Using the Groebner basis algorithm to find proofs of unsatisfiability

Matthew Clegg*
Computer Science Dept.
UCSD
9500 Gilman Drive
La Jolla, CA 92093-0114
mclegg@cs.ucsd.edu

Jeffery Edmonds†
Computer Science Dept.
York University
Toronto, Ontario Canada
jeff@cs.yorku.ca

Russell Impagliazzo‡
Computer Science and Engineering
UC, San Diego
9500 Gilman Drive
La Jolla, CA 92093-0114
russell@cs.ucsd.edu

Abstract

A propositional proof system can be viewed as a non-deterministic algorithm for the (co-NP complete) unsatisfiability problem. Many such proof systems, such as resolution, are also used as the basis for heuristics which deterministically search for short proofs in the system.

We discuss a propositional proof system based on algebraic reasoning, which we call the Groebner proof system because of a tight connection to the Groebner basis algorithm. For an appropriate measure of proof size, we show that (a degree-limited implementation of) the Groebner basis algorithm finds a Groebner proof of a tautology in time polynomial in the size of the smallest such proof, In other words, unlike most proof systems, the non-deterministic algorithm can be converted to a deterministic one without loss in power.

We then compare the power of the Groebner proof system to more studied systems. We show that the Groebner system polynomially simulates Horn clause resolution, quasi-polynomially simulates tree-like resolution, and weakly exponentially simulates resolution. Thus, Groebner proofs will have better than worst-case behaviour on the same classes of inputs that resolution does. On the other hand, there are simple tautologies which have polynomial-size Groebner proofs but which require exponential-size resolution proofs.

We also compare the Groebner proof system to the similar Nullstellensatz proof system introduced in [BIK+94]. We show a family of tautologies that have degree 3 (and hence polynomial-size) Groebner refutations, but which require $\Theta(\sqrt{n})$ degree Nullstellensatz refutations. Thus, there is an exponential separation between the two systems.

These results suggest that the Groebner basis algorithm might replace resolution as a basis for heuristics for NP-complete problems. We have done some preliminary experiments comparing [CW]'s implementation of the Groebner basis algorithm to [KS]'s implementation of the Putnam-Davis procedure for unsatisfiability. [KS]'s program is superior on the instances tested. However, the difference is small enough to hope that further optimization of the Groebner basis algorithm for unsatisfiability could make it the method of choice.

1 Introduction

Propositional proof systems are formal systems for reasoning about the consequences of known relationships between propositional (true/false) variables. Understanding such systems is highly important not just for logic but also for complexity theory and artificial intelligence.

First, lower bounds for such systems can be thought of as a concrete step towards proving $P \neq NP$. Many of the best algorithms for NP-complete problems use the back-tracking approach. In an NP-complete problem, one is typically trying to find a "solution" to a problem that satisfies certain constraints. In the back-tracking approach, the algorithm adds a new constraint, and then recursively tries to find a solution that meets this new constraint as well. If it cannot, it backtracks, and attempts to find a solution that also meets the negation of the constraint. Hopefully, the added constraint narrows the search space, so that recursive problems are easier than the original. If you examine the computation of a back-tracking program on a problem with no solution, it gives you a proof by contradiction that the problem has no solution. The proof has the format: Assume (first back-tracking constraint) . Obtain contradiction (recursively). Assume (negation of constraint). Obtain contradiction.

The formal proof system corresponding to a back-tracking algorithm will have as "lines" the constraints that the algorithm branches on, "axioms" and "inference rules" corresponding to the (hopefully, contradictory) situations in

*Research supported by Office of Naval Research AASERT grant $\#N0014-94-1-0997$

†Research partially done while at ICSI supported by NSERC and NSF Post-doctoral Fellowships

‡Research Supported by NSF YI Award CCR-92-570979, Sloan Research Fellowship BR-3311, grant #93025 of the joint US-Czechoslovak Science and Technology Program, and USA-Israel BSF Grant 92-00043

which the recursion bottoms out, and the size of the proof is proportional to the ronning time of the algorithm. Thus, lower bounds on proof size for formal proof systems give you corresponding lower bounds for any program that uses the corresponding type of back-tracking. Such lower bounds give us large, natural classes of algorithms that provably cannot solve NP-complete problems quickly. Separations of different proof systems give us examples of inputs where certain types of branching or termination rules give better performance than others.

On the other hand, proof systems are used in artificial intelligence and "logic programming" languages such as Prolog. as a method of computation and as a way to represent knowledge. This is basically the contrapositive of the above discussion. If the heart of an algorithm for an NP problem is finding a proof, then proof systems should be used to develop better algorithms. These approaches typically make use of simple but weak proof systems such as resolution, because the size of the smallest proof is often less of an obstacle than the complexity of actually finding small proofs if they exist. This means such algorithms are doubly heuristic: one must hope that tautology one is interested in has a relatively small proof, and then also hope that the proof search heuristic actually finds it. From this perspective, the goal of studying various proof systems is to find systems that are strong (i.e., short proofs exist for many tautologies), simple, and for which searching for proofs is computationally feasible. Although the exact complexity of proof-searching for resolution is unknown, there is some work indicating that it is itself a hard problem ([IM]).

In this paper, we "introduce" a "new" proof system based on algebraic reasoning which seems quite promising as a potential replacement for resolution. (Actually, the system will be basically familiar to anyone who has taken a course in algebra.) We then show that the Groebner basis algorithm, well-known to algebraicists, can be used to approximate the smallest proof in the system in polynomial time in the size of the proof. For this reason, we refer to it as the "Groebner proof system". This shows that, unlike for more conventional proof systems, for Groebner proofs, the search for proofs can be feasibly done provided small proofs exist.

Of course, this is not very interesting if short Groebner proofs do not exist for the relevant tautologies (i.e., those for which resolution-based methods are currently used.) We show that in fact the Groebner proof system should in principle be useful on for the same kinds of instances for which resolution methods are sub-exponential. More precisely, we show the Groebner system polynomially simulates Horn clause resolution, quasi-polynomially simulates tree-like resolution, and weakly exponentially simulates arbitrary resolution proofs. (We say that a system polynomially simulates another if whenever there are polynomial ($n^{O(1)}$) size proofs in the second system for a family of tautologies, there are also polynomial size proofs in the other. Quasi-polynomial ($2^{log^{O(1)}n}$) and weakly exponetial ($2^{o(n)}$) simulations are defined analogously.) Thus, for a tautology that for which a sub-exponential size resolution proof *exists*, the Groebner basis algorithm will *find* a Groebner proof in sub-exponential time. The simulation is even better if the proof is tree-like, and almost all implemented proof-searching heuristics (e.g. the Putnam-Davis procedure) find tree-like proofs. This is a theoretical justification for our belief that our method should be competitive against the current best ones, which use a resolution-based backtracking approach, and better for some applications.

Using Clegg and Wallach's implementation of the Groebner basis algorithm ([CW]), we have begun some testing of our method as a heuristic for unsatisfiability using benchmark data from Selman [S]. Most recently, our approach took 10 minutes of CPU time on a SPARC 20 Model 61 to solve a problem that took 5 minutes using Kirkpatrick and Selman's backtracking algorithm [KS]. (It should be noted that the [KS] algorithm is specifically geared to solving satisfiability, while [CW] is intended for generic algebra applications.) This suggests that, while further work at implementation is needed, the method may be competitive or superior to the best known techniques for this problem.

The Groebner proof system generalizes a similar proof system, the Nullstellensatz system, introduced in [BIK+94]. However, we give a simple family of tautologies based on the principle of strong induction for which there is an exponential separation between the two systems,

2 The Groebner proof system

The main factors defining a proof system are the allowable *lines*, i,e., the types of relationships between propositional variables the system reasons about, and the *inference rules*, i.e., how new relationships are deduced from known ones. Let F be a field. Allowable lines in a Groebner proof over F are polynomial equations over F, i.e., $g(x_1, ...x_n) = 0$ for $g \in F[x_1, ..x_n]$. The equation $x^2 = x$ for each variable x is an axiom; this says that each variable is Boolean, either 0 or 1. Inference rules are as follows: from $g_1(\vec{x}) = 0$ and $g_2(\vec{x}) = 0$ infer $ag_1(\vec{x}) + bg_2(\vec{x}) = 0$ where a, b are constants from F; and from $g(\vec{x}) = 0$ infer $xg(\vec{x}) = 0$ for x a variable.

As an example, consider the following valid inference; given $x_1 \rightarrow x_2$,$x_2 \rightarrow x_3$,..$x_{n-1} \rightarrow x_n$, infer $x_1 \rightarrow x_n$. We can represent $x \rightarrow y$ by the equation : $xy = x$ (if $x = 1$, then $y = 1$, if $x = 0$, the equation gives no constraint). Thus, we translate the given relationships as the system of equations: $x_1 - x_1x_2 = 0$, $x_2 - x_2x_3 = 0$,...$x_{n-1} - x_{n-1}x_n = 0$. Multiplying the second by x_1 gives us $x_1x_2 - x_1x_2x_3 = 0$, adding the first gives us (*)$x_1 - x_1x_2x_3 = 0$ Multiplying the first given by x_3 gives $x_1x_3 - x_1x_2x_3 = 0$, subtracting from (*) gives us $x_1 - x_1x_3 = 0$, i.e., $x_1 \rightarrow x_3$. We then repeat this process to get $x_1 \rightarrow x_4$, etc. For this inference, we did not use any line of degree more than 3. (On the other hand, Buss and Pitassi have recently shown that the above tautology requires degree $O(logn)$ for the Nullstellensatz system [BP].)

The effect of the axioms $x^2 = x$ is, intuitively, to allow each line in the proof to be multi-linear, i.e., if a variable appears in any term with an exponent greater than 1, we replace the exponent by 1 and combine like terms. Define $m(p)$ to be the multi-linear version of polynomial p. We require that polynomials be represented explicitly, i.e., that zero co-efficients be listed as co-efficients, so a polynomial of degree d is represented as the vector of its co-efficients, whose elements are indexed by the $\sum_{i=0}^{i=d} \binom{n}{i}$ multi-linear terms of degree $i \leq d$, i.e., since the exponent of each variable in each term is either 0 or 1, a term of degree at most d is determined by a subset of variables with at most d members. We use the notation $\binom{n}{\leq d}$ for the above sum. We use as a measure of the size of a proof $\binom{n}{\leq d} = O(n^d)$, where d is the maximum degree of a polynomial equation appearing in the proof, since (as we will see later) this number also bounds the number of lines in any minimal proof. So a tautology has a polynomial-size proof (under this measure of size) if and only if it has one where each line is at most a constant degree d. We write $p_1, ..., p_k \vdash_d q$ to mean that there is a Groebner proof of polynomial q from $p_1, ..., p_k$ where every line has degree at most d.

3 A characterization of provable polynomials

In this section, we give a characterization of those polynomials provable from a given set of polynomials $p_1, ..., p_k$ with a degree d Groebner proof. Consider multi-linear polynomials of degree d over F as a vector space whose co-ordinates are indexed by the multi-linear terms of degree at most d. Let $V_d(p_1, ..., p_k)$ be the smallest sub-space V of this space that includes $p_1, ..., p_k$ and so that if $p \in V$ and p has degree at most $d-1$, and x is any variable, then $m(xp)$ is in V (i.e., if $p = xp_1 + p_2$ then $xp_1 + xp_2 \in V$.) Since the intersection of sub-spaces with these closure properties also has the properties, such a smallest sub-space exists.

Theorem 1: For any multi-linear polynomials $p_1, ..., p_k, q$ of degree at most d, $p_1, ..., p_k \vdash_d q$ if and only if $q \in V_d(p_1, ..., p_k)$.

Proof: Let V be the set of multi-linear polynomials q so that $p_1, ..., p_k \vdash_d q$. Then V is a vector space, since if we can prove q_1 and q_2 with a degree d Groebner proof, we can prove $aq_1 + bq_2$ for any $a, b \in F$ by concatenating the two proofs and then adding this line as a conclusion. V certainly contains $p_1, ..., p_k$ Moreover, if q has a degree d Groebner proof and is of degree at most $d-1$, then adding xq to the end of this proof is still a Groebner proof of degree d, and then we can use the axiom $x^2 - x$ to derive $m(xq)$ from xq, Thus by definition , $V_d(p_1, ,.., p_k) \subseteq V$.

Conversely, assume $q \in V - V_d(p_1, ..., p_k)$. Let $p_1, ..., p_k, r_{k+1}, ..r_t$ be a degree d Groebner proof of q, and let r_i be the first line of the proof with $m(r_i) \notin V_d(p_1, ,.., p_k)$ Then r_i cannot be one of the p_j's, and cannot be an axiom $x^2 - x$ either, since $m(x^2 - x) = 0$. It cannot be the weighted sum of two previous lines since $V_d(\vec{p})$ is a vector space (and $m(ar + bs) = am(r) + bm(s)$). It cannot be xr_j for some previous line r_j since for the degree of xr_j to be at most d, the degree of r_j must be at most $d-1$, and then $m(xr_j) \in V_d(\vec{p})$. Thus, there can be no such line, and so $q \in V_d(\vec{p})$, a contradiction. So $V = V_d(\vec{p})$. □

4 A proof-searching algorithm

In this section, we give an algorithm to compute a basis for $V_d(\vec{p})$ and hence to determine whether q is provable from $\vec{p}$ by a degree d Groebner proof. In the next section, we will show how the well-known Groebner basis algorithm can be used for the same purpose. We include this section to present some ideas used in the next, and because the worst-case time of the Groebner basis algorithm seems to be worse than that of the algorithm presented here.

Fix an order on multi-linear monomials that respects degree, i.e. degree d monomials are larger than degree $d-1$ in the sense of the order. Define the *leading term* $\mathrm{LT}(p)$ of a multi-linear polynomial p to be the largest monomial with a non-zero co-efficient.

The algorithm we describe in this section basically closes the vector space of polynomials spanned by $p_1, ..., p_k$ under multiplication by variables. In doing so, it uses Gaussian elimination to keep all polynomials in the space linearly independent.

More precisely, the algorithm maintains a set B of multi-linear polynomials with the property that no two elements of B have the same leading term, and a set S of polynomials that will eventually be added to B. Every element of S and B will be derivable from $p_1, ..., p_k$ by a degree d Groebner proof, and $p_1, ..., p_k$ will be in the space spanned by $B \cup S$.

1. Initially, $S = \{p_1, ..., p_k\}$ and B is empty.
2. Repeat the following until S is empty.
   (a) Arbitrarily select and remove a polynomial p from S.
   (b) Perform a *Gaussian reduction* of p by B to get a polynomial p' as follows: Check to see if $\mathrm{LT}(p) = \mathrm{LT}(q)$ for some $q \in B$. If not, return p. If so, compute $p - aq$ for $a \in F$ which causes the leading terms to cancel. Then recursively reduce $p - aq$.
   (c) If the reduced polynomial $p' \neq 0$ add it to B. If in addition it has degree $\leq d-1$, add $m(xp')$ to S for each variable x.

The algorithm halts when S is empty. Note that each p' that causes n polynomials to be added to S adds a new polynomial of degree $d-1$ or smaller to B. Thus, since no two elements of B have the same leading term, there are at most $k + n\binom{n}{\leq d-1} = O(n\binom{n}{\leq d-1})$ polynomials that are ever in S. Assuming that a constant cost hash table of leading terms can be maintained and that polynomials are stored as $\binom{n}{\le d}$ dimensional vectors, reducing a polynomial requires time $O(\binom{n}{\le d}^2)$. So the total time for the algorithm is $O(n\binom{n}{\le d-1}(\binom{n}{\le d})^2) = O(d\binom{n}{\le d}^3)$

Lemma 2: At the end of this algorithm, B is a basis for $V_d(p_1, ..., p_k)$.

Proof: The steps of the algorithms form degree d Groebner proofs of each of the elements of B, so $Span(B) \subseteq V_d(\vec{p})$.

B has no two members with the same leading terms, so B is linearly independent. It is easy to see that $Span(S \cup B)$ only increases at every stage of the algorithm. This is because reducing a polynomial p by B does not change the span of $B \cup \{p\}$ and otherwise we only add elements to S. Therefore, at the end, $p_1, ..., p_k \in Span(B)$.

Thus, if we can show that for any degree $d-1$ or smaller polynomial $q \in Span(B)$, $m(xq) \in Span(B)$ then we can conclude that $V_d(\vec{p}) \subseteq B$ by the definition of $V_d(\vec{p})$ Since the leading terms of B are all distinct, $q = \sum a_i b_i$ where $a_i \in F$ and $b_i \in B$ are *of degree at most* $d-1$. To see this, consider any linear combination of elements of B that has any non-zero term from a member of degree d. Consider the maximum leading term of a member of the linear combination. Every other member has co-efficient 0 for this term, so this term has non-zero co-efficient in the linear combination. Thus any linear combination that involves polynomials of degree d from B is itself degree d. Each b_i was added to B at some point, after which $m(xb_i)$ was added to S, since its degree is at most $d-1$. Since $Span(B \cup S)$ only increases, this means each $m(xb_i) \in Span(B)$ at the end of the algorithm, and hence so is $m(xq)$. So $Span(B) \subseteq V_d(\vec{p}) \subseteq Span(B)$, and B is a basis for $V_d(\vec{p})$. □

To see whether $q \in V_d(\vec{p})$ we run the above algorithm and then reduce q, and see if the result is 0.

Thus we have

Theorem 3: There is an algorithm that runs in time $O(d\binom{n}{\le d}^3) = O(n^{3d})$ which determines whether q is derivable from $p_1, ..., p_k$ via a degree d Groebner proof.

5 The Groebner Basis Algorithm

The Groebner Basis Algorithm [B65] is an algorithm for deciding membership in a polynomial ideal. In this section, we formulate a variant of the Groebner Basis Algorithm called the *degree-d Groebner Basis Algorithm.* The degree-d Groebner Basis Algorithm allows us to decide membership in V_d without explicitly computing all of the polynomials in B. For example, if the first polynomial were $xy - x$, our previous algorithm would mindlessly add all the polynomials $xyz - xz$, $xyw - xw$, $xyzw - xzw$, etc. to B. These polynomials do not really add any additional information over the original. In the (degree-d) Groebner basis algorithm, we would only *explicitly* compute the first polynomial. The algorithm then considers all the products as being *virtually* in the basis B.

The output of the degree-d Groebner Basis Algorithm is a set G of polynomials which compactly represents the basis B. The algorithm requires that a term order be chosen where $1 \prec t$ for every term t and which respects multiplication, i.e. if $t_1 \prec t_2$, then $xt_1 \prec xt_2$ for any variable x. The output of the algorithm is dependent upon the choice of the term order.

Given a degree-d Groebner Basis G, membership in V_d can be decided through a reduction algorithm. The definition of a reduction is as follows. Given a term t, define $p(t)$ to be an arbitrary but fixed element $g \in G$ such that $\mathrm{LT}(g) \mid t$. If no such g exists, we define $p(t) = \perp$. If q is a polynomial such that $p(\mathrm{LT}(q)) \ne \perp$, we say that q is *directly Groebner reducible* to r if $r = q - \frac{\mathrm{LT}(q)}{\mathrm{LT}(p(\mathrm{LT}(q)))} p(\mathrm{LT}(q))$. We say that q is *Groebner reducible* to q' if $q = q'$ or if there exists an r such that q is directly Groebner reducible to r and r is Groebner reducible to q'. The reduction process is deterministic, and each direct reduction results in a polynomial whose leading term is less than the leading term of the previous polynomial. So after a finite number of steps, a unique polynomial q' will be found which cannot be further reduced. If q' is 0, we will omit reference to it and simply say that q is *reducible* with respect to G.

Given a polynomial f and a set of polynomials G, we will say that f has a *degree-d representation* with respect to G if it is possible to write $f = \sum g_i h_i$ where $g_i \in G$, h_i is an arbitrary polynomial and deg $g_i h_i \le d$ for each i. If in fact $\mathrm{LT}(g_i h_i) \preceq \mathrm{LT}(f)$ for each i, then we will say that f is *semi-reducible* with respect to G.

Note that if the degree of f is less than or equal to d and f is semi-reducible, then f has a degree-d representation. Also, if f is reducible, then it is semi-reducible. For arbitrary sets G, though, the converses of these statements are not in general true. This motivates our definition of a degree-d Groebner basis. We will say that G is a *degree-d Groebner basis* if f is reducible whenever f has a degree-d representation.

A key part of the Groebner basis algorithm is the computation of S-remainders. The S-remainder of two polynomials f and g is defined to be

$$\mathrm{Srem}(f, g) = \frac{\mathrm{lcm}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LT}(f)} f - \frac{\mathrm{lcm}(\mathrm{LT}(f), \mathrm{LT}(g))}{\mathrm{LT}(g)} g.$$

Computing S-remainders allows us to find polynomials which have degree-d representations but which are not semi-reducible. It turns out that all such polynomials can be found through repeated computations of S-remainders and reductions, so a degree-d Groebner basis can be found by computing a set which is closed under the formation of S-remainders and reductions.

In the rest of this section, we will describe an algorithm for computing a degree-d Groebner basis. The algorithm which we present is a simple variant of the standard Groebner basis algorithm. The underlined part is our modification of the algorithm to keep degrees bounded.

1. Initialize $G = \{\}$, and $S = \{x_i^2 - x_i \mid i = 1, ..., n\} \cup \{p_1, ..., p_k\}$.
2. Repeat the following until S is empty:
   (a) Pick $p \in S$
   (b) Groebner reduce p by G, getting p'.
   (c) If $p' \neq 0$, add p' to the end of G and for each g_i <u>if deg $(\mathrm{lcm}(\mathrm{LT}(g_i), \mathrm{LT}(p')) \leq d$, add $\mathrm{Srem}(p', g_i)$ to S.</u>

After each iteration of the main loop, note that either S gets smaller or G gets larger. If G gets larger, the polynomial p' which is added to G has a leading term which is not divisible by the leading term of any other polynomial in G. Since there are only $\binom{n}{\leq d}$ choices for distinct leading terms, G can only increase this many times. The total number of entries added to S is $n + k + \binom{m}{2}$, where m is the final size of G. Consequently, the running time of the algorithm is proportional to $\binom{n}{\leq d}^2$ times the cost of a single reduction, or $O(\binom{n}{\leq d}^4)$.

To prove that the output of the algorithm is a degree-d Groebner basis, we first prove a few lemmas.

Lemma 4: Suppose that a polynomial f is added to S at some stage of the algorithm. Then, f is semi-reducible with respect to the output set G.

Proof: More generally, let H be an arbitrary set of polynomials, and suppose that a polynomial g is reducible to g' with respect to H. If f is semi-reducible with respect to $H \cup \{g\}$, then f is also semi-reducible with respect to $H \cup \{g'\}$.

Under the hypotheses of the lemma, f is semi-reducible with respect to $G \cup S$ at some stage of the algorithm. Since the algorithm executes by replacing $G \cup S$ with $G \cup S - \{p\} \cup \{p'\}$ for polynomials p and p', the invariant of semi-reducibility with respect to $G \cup S$ is maintained. Since S is empty when the algorithm terminates, the lemma follows. $\square$

This lemma allows us to make the following observations:

- $x_1^2 - x_1, ..., x_n^2 - x_n$ are semi-reducible.
- $p_1, ..., p_k$ are semi-reducible.
- If $g_i, g_j \in G$ and deg $(\mathrm{lcm}(\mathrm{LT}(g_i), \mathrm{LT}(g_j))) \leq d$, then $\mathrm{Srem}(g_i, g_j)$ is semi-reducible.

Lemma 5: Suppose that f has a degree-d representation. Then, f is semi-reducible.

Proof: This is a direct application of Lemma 5 (p. 83) from [CLO]. Let $f = \sum g_i h_i$ be a representation of f, and let $t = \max\{\mathrm{LT}(g_i h_i)\}$. If $\mathrm{LT}(f) \prec t$, then Lemma 5, combined with our observation that degree-d S-remainders are semi-reducible, allows us to rewrite $f = \sum g_i h_i'$ such that $\max\{\mathrm{LT}(g_i h_i')\}$ is smaller than t in the term order. Consequently, we can find a representation for f where $\max\{\mathrm{LT}(g_i h_i')\} = \mathrm{LT}(f)$. But this says that f is semi-reducible. $\square$

Theorem 6: The set G computed by the above algorithm is a degree-d Groebner basis.

Proof: Suppose to the contrary that there exists a polynomial f which has a degree-d representation but which is not reducible. Assume that f is chosen such that $\mathrm{LT}(f)$ is minimal among all such polynomials with this property. Let $\sum g_i h_i$ be a representation of f, and let $t = \max\{\mathrm{LT}(g_i h_i)\}$. By the previous lemma, we may assume that $t = \mathrm{LT}(f)$. But this implies that there is an i such that $\mathrm{LT}(g_i) \mid \mathrm{LT}(f)$. Consequently, $p(\mathrm{LT}(f)) \neq \perp$. Therefore, there exists a polynomial f' such that f is directly reducible to f'. By the minimality of our choice of f, f' is reducible. But then f is reducible, so we have a contradiction. $\square$

Let B_G be the set of polynomials which are reducible with respect to G. By the Theorem, B_G is the same as the set of polynomials which have degree-d representations. We have already noted that $x_i^2 - x_i \in B_G$ for each variable x_i and that $p_1, ..., p_k \in B_G$. In addition, it is easy to see that a linear combination of degree-d representations is a degree-d representation, so B_G is a vector space. Now suppose that $f \in B_G$ is of degree at most $d - 1$. Since f is reducible, it has a degree $d - 1$ representation. Consequently, $x_i f$ has a degree d representation, so B_G is closed under multiplication of degree $d - 1$ polynomials by variables. We therefore have the following:

Theorem 7: $V_d(p_1, ..., p_k) \subseteq B_G$.

As noted above, the running time of the degree-d Groebner basis algorithm is $O(\binom{n}{\leq d}^4)$. While this would appear to be worse than the running time of the algorithm described in the previous section, in practice it performs quite a bit better. It is possible to show that many of the S-remainders need not be computed [B79]. In addition, the running time of the algorithm can be improved through using heuristics for choosing a suitable term order, for ordering the set S, for computing the function $p(t)$, and for computing the reduction of a polynomial. An implementation of these heuristics in a distributed setting is described in [CW]. There remain many questions about the optimal methods to implement this algorithm.

6 The power of the proof system

In this section, we compare the power of the Groebner proof system to that of traditional proof systems such as resolution. Resolution is a system for reasoning about clauses, or's of literals. The or of k literals requires degree k to express as a polynomial. Since we assume that the input polynomials to a Groebner proof are low degree polynomials, we need to either restrict to proofs of tautologies involving small clauses, or to give an indirect formulation for arbitrary clause size. One such indirect formulation is given below. Our results hold for either way of handling this issue. Resolution proofs are typically presented as refutations; one proves that the negation of the and of a set of clauses is a tautology by deriving a contradiction from the set, The equivalent for Groebner proofs is to derive 1 (i.e., $1 = 0$) from the translations of the clauses.

We give several simulations of traditional proof systems by Groebner proofs. First, we show that if the clauses are Horn clauses or the duals of Horn clauses, and are unsatisfiable, then their translation has a Groebner proof of degree 3, and hence the Groebner basis algorithm will derive a contradiction in polynomial-time. Since the translation from clauses to polynomials can be done in log space, we have as a consequence that distinguishing between satisfiable polynomials and those with a degree 3 Groebner proof is P-complete.

Second, if a set of clauses have a tree-like resolution refutation with S lines, then they have a degree $\log S$ Groebner refutation. If S is quasi-polynomial, then this refutation will be of quasi-polynomial size and findable in quasi-polynomial time by the Groebner basis algorithm. The Putnam-Davis procedure is an algorithm (with many variations in implementation) to find tree-like resolution proofs. Thus, for instances on which any version of the Putnam-Davis procedure finds a contradiction in time $2^{\epsilon n}$, the algorithm in section 4 will find a contradiction in time approximately $2^{(3\epsilon \log(1/\epsilon)+3\epsilon)n}$. So any instances on which even an optimal Putnam-Davis procedure is sub-exponential, our algorithm will also be sub-exponential.

Finally, if a set of clauses has any resolution refutation with S lines, then it has a degree $3(n \ln S)^{1/2}$ Groebner refutation. In particular, if $S = 2^{\epsilon n}$, our algorithm finds a refutation in time $2^{O(\epsilon^{1/2} \log(1/\epsilon) n)}$. So if weakly exponential ($S = 2^{o}(n)$) size resolution proofs exist for a class of instances, then our algorithm also has weakly exponential behaviour on these instances.

Represent literal $\overline{x}$ by $1 - x$. For tautologies with large clauses, we can translate the clause $C_i = x_{i,1} \vee x_{i,2} \vee ..x_{i,m}$ into the polynomial $r_{i,1}x_{i,1} + r_{i,2}x_{i,2} + ..r_{i,m}x_{i,m} = 1$ where each $r_{i,j}$ is a new variable. If $x_{i,j} = 1$, we can assign value 1 to $r_{i,j} = 1$ and 0 to all the other $r_{i,k}$ to satisfy the equation. However, if all $x_{i,j} = 0$, we cannot satisfy the equation no matter how we pick $r_{i,j}$. Thus, the original set clauses are satisfiable if and only if the translations are mutually satisfiable. If the clause size is small, and we represent the polynomials directly, we can use arguments based on the above translation by letting $r_{i,j} = (-1)^{m+j}(x_{i,j+1} - 1)(x_{i,j+2} - 1)..(x_{i,m} - 1)$. The or of the variables is given by $r_{i,1}x_{i,1} + r_{i,2}x_{i,2} + ..r_{i,m}x_{i,m}$ when we make the given substitution for $r_{i,j}$. Doing this substitution for every line of a proof from the translations will then at most multiply the refutation degree by the clause size -1. (Actually, using the arguments given below with the direct translation only increases the degree by an additive factor of the clause size.)

If all the clauses are duals of Horn clauses (for simplicity, Horn clauses work out identically), i.e., at most one variable is negated per clause, we proceed as follows to find a degree 3 proof. Take all the variables we know to be false, i.e. where we've been given or derived $x_j = 0$. If they cover all the non-negated variables in a clause, but not the negated variable in the clause, this allows us to make the following series of deductions: We know $r_1 x_1 + ... r_k x_k + r_{k+1}(1 - x_{k+1}) = 1$ and each $x_i = 0$ for $i \leq k$. We use this to derive $r_{k+1}(1 - x_{k+1}) = 1$. Multiply by x_{k+1} to get $r_{k+1}(x_{k+1} - x_{k+1}^2) = x_{k+1}$ Then multiply the axiom $x_{k+1} - x_{k+1}^2$ by r_{k+1} and subtract to get $x_{k+1} = 0$. So we've added a new variable to the list of known false variables. Similarly, if our list of known false variables covers an entire clause with no negated variables, we can derive $0 = 1$, and we're done. If at some point, neither is true, set all remaining variables to 1, and we've satisfied all clauses. So if the Horn clauses were unsatisfiable we find a degree 3 proof of $0 = 1$.

It follows that:

Theorem 8: It is P-hard to distinguish satisfiable polynomials from those with a degree 3 Groebner refutation.

It is actually possible to replace degree 3 with degree 2 in the above theorem, by using a translation that distinguishes between negated and non-negated variables. We would not have a co-efficient $r_{i,k+1}$ for the only non-negated variable. It is easy to see that the same argument holds. This was suggested by Pitassi [P].

The following simple properties of the Groebner proof system will be used in showing simulations of resolution proofs for general clauses. For p a polynomial and x a variable, let $p|_{x=1}$ be the polynomial in the other variables obtained by setting x to 1, and similarly for $p|_{x=0}$.

Lemma 9: Let x be a variable and $p, p_1, ..p_k, q, q'$ be multilinear polynomials of degree at most d.

1. If $p_1, ..., p_k, x \vdash_d 1$ then $p_1, ..., p_k \vdash_{d+1} 1 - x$.
2. If $p_1, ..., p_k, 1 - x \vdash_d 1$ then $p_1, ..., p_k \vdash_{d+1} x$.
3. $p, x \vdash_d p|_{x=0}$
4. $p, 1 - x \vdash_d p|_{x=1}$.
5. If $p_1, ..., p_k \vdash_d q$ and $p_1, ..., p_k, q \vdash_d q'$, then $p_1, ..., p_k \vdash_d q'$
6. If $p_1|_{x=0}, ..., p_k|_{x=0} \vdash_d 1$ and $p_1|_{x=1}, ..., p_k|_{x=1} \vdash_{d+1} 1$, then $p_1, ..., p_k \vdash_{d+1} 1$.

7. If $p_1|_{x=1}, ..., p_k|_{x=1} \vdash_d 1$ and $p_1|_{x=0}, ..., p_k|_{x=0} \vdash_{d+1} 1$, then $p_1, ..., p_k \vdash_{d+1} 1$.

Proof:

1. Assume $p_1, ..., p_k, x, r_1, ..., r_t, 1$ is a Groebner refutation of degree d of $p_1, ..., p_k, x$. Then $p_1, ..p_k, p_1(1-x), ..., p_k(1-x), x(1-x), r_1(1-x), ..r_t(1-x), 1-x$ is a degree $d{+}1$ Groebner proof from $p_1, ..., p_k$ (since $x(1-x)$ is an axiom.)
2. Similar to the previous claim.
3. Let $p = xp_1 + p_0$, where p_1, p_0 are independent of x. Then $p|_{x=0} = p_0 = p - p_1 x$. So to prove p_0 from p and x, multiply x by each term of p_1 and subtract from p. Since p has degree at most d, p_1 has degree at most $d-1$, so no line will have degree more than d.
4. Similar to the previous claim, $p + p_1(1-x) = p_1 + p_0 = p|_{x=1}$. .
5. Concatenate the two proofs.
6. $p_1|_{x=0}, ..p_k|_{x=0} \vdash_d 1$, so $p_1, ..., p_k, x \vdash_d p_1|_{x=0}, ..., p_k|_{x=0} \vdash_d 1$. Then by claim 1, $p_1, ..., p_k \vdash_{d+1} 1-x$. Similarly, since $p_1|_{x=1}, ..., p_k|_{x=1} \vdash_{d+1} 1$, $p_1, ..., p_k, 1-x \vdash_{d+1} 1$. Then $p_1, ..p_k \vdash_{d+1} 1$ by the previous claim.
7. Similar to the previous claim.

□

It is particularly useful that the last two claims of the lemma are not symmetric with respect to the two degrees required.

Resolution is a proof system whose lines are *or*'s of sets of literals, and whose one inference rule is, from $x \vee (\vee A)$ and $\neg x \vee (\vee B)$, infer $A \vee B$, where A and B are sets of literals, and $\vee S$ represents the or of all elements of S. The graph of a proof has lines of the proof as nodes, and a directed edge from each line to the previous lines used to derive it. (Axioms and initial assumptions are sinks.) A proof is *tree-like* if the graph of the proof is a tree; we may make copies of the original clauses in order to make a proof tree-like. In general, a backtracking algorithm that does not use memoization, such as the Putnam-Davis procedure in most incarnations, will produce tree-like proofs, since it will re-derive results for identical sub-problems on different branches of the program.

The next theorem gives a quasi-polynomial simulation of tree-like resolution proofs by Groebner proofs. This gives the best algorithm we know of to separate sets of clauses with small tree-like resolution refutations from satisfiable clauses. Computing the minimal size of a tree-like resolution refutation of a set of clauses is NP-hard ([IM]). It should be noted that the ideas here can also be used to produce a direct quasi-polynomial time, quasi-polynomial approximation algorithm for this problem. However, it appears that this algorithm will be slower than finding the corresponding Groebner refutation if the size of the minimal tree-like resolution refutation is close to exponential.

Theorem 10: If a set of clauses of size at most k has a tree-like resolution proof with S lines, then the corresponding polynomials have a Groebner refutation of degree $k + \log_2 S\}$. Their translations using new variables have a Groebner refutation of degree $2 + \log_2 S$.

Proof:

We prove the claim by induction on S. Let $C_1, ..C_k$ be the clauses and $p_1, ..., p_k$ be the direct or implicit translations into polynomials. It should be noted that if x is a literal in C_i, then for either kind of translation, $p_i|_{x=0}$ is the translation of $C_i - x$, the clause when x is restricted to be false. Let m be the maximum degree of the p_i, either $m = k$ if we represent the clauses directly or $m = 2$ if we represent them implicitly.

Consider the derivation of the last line, the empty clause, in the refutation. It must be either one of the C_i's, or obtained by resolving x with $\neg x$ for some variable x. If the former , one of the C_i's is empty, and the corresponding $p_i = 1$. This is a degree 0, 1 line Groebner refutation. If the latter, x was derived by a tree-like resolution proof with S_1 lines and $\neg x$ by a tree-like resolution proof with S_2 lines, where $S_1 + S_2 = S - 1$. Then restricting $x = 0$ at all lines of the first sub-proof gives a S_1 line tree-like refutation from the restrictions of the C_i's substituting $x = 0$. Inductively, $p_1|_{x=0}, ..., p_k|_{x=0} \vdash_{m+\log_2 S_1} 1$ and $p_1|_{x=1}, ..., p_k|_{x=1} \vdash_{m+\log_2 S_2} 1$. Assume $S_1 < S/2$. Then applying Claim 6 of Lemma 9, with $d = m + \log_2 S - 1 \geq m + log_2 S_1$, we have $p_1, ..p_k \vdash_{m+\log_2 S} 1$ as desired. The case $S_2 < S_2$ is symmetric, applying Claim 7 instead. □

This gives an $\binom{n}{\leq m+\log_2 S}^{O(1)}$ algorithm for unsatisfiability , where S is the smallest tree-like resolution proof. The above approach (to find a refutation given that there is a tree-like refutation of size S, search through all literals for a refutation of the restricted version aassuming that a tree-like refutation of size $S/2$ exists. If we find a refutation for x, we restrict our clauses by $\neg x$ and recursively search for a refutation assuming that one of size S exists) directly gives an $n^{O(log_2 S)}$ algorithm, This is approximately the same when S is relatively small, but is more than polynomially better when $S = 2^{n^{1-o(1)}}$.

We prove a similar but weaker result for general resolution proofs.

Theorem 11: If a set of clauses of size at most k has a resolution proof with S lines, then the corresponding polynomials have a Groebner refutation of degree $3(n \ln S)^{1/2} + k + 1$. Their translations using new variables have a Groebner refutation of degree $3 + 3(n \ln S)^{1/2}$.

Proof:

Let $b > 0$. Let m be as before. We prove by induction on S' and the number of variables that if there are at most $S' \geq 1$ lines larger than b in the resolution proof, then there is a refutation of degree at most $b + m + 1 - \log_{1-b/2n}(S')$ If there are no such lines, it is easy to see that the proof can be directly simulated with a degree $b+1$ Groebner proof. (Each derivation is sound and involves at most $b+1$ variables; Groebner proofs are a complete system and the maximum degree of a proof is the number of variables.) If there are S' such lines then there must be a literal x in at least $S'b/2n$ lines. Restricting the proof by setting $x=1$ sets all of these lines to true, and only reduces the size of other lines. So there are at most $(1-b/2n)S'$ lines larger than b in the restricted proof. Inductively then, there is a Groebner refutation of $p_1|_{x=1}, ..p_k|_{x=1}$ of degree $b+m+1-\log_{1-b/2n}((1-b/2n)S') = (b+m+1-log_{1-b/2n}(S'))-1$. Also, by induction on the number of variables, $p_1|_{x=0}, ..., p_k|_{x=0}$ has a Groebner refutation of degree $b+m+1-\log_{1-b/2n} S'$. Thus, applying claim 6 or 7 of Lemma 9 depending on whether x is a variable or negated variable, we have a degree $b+m+1-\log_{1-b/2n} S'$ degree refutation of $p_1, ..., p_k$.

The theorem then follows by setting $b=(n \ln S)^{1/2}$ in the inductive claim. □

Again, the above argument can be made into a direct proof search procedure for resolution, but it will be less efficient than using the Groebner proof searching algorithms when S is almost exponential. (In this case, if $S=2^{\epsilon n}$, using our algorithm takes time at most $2^{O(\epsilon^{1/2}\log(1/\epsilon)n)}$, but the direct procedure will take $2^{O(\epsilon^{1/2} n \log n)}$.)

When we use the above simulations to get time bounds for our algorithms, the constant factors in the exponents are enough to make the bounds of no practical importance. However, these simulations do provide a theoretical justification for our belief that the Groebner basis algorithm will do relatively well on those instances of unsatisfiability that resolution methods do well on. We have begun experimental testing to see if this belief holds up for real instances. Experimentally, the time taken by the Groebner basis algorithm seems to heavily depend on the term order and data structures used. It is not clear how or why. For example, we ran [CW]'s Groebner basis program (on a SPARC 20 Model 61) on an unsatisfiable 3-CNF formula which [KS]'s method solved in 5 minutes of CPU time using back-tracking([S]). With good choices for the representation of the problem and the term order, the Groebner program took 10 minutes; other runs, with other representations and term orders, failed to find a refutation after several days. Once we understand how to optimize the Groebner basis algorithm for unsatisfiability problems, we hope to substantially improve the difference in times.

7 Groebner vs. Nullstellensatz

In this section we show:

Theorem 12: There is a refutation on $n(n+1)$ variables that can be proved in the Grobner proof system with degree 3 yet that requires degree n in the Nullstellensatz proof system.

(See [BIK+94] for the definition of the Nullstellennsatz system.)

The separating refutation is a generalization of the pigeon hole principle. We call it the house sitting problem. The problem consists of $n+1$ pigeons indexed by $[0..n]$ and n holes $[1..n]$. Each pigeon i owns hole i, except for pigeon 0 that owns no hole. The holes are ordered in terms of how nice they are. Every pigeon must be in exactly one hole. A pigeon is either home or is house sitting in a hole nicer then its own, i.e. in hole j for some $j \geq i$. If the owner is home, then no other pigeon may be there. On the other hand, if the owner is not home, then any number of pigeons may be there. It is easy using strong induction to see that this is a refutation. Pigeon n must be home because there are no nicer holes to stay in. Now assume by way of induction that pigeons $[i+1..n]$ are all home. Because these owners are home, pigeon i can't stay any of the holes $[i+1..n]$, yet it must stay in one of the holes $[i..n]$. Hence, it too must be home. The conclusion is that pigeon 0 is home, but it does not have a home.

This can be represented by the following polynomials. For each pigeon $i \in [0..n]$ and hole $j \in [1..n]$, let $x_{\langle i,j \rangle}$ indicate that pigeon i is in hole j. (For simplicity, we include $x_{\langle 0,0 \rangle}$ as a variable).

- For each $i \in [0..n]$, let Q_i be : $(\sum_{j \in [i..n]} x_{\langle i,j \rangle}) - 1$. $Q_i = 0$ implies that pigeon i must be in (at least) one hole that is at least as nice as its own.
- For each $i \in [0..n]$ and $j,k \in [1..n]$, let $Q_{\langle i,j,k \rangle}$ be $x_{\langle i,j \rangle} x_{\langle i,k \rangle}$. $Q_{\langle i,j,k \rangle} = 0$ states that pigeon i can't be in both holes.
- For each $i \in [0..n]$ and $j \in [i+1..n]$, let $Q_{\langle i,j \rangle}$ be $x_{\langle i,j \rangle} x_{\langle j,j \rangle}$. $Q_{\langle i,j \rangle} = 0$ states that pigeon i cannot be in hole j if pigeon j is home.
- Let Q be $x_{\langle 0,0 \rangle}$. $Q=0$ states that pigeon 0 is not home.
- For each $i \in [0..n]$ and $j \in [1..n]$, let $Q'_{\langle i,j \rangle}$ be $x^2_{\langle i,j \rangle} - x_{\langle i,j \rangle}$. $Q_{\langle i,j,k \rangle} = 0$ insures that the variable $x_{\langle i,j \rangle}$ takes on $\{0,1\}$ values.

The proof that this set of polynomials cannot all simultaneously be zero in the Grobner proof system parallels the above prove by strong induction.

In reverse order on the value of i, we derive the polynomials $x_{i,j}$ for $j > i$ and $x_{i,i} - 1$, i.e., that the pigeon i is not in any hole other than its home, and hence is home. For $i=n$, $x_{n,n} = Q_n$ is given. Suppose that we have derived the equations $x_{i+1,i+1} - 1, \ldots, x_{n,n} - 1$. For each $j \in [i+1..n]$, derive $x_{\langle i,j \rangle}$ as $-x_{\langle i,j \rangle} \cdot (x_{j,j} - 1) + Q_{\langle i,j \rangle}$ and then derive $x_{i,i}$ as $Q_i - \sum_{j \in [i+1..n]} x_{i,j}$. Eventually we derive $x_{0,0}$, and $Q - x_{0,0}$ gives the desired contradiction. Note no polynomial in the proof has degree higher then two.

The lower bound that the Nullstellensatz proof system requires degree n is of course harder. We use the technique developed in [BCE+95], using linear algebra duality to reduce the issue to the question of the existence of a certain combinatorial design.

Suppose there are polynomials P of degree $d = n - 1$ so that $\sum_{i\in[0..n]} P_i Q_i + \sum_{i\in[0..n],j,k\in[1..n]} P_{\langle i,j,k\rangle} Q_{\langle i,j,k\rangle} + \sum_{i\in[0..n],j\in[i+1..n]} P_{\langle i,j\rangle} Q_{\langle i,j\rangle} + PQ + \sum_{i\in[0..n],j\in[1..n]} P'_{\langle i,j\rangle} Q'_{\langle i,j\rangle} = 1$. The first thing to observe is that this statement is equivalent to $\sum_{i\in[0..n]} P_i Q_i \equiv 1 (\mathrm{mod}\ Q_{\langle i,j,k\rangle}, Q_{\langle i,j\rangle}, Q, Q'_{\langle i,j\rangle})$. From now on all our algebra will be polynomials modulo $(x_{\langle i,j\rangle} x_{\langle i,k\rangle},\ x_{\langle i,j\rangle} x_{\langle j,j\rangle},\ x_{\langle 0,0\rangle},\ x^2_{\langle i,j\rangle} - x_{\langle i,j\rangle})$ with GF[2] as the co-efficient field. Given this modulus, we may assume that if $X_M = \prod_{\langle i',j'\rangle\in M} x_{i',j'}$ is a monomial in one of the polynomials P_i, then M is a *partial legal mapping from pigeons to holes*. Specifically, M maps a subset $dom(M) \subset [0..n]$ of the pigeons to holes with the requirements that each mapped pigeon is mapped to hole that is at least as nice as his own home and no other pigeon is mapped to a hole if the owner is home. Let the coefficient in P_i of the monomial X_M corresponding to matching M be a^i_M.

DEFINITION 7.1: Matching M *matches* i if $\langle i,j\rangle \in M$ for some $j \in [1,N]$. We write this formally as $i \in M$. If $i \in M$, we write $M - i$ for the matching $M - \{\langle i,j\rangle\}$ where j is the unique value such that $\langle i,j\rangle \in M$.

One useful observation is that we can assume that $a^i_M = 0$ if $i \in M$, since for $M = \{\langle i,k\rangle\} \cup (M - i)$ and $X_M = X_{M-i} x_{i,k}$, we have $X_M \cdot Q_i \equiv 0$.

We can view the equation $\sum_{i\in[0..n]} P_i Q_i = \sum_{i\in[0..n]} P_i[(\sum_{j\in[i..n]} x_{\langle i,j\rangle}) - 1] \equiv 1$ as a set of restrictions on the coefficients a^i_M of P_i. For each legal partial mapping M, we obtain a constraint by equating the coefficients for the monomial X_M on the two sides of the equation. These constraints are:

(1) $-\sum_{i\in[1,N+s]} a^i_\emptyset = 1$

(2) $\sum_{i\in M} a^i_{M-i} - \sum_{i\notin M} a^i_M = 0$, for all $M \neq \emptyset$.

That the polynomials are of maximum degree $d = n-1$ adds the constraints:

(3) $a^i_M = 0$, for $|M| \geq d+1 = n$.

We will now show that the above system of equations (1)-(3) has a solution over GF[2] if and only if there does not exist a particular combinatorial design.

DEFINITION 7.2: For each subset of pigeons S, let $\mathcal{M}_S$ denote a collection of legal matchings where each of the matchings $M \in M_S$ matches only those pigeons in S. Then $\mathcal{M}_S \bigoplus \mathcal{M}'_S$ is defined to be the collection of matches that appear an odd number of times in $\mathcal{M}_S \cup \mathcal{M}'_S$. For $i \in S$, define $\mathcal{M}_S - i$ to be the collection of matchings that is formed by removing pigeon i from each of the matchings in $\mathcal{M}_S$ and then keeping those matchings (on pigeons $S - i$) that appear an odd number of times. More formally $\mathcal{M}_S - i = \bigoplus_{M\in\mathcal{M}} \{M - i\}$.

DEFINITION 7.3: A *d-design* is a collection of legal matchings, $\mathcal{M} = \cup_{S\subset[0..n],|S|\leq d+1} \mathcal{M}_S$, such that each matching in $\mathcal{M}$ has size at most $d+1$ and such that the following conditions hold.

(a) The empty matching $M = \phi$ is in $\mathcal{M}$.

(b) For any $S \subset [0..n]$, $|S| \leq d+1$, and $i \in S$, the collections satisfy $\mathcal{M}_{S-i} = \mathcal{M}_S - i$.

Lemma 13: If there is an n-design then equations (1)-(3) have no solution over GF[2].

Proof: Suppose we have an n-design $\mathcal{M}$ and a solution for (1)-(3). We view the matchings $M \in \mathcal{M}$ as selecting a subset of the equations in (1)-(3), since there is one equation for each matching. We consider the GF[2] sum of the selected equations. Condition (a) in the definition of an n-design requires that equation (1) is selected so the right-hand side of the sum is 1.

We will show that condition (b) in the definition of an n-design implies that the left-hand side of this sum is 0 which is a contradiction. Consider the coefficient of a^i_M in the sum. It occurs once (with coefficient -1) if $M \in \mathcal{M}$. It also occurs once (with coefficient +1) for each j such that $M \cup \{\langle i,j\rangle\} \in \mathcal{M}$. In other words, there is a contribution of -1 if $M \in \mathcal{M}_S$ and a contribution of +1 if there are an odd number of j such that $M \cup \{\langle i,j\rangle\} \in \mathcal{M}_{S\cup\{i\}}$. The latter is true if and only if $M \in \mathcal{M}_{S\cup\{i\}} - i$. By condition (b) of the definition of an n-design, $\mathcal{M}_S = \mathcal{M}_{S\cup i} - i$ so the net coefficient of a^i_M is 0. □

What remains is to construct such a design.

Lemma 14: There exists an n-design.

The proof will be given in the journal version of the paper.

8 Future directions

We do not know of any non-trivial lower bounds for the degrees of Groebner proofs. Besides giving insight into which are the hard instances of unsatisfiability for our methods, such a lower bound might have consequences for traditional proof systems. Pitassi [P] has pointed out that a lower bound for Groebner proofs for one of the matching principles where a switching lemma is known would yield a corresponding lower bound for Frege proofs whose formulas are parities of AC_0 formulas.

If the (non-onto) pigeonhole-principle had a constant or low degree Groebner proof, then this would give a new method for solving bipartite mathching and for separating k-colorable graphs from graphs with a $k+1$-clique. Lovasz [L] has such an algorithm using semi-definite programming, but it would be interesting to see if the Groebner basis algorithm could also be used for this purpose.

Can our simulations of resolution be used to show that there is no resolution proof that exponentially hard one-way functions exist? Such a result might be possible using the "natural proofs" technique of Razbarov and Rudich, and might be usable in showing that the existence of one-way functions is not provable in certain constructive fragments of arithmetic ([RR]).

Clegg and Impagliazzo [CI] have observed that the Nullstellensatz degree of a system of polynomials is characterized by a Groebner basis for homogenized versions of the polynomials. This gives a simpler technique for proving Nullstellensatz lower bounds, and raises questions about the efficiency of the commonly used heuristic of homogenizing polynomials before running the Groebner basis algorithm.

References

[BIK+94] P. Beame, R. Impagliazzo, J. Krajicek, T. Pitassi, and P. Pudlak Lower bounds on Hilbert's Nullstellensatz and propositional proofs In *35th Annual Symposium on Foundations of Computer Science*, pages 794–806, Santa Fe, November 1994.

[BCE+95] P. Beame, S. Cook,J. Edmonds, R. Impagliazzo, T. Pitassi, The Relativized Complexity of Search Problems, Proc. of 27'th STOC, 1995, pp. 303-314.

[B65] B. Buchberger, Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal, (An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal). Doctoral Dissertation Math. Inst. University of Innsbruck, Austria.

[B79] B. Buchberger, A criterion for detecting unnecessary reductions in the construction of Gröbner bases. In: Ng, E.W. (ed.), *EUROSAM '79, An International Symposium on Symbolic and Algebraic Manipulation*, Springer LNCS 72, 3-21.

[BP] S. Buss and T. Pitassi, to appear in 11'th Computational Complexity, 1996.

[CI] M. Clegg and R. Impagliazzo. "Homogenization and Nullstellensatz Degrees". In preparation.

[CW] M. Clegg and N. Wallach, "Groebner: A Commutative Algebra Package for Distributed Networks of Workstations". In preparation.

[CLO] D. Cox, J. Little, D. O'Shea, *Ideals, Varieties and Algorithms*, Springer, 1992.

[IM] K. Iwana, and E. Miyano. Intractability of Read-Once Resolution, 10'th Structures, 1995. pp. 29-36.

[KS] S. Kirkpatrick and B. Selman, Critical Behavior in the Satisfiability of Random Boolean Expressions. *Science*, Vol. 264, May 1994, 1297–1301.

[L] L. Lovasz, On the Shannon Capacity of a graph, Transactions on Information Theory, vol. 25, No. 1, January, 1979

[P] T. Pitassi, personal communication, 1995.

[RR] A. Razbarov and S. Rudich, Natural Proofs, Proc. of 26th STOC, 1994, pp. 204-213.

[S] B. Selman, personal communication, 1995.