\mathbf{F^+(T)} \in \mathcal{CLS}$ be the set of clauses labelling the nodes of T, that is $\mathrm{F}^+(T) := \{C(w) :$

[9)] From (Kullmann, 1999b), Lemma 8.9 it follows, that in the case of propositional clause-sets $\mathcal{W}_6(\mathcal{U}_0)$ is not contained in any $\mathcal{G}_k(\mathcal{U}_0)$ (in fact we just need bounded resolution, and the size of all clauses can be restricted to 4 as pointed out to my be Eli Ben-Sasson).

$w \in V(T)\}$. Now for an instance $P \in \mathcal{UPI}$ we define the (relativised) resolution complexity of P as

$$\mathbf{Comp_{R(\mathcal{U})}(P)} := \min\{\, c(\mathrm{F}^+(T)) \mid T : P \overset{\mathcal{U}}{\vdash} \bot \,\},$$

the minimal number of different clauses in a resolution tree refuting P. Alternatively we could have used dag's (directed acyclic graphs) for the presentation of (general) resolution proofs, and then $\mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P)$ becomes the minimal number of nodes in a resolution dag. For the study of the complexity measure $c(\mathrm{F}^+(T))$ we need the notion of a *normalised resolution tree* T, where for all nodes $w, w' \in V(T)$ in case of $C(w) = C(w')$ the subtrees T_w and $T_{w'}$ are isomorphic as trees and are labelled with the same clauses — when counting only *different* clauses, there is no need to use different proofs for the same clause. Obviously for every resolution tree $T : F \vdash C$ there is a normalised resolution tree $T' : F \vdash C$ with $\mathrm{F}^+(T') \subseteq \mathrm{F}^+(T)$.

A fundamental process for the study of resolution is the process of strengthening (i.e., shortening) the axiom clauses in a resolution tree. In Lemma 5.1, part 1 we have shown that this process does not increase the "geometric complexity", while Lemma 6.1, part 1 relies on the fact that the length of the clauses in the resolution tree is not increased by this process. Now we formulate those properties of the process of strengthening which are relevant in the present context.

Lemma 6.10 *For every normalised resolution tree $T : F \vdash C$ and every $F' \xrightarrow{\mathcal{U}_0} F$ there is a normalised resolution tree $T' \leq T$ and a clause $C' \subseteq C$ with $T' : F' \vdash C'$, such that an embedding $f : V(T') \to V(T)$ exists with*

- $C(w) \subseteq C(f(w))$
- $C(f(w_1)) = C(f(w_2)) \Rightarrow C(w_1) = C(w_2)$

for all nodes $w, w_1, w_2 \in V(T')$. It follows $c(\mathrm{F}^+(T')) \leq c(\mathrm{F}^+(T))$.

Thus for a clause-set $F \in \mathcal{CLS}$ the complexity $\mathrm{Comp}_{\mathrm{R}(\mathcal{U}_0)}(F)$ is just the ordinary complexity of ("dag-") resolution (considering the propositional case). For the following corollary, the counterpart of Lemma 5.1, part 2, we can use essentially the same proof as in Lemma 5.1, part 2, only this time relying on Lemma 6.10, and not on Lemma 5.1, part 1.

Corollary 6.11 *For every normalised resolution tree $T : F \vdash C$ and every partial assignment $\varphi \in \mathcal{PASS}$ with $\varphi * \{C\} \neq \top$ there is a resolution tree $T' \leq T$ and a clause $C' \subseteq C$ with $T' : \varphi * F \vdash C'$, such that an embedding $f : V(T') \to V(T)$ exists with*

- $\varphi * \{C(f(w))\} \neq \top$ *and* $C(w) \subseteq \varphi * C(f(w))$
- $C(f(w_1)) = C(f(w_2)) \Rightarrow C(w_1) = C(w_2)$

for all nodes $w, w_1, w_2 \in V(T')$.

Now assume that $\mathcal{G}_0(\mathcal{U}) = \mathcal{U}$ holds, so that no resolution on trivial variables is needed. Then for any $P \in \mathcal{PI}$ the general relation

$$\mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P) \leq 2 \cdot \mathrm{Comp}_{\mathrm{tR}(\mathcal{U})}(P) - 1$$

between tree resolution and (full) resolution holds true. Thus the upper bound from Corollary 5.9 yields $\mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P) \leq 2 \cdot d^{n(P)} - 1$; in fact this upper bound holds without the assumption $\mathcal{G}_0(\mathcal{U}) = \mathcal{U}$, since in any branching tree for an instance P there are at most $2 \cdot d^{n(P)} - 1$ nodes at all.

6.4. A GENERAL LOWER BOUND FOR RESOLUTION

In (Clegg et al., 1996) a general lower bound for (full) resolution was given, depending on the necessary degree for refutations in "degree restricted Groebner bases proof systems." Is is not hard to see, that their construction does not need the power of the Groebner bases attempt, but exploits only simple properties of resolution. This was first noted in the literature by (Beame and Pitassi, 1996), and systematically worked out in (Ben-Sasson and Wigderson, 1999), using bounded resolution, that is, restricting the size of *all clauses* used in the resolution refutation. The presence of "long" clauses in the clause-set F to be refuted (that is, of length comparable to the necessary size of clauses needed in a resolution refutation) prevent the methods from (Ben-Sasson and Wigderson, 1999) to be used, and so in these cases F first has to be transformed into a 3-CNF, which makes the proof of the lower bound much harder. To overcome this restriction (for full resolution as well as for tree resolution) in (Kullmann, 1999b) a stronger form of width restricted resolution has been introduced, called "CUBI resolution" in (Kullmann, 2000a); see Lemma 8.13 in (Kullmann, 1999b) for applications. We now improve these results a

bit by using k-resolution instead (and, of course, proving the bound for general fipa-systems).

Theorem 6.12 *For any instance $P \in \mathcal{UPI}$ with $n(P) > 0$ we have*

$$\ln \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P) > \frac{1}{8} \frac{\mathrm{wid}_{\mathcal{U}}(P)^2}{n(P)}.$$

Proof. We need a couple of notations: For a clause-set F and a variable v let $\#_v(F) := c(\{C \in F : v \in \mathrm{var}(C)\})$ be the number of occurrences of v, while for a literal x we set $\#_x(F) := c(\{C \in F : x \in C\})$. Just to make sure that these notions are clear, the reader should verify

$$\sum_{C \in F} |C| = \sum_{v \in \mathrm{var}(F)} \#_v(F) = \sum_{v \in \mathrm{var}(F),\, \varepsilon \in D_v} \#_{(v,\varepsilon)}(F).$$

Let $e \in \mathbb{R}_{>0}$ be any positive number. For a clause-set $F \in \mathcal{CLS}$ let $L(F) := \{C \in F : |C| > e\}$ be the set of "large clauses" in F, and for a resolution tree T let $\mu(T) := c(L(\mathrm{F}^+(T)))$ be the number of large clauses in T. For a problem instance $P \in \mathcal{PI}$ we define

$$\mu(P) := \min\{\mu(T) \mid T : P \overset{\mathcal{U}}{\vdash} \bot\}$$

$$b(P) := \begin{cases} 0 & \text{if } 2 \cdot n(P) \le e \\ 1 + \frac{e}{2 \cdot n(P) - e} & \text{if } 2 \cdot n(P) > e \end{cases}.$$

Note that $b(P)$ only depends on $n(P)$, and that $b(P) > 1$ is increasing with decreasing $n(P)$ as long as $n(P) > 0$. Finally for $k \in \mathbb{N}_0$ let

$$\Gamma_k := \{P \in \mathcal{UPI} : \mu(P) \le b(P)^{k+1}\}.$$

Note that for $P \in \mathcal{UPI}$ with $\mu(P) = 0$ we have $P \in \Gamma_0$, and that for $P \in \mathcal{UPI}$ with $n(P) \le e$ trivially $\mu(P) = 0$ holds. It follows that for $P \in \mathcal{UPI} \setminus \Gamma_0$ we have $b(P) > 1$, and thus there is $k \ge 1$ with $P \in \Gamma_k$. For any $P \in \mathcal{UPI}$ let $\gamma(P)$ be the minimal $k \in \mathbb{N}_0$ with $P \in \Gamma_k$.

Claim I: For all $P \in \mathcal{UPI}$ we have $h_{\Gamma_0}(P) \le \gamma(P)$.

We prove Claim I by induction on $n(P)$. If $\gamma(P) = 0$ is the case (including $n(P) = 0$), then the assertion is trivial. So assume $\gamma(P) \ge 1$. Consider a resolution tree $T : P \overset{\mathcal{U}}{\vdash} \bot$ with $\mu(T) = \mu(P)$ and let $F := L(\mathrm{F}^+(T))$ be the set of large clauses in T (so that $c(F) = \mu(T)$).

Without loss of generality by Lemma 6.10 we may assume $\mathrm{var}(F) \subseteq \mathrm{var}(P)$ (and thus $n(F) \leq n(P)$).

Choose a variable $v \in \mathrm{var}(F)$ with $\#_v(F)$ maximal, so that we have

$$\#_v(F) \geq \frac{\sum_{C \in F} |C|}{n(F)} > \frac{e \cdot \mu(P)}{n(P)}.$$

Choose $\varepsilon \in D_v$ with $\#_{(v,\varepsilon)}(F)$ maximal. We want to show that for all $\varepsilon' \in D_v \setminus \{\varepsilon\}$ we have $\gamma(\langle v \to \varepsilon' \rangle * P) \leq \gamma(P) - 1$, since then by induction hypothesis we have $h_{\Gamma_0}(\langle v \to \varepsilon' \rangle * P) \leq \gamma(P) - 1$, while by induction hypothesis also $h_{\Gamma_0}(\langle v \to \varepsilon \rangle * P \leq \gamma(P)$ holds, and thus by Theorem 3.3 we get $h_{\Gamma_0}(P) \leq \gamma(P)$. So consider $\varepsilon' \in D_v \setminus \{\varepsilon\}$ and assume $n(\langle v \to \varepsilon' \rangle * P) \neq 0$ (otherwise the assertion is trivial).

Let $\delta_{\varepsilon'}$ be the number of clauses of F eliminated by applying the assignment $\langle v \to \varepsilon' \rangle$:

$$\delta_{\varepsilon'} := c(F) - c(\langle v \to \varepsilon' \rangle * F) \geq \sum_{\varepsilon'' \in D_v \setminus \{\varepsilon'\}} \#_{(v,\varepsilon'')}(F).$$

Now $\sum_{\varepsilon'' \in D_v \setminus \{\varepsilon'\}} \#_{(v,\varepsilon'')}(F) \geq \frac{1}{2}\#_v(F)$ holds, since otherwise the inequality $\#_{(v,\varepsilon')}(F) > \frac{1}{2}\#_v(F)$ would follow, and thus $\#_{(v,\varepsilon')}(F) > \#_{(v,\varepsilon)}(F)$. We conclude

$$\delta_{\varepsilon'} > \frac{e \cdot \mu(P)}{2 \cdot n(P)}.$$

By Corollary 6.11 there is a resolution tree $T' : \langle v \to \varepsilon' \rangle * F \vdash \bot$ such that $\mu(T') \leq \mu(T) - \delta_{\varepsilon'}$ holds. By Lemma 5.2, part 1 we have $T' : \langle v \to \varepsilon' \rangle * P \vdash \bot$, whence

$$\gamma(\langle v \to \varepsilon' \rangle * P) \leq \mu(T') \leq \mu(P) - \delta_{\varepsilon'} < \mu(P) - \frac{e \cdot \mu(P)}{2 \cdot n(P)} =$$
$$\mu(P) \cdot b(P)^{-1} \leq b(P)^{\gamma(P)+1} \cdot b(P)^{-1} = b(P)^{\gamma(P)} \leq b(\langle v \to \varepsilon' \rangle * P)^{\gamma(P)}.$$

It follows $\gamma(\langle v \to \varepsilon' \rangle * P) \leq \gamma(P) - 1$, completing the proof of claim I.

Claim II: For all $P \in \Gamma_0$ we have $\mathrm{wid}_{\mathcal{U}}(P) \leq \lfloor e \rfloor$.

For the proof of Claim II consider $P \in \Gamma_0$ (and thus $\mu(P) \leq b(P)$). If $\mu(P) = 0$ then by definition we have $\mathrm{wid}_{\mathcal{U}}(P) \leq \lfloor e \rfloor$. So assume $\mu(P) > 0$. It follows $n(P) > e$, thus $b(P) < 2$, and so $\mu(P) = 1$ must be the case. In other words, there is a resolution tree $T : P \overset{\mathcal{U}}{\vdash} \bot$ with $\mu(T) = 1$. Since all parent clauses in resolution steps must be different,

we conclude that all resolution steps in T must be $\lfloor e \rfloor$-resolutions, so that we get $\mathrm{wid}_{\mathcal{U}}(P) \leq \lfloor e \rfloor$, completing the proof of claim II.

From claim I and claim II we get $\Gamma_k \subseteq \mathcal{G}_k(\mathcal{W}_{\lfloor e \rfloor}(\mathcal{U}))$, and thus by Lemma 6.8, part 1 for all $P \in \mathcal{UPI}$

$$\mathrm{wid}_{\mathcal{U}}(P) \leq \lfloor e \rfloor + \gamma(P)$$

follows. The definition of $\gamma(P)$ in turn says

$$\gamma(P) = \begin{cases} 0 & \text{if } \mu(P) = 0 \\ \max(\lceil \log_{b(P)} \mu(P) \rceil - 1, 0) & \text{else} \end{cases},$$

and so, using $\mu(P) \leq \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P)$, we obtain for all $P \in \mathcal{UPI} \setminus \mathcal{UPI}_0$

$$(*)\ \ \mathrm{wid}_{\mathcal{U}}(P) \leq \inf_{0<e<2n(P)} \big(\lfloor e \rfloor + \max(\lceil \log_{b(P)} \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P) \rceil - 1, 0) \big).$$

In order to estimate $(*)$, consider the function

$$f(e) := e + \log_{\frac{s}{s-e}} c$$

for $0 < e < s$ and parameters $s, c \geq 2$ fulfilling $\ln c < s$. Here c stands for $\mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P)$ and s for $2n(P)$, and thus the restrictions are justified as follows:

- if $\mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P) = 1$, then we have $\mathrm{wid}_{\mathcal{U}}(P) = 0$, and the assertion of the theorem holds;
- if $\ln \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P) \geq 2n(P)$, then also the assertion of the theorem holds, since $\frac{1}{8}\frac{\mathrm{wid}_{\mathcal{U}}(P)^2}{n(P)} \leq \frac{1}{8}\frac{n(P)^2}{n(P)} = \frac{1}{8}n(P)$.

We "guess" $e_0 := \sqrt{s \cdot \ln c}$.[10] For the second addend in $f(e_0)$ we get

$$(**)\ \ \log_{\frac{s}{s-e_0}} c = \frac{\ln c}{-\ln(1 - \sqrt{\frac{\ln c}{s}})} \leq \frac{\ln c}{\sqrt{\frac{\ln c}{s}}} = e_0,$$

using $\forall 0 \leq x < 1 : e^x \leq \frac{1}{1-x}$, which is equivalent to $-\ln(1-x) \geq x$.

[10] We have $f'(e) = 1 - \frac{\ln c}{s-e}\frac{1}{(\ln(1-\frac{e}{s}))^2}$. Doing the formulation $e \ll s$ we get $s - e \approx s$ and $-\ln(1-\frac{e}{s}) \approx \frac{e}{s}$, and thus the equation $f'(e) = 0$ becomes $1 - \frac{\ln c}{s} \cdot \frac{1}{(\frac{e}{s})^2} = 0$, $\Rightarrow e = \sqrt{s \cdot \ln c}$.

Now finally by assertions $(*)$ and $(**)$ we get, using $s := 2n(P)$, $c := \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P)$ and excluding the trivial cases mentioned above:

$$\begin{aligned}\mathrm{wid}_{\mathcal{U}}(P) &\leq \lfloor e_0 \rfloor + \max(\lceil \log_{\frac{2n(P)}{2n(P)-e_0}} \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P) \rceil - 1, 0) \\ &\leq \lfloor e_0 \rfloor + \lceil e_0 \rceil - 1 < 2 \cdot e_0 = 2\sqrt{2n(P) \cdot \ln \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P)}.\end{aligned}$$

Easy transformations now yield the assertion of the theorem. □

6.5. Polynomial time decision for (nearly) k-resolution

In this subsection we present an algorithm for deciding "$P \in {}^{\mathrm{i}}\mathcal{W}_k(\mathcal{U})$?" in polynomial time (for constant k), where the hierarchy $({}^{\mathrm{i}}\mathcal{W}_k(\mathcal{U}))_{k\in\mathbb{N}_0}$ is a slight strengthening of the hierarchy $(\mathcal{W}_k(\mathcal{U}))_{k\in\mathbb{N}_0}$, that is for all $k \geq 0$ we have $\mathcal{W}_k(\mathcal{U}) \subseteq {}^{\mathrm{i}}\mathcal{W}_k(\mathcal{U}) \subseteq \mathcal{W}_{k+1}(\mathcal{U})$; the "i" expresses that we use a form of width restricted resolution based on (iterated) *input* resolution. For all results in this subsection we assume stability of $\mathcal{U}$ under application of partial assignments.

Let $\boldsymbol{k\text{-}\mathcal{CL}(V)} := \{C \in \mathcal{CL} : |C| \leq k \wedge \mathrm{var}(C) \subseteq V\}$ for $k \in \mathbb{N}_0$ and $V \subseteq \mathcal{VA}$ be the set of all clauses of length k using only variables from V, and let $\boldsymbol{k\text{–}\mathcal{CLS}(V)} := \{F \in \mathcal{CLS} : F \subseteq k\text{-}\mathcal{CL}(V)\}$ be the corresponding set of clause-sets. "Living" in the system $\mathcal{PI} \oplus \mathcal{CLS}$ of problem instances, we define the derived oracle

$$U' := (U \times \mathcal{CLS}) \cup (\mathcal{PI} \times \mathcal{U}_0)$$

which is also stable under the operation of $\mathcal{PASS}$. Now consider the algorithm $\mathfrak{W}(P,k)$ given in Figure 7, and let $\boldsymbol{{}^{\mathrm{i}}\mathcal{W}_k(\mathcal{U})}$ for $k \in \mathbb{N}_0$ be the set of instances $P \in \mathcal{PI}$ where $\mathfrak{W}(P,k) =$ "yes". Starting with the empty clause-set $F = \top$, the algorithm searches for clauses C of length at most k, such that C follows by input resolution from (P,F) w.r.t. oracle $\mathcal{U}'$, and adds these C to F. Since addition of a new clause to the database F can only extend the possibilities for input resolution, the final clause-set $F \in k\text{–}\mathcal{CLS}(\mathrm{var}(P))$ obtained by repeating this process as long as possible is uniquely determined, and we denote it by $\boldsymbol{F(\mathcal{U},k,P)} \in k\text{–}\mathcal{CLS}(\mathrm{var}(P))$ in the sequel (so that we have $P \in {}^{\mathrm{i}}\mathcal{W}_k(\mathcal{U})$ iff $\bot \in F(\mathcal{U},k,P)$ holds). Finally for $P \in \mathcal{UPI}$ let $\boldsymbol{{}^{\mathrm{i}}\mathrm{wid}_{\mathcal{U}}(P)}$ be the minimal $k \in \mathbb{N}_0$ with $P \in {}^{\mathrm{i}}\mathcal{W}_k(\mathcal{U})$; by the following lemma we see that in fact for each unsatisfiable P there is such a k.

Lemma 6.13 *For any $P \in \mathcal{UPI}$ we have ${}^{\mathrm{i}}\mathrm{wid}_{\mathcal{U}}(P) \leq \mathrm{wid}_{\mathcal{U}}(P)$.*

PROCEDURE $\mathfrak{W}(P \in \mathcal{PI},\ k \in \mathbb{N}_0) : \{\text{“yes”},\ \text{“no”}\}$;
BEGIN
  $F := \top$; { the set of derived k-clauses }
  WHILE there is $C \in k\text{-}\mathcal{CL}(\mathrm{var}(P)) \setminus F$ with
    $\varphi_C * (P, F) \in \mathcal{G}_1(\mathcal{U} \times \mathcal{CLS} \cup \mathcal{PI} \times \mathcal{U}_0)$ DO
    $F := F \cup \{C\}$;
  IF $\bot \in F$ THEN RETURN “yes”
  ELSE RETURN “no”
END $\mathfrak{W}$.

Figure 7. Algorithm $\mathfrak{W}$, deciding in polynomial time whether a slight strengthening of relativised k-resolution is able to derive the empty clause

Proof. Consider an instance $P \in \mathcal{W}_k(\mathcal{U})$ for some $k \in \mathbb{N}_0$. We want to show $\bot \in F := F(\mathcal{U}, k, P)$. For that purpose consider some resolution tree $T : P \overset{\mathcal{U},\mathrm{k}}{\vdash} \bot$ with $\mathrm{var}(\mathrm{F}^+(T)) \subseteq \mathrm{var}(P)$. We prove

$$\forall\, w \in V(T) : \varphi_{C(w)} * (P, F) \in \mathcal{G}_1(\mathcal{U}')$$

by induction on $\mathrm{ht}(T_w)$ (the height of the subtree of T rooted at w), which will prove the lemma, since $C(\mathrm{root}(T)) = \bot$ and thus $\bot \in F$ by definition.

If $\mathrm{ht}(T_w) = 0$ (i.e., w is a leaf), then $\varphi_{C(w)} * P \in \mathcal{U}$ holds, and thus by definition $\varphi_{C(w)} * (P, F) \in \mathcal{U}'$.

Now consider the case $\mathrm{ht}(T_w) > 0$. By induction hypothesis for all $u \in \mathrm{ds_T}(w)$ we have $\varphi_{C(u)} * (P, F) \in \mathcal{G}_1(\mathcal{U}')$. By definition of k-resolution there is $u_0 \in \mathrm{ds_T}(w)$ such that for all other $u \in \mathrm{ds_T}(w) \setminus \{u_0\}$ we have $|C(u)| \leq k$ and thus $C(u) \in F$. By Corollary 5.12 there is a resolution tree $T_0 : (P, F) \overset{\mathcal{U}'}{\vdash} C(u_0)$ with $h(T_0) \leq 1$. Now consider the resolution tree T' obtained from T_w by replacing the subtrees rooted at $u \in \mathrm{ds_T}(w) \setminus \{u_0\}$ by the trivial trees labelled with $C(u)$, and replacing the subtree rooted at u_0 by T_0. By definition we have

$$T' : (P, F) \overset{\mathcal{U}'}{\vdash} C(w),\ h(T') \leq 1,$$

and thus by Corollary 5.12 we get $\varphi_{C(w)} * (P, F) \in \mathcal{G}_1(\mathcal{U}')$. □

For the other direction of the simulation we need another form of “transitivity” of k-resolution.

Lemma 6.14 *Consider $P \in \mathcal{PI}$, $F \in \mathcal{CLS}$, $C \in \mathcal{CL}$ and $k \in \mathbb{N}_0$, and assume $(P,F) \overset{\mathcal{U}',k}{\vdash} C$ and that for all $D \in F$ there is a clause $D' \subseteq D$ with $P \overset{\mathcal{U},\mathrm{k}}{\vdash} D'$. Then there is a clause $C' \subseteq C$ with $P \overset{\mathcal{U},\mathrm{k}}{\vdash} C'$.*

Proof. Consider a clause-set G with $(P,F) \xrightarrow{\mathcal{U}'} G$ and a resolution tree $T : G \overset{\mathrm{k}}{\vdash} C$. For a leaf w of T with $\varphi_{C(w)} * P \notin \mathcal{U}$ by definition of $\mathcal{U}'$ there is a clause $D \in F$ with $D \subseteq C(w)$, while now by assumption there is a clause $D' \subseteq D$ with $P \overset{\mathcal{U},\mathrm{k}}{\vdash} D'$. Thus by Lemma 6.4 (applied with $F := G$ and $C_0 := C$) we obtain $C' \subseteq C$ with $P \overset{\mathcal{U},\mathrm{k}}{\vdash} C'$. □

Lemma 6.15 *For any $P \in \mathcal{UPI}$ we have $\mathrm{wid}_{\mathcal{U}}(P) \leq {}^{\mathrm{i}}\mathrm{wid}_{\mathcal{U}}(P) + 1$.*

Proof. Let $k := {}^{\mathrm{i}}\mathrm{wid}_{\mathcal{U}}(P)$. We prove by induction on the composition of $F(\mathcal{U},k,P)$, that for all clauses $C \in F(\mathcal{U},k,P)$ there is a clause $C' \subseteq C$ with $P \overset{\mathcal{U},k+1}{\vdash} C'$, from which the lemma immediately follows. So consider a clause-set $F \in \mathcal{CLS}$ and a clause $C \in \mathcal{CL}$ with $|C| \leq k$ and $\mathrm{var}(C) \subseteq \mathrm{var}(P)$ such that $\varphi_C * (P,F) \in \mathcal{G}_1(\mathcal{U}')$ holds, while for all $D \in F$ there is a $D' \subseteq D$ with $P \overset{\mathcal{U},k+1}{\vdash} D'$. Now by Lemma 6.8, part 2 we have $\varphi_C * (P,F) \in \mathcal{W}_1(\mathcal{U}')$, hence $(P,F) \overset{\mathcal{U}',k+1}{\vdash} C$ by Lemma 6.3, part 2, and thus by Lemma 6.14 there is a clause $C' \subseteq C$ with $P \overset{\mathcal{U},k+1}{\vdash} C'$. □

Theorem 6.16 *Assume $\mathcal{U}$ is stable under application of partial assignments. If for a problem instance $P \in \mathcal{PI}$ and $k \in \mathbb{N}_0$ the output of algorithm $\mathfrak{W}$ is $\mathfrak{W}(P,k) =$ "yes", then $\mathrm{wid}_{\mathcal{U}}(P) \leq k+1$ holds (and thus $P \in \mathcal{UPI}$), while otherwise $\mathrm{wid}_{\mathcal{U}}(P) \geq k+1$ is the case (possibly P is satisfiable). The running of algorithm $\mathfrak{W}$ on inputs $P \in \mathcal{PI}$ and $k \in \mathbb{N}_0$ in the oracle model (i.e., counting computation of $\mathrm{var}(P)$ and $\varphi * P$ as well as a decision of "$P \in \mathcal{U}$?" as one step) is bounded by $O(n(P)^{3k+2}d^{3k+1})$.*

Proof. Due to Lemmas 6.13 and 6.15 only the estimation of the running time remains. Let $n := n(P)$ and $F := F(\mathcal{U},k,P)$. We have $c(F) \leq O(n^k d^k)$, while there are at most $c(F)$ many cycles of the while-loop in algorithm $\mathfrak{W}$ (see Figure 7), and the search for a new clause C

to be added needs at most $O(c(F)\cdot(c(F)n^2d))$ steps, where $O(c(F)n^2d)$ is the running time for decision of "$\varphi_C * (P,F) \in \mathcal{G}_1(\mathcal{U}')$?". □

6.6. Upper bounds on (full) resolution complexity

Lemma 6.17 *Assume $\mathcal{U}$ is stable under application of partial assignments. Then for all $P \in \mathcal{UPI}$ we have the following upper bound on the resolution complexity of P:*

$$\mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P) \le O(n(P)^{\mathrm{wid}_{\mathcal{U}}(P)+1}\, d^{\mathrm{wid}_{\mathcal{U}}(P)}).$$

Proof. Let $k := \mathrm{wid}_{\mathcal{U}}(P)$. By Lemma 6.13 we have $\bot \in F := F(\mathcal{U},k,P)$, where $F(\mathcal{U},k,P)$ is as specified at the beginning of Subsection 6.5. Each clause in F is derived from other clauses in F and clauses directly entailed by P by an input resolution tree T_C (i.e., $h(T_C) \le 1$), where by Lemma 5.7 w.l.o.g. we may assume $\mathrm{ht}(T_C) \le n(P)$ (or one uses the regularisation Lemma 5.4), so that T_C has at most $2n(P)+1$ nodes. Putting all these trees T_C together (in the order given by the computation of F) we get a resolution refutation $T : P \overset{\mathcal{U}}{\vdash} \bot$ with $c(\mathrm{F}^+(T)) \le c(F)\cdot(2n(P)+1)$, where $c(F) = O(n(P)^k d^k)$. □

Putting Lemma 6.17 together with Theorem 6.16 and Theorem 6.12 we get

Theorem 6.18 *Assume $\mathcal{U}$ is stable under application of partial assignments, and consider an unsatisfiable problem instance $P \in \mathcal{UPI}$ with $n := n(P) > 0$. By computing $\mathfrak{W}(P,0), \mathfrak{W}(P,1), \ldots$ until we get the output "yes" , we can compute the first level $k := \mathrm{wid}'_{\mathcal{U}}(P)$ of inclusion in time*

$$\begin{aligned} O(n^{3k+2}\, d^{3k+1}) &\le O(2^{2\ln n+\ln d+6\sqrt{2n\ln \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P)}(\ln n+\ln d)}) \\ &\le 2^{O(\sqrt{n}\sqrt{\ln \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P)}(\ln n+\ln d))}. \end{aligned}$$

Only impairing the running time by a constant factor we can also obtain a resolution refutation $T : P \overset{\mathcal{U}}{\vdash} \bot$ with

$$\begin{aligned} c(\mathrm{F}^+(T)) &\le O(n^{k+1}\, d^k) \\ &\le O(2^{\ln n+2\sqrt{2n\ln \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P)}(\ln n+\ln d)}) \\ &\le 2^{O(\sqrt{n}\sqrt{\ln \mathrm{Comp}_{\mathrm{R}(\mathcal{U})}(P)}(\ln n+\ln d))}. \end{aligned}$$

For an unsatisfiable instance $P \in \mathcal{UPI}$ let $^{\mathrm{b}}\mathbf{wid_{\mathcal{U}}(P)}$ be the minimal number $k \in \mathbb{N}_0$ such that there exists a resolution tree $T : P \overset{\mathcal{U}}{\vdash} \bot$ with $\mathrm{F}^+(T) \in k\text{-}\mathcal{CLS} := k\text{-}\mathcal{CLS}(\mathcal{VA})$ (the "b" reminds of "bounded resolution" as introduced in (Galil, 1975)). Generalising the notion of a maximal input clause-length for clause-sets, by $\boldsymbol{p_{\mathcal{U}}(P)}$ we denote the maximal $|C|$ for clauses $C \in \mathcal{CL}(\mathrm{var}(P))$ such that $\varphi_C * P \in \mathcal{U}$ holds, and there is no clause $C' \subset C$ with $\varphi_{C'} * P \in \mathcal{U}$. In the case of an unsatisfiable clause-set $F \in \mathcal{USAT}$ we have $p_{\mathcal{U}_0}(F) = 0$ iff $\bot \in F$, while otherwise $p_{\mathcal{U}_0}(F)$ is equal to the maximal $|C|$ for $C \in F$ such that there is no $C' \in F$ with $C' \subset C$.

Lemma 6.19 *For all $P \in \mathcal{G}_1(\mathcal{U})$ we have $^{\mathrm{b}}\mathrm{wid}_{\mathcal{U}}(P) \le p_{\mathcal{U}}(P)$.*

Proof. From $P \in \mathcal{G}_1(\mathcal{U})$ we get $P \in \mathcal{W}_1(\mathcal{U})$ by Lemma 6.8. So consider a resolution tree $T : P \overset{\mathcal{U},1}{\vdash} \bot$, where by Lemma 6.1, part 1 we may assume that all axiom clauses in T have length at most $p_{\mathcal{U}}(P)$. Now by definition of 1-resolution we get $\mathrm{F}^+(T) \in p_{\mathcal{U}}(P)\text{-}\mathcal{CLS}(P)$. □

Before considering the relation between k-resolution and bounded resolution in Lemma 6.14, we state two easy auxiliary lemmas, proven the same way as the corresponding Lemma 6.3, part 2 and Lemma 6.14.

Lemma 6.20 *If $^{\mathrm{b}}\mathrm{wid}_{\mathcal{U}}(\varphi * P) \le k \in \mathbb{N}_0$ holds, then there is a resolution tree $T : P \overset{\mathcal{U}}{\vdash} C_\varphi$ with $\mathrm{F}^+(T) \in (k + n(\varphi))\text{-}\mathcal{CLS}$.*

Lemma 6.21 *Given a resolution tree $T_0 : (P, F) \overset{\mathcal{U}'}{\vdash} C$ using only clauses of length at most k (that is $\mathrm{F}^+(T_0) \in k\text{-}\mathcal{CLS}$) such that for all clauses $D \in F$ there is a resolution tree $T_D : P \overset{\mathcal{U}}{\vdash} D$ also with $\mathrm{F}^+(T_D) \in k\text{-}\mathcal{CLS}$, then there is a resolution tree $T : P \overset{\mathcal{U}}{\vdash} C$ with $\mathrm{F}^+(T) \subseteq k\text{-}\mathcal{CLS}$.*

Lemma 6.22 *Assume $\mathcal{U}$ is stable under application of partial assignments. Then for all $P \in \mathcal{UPI}$ we have*

$$\begin{aligned}\mathrm{wid}_{\mathcal{U}}(P) \le {}^{\mathrm{b}}\mathrm{wid}_{\mathcal{U}}(P) &\le \max({}^{\mathrm{i}}\mathrm{wid}_{\mathcal{U}}(P), p_{\mathcal{U}}(P)) + {}^{\mathrm{i}}\mathrm{wid}_{\mathcal{U}}(P) \\ &\le \max(\mathrm{wid}_{\mathcal{U}}(P), p_{\mathcal{U}}(P)) + \mathrm{wid}_{\mathcal{U}}(P).\end{aligned}$$

Proof. The first inequality is trivial, while the last one follows by Lemma 6.13, so consider the second one. Let $k := {}^{\mathrm{i}}\mathrm{wid}_{\mathcal{U}}(P)$ and define $k' := \max(k, p_{\mathcal{U}}(P)) + k$. We want to show that for all clauses $C \in F(\mathcal{U}, k, P)$ there is a resolution tree $T : P \overset{\mathcal{U}}{\vdash} C$ with $\mathrm{F}^+(T) \in k'$–$\mathcal{CLS}$ (yielding immediately the second inequality due to $\bot \in F(\mathcal{U}, k, P)$). We prove the claim by induction on the construction of $F(\mathcal{U}, k, P)$. So consider a clause-set $F \in k$–$\mathcal{CLS}$ and a clause $C \in k$-$\mathcal{CL}(\mathrm{var}(P))$ with $\varphi_C * (P, F) \in \mathcal{G}_1(\mathcal{U}')$, such that for all clauses $D \in F$ there is a resolution tree $T_D : P \overset{\mathcal{U}}{\vdash} D$ with $\mathrm{F}^+(T_D) \subseteq k'$–$\mathcal{CLS}$. By Lemma 6.19 we have

$$ {}^{\mathrm{b}}\mathrm{wid}_{\mathcal{U}'}(\varphi_C * (P, F)) \leq p_{\mathcal{U}'}(P, F), $$

while by definition we have $p_{\mathcal{U}'}(P, F) \leq \max(p_{\mathcal{U}}(P), k)$. Now Lemma 6.20 yields that there is a resolution tree

$$ T_0 : (P, F) \overset{\mathcal{U}'}{\vdash} C, \quad \mathrm{F}^+(T_0) \in k'\text{–}\mathcal{CLS} $$

and thus by Lemma 6.21 we obtain $T : P \overset{\mathcal{U}}{\vdash} C$ with $\mathrm{F}^+(T) \in k'$–$\mathcal{CLS}$. □

6.8. Induced width of constraint satisfaction problems

The concept of *induced width* for constraint satisfaction problems P has been introduced in (Dechter and Pearl, 1988), considering the *constraint graph* $G(P)$ with the variables of P as its nodes and an edge joining two variables if they are mentioned together by some constraint of P, and defining the induced width of P as the induced width of $G(P)$, which in turn can be seen as an upper bound on the length of the clauses the (original) DP-procedure (Davis and Putnam, 1960) supplied with an optimal variable ordering could produce in the worst case (cf. (Dechter and Rish, 1994)). By definition the induced width of a graph G equals the *dimension* of G, which is the maximal number of neighbours a node may have in an optimal elimination process as considered in (Rose, 1970; Arnborg, 1985; Bertele and Brioschi, 1972) (eliminating one node after another and adding edges between the (remaining) neighbours of the eliminated vertex so that they form a clique). In (Arnborg, 1985) one also find (hints for) the proof that a graph G has dimension at most k iff G is a *partial k-tree* (a subgraph of a k-tree), which in turn holds iff G has *treewidth* at most k (cf. chapter 7 in (Bodlaender, 1998)). In this subsection we show that (as one

would expect) bounded resolution can refute unsatisfiable constraint satisfaction problem with bounded induced width (motivated by the proof of proposition 4.9 in (Baker, 1995)). For more information on the connection between resolution and treewidth see (Alekhnovich and Razborov, 2002).

We start with the definition of the induced width $\mathbf{{}^{ind}wid(G)}$ of a simple undirected graph G. If G is empty, then ${}^{\text{ind}}\text{wid}(G) := -1$. Otherwise consider a linear order $\leq$ on the set $V(G)$ of vertices of G. Let $\mathbf{ind_{\leq}(G)}$ be the smallest (undirected) graph G' with $V(G') = V(G)$ and $E(G') \supseteq E(G)$ such that for all edges $\{x,a\},\{y,a\} \in E(G')$ with $x \neq y$ and $x < a$ as well as $y < a$ also $\{x,y\} \in E(G')$ holds. Now ${}^{\text{ind}}\text{wid}(G,\leq)$ is the maximum over all nodes $w \in V(G)$ of the number of nodes w' such that w' is joined by an edge with w in $\text{ind}_{\leq}(G)$ and $w' < w$ holds, while ${}^{\text{ind}}\text{wid}(G)$ is the minimum of ${}^{\text{ind}}\text{wid}(G,\leq)$ over all linear orders $\leq$ on $V(G)$.

Now consider a constraint satisfaction problem P as defined in Subsection 2.4. For a constraint $C = (\vec{x}, R, \vec{a})$ define $\mathbf{var(C)}$ as the set of variables $v \in \mathcal{VA}$ such that there is an index i with $\vec{x}_i = v$ and $\vec{a}_i =$ "?". Obviously $\text{var}(P) = \bigcup_{C \in P} \text{var}(C)$ holds. Let the *constraint graph* $\boldsymbol{G(P)}$ be the simple undirected graph with vertex set $\text{var}(P)$ and an edge joining variables $v_1, v_2 \in \text{var}(P)$ $(v_1 \neq v_2)$ if there is a constraint $C \in P$ with $v_1, v_2 \in \text{var}(C)$. Define the induced width of P as $\mathbf{{}^{ind}wid(P)} := {}^{\text{ind}}\text{wid}(G(P))$.

We remind the reader at the definition of the basic oracle $\mathcal{U}_0$ for unsatisfiable constraint satisfaction problems as the set of all constraint satisfaction problems P (with substitutions) containing a constraint $C \in P$ with $\text{var}(C) = \emptyset$ such that already $\{C\}$ is unsatisfiable (cf. the introduction to Section 3).

For the refutation of constraint satisfaction problems a natural modification of bounded resolution comes in handily, where we do not bound the length of *all* clauses appearing in the refutation but only the length of *resolvents*. So let $\mathbf{{}^{br}wid_{\mathcal{U}}(P)}$ for $P \in \mathcal{UPI}$ (the "r" stands for "resolvents") be the minimal $k \in \mathbb{N}_0 \cup \{-1\}$ such that there is a resolution tree $T : P \overset{\mathcal{U}}{\vdash} \bot$ with the property that all non-leaf nodes $w \in V(T) \setminus \text{lvs}(T)$ are labelled with clauses of length at most k (i.e., $|C(w)| \leq k$). Obviously we have

$$ {}^{\text{i}}\text{wid}_{\mathcal{U}}(P) \leq {}^{\text{br}}\text{wid}_{\mathcal{U}}(P) \leq {}^{\text{b}}\text{wid}_{\mathcal{U}}(P) \leq {}^{\text{br}}\text{wid}_{\mathcal{U}}(P) + 1. $$

For example consider the clause-set $F = \{\{a, x\}, \{\overline{a}, x\}, \{b, \overline{x}\}, \{\overline{b}, \overline{x}\}\}$ for the boolean case (using pairwise different variables a, b, x), where we have ${}^{\rm b}\mathrm{wid}_{\mathcal{U}_0}(F) = \mathrm{wid}_{\mathcal{U}_0}(F) = 2$ and ${}^{\rm br}\mathrm{wid}_{\mathcal{U}_0}(F) = {}^{\rm i}\mathrm{wid}_{\mathcal{U}_0}(F) = 1$.

Lemma 6.23 *For all unsatisfiable constraint satisfaction problems P we have the inequality ${}^{\rm br}\mathrm{wid}_{\mathcal{U}_0}(P) \leq {}^{\rm ind}\mathrm{wid}(P)$.*

Proof. If $P \in \mathcal{U}_0$ holds, then ${}^{\rm br}\mathrm{wid}_{\mathcal{U}_0}(P) = -1 \leq {}^{\rm ind}\mathrm{wid}(P)$. So assume $P \notin \mathcal{U}_0$ and consider an optimal linear order $\leq$ for $\mathrm{var}(P)$ (with ${}^{\rm ind}\mathrm{wid}(G(P), \leq) = {}^{\rm ind}\mathrm{wid}(G(P))$). Let $G' := \mathrm{ind}(G(P))$, and for a variable $v \in \mathrm{var}(P)$ let $\mathrm{dp}(v)$ be the set of direct predecessors of v in G' w.r.t. order $\leq$, that is

$$\mathrm{dp}(v) := \{w \in \mathrm{var}(P) : w < v \wedge \{w, v\} \in E(G')\}.$$

By definition of induced width we have $|\mathrm{dp}(v)| \leq {}^{\rm ind}\mathrm{wid}(P)$ for all variables $v \in \mathrm{var}(P)$. Now consider a resolution tree $T : P \overset{\mathcal{U}_0}{\vdash} \bot$ using only minimal axioms, that is for all leaves $w \in \mathrm{lvs}(T)$ there is no clause $C' \subset C(w)$ with $\varphi_{C'} * P \in \mathcal{U}_0$. Furthermore we require that for any two (different) resolution variables v_1, v_2 in T such that v_2 is a successor of v_1 in T we have $v_1 < v_2$ (recall that the edges in our trees are directed towards the leaves). Such a resolution tree exists by Lemma 5.7 and the proof of Lemma 5.1, part 1.

We show by induction that for all nodes $w \in V(T) \setminus \mathrm{lvs}(T)$ we have

$$\mathrm{var}(C(w)) \subseteq \mathrm{dp}(v(w))$$

(where $v(w)$ is the resolution variable at node w) proving the assertion. Let $\{w_1, \ldots, w_m\}$ be the set of direct successors of w in T. For each $1 \leq i \leq m$ we have $v(w) \in \mathrm{var}(C(w_i))$ due to the definition of the resolution rule, and furthermore for all variables $v' \in \mathrm{var}(C(w_i)) \setminus \{v(w)\}$ we have $v' < v(w)$, since all variables in $C(w_i)$ must be resolved upon the path from the root to w_i in T. First consider the case that w_i is a leaf of T. Due to $\varphi_{C(w_i)} * P \in \mathcal{U}_0$ and the minimality of the axioms there must be a constraint $C \in P$ with $\mathrm{var}(C(w_i)) = \mathrm{var}(C)$. So by definition of the constraint graph $G(P)$ and $E(G') \supseteq E(G(P))$ we get

$$\mathrm{var}(C(w_i)) \subseteq \{v(w)\} \cup \mathrm{dp}(v(w)).$$

Now consider the case that w_i is not a leaf, so that by induction hypothesis we have $\mathrm{var}(C(w_i)) \subseteq \mathrm{dp}(v(w_i))$. Consider a variable $v' \in \mathrm{var}(C(w_i))$, $v \neq v(w)$. By definition of " direct predecessor" we have

$\{v', v(w_i)\}, \{v(w), v(w_i)\} \in E(G')$ as well as $v' < v(w_i)$ and $v(w) < v(w_i)$, and thus also $\{v', v(w)\} \in E(G')$ must hold (due to the definition of the induced graph). Using $v' < v(w)$ in fact we get $v' \in \mathrm{dp}(v(w))$, whence again

$$\mathrm{var}(C(w_i)) \subseteq \{v(w)\} \cup \mathrm{dp}(v(w))$$

holds true. Altogether the induction statement follows. □

We have shown in this final subsection only the very starting point of investigations into the important connection between resolution theory and and various notions of width from graph theory (where often due to technicalities the very intuitive basis for this relation is overseen). Just to mention a few interesting developments, there is a close relation of width as studied in this section and the notion of bounded width in Datalog (see (Feder and Vardi, 1998)), while a generalisation of tree width is studied in (Courcelle et al., 1999).

References

Alekhnovich, M., S. Buss, S. Moran, and T. Pitassi: 2001, 'Minimum Propositional Proof Length is NP-Hard to Linearly Approximate'. *Journal of Symbolic Logic* **66**, 171–191.

Alekhnovich, M. and A. Razborov: 2001, 'Resolution is Not Automatizable Unless W[P] is Tractable'. In: *Proc. of the 42nd IEEE FOCS*. pp. 210–219.

Alekhnovich, M. and A. A. Razborov: 2002, 'Satisfiability, Branch-width and Tseitin Tautologies'. Unpublished (?).

Arnborg, S.: 1985, 'Efficient algorithms for combinatorial problems on graphs with bounded decomposability - a survey'. *BIT* **25**, 2–23.

Baker, A. B.: 1995, 'Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results'. Ph.D. thesis, University of Oregon. The ps-file can be down loaded using the URL `http://www.cirl.uoregon.edu/baker/thesis.ps.gz`.

Beame, P. and T. Pitassi: 1996, 'Simplified and Improved Resolution Lower Bounds'. In: *37th Symposium on Foundations of Computer Science (FOCS' 96)*. pp. 274–282.

Ben-Sasson, E. and A. Wigderson: 1999, 'Short Proofs are Narrow — Resolution made Simple'. Technical Report TR99-022, ECCC Electronic Colloquium on Computational Complexity.

Bertele, U. and F. Brioschi: 1972, *Nonserial Dynamic Programming*. New York: Academic Press.

Bodlaender, H. L.: 1998, 'A partial k-arboretum of graphs with bounded treewidth'. *Theoretical Computer Science* **209**, 1–45.

Bonet, M. L., C. Domingo, R. Gavaldá, A. Maciel, and T. Pitassi: 1999, 'Non-automatizability of bounded depth Frege proofs'. In: *Fourteenth Annual IEEE Conference on Computational Complexity*.

Bonet, M. L. and N. Galesi: 1999, 'A Study of Proof Search Algorithms for Resolution and Polynomial Calculus'. In: *40th Annual Symposium on Foundations of Computer Science (FOCS)*. pp. 422–431.

Bonet, M. L., T. Pitassi, and R. Raz: 1997, 'No Feasible Interpolation for TC^0-Frege Proofs'. In: *38th Symposium on Foundations of Computer Science (FOCS' 97)*. pp. 254–263.

Bourbaki, N.: 1989, *Algebra I: Chapters 1-3*, Elements of Mathematics. Springer-Verlag.

Büning, H. K.: 1993, 'On generalized Horn formulas and k-resolution'. *Theoretical Computer Science* **116**, 405–413.

Buro, M. and H. K. Büning: 1996, 'On Resolution with Short Clauses'. *Annals of Mathematics and Artificial Intelligence* **18**(2-4), 243–260.

Chang, C. L.: 1970, 'The unit proof and the input proof in theorem proving'. *Journal of the Association for Computing Machinery* **17**(4), 698–707.

Clegg, M., J. Edmonds, and R. Impagliazzo: 1996, 'Using the Groebner basis algorithm to find proofs of unsatisfiability'. In: *Proceedings of the 28th ACM Symposium on Theory of Computation*. pp. 174–183.

Courcelle, B., J. A. Makowsky, and U. Rotics: 1999, 'Linear Time Solvable Optimization Problems on Graphs of Bounded Clique Width'. Can be obtained from `http://www.cs.technion.ac.il/~janos/`.

Davis, M. and H. Putnam: 1960, 'A Computing procedure for quantification theory'. *Journal of the ACM* **7**, 201–215.

Dechter, R. and J. Pearl: 1988, 'Network-Based Heuristics for Constraint-Satisfaction Problems'. *Artificial Intelligence* **34**, 1–38.

Dechter, R. and I. Rish: 1994, 'Directional Resolution: The Davis-Putnam Procedure, Revisited'. In: *Proceedings of KR-94*. An extended version can be obtained from `http://www.ics.uci.edu/~irinar`.

Diestel, R.: 2000, *Graph Theory*, Vol. 173 of *Graduate Texts in Mathematics*. New York: Springer, second edition.

Esteban, J. L. and J. Torán: 1999, 'Space Bounds for Resolution'. In: C. Meinel and S. Tison (eds.): *16th Annual Symposium on Theoretical Aspects of Computer Science*, Vol. 1563 of *LNCS*. Springer.

Feder, T. and M. Y. Vardi: 1998, 'The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory'. *SIAM J. Comput.* **28**(1), 57–104.

Galil, Z.: 1975, 'On the validity and complexity of bounded resolution'. In: *Proceedings of seventh annual ACM Symposium on Theory of Computing*. pp. 72–82.

Gallo, G. and M. G. Scutellà: 1988, 'Polynomially solvable satisfiability problems'. *Information Processing Letters* **29**, 221–227.

Goerdt, A.: 1993, 'Regular Resolution Versus Unrestricted Resolution'. *SIAM Journal on Computing* **22**(4), 661–683.

Groote, J. F. and J. P. Warners: 2000, 'The Propositional Formula Checker HeerHugo'. *Journal of Automated Reasoning* **24**, 101–125.

Harrison, J.: 1996, 'Stålmarck's Algorithm as a HOL Derived Rule'. In: *Theorem proving in higher order logics: 9th International Conference, TPHOLs'96*. pp. 221–234.

Iwama, K.: 1997, 'Complexity of Finding Short Resolution Proofs'. In: *Mathematical Foundations of Computer Science 1997 (MFCS'97), 22nd International Symposium*. pp. 309–318, Springer.

Krishnamurthy, B. and R. N. Moll: 1981, 'Examples of hard tautologies in the propositional calculus'. In: *13th ACM Symposium on Theory of Computing (STOC'81)*. pp. 28–37.

Kullmann, O.: 1999a, 'Heuristics for SAT algorithms: Searching for some foundations'. Manuscript. 23 pages, previous version in *SAT'98*.

Kullmann, O.: 1999b, 'Investigating a general hierarchy of polynomially decidable classes of CNF's based on short tree-like resolution proofs'. Technical Report TR99-041, Electronic Colloquium on Computational Complexity (ECCC).

Kullmann, O.: 2000a, 'An improved version of width restricted resolution'. In: *Electronical Proceedings of Sixth International Symposium on Artificial Intelligence and Mathematics*. 11 pages; see "Accepted Contributions" at `http://rutcor.rutgers.edu/~amai/`.

Kullmann, O.: 2000b, 'Investigations on autark assignments'. *Discrete Applied Mathematics* **107**, 99–137.

Kullmann, O.: 2001, 'On the use of autarkies for satisfiability decision'. In: H. Kautz and B. Selman (eds.): *LICS 2001 Workshop on Theory and Applications of Satisfiability Testing (SAT 2001)*, Vol. 9 of *Electronic Notes in Discrete Mathematics (ENDM)*. Elsevier Science.

Kullmann, O.: 2002, 'Some general considerations on heuristics for search problems'. In preparation.

Kullmann, O.: 2003, 'Autarkies — a tool kit for generalised satisfiability decision'. Journal version of (Kullmann, 2001); invited issue for Annals of Mathematics and Artificial Intelligence.

Li, C. M. and Anbulagan: 1997, 'Look-ahead versus look-back for satisfiability problems'. In: *Proceedings of CP-97*, Vol. 1330 of *Lecture Notes in Computer Science*. pp. 342–356, Springer.

Liberatore, P.: 2000, 'On the complexity of choosing the branching literal in DPLL'. *Artificial Intelligence* **116**, 315–326.

Lovász, L., M. Naor, I. Newman, and A. Wigderson: 1991, 'Search Problems in the Decision Tree Model'. In: *32nd Symposium on Foundations of Computer Science (FOCS' 91)*. pp. 576–585.

Mitchell, D. G.: 1998, 'Hard Problems for CSP Algorithms'. In: *Proceedings Fifteenth National Conference on Artificial Intelligence (AAAI-98)*.

Pretolani, D.: 1996, 'Hierarchies of Polynomially Solvable Satisfiability Problems'. *Annals of Mathematics and Artificial Intelligence* **17**(3-4), 339–357.

Reckhow, R. A.: 1975, 'On the lengths of proofs in the propositional calculus'. Ph.D. thesis, University of Toronto, Department of Computer Science.

Rose, D. J.: 1970, 'Triangulated Graphs and the Elimination Process'. *Journal of Mathematical Analysis and Applications* **32**, 597–609.

Sheeran, M. and G. Stålmarck: 1998, 'A Tutorial on Stålmarck's Proof Procedure for Propositional Logic'. In: *FMCAD'98*, Vol. 1522 of *Lecture Notes in Computer Science*. pp. 82–99.

Silva, J. P. M. and K. A. Sakallah: 1996, 'GRASP—A New Search Algorithm for Satisfiability'. Technical Report CSE-TR-292-96, University of Michigan, Department of Electrical Engineering and Computer Science.

Stålmarck, G. and M. Säflund: 1990, 'Modeling and verifying systems and software in propositional logic'. In: B. Daniels (ed.): *Safety of Computer Control Systems (SAFECOMP'90)*. pp. 31–36.

Torán, J.: 1999, 'Lower Bounds for Space in Resolution'. In: *Computer Science Logic, 13th International Workshop, CSL'99*, Vol. 1683 of *Lecture Notes in Computer Science*. pp. 362–373, Springer.

Tseitin, G.: 1968, 'On the complexity of derivation in propositional calculus'. In: *Seminars in Mathematics*, Vol. 8. V.A. Steklov Mathematical Institute, Leningrad. English translation: Studies in mathematics and mathematical logic, Part II (A.O. Slisenko, editor), 1970, pages 115-125.

Urquhart, A.: 1995, 'The complexity of propositional proofs'. *The Bulletin of Symbolic Logic* **1**(4), 425–467.

Yamasaki, S. and S. Doshita: 1983, 'The Satisfiability Problem for a Class Consisting of Horn Sentences and Some Non-Horn Sentences in Propositional Logic'. *Information and Control* **59**, 1–12.

Zhang, H.: 1997, 'SATO: an Efficient Propositional Prover'. In: *Proc. of International Conference on Automated Deduction (CADE-97)*.