On the Automatizability of Resolution and Related Propositional Proof Systems*

Albert Atserias and María Luisa Bonet

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Barcelona
C/Jordi Girona Salgado, 1-3, Edif. C6.
08034 Barcelona - Spain.
Tel: +34 93 401 69 94
Fax: +34 93 401 70 14
{atserias,bonet}@lsi.upc.es

Abstract. We analyse the possibility that a system that simulates Resolution is automatizable. We call this notion "weak automatizability". We prove that Resolution is weakly automatizable if and only if Res(2) has feasible interpolation. In order to prove this theorem, we show that Res(2) has polynomial-size proofs of the reflection principle of Resolution (and of any Res(k)), which is a version of consistency. We also show that Resolution proofs of its own reflection principle require slightly subexponential size. This gives a better lower bound for the monotone interpolation of Res(2) and a better separation from Resolution as a byproduct. Finally, the techniques for proving these results give us a new complexity measure for Resolution that refines the width of Ben-Sasson and Wigderson. The new measure and techniques suggest a new algorithm to find Resolution refutations, and 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.

* Partially supported by CICYT TIC2001-1577-C03-02, ALCOM-FT IST-99-14186 and HA2000-41.

1 Introduction

In several areas of Computer Science there has been important efforts in studying algorithms for satisfiability, despite the problem is NP-complete, and also in studying the complementary problem of verifying tautologies.

By the theorem of Cook and Reckhow [14], there is strong evidence that for every propositional proof system there is a class of tautologies whose shortest proofs are super-polynomial in the size of the tautologies. From this we conclude that given a propositional proof system S, there will not be an algorithm that will produce S-proofs of a tautology in time polynomial in the size of the tautology. This is because in some cases we might require exponential time just to write down the proof. Considering this limitation of proof systems, Bonet, Pitassi and Raz [12] proposed the following definition. A propositional proof system S is *automatizable* if there exists an algorithm that, given a tautology, it produces an S-proof of it in time polynomial in the size of the smallest S-proof of the tautology. The idea behind this definition is that if short S-proofs exist, an automatization algorithm for S should find them quickly. In the sequel of papers [24, 13, 9] it was proved that no proof system that simulates AC^0-Frege is automatizable, unless some widely accepted cryptographic conjecture is violated. Later, Alekhnovich and Razborov [1] proved that Resolution is not automatizable under a reasonable assumption in parameterized complexity. The drawback of this result is that it is weaker than the others in the sense that we do not know whether a system that simulates Resolution can be automatizable. This problem suggests the following definition. We say that a proof system S is *weakly automatizable* if there is a proof system that polynomially simulates S and is automatizable. At this point it is still open whether Resolution is weakly automatizable.

In this paper we characterize the question of whether Resolution is weakly automatizable as whether the extension of Resolution Res(2) (or Res(k) for k constant) has feasible interpolation. This notion will be defined in Section 4. Let us say for the moment, that Resolution, Cutting Planes, Relativized Bounded Arithmetic, Polynomial Calculus, Lovász-Schrijver and Nullstellensatz have feasible interpolation (see [20, 12, 26, 15, 22, 30, 29, 27]). On the other hand, the stronger system Frege, and any system that simulates AC^0-Frege do not have feasible interpolation under a cryptographic conjecture. To obtain this characterization 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 also show that Resolution requires almost exponential size to prove its own reflection principle. As a corollary we get an almost exponential lower bound for the monotone interpolation of Res(2) improving over the quasipolynomial lower bound in [4].

Despite the discouraging results in [1] mentioned before, there is still some effort put in finding good algorithms for proof systems such as Resolution. The first implementations were variants of the Davis-Putnam procedure [18, 17] for testing unsatisfiability that consists of either producing a tree-like Resolution refutation (if one exists), or giving a satisfying assignment. For various versions of this algorithm, one can prove that is it not an automatization procedure even for tree-like Resolution. A better algorithm for finding tree-like Resolution refutations was proposed by Beame and Pitassi [5]. They give an algorithm that works in time quasipolynomial in the size of the smallest proof of the tautology. So tree-like Resolution is automatizable in quasipolynomial time, but the algorithm is not a good automatization procedure for general Resolution (see [10, 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 gave a class of tautologies for which the algorithm will take subexponential 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 [1]. As they claim, this is a necessary step towards proving that Resolution is not automatizable in quasipolynomial-time. Our techniques also suggest a new complexity measure for Resolution that refines the width of Ben-Sasson and Wigderson, and that gives rise to a new algorithm to find Resolution refutations.

2 Definitions

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

$$\frac{A \vee l \quad \neg l \vee B}{A \vee B}$$

where l is a literal, 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 clause is used at most once in the proof. When this restriction is not fulfilled, we say that the refutation is in DAG form.

Following [7] the *width* of a refutation Π is defined as the maximum number of literals of the clauses appearing in Π. The main result in [7] is a relation between the size and the width of Resolution refutations. They show that if a set of 3-clauses has a tree-like Resolution refutation of size S_T, then it has a Resolution refutation of width $\log S_T$. Similarly, if it has a Resolution refutation of size S_R, then it has a Resolution refutation of width $O(\sqrt{n \log S_R})$. Ben-Sasson and Wigderson used this size-width trade-off to obtain an algorithm that finds Resolution refutations. It consists in deriving all posible clauses of increasing width until the empty clause is found. The time of the algorithm is $n^{O(w)}$ where w is the minimal width of a Resolution refutation of the initial set of clauses. Notice that the space used by the algorithm can only be bounded by $n^{O(w)}$ since all derivable clauses of width $v < w$ are needed to obtain the clauses of width w. Recall that the minimal width w is at most $\log S_T$ in the tree-like case, where

S_T is the minimal tree-like size to refute the initial set of clauses. Therefore, the algorithm takes time $S_T^{O(\log n)}$ in this case. Also, the minimal width w is at most $\sqrt{n \log S_R}$ in the general case, where S_R is the minimal size to refute the set of clauses in general Resolution. This gives an $n^{O(\sqrt{n \log S_R})}$ bound on the running time.

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 [23], works with k-disjunctions. There are three inference rules in Res(k): Weakening, $\wedge$-Introduction, and Cut.

$$\frac{A}{A \vee B} \qquad \frac{A \vee l_1 \quad B \vee (l_2 \wedge \ldots \wedge l_s)}{A \vee B \vee (l_1 \wedge \ldots \wedge l_s)} \qquad \frac{A \vee (l_1 \wedge \ldots \wedge l_s) \quad B \vee \neg l_1 \vee \ldots \vee \neg l_s}{A \vee B}$$

Here A and B are k-disjunctions and the l_i's are literals. As usual, if l is a literal, $\neg l$ denotes the oposite literal. We also allow the axioms $l \vee \neg l$. 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 k-disjunction in the proof to be used only once.

3 Some Simple Lemmas and a New Measure

For every set of literals $l_1, \ldots, l_s$ we define a new variable $z_{l_1,\ldots,l_s}$ meaning $l_1 \wedge \ldots \wedge l_s$. The following clauses define $z_{l_1,\ldots,l_s}$:

$$\neg z_{l_1,\ldots,l_s} \vee l_i \quad \text{for every } i \in \{1, \ldots, s\} \tag{1}$$
$$\neg l_1 \vee \ldots \vee \neg l_s \vee z_{l_1,\ldots,l_s} \tag{2}$$

Let $\mathcal{C}$ be a set of clauses on the variables $v_1, \ldots, v_n$. For every integer $k > 0$, we define $\mathcal{C}_k$ as the union of $\mathcal{C}$ with all the defining clauses for the variables $z_{l_1,\ldots,l_s}$ for all $s \leq k$.

Lemma 1. *If the set of clauses $\mathcal{C}$ has a Res(k) refutation of size S, then $\mathcal{C}_k$ 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 of Lemma 1: Let Π be a Res(k) refutation of size S. To get a Resolution refutation of $\mathcal{C}_k$, we will first get a clause for each k-disjunction of Π. The translation consists in substituting each conjunction $l_1 \wedge \ldots \wedge l_s$ for $s \leq k$ in a clause of Π 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 Π is tree-like, then the new refutation will also be. We have the following cases:

Case 1: In Π we have the step:

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

The corresponding clauses in the translation will be: $C' \vee z_{l_1,\ldots,l_s}$, $D' \vee \neg l_1 \vee \ldots \vee \neg l_s$ 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 $D' \vee \neg l_1 \vee \ldots \vee \neg l_s$ 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 Π we have the step:

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

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 $\mathcal{C}_k$. 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 $\mathcal{C}_k$ that may use axioms of the type $l \vee \neg l$. These can be eliminated easily too. □

Lemma 2. *If the set of clauses $\mathcal{C}_k$ has a Resolution refutation of size S, then $\mathcal{C}$ 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 change each clause of the Resolution refutation by a k-disjunction of Res(k) by translating $z_{l_1,\ldots,l_s}$ by $l_1 \wedge \ldots \wedge l_s$ and $\neg z_{l_1,\ldots,l_s}$ by $\neg l_1 \vee \ldots \vee \neg l_s$. 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 $\neg l_1 \vee \ldots \vee \neg l_s \vee l_i$, which is a weakening of the axiom $l_i \vee \neg l_i$. The clause $\neg l_1 \vee \ldots \vee \neg l_s \vee z_{l_1,\ldots,l_s}$ gets translated into $\neg l_1 \vee \ldots \vee \neg l_s \vee (l_1 \wedge \ldots \wedge l_s)$ which can be proved form the axioms $l_i \vee \neg l_i$ using the rule for the introduction of the $\wedge$. □

The next lemmas are essentially Proposition 1.1 and 1.2 of [21].

Lemma 3. *Any Resolution refutation of width k and size S can be translated into a tree-like Res(k) refutation of size $O(kS)$.*

Proof sketch: Let Π be a Resolution refutation of width k and size S. Every non-initial clause C of Π is derived from two other clauses, say C_1 and C_2. Note that the k-disjunction $\neg C_1 \vee \neg C_2 \vee C$, where $\neg C_i$ is the conjunction of the negated literals of C_i, has a very simple tree-like Res(k) proof. The rest of the proof goes as in [21]. □

Lemma 4. *([21, 25, 19]) Any tree-like Res(k) refutation of size S can be translated into a Resolution refutation of size $O(S^2)$*

These lemmas suggest a refinement of the width mesure that we discuss next. Following [7], for an unsatisfiable set of clauses $\mathcal{C}$, let $w(\mathcal{C})$ be the minimal width of the Resolution refutations of $\mathcal{C}$. We define $k(\mathcal{C})$ to be the minimal k such that $\mathcal{C}$ has a tree-like Res(k) refutation of size n^k, where n is the number of variables of $\mathcal{C}$. We will prove that $k(\mathcal{C})$ is at most linear in $w(\mathcal{C})$, and that in some cases, $k(\mathcal{C})$ is significantly smaller than $w(\mathcal{C})$.

Lemma 5. $k(\mathcal{C}) = O(w(\mathcal{C}))$.

Proof: Let $w = w(\mathcal{C})$. Then $\mathcal{C}$ has a Resolution refutation of size $n^{O(w)}$ and width w since there are less than $n^{O(w)}$ clauses of width at most w and each clause needs to be derived only once since we are in the dag-like case. By Lemma 3, $\mathcal{C}$ a tree-like Res(w) refutation of size $O(wn^{O(w)})$. Taking $k = O(w)$, we see that $k(\mathcal{C}) = O(w(\mathcal{C}))$. □

Lemma 6. *There are sets of 3-clauses* $\mathcal{F}_n$ *such that* $k(\mathcal{F}_n) = O(1)$ *but* $w(\mathcal{F}_n) = \Omega(\log n / \log\log n)$.

Proof: Let $\mathcal{F}_n$ be the set of 3-clauses $E\text{-}PHP^m_{m'}$ where $m' = \log m / \log\log m$. Let n be the number of variables of $E\text{-}PHP^m_{m'}$. Dantchev and Riis [16] proved that $\mathcal{F}_n$ has tree-like Resolution refutations of size $2^{O(m' \log m')}$ which in this case is $n^{O(1)}$. Therefore, $k(\mathcal{F}_n) = O(1)$. On the other hand, a standard width lower bound argument proves that $w(\mathcal{F}_n) = \Omega(m')$ which in this case is $\Omega(\log n / \log\log n)$. □

These Lemmas give rise to an algorithm to find Resolution refutations that improves the width algorithm of Ben-Sasson and Wigderson. Due to space limitations, we omit the precise description of this algorithm (see [3] instead). In a nutshell, the algorithm consists in using the algorithm of Beame and Pitassi [5] to find tree-like Resolution refutations of $\mathcal{C}_k$ of size n^k for increasing values of k until one is found. By Lemma 6, this algorithm improves Ben-Sasson and Wigderson in terms of space usage, and by Lemma 5 its running time is never worse for sets of clauses with relatively small (subexponential) Resolution refutations.

4 Reflection Principles and Weak Automatizability

Let $\mathcal{S}$ be a refutational proof system. Following Razborov [30] (see also [28]), let $REF(\mathcal{S})$ be the set of pairs $(\mathcal{C}, m)$, where $\mathcal{C}$ is a CNF formula that has an $\mathcal{S}$-refutation of size m. Furthermore, let SAT^* be the set of pairs $(\mathcal{C}, m)$ where $\mathcal{C}$ is a satisfiable CNF. Observe that when m is given in unary, both $REF(\mathcal{S})$ and SAT^* are in the complexity class NP. Pudlák called $(REF(\mathcal{S}), SAT^*)$ the *canonical NP-pair* of $\mathcal{S}$. Note also that $REF(\mathcal{S}) \cap SAT^* = \emptyset$ since $\mathcal{S}$ is supposed to refute unsatisfiable CNF formulas only. Interestingly enough, there is a tight connection between the complexity of the canonical NP-pair of $\mathcal{S}$ and the weak automatizability of $\mathcal{S}$. Namely, Pudlák [28] showed that $\mathcal{S}$ is weakly automatizable if and only if the canonical NP-pair of $\mathcal{S}$ is polynomially separable, which means that a polynomial-time algorithm returns 0 on every input from $REF(\mathcal{S})$ and returns 1 on every input from SAT^*. We will use this connection later.

The disjointness of the canonical NP-pair for a proof system $\mathcal{S}$ is often expressible as a contradictory set of clauses. Suppose that one is able to write down a CNF formula $SAT^n_r(x, z)$ meaning that "z encodes a truth assignment that satisfies the CNF encoded by x. The CNF is of size r and the underlying variables are $v_1, \ldots, v_n$". Similarly, suppose that one is able to write down a CNF formula $REF^n_{r,m}(x, y)$ meaning that "y encodes an $\mathcal{S}$-refutation of the CNF encoded by x. The size of the refutation is m, the size of the CNF is r, and the underlying variables are $v_1, \ldots, v_n$". Under these two assumptions, the disjointness of the canonical NP-pair for $\mathcal{S}$ is expressible by the contradictions $REF^n_{r,m}(y, z) \wedge SAT^n_r(x, z)$. This collection of CNF formulas is referred to as the *Reflection Principle* of $\mathcal{S}$. Notice that $REF^n_{r,m}(y, z) \wedge SAT^n_r(x, z)$ is a form of consistency of S.

We turn next to the concept of Feasible Interpolation introduced by Krajicek [22] (see also [12, 26]). 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. We say that a proof system $\mathcal{S}$ has the *Interpolation Property* in time $T = T(m)$ if there exists an algorithm that, given a truth assignment a for the common variables x, returns an $i \in \{0, 1\}$ such that $A_i(a, y_i)$ is contradictory, and the running time is bounded by $T(m)$ where m is the minimal size of an $\mathcal{S}$-refutation of $A_0(x, y_0) \wedge A_1(x, y_1)$. Whenever $T(m)$ is a polynomial, we say that $\mathcal{S}$ has *Feasible Interpolation.*

The following result by Pudlák connects feasible interpolation with the reflection principle and weak automatizability.

Theorem 1. *[28] If the reflection principle for $\mathcal{S}$ has polynomial-size refutations in a proof system that has the feasible interpolation, then the canonical NP-pair for $\mathcal{S}$ is polynomially separable, and therefore $\mathcal{S}$ is weakly automatizable.*

For the rest of this section, we will need a concrete encoding of the reflection principle for Resolution. We start with the encoding of $SAT^n_r(x, z)$. 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 $e \in \{0, 1\}$, $i \in \{1, \ldots, n + 1\}$ 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 truth assignment due to a literal among $v_1, \neg v_1, \ldots, v_{i-1}, \neg v_{i-1}$. Similarly, the meaning of $z_{1,i,j}$ is that clause j is satisfied by the truth assignment due to a literal among $v_1, \neg v_1, \ldots, v_{i-1}, \neg v_{i-1}, v_i$. We formalize this as a set of clauses as follows:

$$\neg z_{0,1,j} \quad (3) \qquad z_{0,n+1,j} \quad (4)$$
$$z_{0,i,j} \vee \neg x_{0,i,j} \vee z_i \vee \neg z_{1,i,j} \quad (5) \qquad z_{1,i,j} \vee \neg x_{1,i,j} \vee \neg z_i \vee \neg z_{0,i+1,j} \quad (6)$$
$$z_{0,i,j} \vee x_{0,i,j} \vee \neg z_{1,i,j} \quad (7) \qquad z_{1,i,j} \vee x_{1,i,j} \vee \neg z_{0,i+1,j} \quad (8)$$

The encoding of $REF^n_{r,m}(x,y)$ is also quite standard. The encoding of the set of clauses by the variables in $\overline{x}$ is as before. The encoding of the Resolution refutation by the variables in $\overline{y}$ 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 was obtained from clause C_j and some other clause, and C_j contains the resolved variable positively (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 was obtained by resolving upon v_i. We formalize this by the following set of clauses:

$\neg x_{e,i,j} \vee y_{e,i,j}$	(9)	$\neg y_{e,i,m}$	(10)
$\neg y_{0,i,j} \vee \neg y_{1,i,j}$	(11)	$p_{1,k} \vee \ldots \vee p_{k-1,k}$	(12)
$q_{1,k} \vee \ldots \vee q_{k-1,k}$	(13)	$\neg p_{j,k} \vee \neg q_{j,k}$	(14)
$\neg p_{j,k} \vee \neg p_{j',k}$	(15)	$\neg q_{j,k} \vee \neg q_{j',k}$	(16)
$\neg p_{j,k} \vee \neg w_{i,k} \vee y_{0,i,j}$	(17)	$\neg q_{j,k} \vee \neg w_{i,k} \vee y_{1,i,j}$	(18)
$\neg p_{j,k} \vee w_{i,k} \vee \neg y_{e,i,j} \vee y_{e,i,k}$	(19)	$\neg q_{j,k} \vee w_{i,k} \vee \neg y_{e,i,j} \vee y_{e,i,k}$	(20)
$w_{1,k} \vee \ldots \vee w_{n,k}$	(21)	$\neg w_{i,k} \vee \neg w_{i',k}$	(22)

Notice that this encoding has the appropriate form for the monotone interpolation theorem.

Theorem 2. *The reflection principle for Resolution $SAT^n_r(x,z) \wedge REF^n_{r,m}(x,y)$ 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 (10). We distinguish two cases: $k \leq r$ and $r < k \leq m$. Since the case $k \leq r$ is easier but long, we leave it to Appendix A.

For the case $r < k \leq m$, 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 (18) and (11) 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

$$\neg q_{l,k} \vee \neg w_{q,k} \vee (y_{1,q,l} \wedge \neg z_q) \vee \bigvee_{\substack{i=1 \\ i \neq q}}^{n} (y_{0,i,l} \wedge z_i) \vee (y_{1,i,l} \wedge \neg z_i). \tag{23}$$

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

$$\neg q_{l,k} \vee \neg w_{q,k} \vee \neg z_q \vee \bigvee_{\substack{i=1 \\ i \neq q}}^{n} (y_{0,i,l} \wedge z_i) \vee (y_{1,i,l} \wedge \neg z_i). \tag{24}$$

Let $q' \neq q$. A cut with $z_{q'} \vee \neg z_{q'}$ on $y_{0,q',l} \wedge z_{q'}$ gives

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

From (20) and (22) we get $\neg q_{l,k} \vee \neg w_{q,k} \vee \neg y_{0,q',l} \vee y_{0,q',k}$. Resolving with (24) on $y_{0,q',l} \wedge z_{q'}$ gives

$$\neg q_{l,k} \vee \neg w_{q,k} \vee \neg z_q \vee y_{0,q',k} \vee (y_{1,q',l} \wedge \neg z_{q'}) \vee \bigvee_{i \neq q,q'} (y_{0,i,l} \wedge z_i) \vee (y_{1,i,l} \wedge \neg z_i). \quad (26)$$

An introduction of conjunction between (25) and (26) gives

$$\neg q_{l,k} \vee \neg w_{q,k} \vee \neg z_q \vee (y_{0,q',k} \wedge z_{q'}) \vee (y_{1,q',l} \wedge \neg z_{q'}) \vee \bigvee_{i \neq q,q'} (y_{0,i,l} \wedge z_i) \vee (y_{1,i,l} \wedge \neg z_i). \quad (27)$$

From (20) and (22) 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

$$\neg q_{l,k} \vee \neg w_{q,k} \vee \neg z_q \vee (y_{0,q',k} \wedge z_{q'}) \vee (y_{1,q',k} \wedge \neg z_{q'}) \vee \bigvee_{i \neq q,q'} (y_{0,i,l} \wedge z_i) \vee (y_{1,i,l} \wedge \neg z_i). \quad (28)$$

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

$$\neg q_{l,k} \vee \neg w_{q,k} \vee \neg z_q \vee \bigvee_{i \neq q} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,l} \wedge \neg z_i). \quad (29)$$

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 (29) 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 $\neg p_{j,k} \vee \neg q_{l,k} \vee \neg w_{q,k} \vee D_k$. Resolving with (21) gives $\neg p_{j,k} \vee \neg q_{l,k} \vee D_k$. Coming to the end, we resolve this with (12) to get $p_{l,k} \vee \neg q_{l,k} \vee D_k$. Then resolve it with (14) to get $\neg q_{l,k} \vee D_k$, and resolve it with (13) to get D_k. □

An immediate consequence of Theorems 2 and 1 is that if Res(2) has feasible interpolation, then Resolution is weakly automatizable. The reverse implication holds too.

Theorem 3. *Resolution is weakly automatizable if and only if Res(2) has feasible interpolation.*

Proof: Suppose Resolution is weakly automatizable. Then by Corollary 10 in [28], the NP-pair of resolution is polynomially separable. We claim that the canonical pair of Res(2) is also polynomially separable. Here is the separation algorithm: Given a set of clauses $\mathcal{C}$ and a number S, we build $\mathcal{C}_2$ and run the separation algorithm for the canonical pair of Resolution on $\mathcal{C}_2$ and $c \cdot 2S$, where c is the hidden constant in Lemma 1. For the correctness, note that if $\mathcal{C}$ has a Res(2) refutation of size S, then $\mathcal{C}_2$ has a Resolution refutation of size $c \cdot 2S$ by Lemma 1, and the separation algorithm for the canonical pair of Resolution will return 0 on it. On the other hand, if $\mathcal{C}$ is satisfiable, so is $\mathcal{C}_2$ and the separation algorithm for Resolution will return 1 on it. Now, for the feasible interpolation of Res(2), consider the following algorithm. Let $A_0(x,y) \wedge A_1(x,z)$ be a contradictory set of clauses with a Res(2) refutation Π of size S. Given a truth assignment a for the variables x, run the separation algorithm for the canonical pair of Res(2) on inputs $A_0(a,y)$ and S. For the correctness, observe that if $A_1(a,z)$ is satisfiable, say by $z = b$, then $\Pi|_{x=a,z=b}$ is a Res(2) refutation of $A_0(a,y)$ of size at most S and the separation algorithm will return 0 on it. On the other hand, if $A_0(a,y)$ is satisfiable, the separation algorithm will return 1, which is correct. If both are unsatisfiable, any answer is fine. □

The previous theorem works for any k constant. If $k = \log n$, then we get that if Resolution is weakly automatizable then Res(log) has feasible interpolation in quasipolynomial time. The positive interpretation of these results is that to show that Resolution is weakly automatizable, then we only have to prove that Res(2) has feasible interpolation. The negative interpretation is that to show that resolution is not weakly automatizable we only have to prove that Res(log) doesn't have feasible interpolation in quasipolynomial time.

It is not clear whether Res(2) has feasible interpolation. We know, however, that Res(2) does not have monotone feasible interpolation (see [4] and Corollary 1 in this paper). On the other hand, tree-like Res(2) has feasible interpolation (even monotone) since Resolution polynomially simulates it by Lemma 4.

A natural question to ask is whether the reflection principle for Resolution has Resolution refutations of moderate size. Since Resolution has feasible interpolation, a positive answer would imply that Resolution is weakly automatizable by theorem 1. Unfortunately, as the next theorem shows, this will not happen. The proof of this result uses an idea due to Pudlak.

Theorem 4. *For some choice of n, r, and m of the order of a quasipolynomial $s^{O(\log s)}$ on the parameter s, every Resolution refutation of $REF^n_{r,m}(x,y) \wedge SAT^n_r(x,z)$ 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 $\{p_{i,j}\}$. 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 $\neg q_{il} \vee \neg q_{jl} \vee \neg p_{ij}$. 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^{2k}_k 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 $c\log k$ literals on the q-variables. Here, c is a constant so that the $k^{O(\log k)}$ upper bound on PHP^{2k}_k of [25] can be done in Res($c\log k$). From its very definition and Lemma 1, if G contains a $2k$-clique, then $\mathcal{F}(G)$ has a Resolution refutation of size $k^{O(\log k)}$. Finally, for every graph G, let $x(G)$ be the encoding of the formula $\mathcal{F}(G)$. With all this notation, we are ready for the argument.

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 $m = k^{O(\log k)}$. By assumption, the formulas $REF^n_{r,m}(x(G), y) \wedge SAT^n_r(x(G), z)$ have Resolution refutations of size at most S. Let C be the monotone circuit that interpolates these formulas given $x(G)$. The size of C is $S^{O(1)}$. Moreover, if G is k-colorable, then $SAT^n_r(x(G), z)$ is satisfiable, and C must return 0 on $x(G)$. Also, if G contains a $2k$-clique, then $REF^n_{r,m}(x(G), y)$ 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 formula $x(G)$ (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 3.11 of Alon and Boppana [2].

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

- 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 plugging the constant 1 to the outputs $x_{1,q_{i1},t}, \ldots, x_{1,q_{ik},t}$. The rest of outputs of clause t get plugged the constant 0.
- 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 $x_{0,q_{il},t} = 1$, $x_{0,q_{jl},t} = 1$, $x_{0,p_{ij},t} = \neg p_{ij}$ 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.
- Finally, the clauses defining the conjunctions of up to $c \log k$ 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^{2k}_k and the definitions of the conjunctions up to $c \log k$ literals. Therefore $REF(x(G), y)$ is satisfiable given that PHP^{2k}_k has a small Res($c \log k$) 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 [4].

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

Theorem 4 is in sharp contrast with the fact that an appropriate encoding of the reflection principle for Res(2) has polynomial-size proofs in Res(2). This encoding incorporates new z-variables for the truth values of conjunctions of two literals, and new y-variables encoding the presence of conjunctions in the 2-disjunctions of the proof. The resulting formula preserves the form of the feasible interpolation. We leave the tedious details to the interested reader.

Theorem 5. *The reflection principle for Res(2) has Res(2) refutations of size* $(n^2 r + mr)^{O(1)}$. *More strongly, the reflection principle for Res(k) has Res(2) refutations of size* $(n^k r + mr)^{O(1)}$.

We observe that there is *a version* of the reflection principle for Resolution that has polynomial-size proofs in Resolution. Namely, let $\mathcal{C}$ be the CNF formula $SAT^n_r(x, z) \wedge REF^n_{r,m}(y, z)$. Then, $\mathcal{C}_2$ has polynomial-size Resolution refutations by Lemma 1 and Theorem 2. However, this does not imply the weak automatizability of Resolution since the set of clauses does not have the appropriate form for the feasible interpolation theorem.

5 Short Proofs that Require Large Width

Bonet and Galesi [11] gave an example of a CNF expressed in constant width, with small Resolution refutations, and requiring 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 [1] 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 $k \leq \log n$. Then the example consists of adding to the formula the clauses defining the extension variables for all the conjunctions of at most k literals. Below we ilustrate this technique by giving a large class of examples that have small Resolution refutations, 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^m_n, defined by Ben-Sasson and Wigderson [7], 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:

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

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 1. [7] *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 $|\partial U'| \geq f \cdot |U'|$.*

The proof of the following statement is the same as in [7] for Resolution.

Theorem 6. [7] *Let $\mathcal{S}$ be a sound refutation system with all rules having fan-in at most two. Then, if G is (m,n,r,f)-expanding, every $\mathcal{S}$-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 $m \geq 2n$, let $\mathcal{C}(G)$ be the set of clauses defining G-PHP_n^m together with the clauses defining all the conjunctions up to $c \log n$ literals, where c is a large constant.

Theorem 7. *Let G be an $(m,n,\Omega(n/\log m),\frac{3}{4}\log m)$-expander with $m \geq 2n$ 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 $\Omega(n/\log n)$, and (iii) $\mathcal{C}(G)$ has polynomial-size Resolution refutations.*

Proof: Part (i) is obvious. For (ii), suppose for contradiction that $\mathcal{C}(G)$ has a Resolution refutation of width $w = o(n/\log n)$. Then, by the proof of Lemma 2, G-PHP_n^m has a Res($c \log n$) refutation in which every $(c \log n)$-disjunction involves at most $wc \log n = o(n)$ literals. This contradicts Theorem 6. For (iii), recall that PHP_n^m has a Res($c \log n$) refutation of size $n^{O(\log n)}$ by [25] since $m \geq 2n$. Now, setting to zero the appropriate variables of PHP_n^m, we get a Res($c \log n$) 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. □

It is known that deciding whether a bipartite graph is an expander (for a slightly different definition than ours) is coNP-complete [8]. Although we have not checked the details, we suspect that deciding whether a bipartite graph is an (m, n, r, f)-expander in the sense of Definition 1 is also coNP-complete. However, we should note that the class of formulas $\{\mathcal{C}(G) : G \text{ expander}, m \geq 2n\}$ is contained in $\{\mathcal{C}(G) : G \text{ bipartite}, m \geq 2n\}$ which is decidable in polynomial-time, and that all formulas of this class have short Resolution refutations that are easy to find. This is so because the proof of PHP_n^{2n} in [25] is given explicitely.

6 Conclusions and Open Problems

We showed that the new measure $k(\mathcal{C})$ introduced in section 3 is a refinement of the width $w(\mathcal{C})$. Actually, we believe that a careful analysis in Lemma 5 could even show that $k(\mathcal{C}) \leq w(\mathcal{C}) + 1$ for sets of clauses $\mathcal{C}$ with sufficiently many variables. On the other hand, we proved a logarithmic gap between $k(\mathcal{C})$ and $w(\mathcal{C})$ for a concrete class of 3-clauses $\mathcal{C}_n$. We do not know if a larger gap is possible.

It is surprising that the weak pigeonhole principle PHP_n^{2n} has short Resolution proofs when encoded with the clauses defining the extension variables. This suggests that to prove Resolution lower bounds that are robust, one should prove Res(k) lower bounds for relatively large k. In fact, at this point the only robust lower bounds we know are the ones for AC^0-Frege.

Of course, it remains open whether Resolution is weakly automatizable, or automatizable in quasipolynomial-time.

Acknowledgement. We are grateful to Pavel Pudlák for stimulating discussions on the idea of Theorem 4.

References

1. M. Alekhnovich and A. A. Razborov. Resolution is not automatizable unless W[P] is tractable. In *42nd Annual IEEE Symposium on Foundations of Computer Science*, 2001.
2. N. Alon and R. B. Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7:1–22, 1987.
3. A. Atserias and M. L. Bonet. On the automatizability of resolution and related propositional proof systems. ECCC TR02-010, 2002.
4. A. Atserias, M. L. Bonet, and J. L. Esteban. Lower bounds for the weak pigeonhole principle and random formulas beyond resolution. Accepted for publication in *Information and Computation.* A preliminary version appeared in ICALP'01, *Lecture Notes in Computer Science* 2076, Springer, pages 1005–1016., 2001.
5. P. Beame and T. Pitassi. Simplified and improved resolution lower bounds. In *37th Annual IEEE Symposium on Foundations of Computer Science*, pages 274–282, 1996.
6. E. Ben-Sasson, R. Impagliazzo, and A. Wigderson. Near-optimal separation of general and tree-like resolution. To appear, 2002.
7. E. Ben-Sasson and A. Wigderson. Short proofs are narrow–resolution made simple. *J. ACM*, 48(2):149–169, 2001.

8. M. Blum, R. M. Karp, O. Vornberger, C. H. Papadimitriou, and M. Yannakakis. The complexity of testing whether a graph is a superconcentrator. *Information Processing Letter*, 13:164–167, 1981.
9. M. L. Bonet, C. Domingo, R. Gavaldà, A. Maciel, and T. Pitassi. Non-automatizability of bounded-depth Frege proofs. In *14th IEEE Conference in Computational Complexity*, pages 15–23, 1999. Accepted for publication in the Journal of Computational Complexity.
10. M. L. Bonet, J. L. Esteban, N. Galesi, and J. Johansen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM Journal of Computing*, 30(5):1462–1484, 2000. A preliminary version appeared in FOCS'98.
11. M. L. Bonet and N. Galesi. Optimality of size-width trade-offs for resolution. *Journal of Computational Complexity*, 2001. To appear. A preliminary version appeared in FOCS'99.
12. M. L. Bonet, T. Pitassi, and R. Raz. Lower bounds for cutting planes proofs with small coefficients. *Journal of Symbolic Logic*, 62(3):708–728, 1997. A preliminary version appeared in STOC'95.
13. M. L. Bonet, T. Pitassi, and R. Raz. On interpolation and automatization for Frege systems. *SIAM Journal of Computing*, 29(6):1939–1967, 2000. A preliminary version appeared in FOCS'97.
14. S. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36–50, 1979.
15. S. A. Cook and A. Haken. An exponential lower bound for the size of monotone real circuits. *Journal of Computer and System Sciences*, 58:326–335, 1999.
16. S. Dantchev and S. Riis. Tree resolution proofs of the weak pigeon-hole principle. In *16th IEEE Conference in Computational Complexity*, pages 69–75, 2001.
17. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.
18. M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7:201–215, 1960.
19. J. L. Esteban, N. Galesi, and J. Messner. Personal communication. Manuscript, 2001.
20. R. Impagliazzo, T. Pitassi, and A. Urquhart. Upper and lower bounds for tree-like cutting planes proofs. In *9th IEEE Symposium on Logic in Computer Science*, pages 220–228, 1994.
21. J. Krajícek. Lower bounds to the size of constant-depth propositional proofs. *Journal of Symbolic Logic*, 39(1):73–86, 1994.
22. J. Krajícek. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *Journal of Symbolic Logic*, 62:457–486, 1997.
23. J. Krajícek. On the weak pigeonhole principle. To appear in *Fundamenta Mathematicæ*, 2000.
24. J. Krajícek and P. Pudlák. Some consequences of cryptographical conjectures for s_2^1 and ef. *Information and Computation*, 140(1):82–94, 1998.
25. A. Maciel, T. Pitassi, and A. R. Woods. A new proof of the weak pigeonhole principle. In *32nd Annual ACM Symposium on the Theory of Computing*, 2000.
26. P. Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62(3):981–998, 1997.
27. P. Pudlák. On the complexity of the propositional calculus. In *Sets and Proofs, Invited Papers from Logic Colloquium '97*, pages 197–218. Cambridge University Press, 1999.

28. 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, pages 621–632. Springer-Verlag, 2001.
29. P. Pudlák and J. Sgall. Algebraic models of computation and interpolation for algebraic proof systems. In P. W. Beame and S. R. Buss, editors, *Proof Complexity and Feasible Arithmetic*, volume 39 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 279–296. American Mathematical Society, 1998.
30. A. A. Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. Izvestiya of the RAN, 1995.

A Appendix: Deriving D_k

We consider the case $k \leq r$. We will derive D_k by successive steps as follows. Let $D^0_{q,k}$ be the following 2-disjunction

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

and let $D^1_{q,k}$ be the following 2-disjunction

$$D^1_{q,k} \equiv z_{1,q,k} \vee (y_{1,q,k} \wedge \neg z_q) \vee \bigvee_{i=q+1}^{n} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i).$$

Observe that $D^0_{n+1,k}$ is simply $z_{0,n+1,k}$ which is the clause (4) in $SAT^n_r(\overline{x}, \overline{z})$. We obtain $D^1_{q-1,k}$ from $D^0_{q,k}$ as follows. Cut $D^0_{q,k}$ with (8) and (6) on $z_{0,q,k}$ to get

$$z_{1,q-1,k} \vee x_{1,q-1,k} \vee \bigvee_{i=q}^{n} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i), \tag{30}$$

and

$$z_{1,q-1,k} \vee \neg x_{1,q-1,k} \vee \neg z_{q-1} \vee \bigvee_{i=q}^{n} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i), \tag{31}$$

respectively. A cut between (30) and (31) on $x_{1,q-1,k}$ gives

$$z_{1,q-1,k} \vee \neg z_{q-1} \vee \bigvee_{i=q}^{n} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i). \tag{32}$$

On the other hand, a cut between (30) and (9) on $x_{1,q-1,k}$ gives

$$z_{1,q-1,k} \vee y_{1,q-1,k} \vee \bigvee_{i=q}^{n} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i), \tag{33}$$

Finally, an introduction of conjunction between (32) and (33) on $\neg z_{q-1}$ and $y_{1,q-1,k}$ gives $D^1_{q-1,k}$ as claimed. Next, we show how to get $D^0_{q,k}$ from $D^1_{q,k}$. Cut $D^1_{q,k}$ with (7) and (5) on $z_{1,q,k}$ to get

$$z_{0,q,k} \vee x_{0,q,k} \vee (y_{1,q,k} \wedge \neg z_q) \vee \bigvee_{i=q+1}^{n} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i), \tag{34}$$

and

$$z_{0,q,k} \vee \neg x_{0,q,k} \vee z_q \vee (y_{1,q,k} \wedge \neg z_q) \vee \bigvee_{i=q+1}^{n} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i), \tag{35}$$

respectively. A cut between (34) and (35) on $x_{0,q,k}$ gives

$$z_{0,q,k} \vee z_q \vee (y_{1,q,k} \wedge \neg z_q) \vee \bigvee_{i=q+1}^{n} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i). \tag{36}$$

On the other hand, a cut between (34) and (9) on $x_{0,q,k}$ gives

$$z_{0,q,k} \vee y_{0,q,k} \vee (y_{1,q,k} \wedge \neg z_q) \vee \bigvee_{i=q+1}^{n} (y_{0,i,k} \wedge z_i) \vee (y_{1,i,k} \wedge \neg z_i). \tag{37}$$

Finally, an introduction of conjunction between (36) and (37) on z_q and $y_{0,q,k}$ gives $D^0_{q,k}$ as desired.

We have shown how to obtain $D^0_{1,k}$. In order to obtain D_k, we only need to cut $D^0_{1,k}$ with (3) on $z_{0,1,k}$.