Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information and Computation 189 (2004) 182–201
Information and Computation
www.elsevier.com/locate/ic

# On the automatizability of resolution and related propositional proof systems

Albert Atserias*,[1] and María Luisa Bonet[2]

*Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain*

Received 23 April 2003; revised 15 October 2003

**Abstract**

A propositional proof system is automatizable if there is an algorithm that, given a tautology, produces a proof in time polynomial in the size of its smallest proof. This notion can be weakened if we allow the algorithm to produce a proof in a stronger system within the same time bound. This new notion is called weak automatizability. Among other characterizations, we prove that a system is weakly automatizable exactly when a weak form of the satisfiability problem is solvable in polynomial time. After studying the robustness of the definition, we prove the equivalence between: (i) Resolution is weakly automatizable, (ii) Res($k$) is weakly automatizable, and (iii) Res($k$) has feasible interpolation, when $k>1$. In order to prove this result, we show that Res(2) has polynomial-size proofs of the reflection principle of Resolution, which is a version of consistency. We also show that Res($k$), for every $k>1$, proves its consistency in polynomial size, while Resolution does not. In fact, we show that Resolution proofs of its own consistency require almost exponential size. This gives a better lower bound for the monotone interpolation of Res(2) and a separation from Resolution as a byproduct. Our techniques also give us a way to obtain a large class of examples that have small Resolution refutations but require relatively large width. This answers a question of Alekhnovich and Razborov related to whether Resolution is automatizable in quasipolynomial-time.


*Keywords:* Propositional proof complexity; Feasible interpolation; Automatizability; Reflection principles; Resolution; Res(2)

*Corresponding author. Fax: +34-93-401-70-14.
*E-mail addresses:* atserias@lsi.upc.es (A. Atserias), bonet@lsi.upc.es (M.L. Bonet).

[1]Partially supported by CICYT TIC2001-1577-C03-02, HA2000-41, and the Future and Emerging Technologies programme of the EU under Contract No. IST-1999-14186 (ALCOM-FT).
[2]Partially supported by CICYT TIC2001-1577-C03-02 and HA2000-41.

0890-5401/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2003.10.004

## 1. Introduction

Considerable effort has gone into studying algorithms for propositional satisfiability in several areas of computer science, despite the problem is NP-complete. The complementary problem of verifying propositional tautologies also received considerable attention, despite the fact that the problem is obviously also hard, namely co-NP-complete. As a consequence of the pioneering work of Cook and Reckhow [14], there is strong evidence that no propositional proof system $P$ can prove every tautology in polynomial size. Thus, there is strong evidence that for no proof system $P$, there will be an algorithm that will produce $P$-proofs in time polynomial in the size of the tautology. This is because in some cases we might require super-polynomial time just to write down the proof.

Considering this limitation of proof systems, Bonet et al. [10] proposed the following definition. A propositional proof system $P$ is *automatizable* if there exists an algorithm that, given a tautology, produces a $P$-proof of it in time polynomial in the size of its smallest $P$-proof. The idea is that if short $P$-proofs exist, an automatization algorithm for $P$ should find them quickly. Somewhat surprisingly perhaps, this weaker notion turned out to be a strong requirement too for several natural systems. In the sequence of papers [5,10,19] it was proved that no proof system that simulates bounded-depth Frege is automatizable, unless some widely accepted cryptographic conjecture is violated. Later, Alekhnovich and Razborov [4] proved that Resolution is not automatizable under a reasonable assumption in parameterized complexity theory. The situation remains unknown for proof systems that lie between Resolution and bounded-depth Frege. In particular, it is open whether a proof system that simulates Resolution is automatizable.

In this paper we show the robustness of a notion that we call weak automatizability, and study in detail the particular case of Resolution. We say that a proof system $P$ is *weakly automatizable* if there exists proof system $Q$ and an algorithm that, given a tautology, produces a $Q$-proof of it in time polynomial in its smallest $P$-proof. In such a case we say that $P$ is automatizable in terms of $Q$. Obviously, $P$ is automatizable exactly when it is automatizable in terms of itself. Thus, our notion is a natural extension. In fact, we believe that weak automatizability might be more interesting from the point of view of applications. This is because usually we want to verify tautologies without restricting ourselves to having proofs in a particular system. But at the same time it is desirable to have the time of the proof search algorithm measured in terms of a proof system that we know a lot about (such as Resolution). We discuss the robustness of this new definition by providing several characterizations that we describe next. Let us fix a refutational proof system $P$ and consider the following weak form of the satisfiability problem:

> WEAK SAT for $P$: Given a CNF formula $F$ and a positive integer $m$, return 1 if $F$ is satisfiable, return 0 if $F$ has a $P$-refutation of size at most $m$, and return anything in any other case.

Intuitively, an algorithm that solves this weak form of the satisfiability problem might provide a wrong answer, but only on those CNF formulas that require long refutations in $P$, for a bound $m$ of our choice. Thus, if the unsatisfiable instances of our application have short refutations for some application-dependent reason, a fast algorithm for this problem is of practical importance. Our first result is that $P$ is weakly automatizable if and only if the problem WEAK SAT for $P$ is solvable in time polynomial in $m$ and $|F|$, the size of $F$.

As it turns out, the computational problem WEAK SAT for a system $P$ is not new in the proof complexity literature. Pudlák [27,28], following the work of Razborov [29], considered the

problem of separating the canonical NP-pair of a proof system $P$ by a polynomial-time algorithm. It is almost immediate from the definitions (see Section 2), that our problem is a reformulation of theirs. Moreover, Pudlák [28] showed that the canonical NP-pair of $P$ is polynomially separable if and only if there exists a proof system that simulates $P$ and is automatizable. Summing up, all four concepts introduced so far are equivalent:

(i) $P$ is weakly automatizable.
(ii) There is a proof system that simulates $P$ and is automatizable.
(iii) WEAK SAT for $P$ is solvable in time polynomial in $m$ and $|F|$.
(iv) The canonical NP-pair of $P$ is polynomially separable.

For automatizability, we also present an equivalent notion which is related to the problem of proof size approximation first considered in [3]. We provide details in Section 3.

Let us turn our attention to the particular case of Resolution. As pointed out already, it remains open whether a system that simulates Resolution can be automatizable. Thus, in view of our previous result, it remains open whether Resolution is weakly automatizable. In this paper we show that this question is equivalent to whether Res(2) has feasible interpolation. In fact we show that the following are equivalent for every $k > 1$: (i) Resolution is weakly automatizable, (ii) Res($k$) is weakly automatizable, and (iii) Res($k$) has feasible interpolation. The systems Res($k$) and the notion of feasible interpolation will be defined in Sections 2 and 4. Let us say for the moment, that Resolution, Cutting Planes, Relativized Bounded Arithmetic, Polynomial Calculus, Lovász-Schrijver, and Nullstellensatz all have feasible interpolation (see [9,13,18,20,25,24,26,30]). On the other hand, the stronger system Frege, and any system that simulates bounded-depth Frege, do not have feasible interpolation under a cryptographic conjecture [5,10,19].

To prove the three equivalences mentioned above we show that Res(2) has polynomial-size proofs of the reflection principle of Resolution, which is a form of consistency saying that if a CNF formula is satisfiable, then it does not have a Resolution refutation. We extend this result by proving that Res($k$) has small proofs of its own consistency for $k > 1$. In contrast we show that Resolution requires almost exponential size to prove its own consistency. As a corollary we get an almost exponential lower bound for the monotone interpolation of Res(2) improving over the quasipolynomial lower bound in [2]. It also shows that Res(2) is almost exponentially stronger than Resolution (recently, [31] proved a truly exponential separation).

These results promote the Res($k$) systems not only as proof systems of independent interest, but also as an important tool for studying the complexity of Resolution. As a matter of fact, our techniques strongly depend on the fact, first noted in this paper, that the proofs in the Res($k$) systems are in a sense translatable into Resolution and conversely. The precise results are stated in Section 4. We believe these techniques, although technically simple, will be useful in future work on the algorithmic aspects of Resolution.

Despite the discouraging results in [4] mentioned before, there is still some effort put into finding good algorithms for the proof search problem of Resolution. The first implementations were variants of the Davis–Putnam procedure [15,16] for testing unsatisfiability which either produce a tree-like Resolution refutation (if one exists), or give a satisfying assignment. For various versions of this algorithm, one can prove that it is not an automatization procedure even for tree-like Resolution. A theoretically better algorithm for finding tree-like Resolution refutations was proposed by Beame and Pitassi [8]. They give an algorithm that works in time quasipolynomial in the size of the smallest tree-like refutation. So tree-like Resolution is automatizable in

quasipolynomial-time, but the algorithm is not a good automatization procedure for general Resolution (see [6,11]). A more efficient algorithm is the one of Ben-Sasson and Wigderson based on the width of a refutation. This algorithm weakly automatizes tree-like Resolution in quasipolynomial-time and automatizes Resolution in subexponential time. On the other hand, Bonet and Galesi [7] gave a class of tautologies for which the algorithm will take exponential time to finish, matching the upper bound. Using the techniques introduced in this paper, we show that this is not an isolated example. We describe a method to produce tautologies that have small Resolution refutations but require relatively large width, answering an open problem of Alekhnovich and Razborov [4]. As they claim, this is a necessary step towards proving that Resolution is not automatizable in quasipolynomial-time.

## 2. Automatizability, interpolation, and reflection principles

We reserve the letter $\Sigma$ possibly with subindices for finite alphabets. Let $TAUT$ be the set of propositional tautologies appropriately encoded in some finite alphabet. The following definitions are taken directly from Cook and Reckhow [14].

**Definition 1.** A propositional proof system is a polynomial-time computable function $f : \Sigma^* \to TAUT$ that is onto. We say that the proof system is polynomially bounded if and only if there is a polynomial $p(n)$ such that for all $y \in TAUT$ there is $x \in \Sigma^*$ such that $y = f(x)$ and $|x| \leqslant p(|y|)$, where $|z|$ denotes the length of a string $z$. If $y = f(x)$, then we will say that $x$ is an $f$-proof of $y$, and $x$ is a short proof of $y$ if in addition $|x| \leqslant p(|y|)$.

Cook and Reckhow pointed out that $TAUT$ is in NP if and only if there exists a polynomially bounded proof system. They also introduced a concept that compares the relative power of proof systems. Here is their definition:

**Definition 2.** Let $f$ and $g$ be propositional proof systems. We say that $g$ simulates $f$ if and only if there is polynomial $p$ such that for all $x$ and $y$ such that $f(x) = y$, there exists $x'$ satisfying $|x'| \leqslant p(|x|)$ and $g(x') = y$.

We turn next to the following more recent definitions. For a fixed proof system $f : \Sigma^* \to TAUT$, consider the problem of finding a proof $x \in \Sigma^*$ of a given $y \in TAUT$. Clearly, this problem is solvable in polynomial-time only if $f$ is polynomially bounded, and in fact, only if $TAUT \in \mathrm{P}$. A weaker requirement possibly of wider relevance might be that of solving the problem in time polynomial in the shortest $f$-proof of $y$. With this new requirement, a proof system might not be polynomially bounded in general, but the problem of finding proofs might be solvable efficiently when short proofs exist. A proof system with this property is called *automatizable*. The concept was introduced by Bonet et al. [10].

**Definition 3.** A proof system $f : \Sigma^* \to TAUT$ is automatizable if and only if there is a deterministic Turing machine $M$ and a polynomial $p$ such that for every $y \in TAUT$ it holds that $f(M(y)) = y$ and $M$ on input $y$ halts in time at most $p(|x|)$ where $x \in \Sigma^*$ is the shortest $f$-proof of $y$.

In some practical situations, the only relevant question is that of finding proofs in *some* proof system, not necessarily $f$ itself. If proofs in another proof system can be found in time polynomial in the shortest $f$-proof, we say that $f$ is *weakly automatizable*.

**Definition 4.** A proof system $f$ is weakly automatizable if and only if there exists another proof system $g$, a deterministic Turing machine $M$, and a polynomial $p$ such that for every $y \in TAUT$ it holds that $g(M(y)) = y$ and $M$ on input $y$ halts in time at most $p(|x|)$ where $x \in \Sigma^*$ is the shortest $f$-proof of $y$. In that case, we say that $f$ is automatizable in terms of $g$.

Observe that if $f$ is automatizable in terms of $g$, then $g$ simulates $f$. After all, it could be that $f$ is not automatizable but some proof system that simulates it is. Finally, we consider several related algorithmic problems with respect to a proof system $f$. The first problem that we consider is a separation problem, namely, that of distinguishing a $y \in TAUT$ with a proof of size at most $m$ from a $y$ that does not even belong to $TAUT$. This problem was introduced by Razborov [30] (see also [28]) under the name of the *separation problem for the canonical pair*.

**Definition 5.** The canonical NP-pair of a proof system $f$ is the following pair of languages:

$$A_0^f = \{\langle y, 1^m\rangle : y \in TAUT \text{ and } f(x) = y \text{ for some } x \text{ such that } |x| \leqslant m\},$$
$$A_1^f = \{\langle y, 1^m\rangle : y \notin TAUT\}.$$

Following Pudlák [28], we say that a pair of disjoint languages $L_0 \cap L_1 = \emptyset$ is polynomially separable if there exists a polynomial-time computable function $h : \Sigma^* \to \{0,1\}$ such that $h(L_0) \subseteq \{0\}$ and $h(L_1) \subseteq \{1\}$. That is, $h(x) = 0$ if $x \in L_0$, $h(x) = 1$ if $x \in L_1$, and $h(x)$ can be 0 or 1 elsewhere.

Observe that when considering refutational systems, such as Resolution, one works with the set of unsatisfiable CNF formulas, instead of $TAUT$. In such a case, the canonical NP-pair consists of the set of pairs $\langle F, 1^m\rangle$ where $F$ is a CNF with refutations of size at most $m$, and the set of pairs $\langle F, 1^m\rangle$ where $F$ is a satisfiable CNF. It is immediately seen that the canonical pair is polynomially separable if and only if the problem WEAK SAT for $f$ defined in the introduction is solvable in time polynomial in $m$ and the size of $F$.

The disjointness of the canonical NP-pair for a refutational proof system $f$ is often expressible as a contradictory set of clauses. Suppose that $SAT_r^n(x,z)$ is a CNF formula meaning that "$z$ encodes a truth assignment that satisfies the CNF formula encoded by $x$." Here, $r$ is the number of clauses of the CNF formula encoded by $x$ and $n$ is the number of underlying variables $v_1, \ldots, v_n$. Similarly, suppose that $REF_{r,m}^n(x,y)$ is a CNF formula meaning that "$y$ encodes an $f$-refutation of the CNF formula encoded by $x$." Here, $m$ is the size of the refutation encoded by $y$, and $r$ and $n$ are as before. Under these two assumptions, the disjointness of the canonical NP-pair for $f$ is expressible by the contradictions

$$SAT_r^n(x,z) \wedge REF_{r,m}^n(x,y).$$

This collection of CNF formulas is known as the *reflection principle* of $f$. Notice that it is a form of consistency of $f$.

We define yet another separation problem, this time it is an NP/co-NP-pair.

**Definition 6.** The canonical NP/co-NP-pair of a proof system $f$ and a polynomial $p$ is the following pair of languages:

$$A_0^{f,p} = \{\langle y, 1^m\rangle : f(x) = y \text{ for some } x \in \Sigma^* \text{ such that } |x| \leqslant m\},$$
$$A_1^{f,p} = \{\langle y, 1^m\rangle : \text{for all } x \in \Sigma^*, f(x) = y \text{ implies } |x| > p(m)\}.$$

Notice that the problem of separating the canonical NP/co-NP-pair of a proof system, is equivalent to the problem of approximating minimal proof size within a polynomial.

We turn next to the concept of feasible interpolation introduced by Krajíček [20] (see also [9,10,25]). Suppose that $A_0(x, y_0) \wedge A_1(x, y_1)$ is a contradictory CNF formula, where $x$, $y_0$, and $y_1$ are disjoint sets of variables. Note that for every given truth assignment $a$ for the variables $x$, one of the formulas $A_0(a, y_0)$ or $A_1(a, y_1)$ must be contradictory by itself.

**Definition 7.** A refutational proof system $f$ has feasible interpolation if there exists a Turing machine $M$ and a polynomial $p$ that, given a refutation $r$ of an unsatisfiable CNF formula $A_0(x, y_0) \wedge A_1(x, y_1)$ and given a truth assignment $a$ for the common variables, $M$ on input $\langle r, a\rangle$ halts in $p(|r|)$ steps and returns $i \in \{0, 1\}$ such that $A_i(a, y_i)$ is unsatisfiable.

We note that this definition is a uniform version of feasible interpolation as defined in [10,20,25]. If in the definition we replace the Turing machine by a Boolean circuit we obtain the non-uniform version. Moreover, this allows us to define the monotone version of feasible interpolation. That is, a proof system has monotone feasible interpolation if the interpolating circuit is monotone in $a$ when either $A_0(x, y_0)$ or $A_1(x, y_1)$ are such that the $x$-variables appear only positively or only negatively.

## 3. Robustness of the definitions

The aim of this section is to show that the definitions of Section 2 lead to a nice theory. Our first result establishes an informative characterization of weak automatizability. We note that the equivalence between (ii) and (iii) was first noted by Pudlák [27]. We reproduce it here for completeness:

**Theorem 1.** *If $f : \Sigma^* \to TAUT$ is a proof system, then the following are equivalent*:
(*i*) *$f$ is weakly automatizable.*
(*ii*) *There is a proof system $g$ that simulates $f$ and is automatizable.*
(*iii*) *The canonical NP-pair of $f$ is polynomially separable.*

**Proof.** We close a cycle of implications (iii) implies (ii), (ii) implies (i), and (i) implies (iii).

(iii) implies (ii) [27]. Suppose that the canonical NP-pair of $f$ is polynomially separated by $h$. Let $\Sigma_1$ be a finite alphabet that is suitable to encode pairs of the form $\langle y, 1^m\rangle$. Let $g : \Sigma_1^* \to TAUT$ be the following proof system. If $w \in \Sigma_1^*$ encodes a pair of the form $\langle y, 1^m\rangle$ and $h(\langle y, 1^m\rangle) = 0$, then $g(w) = y$. Otherwise $g(w) = p \vee \neg p$. We show that $g$ simulates $f$. Indeed, if $x$ is an $f$-proof, then $\langle f(x), 1^{|x|}\rangle$ is a $g$-proof of the same formula. We also show that $g$ is automatizable. The algorithm is as follows: given $y$, find by binary search the minimal $m$ such that $h(\langle y, 1^m\rangle) = 0$, and output $\langle y, 1^m\rangle$. Such an $m$ must exist and the running time is polynomial in the length of $\langle y, 1^m\rangle$.

(ii) implies (i). Suppose that $g: \Sigma_2^* \to TAUT$ is an automatizable proof system that simulates $f$. For every $x \in \Sigma^*$, let $h(x)$ be the shortest $g$-proof of $f(x)$. Let $q$ be a polynomial such that $|h(x)| \leqslant q(|x|)$. Such a polynomial exists because $g$ simulates $f$. Let $M$ and $p$ be the Turing machine and the polynomial that witness the fact that $g$ is automatizable. We may assume that $p$ is a monotone non-decreasing function of its argument. We claim that the same machine $M$ and the polynomial $p(q(n))$ witness the fact that $f$ is automatizable in terms of $g$. To see this, suppose that $y \in TAUT$. Let $x \in \Sigma^*$ be the shortest $f$-proof of $y$ and let $x' = h(x)$. Observe that $|x'| \leqslant q(|x|)$. Moreover, $M$ on input $y$ halts in at most $p(|x'|) \leqslant p(q(|x|))$ steps and its output $x''$ is such that $g(x'') = y$. This shows that $f$ is weakly automatizable.

(i) implies (iii). Suppose that $f$ is weakly automatizable. Let $g : \Sigma_2^* \to L$ be the proof system in terms of which $f$ is automatizable. Let $M$ and $p$ be a Turing machine witnessing it. Let $N$ be the Turing machine that behaves as follows. On input $\langle y, 1^m \rangle$, it simulates $M$ on input $y$ for $p(m)$ steps. If $M$ halted and the output is such that $g(M(y)) = y$, then it outputs 0. In any other case, it outputs 1. We claim that $N$ witnesses the fact that $A_0^f$ and $A_1^f$ are polynomially separable. Clearly, $N$ runs in polynomial-time. Suppose now that $y$ has an $f$-proof of length at most $m$. Then $M$ on input $y$ halts in at most $p(m)$ steps and outputs some $x$ such that $g(x) = y$. If follows that $N$ outputs 0 on $\langle y, 1^m \rangle$. On the other hand, if $y \notin TAUT$, then there is no $x$ for which $g(x) = y$. It follows that $N$ outputs 1 on $\langle y, 1^m \rangle$. □

We turn next to a similar characterization of automatizability in terms of canonical pairs. Recall that two systems are equivalent if they simulate each other.

**Theorem 2.** *If $f : \Sigma^* \to TAUT$ is a proof system, then the following are equivalent*:
(*i*) *There is a proof system equivalent to $f$ that is automatizable.*
(*ii*) *The canonical NP/co-NP-pair of $f$ and some polynomial is polynomially separable.*

**Proof.** We prove both implications (i) implies (ii) and (ii) implies (i).

(i) implies (ii). Let $g$ be the automatizable proof system that is polynomially equivalent to $f$. Let $M$ be the automatization algorithm, and $q$ the time in which $M$ works. Given that $f$ and $g$ are polynomially equivalent, there are two polynomials $p'$ and $p''$, such that if $f(x) = y$, there exists $x'$ satisfying $|x'| \leqslant p'(|x|)$ and $g(x') = y$, and if $g(x) = y$, there exists $x'$ satisfying $|x'| \leqslant p''(|x|)$ and $f(x') = y$. Now we need to choose a polynomial $p$ such that $p''^{(-1)}(p(m)) > q(p'(m))$ (if $p'' = n^c$ then $p''^{(-1)} = n^{1/c}$). Now we will show that the canonical NP/co-NP-pair for the polynomial $p$ and the system $f$ is separable. On input $\langle y, 1^m \rangle$, run $M$ on $y$ for $q(p'(m))$ steps. If the algorithm $M$ returns a proof, return 0, and otherwise 1.

If $y$ has an $f$-proof of size at most $m$, then $y$ has a $g$-proof of size at most $p'(m)$, and the algorithm will find a proof in time $q(p'(m))$. On the other hand, if $y$ does not have an $f$-proof of size at most $p(m)$, then $g$ does not have a proof of $y$ of size at most $p''^{(-1)}(p(m))$. In this case, given that $p''^{(-1)}(p(m)) > q(p'(m))$, the algorithm will not return a proof, and the output will be 1.

(ii) implies (i). Suppose that the canonical NP/co-NP-pair of $f$ for some polynomial $p$ is polynomially separated by the function $h$. We define the propositional proof system $g$ as in part (iii) implies (ii) of the previous theorem. We show there that $g$ is an automatizable propositional proof system that simulates $f$. Now we will also show that $f$ simulates $g$. If $g(x) = y \neq p \vee \neg p$,

by definition $x = \langle y, 1^m \rangle$ and $h(\langle y, 1^m \rangle) = 0$. Then there is $x'$ such that $f(x') = y$ and $|x'| \leqslant p(m)$ (otherwise $h(\langle y, 1^m \rangle)$ would be 1). If $g(x) = p \vee \neg p$, then construct a small $f$-proof of $p \vee \neg p$. □

We close this section by recalling the known relationship between the concepts of feasible interpolation, automatizability, and reflection principles. First, let us define the following mild condition on a proof system $f$. We say $f$ is closed under restrictions if, whenever $A(x)$ has an $f$-proof of size $m$, any restriction of $A(x)$ by a partial truth assignment on $x$ has an $f$-proof of size at most polynomial in $m$. We note that most natural proof systems are closed under restrictions, and that the definition applies as well to refutational systems.

**Theorem 3** ([10]). *Let $f$ be a refutational proof system that is closed under restrictions. If $f$ is automatizable or weakly automatizable, then $f$ has feasible interpolation.*

In fact, in [10], this result is stated and proved for $f$ automatizable. We note that a similar proof works for $f$ weakly automatizable.

The following is a partial converse:

**Theorem 4** ([28]). *If the reflection principle of $f$ has polynomial-size refutations in a proof system that has feasible interpolation, and the refutations are given uniformly in polynomial-time, then $f$ is weakly automatizable.*

## 4. Resolution-based propositional proof systems

Resolution is a refutational proof system for CNF formulas, that is, conjunctions of clauses. The system has one inference rule, the *resolution rule*:

$$\frac{A \vee x \quad B \vee \neg x}{A \vee B},$$

where $x$ is a variable, and $A$ and $B$ are clauses. The refutation finishes with the empty clause. The *size* of a Resolution refutation is the number of clauses in it. The system *tree-like Resolution* requires that each occurence of a clause is used at most once. When this restriction is not fulfilled, we say that the refutation is in *DAG* form.

A *k-term* is a conjunction of up to $k$ literals. A *k-disjunction* is an (unbounded fan-in) disjunction of $k$-terms. The refutation system Res($k$), defined by Krajíček [22], works with $k$-disjunctions. There are three inference rules in Res($k$): Weakening

$$\frac{A}{A \vee B},$$

the rule of $\wedge$-Introduction

$$\frac{A \vee l_1 \quad B \vee (l_2 \wedge \cdots \wedge l_s)}{A \vee B \vee (l_1 \wedge \cdots \wedge l_s)},$$

and the Cut rule

$$\frac{A \vee (l_1 \wedge \cdots \wedge l_s) \qquad B \vee \neg l_1 \vee \cdots \vee \neg l_s}{A \vee B}.$$

Here $A$ and $B$ are $k$-disjunctions, the $l_i$'s are literals, and $s \leqslant k$. We also allow the axioms $x \vee \neg x$. Observe that Res(1) is equivalent to Resolution since the axioms and the weakening rule are easy to eliminate in this case. The size of a Res($k$) refutation is the number of $k$-disjunctions in it. As in Resolution, the tree-like version of Res($k$) requires each occurrence of a $k$-disjunction to be used only once.

For every set of literals $l_1, \ldots, l_s$ we define a new variable $z(l_1, \ldots, l_s)$ meaning $l_1 \wedge \cdots \wedge l_s$. The following clauses define $z(l_1, \ldots, l_s)$:

$$\neg l_1 \vee \cdots \vee \neg l_s \vee z(l_1, \ldots, l_s),$$
$$\neg z(l_1, \ldots, l_s) \vee l_i,$$

where $i \in \{1, \ldots, s\}$ in the second case. Let $F$ be a CNF formula on the variables $x_1, \ldots, x_n$. For every integer $k > 0$, we define $F(k)$ as the conjunction of $F$ with all the defining clauses for the variables $z(l_1, \ldots, l_s)$ for all $s \leqslant k$.

**Lemma 1.** *If $F$ has a Res($k$) refutation of size $S$, then $F(k)$ has a Resolution refutation of size* $O(kS)$. *Furthermore, if the Res($k$) refutation is tree-like, then the Resolution refutation is also tree-like.*

**Proof.** Let $\Pi$ be a Res($k$) refutation of size $S$. To get a Resolution refutation of $F(k)$, we will first get a clause for each $k$-disjunction of $\Pi$. The translation consists in replacing each conjunction $l_1 \wedge \cdots \wedge l_s$ for $s \leqslant k$ in a $k$-disjunction of $\Pi$ by $z(l_1, \ldots, l_s)$. Also we have to make sure that we can make this new sequence of clauses into a Resolution refutation so that if $\Pi$ is tree-like, then the new refutation will also be. We have the following cases:

*Case 1:* In $\Pi$ we have the step:

$$\frac{C \vee (l_1 \wedge \cdots \wedge l_s) \qquad D \vee \neg l_1 \vee \cdots \vee \neg l_s}{C \vee D}.$$

The corresponding clauses in the translation will be: $C' \vee z(l_1, \ldots, l_s)$, $D' \vee \neg l_1 \vee \cdots \vee \neg l_s$ and $C' \vee D'$. To get a tree-like proof of $C' \vee D'$ from the two other ones, first obtain $\neg z(l_1, \ldots, l_s) \vee D'$ in a tree-like way from $D' \vee \neg l_1 \vee \cdots \vee \neg l_s$ and the clauses $\neg z(l_1, \ldots, l_s) \vee l_i$. Finally resolve $\neg z(l_1, \ldots, l_s) \vee D'$ with $C' \vee z(l_1, \ldots, l_s)$ to get $C' \vee D'$.

*Case 2:* In $\Pi$ we have the step:

$$\frac{C \vee l_1 \qquad D \vee (l_2 \wedge \cdots \wedge l_s)}{C \vee D \vee (l_1 \wedge \cdots \wedge l_s)}.$$

The corresponding clauses in the translation will be: $C' \vee l_1$, $D' \vee z(l_2, \ldots, l_s)$ and $C' \vee D' \vee z(l_1, \ldots, l_s)$. Notice that there is a tree-like proof of $\neg l_1 \vee \neg z(l_2, \ldots, l_s) \vee z(l_1, \ldots, l_s)$ from the clauses of $F(k)$. Using this clause and the translation of the premises, we get $C' \vee D' \vee z(l_1, \ldots, l_s)$.

*Case 3:* The Weakening rule turns into a weakening rule for Resolution which can be eliminated easily.

At this point we have obtained a Resolution refutation of $F(k)$ that may use axioms of the type $x \vee \neg x$. These can be eliminated easily too. □

**Lemma 2.** *If $F(k)$ has a Resolution refutation of size $S$, then $F$ has a Res($k$) refutation of size* $O(kS)$. *Furthermore, if the Resolution refutation is tree-like, then the Res($k$) refutation is also tree-like.*

**Proof.** We first substitute each clause of the Resolution refutation by a $k$-disjunction by translating $z(l_1, \ldots, l_s)$ into $l_1 \wedge \cdots \wedge l_s$ and $\neg z(l_1, \ldots, l_s)$ into $\neg l_1 \vee \cdots \vee \neg l_s$. At this point the rules of the Resolution refutation turn into valid rules of Res($k$).

Now we only need to produce proofs of the defining clauses of the $z$ variables in Res($k$) to finish the simulation. The clauses $\neg z(l_1, \ldots, l_s) \vee l_i$ get translated into $\neg l_1 \vee \cdots \vee \neg l_s \vee l_i$, which is a weakening of the axiom $l_i \vee \neg l_i$. The clause $\neg l_1 \vee \cdots \vee \neg l_s \vee z(l_1, \ldots, l_s)$ gets translated into $\neg l_1 \vee \cdots \vee \neg l_s \vee (l_1 \wedge \cdots \wedge l_s)$ which can be proved form the axioms $l_i \vee \neg l_i$ using the rule for the introduction of $\wedge$. □

## 5. Resolution: reflection principle and weak automatizability

In this section we establish the equivalence between: (i) Resolution is weakly automatizable, (ii) Res($k$) is weakly automatizable, and (iii) Res($k$) has feasible interpolation, when $k > 1$. In the course of the proof we need to study the provability of the reflection principle of Resolution.

In what follows we will need a concrete encoding of the reflection principle. We start with the encoding of $SAT_r^n(x, z)$. The encoding of the set of clauses by the variables in $x$ is as follows. There are variables $x_{e,i,j}$ for every $e \in \{0,1\}$, $i \in \{1, \ldots, n\}$, and $j \in \{1, \ldots, r\}$. The meaning of $x_{0,i,j}$ is that the literal $v_i$ appears in clause $j$, while the meaning of $x_{1,i,j}$ is that the literal $\neg v_i$ appears in clause $j$.

The encoding of the truth assignment $a \in \{0,1\}^n$ by the variables $z$ is as follows. There are variables $z_i$ for every $i \in \{1, \ldots, n\}$, and $z_{e,i,j}$ for every $e \in \{0,1\}$, $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, r\}$. The meaning of $z_i$ is that variable $v_i$ is assigned true under the truth assignment. The meaning of $z_{0,i,j}$ is that clause $j$ is satisfied by the literal $v_i$, and the meaning of $z_{1,i,j}$ is that clause $j$ is satisfied by the literal $\neg v_i$. Here is the set of clauses that formalizes $SAT_r^n(x, z)$:

$$z_{0,1,j} \vee z_{1,1,j} \vee \cdots \vee z_{0,n,j} \vee z_{1,n,j}, \tag{1}$$

$$\neg z_{0,i,j} \vee z_i, \tag{2}$$

$$\neg z_{e,i,j} \vee x_{e,i,j}, \tag{3}$$

$$\neg z_{1,i,j} \vee \neg z_i. \tag{4}$$

The meaning of clause (1) is that at least one literal is chosen to be satisfied in clause $j$. The meaning of clauses (2) and (4) ensure the consistency of this choice between clauses. The meaning of clause (3) is that if some literal satisfies clause $j$, then it appears in the clause.

The encoding of $REF_{r,m}^n(x, y)$ is quite standard. The encoding of the set of clauses by the variables in $x$ is as before. The encoding of the Resolution refutation by the variables in $y$ is as follows.

There are variables $y_{e,i,j}$ for every $e \in \{0,1\}$, $i \in \{1,\ldots,n\}$, and $j \in \{1,\ldots,m\}$. The meaning of $y_{0,i,j}$ is that the literal $v_i$ appears in clause $j$ of the refutation. Similarly, the meaning of $y_{1,i,j}$ is that the literal $\neg v_i$ appears in clause $j$ of the refutation. There are variables $p_{j,k}$ and $q_{j,k}$ for every $j \in \{1,\ldots,m\}$ and $k \in \{r,\ldots,m\}$. The meaning of $p_{j,k}$ (of $q_{j',k}$) is that clause $C_k$ is obtained from clause $C_j$ (from clause $C_{j'}$), and $C_j$ contains the resolved variable positively ($C_{j'}$ contains it negatively). Finally, there are variables $w_{i,k}$ for every $i \in \{1,\ldots,n\}$ and $k \in \{r,\ldots,m\}$. The meaning of $w_{i,k}$ is that clause $C_k$ is obtained by resolving upon $v_i$. We formalize this by the following set of clauses:

$$\neg x_{e,i,j} \vee y_{e,i,j}, \tag{5}$$

$$\neg y_{e,i,m}, \tag{6}$$

$$\neg y_{0,i,j} \vee \neg y_{1,i,j}, \tag{7}$$

$$p_{1,k} \vee \cdots \vee p_{k-1,k}, \tag{8}$$

$$q_{1,k} \vee \cdots \vee q_{k-1,k}, \tag{9}$$

$$\neg p_{j,k} \vee \neg q_{j,k}, \tag{10}$$

$$\neg p_{j,k} \vee \neg p_{j',k}, \tag{11}$$

$$\neg q_{j,k} \vee \neg q_{j',k}, \tag{12}$$

$$\neg p_{j,k} \vee \neg w_{i,k} \vee y_{0,i,j}, \tag{13}$$

$$\neg q_{j,k} \vee \neg w_{i,k} \vee y_{1,i,j}, \tag{14}$$

$$\neg p_{j,k} \vee w_{i,k} \vee \neg y_{e,i,j} \vee y_{e,i,k}, \tag{15}$$

$$\neg q_{j,k} \vee w_{i,k} \vee \neg y_{e,i,j} \vee y_{e,i,k}, \tag{16}$$

$$w_{1,k} \vee \cdots \vee w_{n,k}, \tag{17}$$

$$\neg w_{i,k} \vee \neg w_{i',k}. \tag{18}$$

The index $j'$ ranges over $\{1,\ldots,m\}$ with $j' \neq j$, and the index $i'$ ranges over $\{1,\ldots,n\}$ with $i' \neq i$. The meaning of clause (5) is that the initial clauses of the refutation are those of the formula. The meaning of clause (6) is that the last clause of the refutation is empty. The meaning of clause (7) is that the clauses of the refutation are not tautologies. Clauses (8) and (9) indicate that clause $C_k$ is obtained from two previous clauses, one with the resolved variable positive and the other negative. Clauses (10–12) ensure the structure of the resolution rule. Clauses (13) and (14) express that if a clause $C_k$ is obtained from $C_j$ by resolving on variable $v_i$ (or $\neg v_i$), then $v_i$ (or $\neg v_i$) appears in $C_j$. Clauses (15) and (16) express that after applying the rule, the non-resolved literals stay. Finally, Clauses (17) and (18) say that $C_k$ is obtained by resolving exactly one variable.

Notice that this encoding has the appropriate form of the monotone feasible interpolation because the $x$-variables appear only positively in $SAT^n_r(x,z)$ and in fact, only negatively in $REF^n_{r,m}(x,y)$ too. This will be of use later.

**Theorem 5.** *The reflection principle for Resolution expressed by the CNF formula* $SAT^n_r(x,z) \wedge REF^n_{r,m}(x,y)$ *has Res*(2) *refutations of size* $(nr+nm)^{O(1)}$.

**Proof.** The goal is to get the following 2-disjunction

$$D_k \equiv \bigvee_{i=1}^{n} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i)$$

for every $k \in \{1, \ldots, m\}$. The empty clause will follow by resolving $D_m$ with (6). We distinguish two cases: $k \leqslant r$ and $r < k \leqslant m$. Since the case $k \leqslant r$ is easier, we leave it to the reader.

For the case $r < k \leqslant m$, we show how to derive $D_k$ from $D_1, \ldots, D_{k-1}$. First, we derive $\neg p_{j,k} \vee \neg q_{l,k} \vee D_k$. From (7) and (14) we get $\neg q_{l,k} \vee \neg w_{q,k} \vee \neg y_{0,q,l}$. Resolving with $D_l$ on $y_{0,q,l}$ we get

$$\neg q_{l,k} \vee \neg w_{q,k} \vee (y_{1,q,l} \wedge \neg z_q) \vee \bigvee_{\substack{i=1 \\ i \neq q}}^{n} (y_{0,i,l} \wedge z_i) \vee (y_{1,i,l} \wedge \neg z_i). \tag{19}$$

A cut with $z_q \vee \neg z_q$ on $y_{1,q,l} \wedge \neg z_q$ gives

$$\neg q_{l,k} \vee \neg w_{q,k} \vee \neg z_q \vee \bigvee_{\substack{i=1 \\ i \neq q}}^{n} (y_{0,i,l} \wedge z_i) \vee (y_{1,i,l} \wedge \neg z_i). \tag{20}$$

Let $q' \neq q$. We use the notation $H$ for $\bigvee_{i \neq q,q'} (y_{0,i,l} \wedge z_i) \vee (y_{1,i,l} \wedge \neg z_i)$ which will appear quite often. A cut between (20) and $z_{q'} \vee \neg z_{q'}$ on $y_{0,q',l} \wedge z_{q'}$ gives

$$\neg q_{l,k} \vee \neg w_{q,k} \vee \neg z_q \vee z_{q'} \vee (y_{1,q',l} \wedge \neg z_{q'}) \vee H. \tag{21}$$

From (16) and (18) we get $\neg q_{l,k} \vee \neg w_{q,k} \vee \neg y_{0,q',l} \vee y_{0,q',k}$. Resolving with (20) on $y_{0,q',l} \wedge z_{q'}$ gives

$$\neg q_{l,k} \vee \neg w_{q,k} \vee \neg z_q \vee y_{0,q',k} \vee (y_{1,q',l} \wedge \neg z_{q'}) \vee H. \tag{22}$$

An introduction of conjunction between (21) and (22) gives

$$\neg q_{l,k} \vee \neg w_{q,k} \vee \neg z_q \vee (y_{0,q',k} \wedge z_{q'}) \vee (y_{1,q',l} \wedge \neg z_{q'}) \vee H. \tag{23}$$

From (16) and (18) we also get $\neg q_{l,k} \vee \neg w_{q,k} \vee \neg y_{1,q',l} \vee y_{1,q',k}$. Repeating the same procedure we get

$$\neg q_{l,k} \vee \neg w_{q,k} \vee \neg z_q \vee (y_{0,q',k} \wedge z_{q'}) \vee (y_{1,q',k} \wedge \neg z_{q'}) \vee H. \tag{24}$$

Now, repeating this two-step procedure for every $q' \neq q$, we get

$$\neg q_{l,k} \vee \neg w_{q,k} \vee \neg z_q \vee \bigvee_{i \neq q} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i). \tag{25}$$

A dual argument yould yield $\neg p_{j,k} \vee \neg w_{q,k} \vee z_q \vee \bigvee_{i \neq q}(y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i)$. A cut with (25) on $z_q$ gives $\neg p_{j,k} \vee \neg q_{l,k} \vee \neg w_{q,k} \vee \bigvee_{i \neq q}(y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i)$. Weakening gives then $\neg p_{j,k} \vee \neg q_{l,k} \vee \neg w_{q,k} \vee D_k$. Since the indices $j$, $l$, and $q$ were arbitrary, we obtain the same formula for every such triple. Using (17) gives $\neg p_{j,k} \vee \neg q_{l,k} \vee D_k$, again for every pair of indices $j$ and $l$. To complete the proof, we use this with (8) to get $\neg q_{l,k} \vee D_k$ for every $l$, and then (9) to get $D_k$.

For the size of the refutation, we consider both cases $k \leqslant r$ and $k > r$. There are $r$ initial clauses and each corresponding $D_k$ has size $O(n)$. Deriving all of these takes size $(nr)^{O(1)}$. The rest of $D_k$'s also have size $O(n)$ and there are $m - r$ of these. Deriving $D_{k+1}$ from $D_1, \ldots, D_k$ takes size $(nm)^{O(1)}$ because $C_{k+1}$ can be derived from any pair of previous clauses. Overall, this is size $(nr + nm)^{O(1)}$. □

The previous theorem can be generalized to reflection principles for Res($k$). This requires a modification of $REF^n_{r,m}(x, y)$ to refutations in Res($k$). For Resolution, the refutation was encoded by the variables $y_{e,i,j}$ meaning that the variable $v_i$ appears in clause $j$ positively or negatively. Now we need variables $y_{e_1,i_1,\ldots,e_l,i_l,j}$ for $l \in \{1, \ldots, k\}$, $e_1, \ldots, e_l \in \{0, 1\}$, $i_1, \ldots, i_l \in \{1, \ldots, n\}$, and $j \in \{1, \ldots, m\}$. Their meaning is that the $k$-disjunction $C_j$ contains the term $v_{i_1}^{(e_1)} \wedge \cdots \wedge v_{i_l}^{(e_l)}$, where $v_i^{(0)}$ is $v_i$ and $v_i^{(1)}$ is $\neg v_i$. The structure of the refutation is encoded by variables $t$, $p$, $q$, and $w$. The variables $t_{1,j}, \ldots, t_{4,j}$ for $j \in \{1, \ldots, m\}$ specify whether $C_j$ is obtained by weakening, introduction of conjunction, cut, or is an axiom, respectively. The variables $p_{j,j'}$ ($q_{j,j'}$) for $j, j' \in \{1, \ldots, m\}$ indicate that the left (right) hypothesis of $C_{j'}$ is $C_j$. The variables $w_{e_1,i_1,\ldots,e_l,i_l,j}$ have different meanings depending on the rule. If $C_j$ is obtained by cut, it means that the resolved term is $v_{i_1}^{(e_1)} \wedge \cdots \wedge v_{i_l}^{(e_l)}$. If $C_j$ is obtained by introduction of conjunction, it means that the introduced conjunction is $v_{i_1}^{(e_1)} \wedge \cdots \wedge v_{i_l}^{(e_l)}$. Notice that the rule has a fixed format, namely, the first literal is in the left hypothesis, and the conjunction of the rest is in the right hypothesis. If $C_j$ is an axiom, it specifies which variable is in the axiom and $l = 1$ in this case. If $C_j$ is obtained by weakening, the values of $w$ are irrelevant. Writing the set of clauses expressing this is straightforward. We only give an example of one of the new clauses. The clause

$$\neg t_{2,j'} \vee \neg p_{j,j'} \vee \neg w_{e_1,i_1,\ldots,e_l,i_l,j'} \vee y_{e_1,i_1,j}$$

means that if $C_{j'}$ is obtained by introduction of conjunction with $C_j$ as left hypothesis, and the introduced conjunction is $v_{i_1}^{(e_1)} \wedge \ldots \wedge v_{i_l}^{(e_l)}$, then the literal $v_{i_1}^{(e_1)}$ appears in $C_j$.

The resulting set of clauses is called $REF^{n,k}_{r,m}(x, y)$. Notice the new superindex $k$ in $REF$ to indicate that the refutation is in Res($k$).

**Theorem 6.** *The reflection principle for Res($k$) expressed by the CNF formula $SAT^n_r(x, z) \wedge REF^{n,k}_{r,m}(x, y)$ has Res($k + 1$) refutations of size $(nr + kn^k m)^{O(1)}$.*

**Proof sketch.** It suffices to prove, for every $j \in \{1, \ldots, m\}$, the following $(k + 1)$-disjunction

$$D_j \equiv \bigvee \left( y_{e_1,i_1,\ldots,e_l,i_l,j} \wedge z_{i_1}^{(e_1)} \wedge \cdots \wedge z_{i_l}^{(e_l)} \right),$$

where the disjunction ranges over $l \in \{1, \ldots, k\}$, $e_1, \ldots, e_l \in \{0, 1\}$, $i_1, \ldots, i_l \in \{1, \ldots, n\}$. The meaning of $D_j$ is that at least one of the $k$-terms that appear in $C_j$ is satisfied by the truth assignment $z_1, \ldots, z_n$. □

We want to point out that a small change in the encoding of the reflection principle for Res($k$) preserves the form of the feasible interpolation and has polynomial-size proofs in Res(2). This change consists in adding new $z$-variables and new clauses to $SAT_r^n(x,z)$. These variables will encode the conjunctions of up to $k$ literals on $z_1,\ldots,z_n$. More precisely, the new variables are $z(l_1,\ldots,l_s)$ for $s \leqslant k$, where $l_i$ is a literal on $z_1,\ldots,z_n$, as defined in Section 4. The new clauses are those exposed in Section 4 defining $z(l_1,\ldots,l_s)$. The resulting formula is called $SAT_r^{n,k}(x,z)$. Again notice the new superindex $k$ in $SAT$.

**Theorem 7.** *The reflection principle for Res($k$) expressed by the CNF formula $SAT_r^{n,k}(x,z) \wedge REF_{r,m}^{n,k}(x,y)$ has Res(2) refutations of size $(n^k r + kn^k m)^{O(1)}$. Moreover, $SAT_r^{n,k}(x,z) \wedge REF_{r,m}^{n,k}(x,y)$ preserves the form of monotone feasible interpolation.*

**Proof sketch.** Again, it suffices to prove, for every $j \in \{1,\ldots,m\}$, the following 2-disjunction

$$D_j \equiv \bigvee \left( y_{e_1,i_1,\ldots,e_l,i_l,j} \wedge z\left(z_{i_1}^{(e_1)},\ldots,z_{i_l}^{(e_l)}\right)\right),$$

where the disjunction ranges over $l \in \{1,\ldots,k\}$, $e_1,\ldots,e_l \in \{0,1\}$, $i_1,\ldots,i_l \in \{1,\ldots,n\}$. □

Let us note that the technique of the previous two theorems cannot be pushed to show that Resolution proves the reflection principle of Res($k$) while preserving the form of feasible interpolation. This is because the formula $D_j$ in the proof has conjunctions that involve $y$- and $z$-variables, which prevents us from defining their conjunction without mixing them.

Our next result shows the exact relationship between feasible interpolation and weak automatizability for Res($k$). In its proof, we will use what we have learned about reflection principles.

**Theorem 8.** *For every constant $k \geqslant 2$, the following are equivalent*:

(*i*) *Resolution is weakly automatizable.*
(*ii*) *Res($k$) is weakly automatizable.*
(*iii*) *Res($k$) has feasible interpolation.*

**Proof.** (ii) implies (iii) is simply Theorem 3. (iii) implies (i) is a consequence of Theorems 4 and 5 because $k \geqslant 2$. We prove (i) implies (ii). Suppose Resolution is weakly automatizable. Then by Theorem 1, the canonical NP-pair of Resolution is polynomially separable. We claim that the canonical NP-pair of Res($k$) is also polynomially separable. Here is the separation algorithm: given $F$ and a number $m$ in unary, we build $F(k)$ and run the separation algorithm for the canonical pair of Resolution on $F(k)$ and $c \cdot k \cdot m$ in unary, where $c$ is the hidden constant in Lemma 1. The running time of this algorithm is polynomial in the size of $F(k)$ and $m$, which is polynomial in the size of $F$ and $m$ because $k$ is a constant. For the correctness, note that if $F$ has a Res($k$) refutation of size $m$, then $F(k)$ has a Resolution refutation of size at most $c \cdot k \cdot m$ by Lemma 1, and the separation algorithm for the canonical pair of Resolution will return 0 on it. On the other hand, if $F$ is satisfiable, so is $F(k)$ and the separation algorithm for Resolution will return 1 on it. □

It is known that Resolution has feasible interpolation, and in fact monotone [20,25]. However, it is not clear whether Res($k$) has feasible interpolation for $k \geqslant 2$. We know, however, that Res($k$)

does not have monotone feasible interpolation (see [2] and Corollary 1 in this paper). On the other hand, tree-like Res($k$) has feasible interpolation, even monotone, since Resolution polynomially simulates it [17].

If $k$ is no longer a constant, say $k = \log n$, the weak automatizability of Resolution would imply the feasible interpolation of Res($k$) in time $n^{O(k)}$. This is because computing $F(k)$ requires time $n^{O(k)}$. Therefore, in order to establish that Resolution is not weakly automatizable it would suffice to prove that Res($\log n$) does not have feasible interpolation in time $n^{O(\log n)}$.

## 6. Resolution: lower bound for the reflection principle

The next natural question to ask is whether the reflection principle for Resolution $SAT_r^n(x,z) \wedge REF_{r,m}^n(x,y)$ has refutations of moderate size in Resolution itself. Since Resolution has feasible interpolation [20,25], a positive answer would imply that Resolution is weakly automatizable. Unfortunately, as the main result of this section shows, this will not happen. Before stating it we need two technical well-known results.

The first is a result due to Alon and Boppana. Let $F(m,k,k')$ be the set of monotone functions that separate $k$-cliques from $k'$-colorings on $m$ nodes.

**Theorem 9** ([1]). *If $f \in F(m,k,k')$ where $3 \leqslant k' \leqslant k$ and $k\sqrt{k'} \leqslant m/(8\log m)$, then*

$$S^+(f) \geqslant \frac{1}{8}\left(\frac{m}{4k\sqrt{k'}\log m}\right)^{(\sqrt{k'}+1)/2},$$

*where $S^+(f)$ is the monotone circuit size of $f$.*

The second result that we need is due to Maciel, Pitassi, and Woods. Let $PHP_n^m$ be the set of clauses encoding the pigeonhole principle with $m$ pigeons and $n$ holes. That is, $PHP_n^m$ is

$$p_{i,1} \vee \cdots \vee p_{i,n}, \tag{26}$$

$$\neg p_{i,k} \vee \neg p_{j,k}, \tag{27}$$

where $i,j \in \{1,\ldots,m\}$, $i \neq j$, and $k \in \{1,\ldots,n\}$.

**Theorem 10** ([23]). *$PHP_n^{2n}$ has Res$((\log n)^{O(1)})$ refutations of size $2^{(\log n)^{O(1)}}$.*

In addition to these two results, we also need the fact that Resolution has monotone feasible interpolation as shown in [21,25]. Recall the definition of monotone feasible interpolation in Section 2 just after Definition 7. We are now ready to state and prove the main result of this section. Its proof uses an idea due to Pudlák [28].

**Theorem 11.** *Let $s$ be a parameter. For a choice of $n$, $r$, and $m$ of the order of a quasipolynomial $2^{(\log s)^{O(1)}}$, every Resolution refutation of $SAT_r^n(x,z) \wedge REF_{r,m}^n(x,y)$ requires size at least $2^{\Omega(s^{1/4})}$.*

**Proof.** Suppose for contradiction that there is a Resolution refutation of size $S = 2^{O(s^{1/4})}$. Let $k = s^{1/2}$, and let $COL_k(p,q)$ be the CNF formula expressing that $q$ encodes a $k$-coloring of the graph on $s$ nodes encoded by $\{p_{ij}\}$. An explicit definition is the following: for every $i \in \{1,\ldots,s\}$, there is a clause of the form $\bigvee_{l=1}^{k} q_{il}$; and for every $i,j \in \{1,\ldots,s\}$ with $i \neq j$ and $l \in \{1,\ldots,k\}$, there is a clause of the form $\neg q_{il} \vee \neg q_{jl} \vee \neg p_{ij}$. Let $COL_k(G,q)$ be the result of replacing the $p_{ij}$-variables in $COL_k(p,q)$ by the values 1/0 depending on whether the edge $\{i,j\}$ is or is not in $G$. Obviously, if $G$ is $k$-colorable, then $COL_k(G,q)$ is satisfiable, and if $G$ contains a $2k$-clique, then $COL_k(G,q)$ is unsatisfiable. More importantly, if $G$ contains a $2k$-clique, then the clauses of $PHP_k^{2k}$ are contained in $COL_k(G,q)$. Now, for every graph $G$ on $s$ nodes, let $\mathcal{F}(G)$ be the clauses $COL_k(G,q)$ together with all clauses defining the extension variables for the conjunctions of up to $(\log k)^c$ literals on the $q$-variables. Here, $c$ is a constant so that the $2^{(\log k)^{O(1)}}$ upper bound on $PHP_k^{2k}$ of Theorem 10 can be done in $\mathrm{Res}((\log k)^c)$. From Theorem 10 and Lemma 1, if $G$ contains a $2k$-clique, then $\mathcal{F}(G)$ has a Resolution refutation of size $2^{(\log k)^{O(1)}}$.

In the following, let $n$ be the number of variables of $\mathcal{F}(G)$, let $r$ be the number of clauses of $\mathcal{F}(G)$, and let $m = 2^{(\log k)^{O(1)}}$. Consider the formulas $SAT_r^n(x(G),z) \wedge REF_{r,m}^n(x(G),y)$ where $x(G)$ is the $x$-encoding of the formula $\mathcal{F}(G)$. By assumption, these formulas have Resolution refutations of size at most $S$. Let $C$ be the monotone circuit that interpolates these formulas given by the monotone feasible interpolation of Resolution. The size of $C$ is $S^{O(1)}$. Moreover, if $G$ is $k$-colorable, then $SAT_r^n(x(G),z)$ is satisfiable, and $C$ must return 0 on $x(G)$. Also, if $G$ contains a $2k$-clique, then $REF_{r,m}^n(x(G),y)$ is satisfiable, and $C$ must return 1 on $x(G)$. Now, an anti-monotone circuit for separating $2k$-cliques from $k$-colorings can be built as follows: given a graph $G$, build the assignment $x(G)$ (anti-monotonically, see below for details), and apply the monotone circuit given by the monotone interpolation. The size of this circuit is $2^{o(s^{1/4})}$, and this contradicts Theorem 9.

It remains to show how to build an anti-monotone circuit that, on input $G = \{p_{uv}\}$, produces $x(G)$, that is, the values of $x_{e,i,j}$ that correspond to the encoding of $\mathcal{F}(G)$ in terms of the $x$-variables. There are three types of clauses in $\mathcal{F}(G)$:

(1) Clauses of the type $\bigvee_{l=1}^{k} q_{il}$: Let $t$ be the numbering of this clause in $\mathcal{F}(G)$. Then, its encoding in terms of the $x$-variables is produced by outputting the constant 1 for $x_{0,q_{i1},t},\ldots,x_{0,q_{ik},t}$. The rest of the outputs of clause $t$ are 0.

(2) Clauses of the type $\neg q_{il} \vee \neg q_{jl} \vee \neg p_{ij}$: let $t$ be the numbering of this clause in $\mathcal{F}(G)$. The encoding is $x_{1,q_{il},t} = 1$, $x_{1,q_{jl},t} = 1$, $x_{1,p_{ij},t} = \neg p_{ij}$ and the rest are zero. Notice that this encoding is anti-monotone in the $p_{ij}$'s. Notice also that the encoded $\mathcal{F}(G)$ contains some $p$-variables (and not only $q$-variables as the reader might have expected) but this will not be a problem since the main properties of $\mathcal{F}(G)$ are preserved as we show below.

(3) Finally, the clauses defining the conjunctions of up to $(\log k)^c$ literals are independent of $G$ since only the $q$-variables are relevant here. Therefore, the encoding is done as in the first case.

The reader can easily verify that when $G$ contains a $2k$-clique, the encoded formula contains the clauses of $PHP_k^{2k}$ and the definitions of the conjunctions up to $(\log k)^c$ literals. Therefore $REF(x(G),y)$ is satisfiable because $PHP_k^{2k}$ has a small $\mathrm{Res}((\log k)^c)$ refutation. Similarly, if $G$ is $k$-colorable, the formula $SAT(x(G),z)$ is satisfiable by setting $z_{p_{ij}} = p_{ij}$ and $q_{il} = 1$ if and only if node $i$ gets color $l$. Therefore, the main properties of $\mathcal{F}(G)$ are preserved, and the theorem follows. □

An immediate corollary of the last two results is that Res(2) is exponentially more powerful than Resolution. In fact, the proof shows a lower bound for the monotone interpolation of Res(2) improving over the quasipolynomial lower bound in [2].

**Corollary 1.** *Monotone circuits that interpolate Res*(2) *refutations require size* $2^{\Omega(s^{1/4})}$ *on Res*(2) *refutations of size* $2^{(\log s)^{O(1)}}$.

## 7. Short proofs that require large width

The width of a Resolution refutation is the number of literals of the largest clause. Ben-Sasson and Wigderson [12] proved that the following relationship holds. Let $F$ be a $k$-CNF formula with $n$ variables and a Resolution refutation of size $S$. Then $F$ has a Resolution refutation of width at most $\sqrt{n \log(S)} + k$. This suggests the following proof search procedure for Resolution: derive all possible clauses using derived clauses of increasing width $1, 2, 3, \ldots$ until the empty clause is found. The running time of this algorithm is bounded by $n^{O(\sqrt{n \log(S)}+k)}$, which is subexponential if $S$ is polynomial in $n$, say.

Bonet and Galesi [7] gave an example of a 3-CNF formula with small Resolution refutations that requires relatively large width (square root of the number of variables). This showed that the size-width trade-off of Ben-Sasson and Wigderson could not be improved. Also it showed that the algorithm of Ben-Sasson and Wigderson for finding Resolution refutations could perform very badly in the worst case. This is because their example requires large width, and the algorithm would take almost exponential time, while we know that there is a polynomial size Resolution refutation.

Alekhnovich and Razborov [4] posed the question of whether more of these examples could be found. They say this is a necessary first step for showing that Resolution is not automatizable in quasipolynomial-time. Here we give a way of producing such bad examples for the algorithm. Basically the idea is finding CNFs that require sufficiently high width in Resolution, but that have polynomial size Res($k$) refutations for small $k$, say $k \leqslant \log n$. Then the example consists in adding to the formula the clauses defining the extension variables for all the conjunctions of at most $k$ literals. Below we illustrate this technique by giving a large class of examples that have small Resolution refutations but that require large width. Moreover, deciding whether a formula is in the class is hard (no polynomial-time algorithm is known).

Let $G = (U \cup V, E)$ be a bipartite graph on the sets $U$ and $V$ of cardinality $m$ and $n$, respectively, where $m > n$. The *G-PHP*$_n^m$, defined by Ben-Sasson and Wigderson [12], states that there is no matching from $U$ into $V$. For every edge $(u, v) \in E$, let $x_{u,v}$ be a propositional variable meaning that $u$ is mapped to $v$. The principle is then formalized as the conjunction of the following clauses:

$$x_{u,v_1} \vee \cdots \vee x_{u,v_r} \quad \text{for } u \in U,\ N_G(u) = \{v_1, \ldots, v_r\},$$
$$\neg x_{u,v} \vee \neg x_{u',v} \quad \text{for } v \in V,\ u, u' \in N_G(v),\ u \neq u'.$$

Here, $N_G(w)$ denotes the set of neighbors of $w$ in $G$. Note that if $G$ has left-degree at most $d$, then the width of the initial clauses is bounded by $d$.

Ben-Sasson and Wigderson proved that whenever $G$ is expanding in a sense defined next, every Resolution refutation of *G-PHP*$_n^m$ must contain a clause with many literals. We observe that this

result is not unique to Resolution and holds in a more general setting. Before we state the precise result, let us recall the definition of expansion:

**Definition 8** ([12]). Let $G = (U \cup V, E)$ be a bipartite graph where $|U| = m$, and $|V| = n$. For $U' \subset U$, the *boundary* of $U'$, denoted by $\partial U'$, is the set of vertices in $V$ that have exactly one neighbor in $U'$; that is, $\partial U' = \{v \in V : |N(v) \cap U'| = 1\}$. We say that $G$ is $(m, n, r, f)$-*expanding* if every subset $U' \subseteq U$ of size at most $r$ is such that $|\partial U'| \geqslant f \cdot |U'|$.

The proof of the following statement is the same as in [12] for Resolution.

**Theorem 12** ([12]). *Let $P$ be a sound refutation system with all rules having fan-in at most two. Then, if $G$ is $(m, n, r, f)$-expanding, every $P$-refutation of G-$PHP_n^m$ must contain a formula that involves at least $rf/2$ distinct literals.*

Now, for every bipartite graph $G$ with $m \geqslant 2n$, let $\mathcal{C}(G)$ be the set of clauses defining G-$PHP_n^m$ together with the clauses defining all the conjunctions up to $(\log n)^c$ literals, where $c$ is a large constant so that the $2^{(\log n)^{O(1)}}$ upper bound on $PHP_n^{2n}$ of Theorem 10 can be done in $\mathrm{Res}((\log n)^c)$.

**Theorem 13.** *Let $G$ be an $(m, n, \Omega(n/\log m), \frac{3}{4}\log m)$-expander with $m \geqslant 2n$ and left-degree at most $\log m$. Then (i) $\mathcal{C}(G)$ has initial width $\log m$, (ii) any Resolution refutation of $\mathcal{C}(G)$ requires width at least $\Omega(n/(\log n)^c)$, and (iii) $\mathcal{C}(G)$ has Resolution refutations of size $2^{(\log n)^{O(1)}}$.*

**Proof.** Part (i) is obvious. For (ii), suppose for contradiction that $\mathcal{C}(G)$ has a Resolution refutation of width $w = o(n/(\log n)^c)$. Then, by the proof of Lemma 2, G-$PHP_n^m$ has a $\mathrm{Res}((\log n)^c)$ refutation in which every $(\log n)^c$-disjunction involves at most $(\log n)^c w = o(n)$ literals. This contradicts Theorem 12. For (iii), recall that $PHP_n^m$ has a $\mathrm{Res}((\log n)^c)$ refutation of size $2^{(\log n)^{O(1)}}$ by Theorem 10 since $m \geqslant 2n$. Now, setting to zero the appropriate variables of $PHP_n^m$, we get a $\mathrm{Res}((\log n)^c)$ refutation of G-$PHP_n^m$ of the same size. By Lemma 1, $\mathcal{C}(G)$ has a Resolution refutation of roughly the same size, which is polynomial in the size of the formula. □

## 8. Conclusions and open problems

It is surprising that the weak pigeonhole principle $PHP_n^{2n}$ has short Resolution proofs when encoded with the clauses defining the extension variables for the small conjunctions. We exploited this fact in the proof of Theorem 11. The astute reader will notice that any small refutation in *any* Frege system could be translated into a short Resolution refutation when the CNF is enriched with clauses defining new variables for each of the intermediate formulas that are used in the Frege proof. However, the crucial difference of this approach is that the new encoding is not *canonical*: there is no uniform method to obtain it. As a matter of fact, although the pigeonhole principle has polynomial-size proofs in Frege, this approach would not allow us to improve the non-monotone interpolation to a truly exponential lower bound because the $2k$-clique of the graph is unknown to us beforehand.

We showed that the connection between weak automatizability of Resolution and interpolation of Res(2) extends to Res($k$) for any $k \geqslant 2$. As a matter of fact, in order to prove that Resolution is not weakly automatizable, it suffices to show, for example, that Res($\log n$) does not have interpolation in subexponential-time. This leaves some hope of proving such a result with current techniques because Res($\log n$) is a fairly strong system (e.g., it proves the weak pigeonhole principle in quasi-polynomial-size). However, let us note that the non-interpolation for bounded-depth Frege of [5] assuming that numbers cannot be factored in subexponential-time does not seem to establish the non-interpolation for any particular constant depth. Notice that Res($\log n$) is a depth-two system.

Of course, it remains open whether Resolution is weakly automatizable, or automatizable in quasipolynomial-time. Let us note that the width method for Resolution leads to some non-obvious results along these lines. For example, the width-based algorithm of Ben-Sasson and Wigderson solves WEAK SAT for tree-like Resolution in time $n^{O(\log m)}|F|^{O(1)}$, where $n$ is the number of variables of $F$. This is quasipolynomial if $m$ is polynomial in $n$. Similarly, it allows us to solve WEAK SAT for general Resolution in time $n^{O(\sqrt{n \log m})}|F|^{O(1)}$. This is subexponential if $m$ is polynomial. The negative results of [4] do not rule out further improvements on these. Also, it would be interesting to get results of this type for other propositional proof systems.

## Acknowledgment

We are grateful to Pavel Pudlák for stimulating discussions on the idea of Theorem 11.

## References

[1] N. Alon, R.B. Boppana, The monotone circuit complexity of boolean functions, Combinatorica 7 (1987) 1–22.
[2] A. Atserias, M.L. Bonet, J.L. Esteban, Lower bounds for teh weak pigeonhole principle and random formulas beyond resoultion, Information and Computaion 176 (2) (2002) 136–152. (A preliminary version appeared in ICALP'01.)
[3] M. Alekhnovich, S. Buss, S. Moran, T. Pitassi, Minimum propositional proof length is NP-hard to linearly approximate, Journal of Symbolic Logic 66 (2001) 171–191.
[4] M. Alekhnovich, A.A. Razborov, Resolution is not automatizable unless W[P] is tractable, in: 42nd Annual IEEE Symposium on Foundations of Computer Science, 2001, pp. 210–219.
[5] M.L. Bonet, C. Domingo, R. Gavaldà, A. Maciel, T.Pitassi, Non-automatizability of bounded-depth Frege proofs, in: 14th IEEE Conference on Computational Complexity, pp. 15–23, 1999. Journal of Computational Complexity (accepted).
[6] M.L. Bonet, J.L. Esteban, N. Galesi, J. Johansen, On the relative complexity of resolution refinements and cutting planes proof systems, SIAM Journal of Computing 30 (5) (2000) 1462–1484. (A preliminary version appeared in FOCS'98.)
[7] M.L. Bonet, N. Galesi, Optimality of size-width trade-offs for resolution, Computational Complexity 10 (4) (2001) 261–276. (A preliminary version appeared in FOCS'99.)
[8] P. Beame, T. Pitassi, Simplified and improved resolution lower bounds, in: 37th Annual IEEE Symposium on Foundations of Computer Science, 1996, pp. 274–282.
[9] M.L. Bonet, T. Pitassi, R. Raz, Lower bounds for cutting planes proofs with small coefficients, Journal of Symbolic Logic 62 (3) (1997) 708–728. (A preliminary version appeared in STOC'95.)
[10] M.L. Bonet, T. Pitassi, R. Raz, On interpolation and automatization for Frege systems, SIAM Journal of Computing 29 (6) (2000) 1939–1967. (A preliminary version appeared in FOCS'97.)

[11] E. Ben-Sasson, R. Impagliazzo, A. Wigderson, Near-optimal separation of general and tree-like resolution. Combinatorica, 2002 (to appear).
[12] E. Ben-Sasson, A. Wigderson, Short proofs are narrow-resolution made simple, Journal of the ACM 48 (2) (2001) 149–169.
[13] S.A. Cook, A. Haken, An exponential lower bound for the size of monotone real circuits, Journal of Computer and System Sciences 58 (1999) 326–335.
[14] S. Cook, R. Reckhow, The relative efficiency of propositional proof systems, Journal of Symbolic Logic 44 (1979) 36–50.
[15] M. Davis, G. Logemann, D. Loveland, A machine program for theorem proving, Communications of the ACM 5 (1962) 394–397.
[16] M. Davis, H. Putnam, A computing procedure for quantification theory, Journal of the ACM 7 (1960) 201–215.
[17] J.L. Esteban, N. Galesi, J. Messner, On the complexity of resolution with bounded conjunctions, in: 29th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science, vol. 2380, Springer-Verlag, 2002, pp. 220–231.
[18] R. Impagliazzo, T. Pitassi, A. Urquhart, Upper and lower bounds for tree-like cutting planes proofs, in: 9th IEEE Symposium on Logic in Computer Science, 1994, pp. 220–228.
[19] J. Krajícek, P. Pudlák, Some consequences of cryptographical conjectures for $S_2^1$ and $EF$, Information and Computation 140 (1) (1998) 82–94.
[20] J. Krajícek, Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic, Journal of Symbolic Logic 62 (1997) 457–486.
[21] J. Krajícek, On methods for proving lower bounds in propositional logic, in: Logic and Scientific Methods, Kluwer Academic Publishers, 1997, pp. 69–83.
[22] J. Krajícek, On the weak pigeonhole principle, Fundamenta Mathematicæ 170 (1–3) (2001) 123–140.
[23] A. Maciel, T. Pitassi, A.R. Woods, A new proof of the weak pigeonhole principle, Journal of Computer and Systems Science 64 (2002) 843–872.
[24] P. Pudlák, J. Sgall, Algebraic models of computation and interpolation for algebraic proof systems, in: P.W. Beame, S.R. Buss (Eds.), Proof Complexity and Feasible Arithmetic, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 39, American Mathematical Society, 1998, pp. 279–296.
[25] P. Pudlák, Lower bounds for resolution and cutting plane proofs and monotone computations, Journal of Symbolic Logic 62 (3) (1997) 981–998.
[26] P. Pudlák, On the complexity of the propositional calculus, in: Sets and Proofs, Invited Papers from Logic Colloquium '97. Cambridge University Press, 1999, pp. 197–218.
[27] P. Pudlák, On reducibility and symmetry of disjoint NP-pairs, in: 26th International Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, Springer-Verlag, 2001, pp. 621–632.
[28] P. Pudlák, On reducibility and symmetry of disjoint NP-pairs, Theoretical Computer Science 295 (2003) 323–339.
[29] A.A. Razborov, On provably disjoint NP-pairs, Technical Report RS-94-36, BRICS, 1994.
[30] A.A. Razborov, Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic, Izvestiya of the RAN 59 (1) (1995) 205–227.
[31] N. Segerlind, S. Buss, R. Impagliazzo, A switching lemma for small restrictions and lower bounds for $k$-DNF resolution, in: 43rd Annual IEEE Symposium on Foundations of Computer Scienc, 2002, pp. 604–614.