Algebraic models of computation and interpolation for algebraic proof systems

Pavel Pudlák and Jiří Sgall

1. Introduction

We consider some algebraic models used in circuit complexity theory and in the study of the complexity of the propositional calculus. This direction of research has been getting attention recently with the hope that the connection to well-developed fields of mathematics like algebra can be helpful in proving lower bounds.

Span programs as a model of computation were introduced in [**13**]. A span program is a device for defining boolean functions, where the function is defined to be 1 iff a fixed vector can be expressed as a linear combination of vectors chosen by the input. Span programs polynomially simulate branching programs (for finite fields they are equivalent to counting branching programs), thus an exponential lower bound on the size of span programs computing a concrete boolean function would solve a major open problem. An important subclass of span programs are monotone span programs; they simulate both monotone formulas and monotone contact switching networks. Recently a superpolynomial lower bound was proved for this model [**1, 2**] (based on a combinatorial condition of [**4**]).

One direction of study of propositional proof systems is to prove lower bounds on the length of proofs in certain restricted proof systems. Exponential lower bounds were obtained for such systems as resolution [**12**], bounded depth Frege systems [**15, 17**], cutting planes [**6, 18**], and Nullstellensatz refutations [**3, 8**]. In this paper we are interested in systems that use uses polynomials instead of boolean formulas. From the previous list this includes the Nullstellensatz refutations. Recently a stronger system using polynomials was proposed, the polynomial calculus, also called the Groebner calculus [**9**].

The proof systems form a similar hierarchy as the complexity classes or classes of circuits in the computational complexity, but there is no direct relation between the two hierarchies. Recently a new method was found which makes it possible to prove lower bounds on the length of proofs for some propositional proof systems using lower bounds on circuit complexity. This method is based on proving computationally efficient versions of Craig's interpolation theorem for the proof system in question [**14, 18**]. For appropriate tautologies the interpolation theorem

1991 *Mathematics Subject Classification.* Primary 68Q99; Secondary 68Q05, 03B05.

This work was partially supported by grant A1019602 of AV ČR, by grant 93025 of the US-Czechoslovak Science and Technology Program and by cooperative research grant INT-9600919/ME-103 from the NSF (USA) and the MŠMT (Czech republic).

may lead to a monotone model of computation, which makes it possible to use available lower bounds for monotone models of computation to prove lower bounds for proof systems. Such interpolation theorem yields a simple proof of exponential lower bounds for resolution using known lower bounds for monotone boolean circuits [**18**]. It was also applied to yield an exponential lower bound for unrestricted cutting planes [**18**], in which case the proof is based on the new bounds for monotone real circuit [**11, 18**].

The main result of this paper is an interpolation theorem for Nullstellensatz refutations. We prove that the interpolants can be computed by span programs, and the span programs are monotone for suitable tautologies. Moreover, this characterization of interpolants is tight, namely every span program is a unique interpolant for some tautology provable by the Nullstellensatz system. In principle, this interpolation theorem can be used for proving lower bounds on Nullstellensatz proofs using lower bounds on monotone span program complexity, but due to the difficulty of proving lower bounds for monotone span programs known direct proofs are simpler and yield better bounds.

We introduce a new computational model, polynomial programs, which bounds the complexity of the interpolants in polynomial calculus in a similar way as span programs do in case of the Nullstellensatz system. The general version of polynomial programs over finite fields is equivalent to boolean circuits. The monotone version simulates both monotone boolean circuits and monotone span programs, thus it is a very strong monotone computational model. Unfortunately the lower bound technique for monotone span programs does not extend to monotone polynomial programs. Our original motivation for studying this model was to get non-trivial lower bounds for the polynomial calculus. During the refereeing process Razborov proved such a bound without using interpolation theorems [**21**]. However, lower bounds for polynomial programs would still be interesting.

We introduce yet another model of computation, dependency programs. These programs are similar to span programs but only linear dependence of chosen vectors is tested, instead of testing whether their span contains some vector. A communication complexity version of this model was studied in [**19**] as the projective dimension of graphs (a concept also related to the affine dimension of graphs of [**20**]). We prove an exponential lower bound for monotone dependency programs, using a simplification of the methods used for monotone span programs in [**1, 4**]. In the non-monotone case over finite fields the dependency programs turn out to be equivalent to span programs. However, they may be useful to consider in lower bound proofs, as they are in some sense simpler (as demonstrated by the monotone lower bound, which is much simpler than the analogous bound for span programs).

Finally, we investigate the closure properties and relations among the algebraic models of computation. We give simple constructions for some closure properties which in some cases generalize the known results; for example we show that span programs are closed under NC^1-reductions for arbitrary fields. One interesting open question is the relative strength of these models of computation when the underlying field is changed. We note that the most straightforward attempt to convert a span program over $\mathbf{R}$ into a span program over $\mathbf{Q}$ fails, since there exist matroids representable over $\mathbf{R}$ but not over $\mathbf{Q}$.

We start by the definitions in Section 2 and the properties of the computational models in Section 3. The main results on interpolation are proved in Section 4. The result on matroids is given in Section 5.

2. Definitions

We assume that a field K is fixed. Most often we consider the finite fields $GF(p)$ with p elements for p prime, or rational numbers $\mathbf{Q}$ or reals $\mathbf{R}$.

As usual, for a matrix $\mathbf{A}$, a_{ij} denotes its entry in ith row and jth column. The column vectors are denoted by $\vec{u} = (u_1, \dots, u_k)^\top$, etc. The dot product of vectors $\vec{u}$ and $\vec{v}$ is $\vec{u}^\top \vec{v}$. Some special vectors are denoted by $\vec{0} = (0, \dots, 0)^\top$, $\vec{1} = (1, \dots, 1)^\top$, and $\vec{e} = (1, 0, \dots, 0)^\top$. The unit matrix is denoted by $\mathbf{I}$ and the matrix with all entries zero is $\mathbf{0}$. The length of these vectors and the dimensions of the matrices will always be clear from the context.

2.1. Algebraic proof systems. A *Nullstellensatz refutation* (shortly *NS refutation*) of a set of polynomial equations $p_1 = 0, \dots, p_m = 0$ is given by a system of polynomials $q_1, \dots, q_m$ such that $\sum q_i p_i = 1$. The degree of such a refutation is $d = \max_i deg(q_i p_i)$.

This refutation system is complete if we look for solutions in an algebraically closed field (by the Hilbert Nullstellensatz). Here we are interested only in 0-1 solutions. To get the completeness for such solutions, we shall assume that for each used variable x_k the equation $x_k^2 - x_k = 0$ is present among $p_1 = 0, \dots, p_m = 0$.

A *polynomial refutation* of a set of polynomial equations $p_1 = 0, \dots, p_m = 0$ is sequence of polynomials $q_1, \dots, q_k$ such that q_k is the constant 1 and each q_i is either some p_j, or a linear combination of the polynomials $q_1, \dots, q_{i-1}$, or $x_t q_j$ for some $j < i$ and some variable x_t. The degree of the refutation is $d = \max_i deg(q_i)$.

2.2. Algebraic models of computation. For our purposes *literals* are denoted by x_i or $1 - x_i$, the latter corresponding to the negation of a variable. A *labelled matrix* is a matrix such that each row is labelled by some literal. Given a labelled matrix $\mathbf{A}$ and a truth assignment $\vec{u}$, we define $\mathbf{A}(\vec{u})$ to be the matrix consisting only of those rows of $\mathbf{A}$ labelled by literals that are true in the assignment $\vec{u}$ (i.e., either the row is labelled by x_i and $u_i = 1$, or the row is labelled by $1 - x_i$ and $u_i = 0$ in the assignment).

A *dependency program* is given by a labelled matrix $\mathbf{A}$. Its value on the assignment $\vec{u}$ is defined to be 1 if the rows of $\mathbf{A}(\vec{u})$ are linearly dependent, and 0 otherwise. The size of a dependency program is the number of columns of the matrix.

A *span program* is given by a labelled matrix $\mathbf{A}$. Its value on the assignment $\vec{u}$ is defined to be 1 if the vector $\vec{e}^\top = (1, 0, \dots, 0)$ is a linear combination of the rows of $\mathbf{A}(\vec{u})$, and 0 otherwise. The size of a span program is the number of columns of the matrix. Note that nothing changes if we choose any non-zero vector instead of $(1, 0, \dots, 0)$, as we can transform the basis of the vector space.

A *polynomial program of degree* d is given by a set of polynomials in $K[y_1, \dots, y_m]$ ($y_1, \dots, y_m$ are new formal variables), where each polynomial is labelled by a literal (x_i or $1 - x_i$). Its value on an assignment $\vec{u}$ is defined to be 1 if there exists a polynomial refutation of degree d from all those vectors that are labelled by literals that are true in the assignment $\vec{u}$. The size of a polynomial program is the number of monomials of $K[y_1, \dots, y_m]$ of degree at most d, which is equal to $\sum_{i=0}^{d} \binom{m}{d}$. In particular, if d is a constant, the size is polynomial in the number of variables m.

A dependency, span, or polynomial program is *monotone*, if all the labels (of the vectors or polynomials) are positive literals. Clearly, the monotone programs compute only monotone functions. Using these definitions it would be impossible to compute the constant 1 function in any of the monotone models. For that reason we augment our definitions so that the empty program is defined to compute the constant 1 in any of the models, and it is considered to be monotone. (An equivalent definition would be to allow also 1 as a label of a row, not only a literal. See Section 3.2 for more discussion of this.)

The minimal size of a dependency, span, or polynomial program computing a function f is denoted $DP_K(f)$, $SP_K(f)$, or $PP_{d,K}(f)$. The monotone variants are denoted $mDP_K(f)$, etc. The index K is omitted if the field is clear from the context; also for finite fields we write $SP_p(f)$ if the field is $GF(p)$. Note that there is some ambiguity in the literature regarding to the measure of size: most of the previous papers measure the number of rows instead of the number of columns. As discussed in Section 3.2, this is not essential for span or dependency programs, as these two measures differ at most by a factor of n. However, for polynomial programs the number of columns can be significantly larger, and hence it is a more appropriate measure.

If the arithmetic operation in the given field can be implemented efficiently (which is true of all finite fields), the dependency and span programs are efficient procedures, as their value can be found using the Gaussian elimination. An efficient decision procedure for polynomial programs follows from [**9**]; it uses the Groebner basis algorithm which generalizes the Gaussian elimination appropriately.

The relations among these models and some other variants will be studied in Section 3.2. It is easy to see that the models are increasingly more powerful. In the non-monotone case the polynomial programs over any finite field are polynomially equivalent to boolean circuits, and the dependency programs are polynomially equivalent to the span programs over the same field $GF(p)$. However, monotone span programs are exponentially stronger than monotone dependency programs.

3. Relations among the algebraic models of computation

3.1. An exponential lower bound for monotone dependency programs. This bound is based on the ideas of the papers [**1, 4**], which prove superpolynomial lower bounds for monotone span programs. Using their methods, we are able to prove an exponential lower bound on the size of monotone dependency programs for a very simple function.

THEOREM 3.1. *Let* $f = (x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge \ldots \wedge (x_{2n-1} \vee x_{2n})$. *Then* $mDP(f) \geq 2^n/n$, *for an arbitrary field.*

We start by a lemma about minterms of any function computed by a monotone dependency program. A *minterm* of a function as is an assignment $\vec{u}$ such that $f(\vec{u}) = 1$ and for any $\vec{v} \leq \vec{u}$, $f(\vec{v}) = 0$ or $\vec{v} = \vec{u}$.

LEMMA 3.2. *Suppose that a boolean function f is computed by a monotone dependency program* $\mathbf{A}$ *with the number of rows smaller than the number of minterms of f. Then there exists a set of minterms U, $|U| \geq 2$, such that for any non-trivial partition $U = V \cup W$, $V, W \neq \emptyset$, $V \cap W = \emptyset$, there exists a minterm $\vec{u}$ of f such that for every i, $u_i = 1$ implies that both $(\exists \vec{v} \in V) v_i = 1$ and $(\exists \vec{w} \in W) w_i = 1$.*

PROOF. For every minterm $\vec{u}$ of f chose some linear dependence $\vec{c}^{\vec{u}}$ of the rows of $\mathbf{A}$ consistent with the labels of $\mathbf{A}$, i.e., $\vec{c}^{\vec{u}}$ is a vector such that $(\vec{c}^{\vec{u}})^\top \mathbf{A} = 0$, and if $c_j^{\vec{u}} \neq 0$ and the jth row of $\mathbf{A}$ is labelled by x_i then $u_i = 1$. Such $\vec{c}^{\vec{u}}$ exists since $\vec{u}$ is accepted by $\mathbf{A}$.

The vectors $\vec{c}^{\vec{u}}$ are linearly dependent, as their length is the number of rows which is smaller than the number of minterms u. Let U be a minimal set of minterms such that $\{\vec{c}^{\vec{u}} \mid \vec{u} \in U\}$ is linearly dependent. Thus $\sum_{\vec{u} \in U} \alpha_{\vec{u}} \vec{c}^{\vec{u}} = \vec{0}$ for some $\alpha_{\vec{u}} \neq 0$. Now for any nontrivial partition $\vec{c} = \sum_{\vec{u} \in V} \alpha_{\vec{u}} \vec{c}^{\vec{u}} = -\sum_{\vec{u} \in W} \alpha_{\vec{u}} \vec{c}^{\vec{u}} \neq \vec{0}$. Define $u'_i = 1$ if there exist j such that $c_j \neq 0$ and jth row of A is labeled by x_i; let $\vec{u}$ be any minterm smaller than or equal to $\vec{u}'$ (i.e., $u_i \leq u'_i$ for all i). This $\vec{u}$ satisfies the conditions in the statement of the lemma. □

PROOF. (of Theorem 3.1.) An assignment $\vec{u}$ is a minterm of f from the statement of the theorem iff $u_{2i-1} + u_{2i} = 1$ for all $i = 1, \ldots, n$; there are 2^n minterms.

Suppose that f is computed by a monotone dependency program of size less than $2^n/n$. Then it is computed also by a monotone dependency program with less than 2^n rows (since all the rows labelled by the same x_i are either independent, or can be replaced by a single vector $\vec{0}$, cf. Section 3.2).

Let U be the set of minterms guaranteed by the lemma. Pick i and $\vec{v}, \vec{w} \in U$ such that $v_{2i-1} = 0$ and $w_{2i} = 0$; these exists because $|U| \geq 2$ and because of the particular structure of the minterms. Now set $V = \{\vec{v} \in U \mid v_{2i-1} = 0\}$ and $W = \{\vec{w} \in U \mid w_{2i} = 0\}$. Let $\vec{u}$ be a minterm guaranteed by the lemma. By the definition of V and W it follows that $u_{2i-1} = u_{2i} = 0$, which is impossible for a minterm, a contradiction. Hence f has no small monotone dependency programs. □

3.2. Closure properties and some variants of the definitions. First we prove that our models are closed under restrictions. For span programs this was noticed already in [**13**], using a more complicated argument.

LEMMA 3.3. *If g is a restriction of a function f, then $mDP(g) \leq mDP(f)$ for an arbitrary field K; similarly for span and polynomial programs and also for the non-monotone versions.*

PROOF. Suppose that x_i is assigned a constant, hence some rows (or polynomials for polynomial programs) are now labelled by 0 or 1 instead of a literal. We remove all rows labelled by 0.

For span and polynomial programs we replace each row labelled by 1 by multiple copies labelled by all the possible literals, using only monotone literals in the monotone case. The new program computes the restriction except for the case when it is the constant 1 function, which is by definition computed by the empty program. Monotonicity of the program is preserved and the size does not increase.

For dependency programs for each row labelled by 1 we change the basis so that it is the first basis vector and remove the first column of the program. This does not change the computed function, as the row can be used on any input and hence anything in the first column can be cancelled. If the row is $\vec{0}$, we cannot perform the previous transformation; however, the computed function is the constant 1 function, which is by definition computed by the empty program. Monotonicity of the program is preserved and the size does not increase. □

There are several variations in the definitions we can make. First, we can allow a row to be labelled by 1 instead of a literal, with the meaning that it can be used for any input. This is essentially the same as taking a restriction, hence by previous lemma it does not change the size of the program. We will use this generalization in our constructions.

Second, we can measure the number of rows instead of the number of columns. For a minimal program, the number of rows is larger than the number of columns by at most a factor of $2n$, as we can take all the rows labelled by the same literal linearly independent. For the dependency or span programs even the number of rows does not increase if we take a restriction of the function; for the dependency programs it follows from the argument in the lemma, and for span programs it is possible to use a similar argument, too. By the same argument the factor between the number of rows and columns can be tightened from $2n$ to n. Also, for dependency and span programs the number of rows is always at least the number of columns, as we can work in the span of all the rows.

Third, it is possible to consider a more general variant of span programs where we have not one target vector but a vector subspace, with the zero vector excluded. If the target vectors are the first k basis vectors and the generalized span program is given by $(\vec{a}^1 \cdots \vec{a}^k\, \mathbf{A})$ (the first k columns of the matrix written separately), then it computes the same function as the disjunction of the k span programs $(\vec{a}^1\, \mathbf{A}),\ldots,$ $(\vec{a}^k\, \mathbf{A})$. In the next lemma we will prove that span programs are closed under disjunctions, hence the size of the usual span program is at most the square of the size of the generalized one.

Now we prove that span programs are closed under NC^1-reductions and monotone span programs are closed under monotone NC^1-reductions. This was known for non-monotone span programs over $GF(p)$, due to their equivalence to counting branching programs [**7, 13**], see also Section 3.3. Our proof is direct and works for an arbitrary field.[1]

We say that a function f is NC^1-reducible to a function g if there exists a family of circuits of depth $O(\log n)$ computing f with gates for NOT, OR and AND of fan-in two, and gates for g; the gates for g of fan-in k count as depth $\log k$. The reduction is monotone if there are no NOT gates in the circuits. We consider non-uniform reductions, since we consider non-uniform models of computation, unlike e.g. [**7, 10**].

THEOREM 3.4. *For arbitrary field K and all boolean functions f, g, g_1, $\ldots$, g_k,*

- $mSP(f \vee g) \leq mSP(f) + mSP(g) - 1$, $SP(f \vee g) \leq SP(f) + SP(g) - 1$,
- $mSP(f \wedge g) \leq mSP(f) + mSP(g)$, $SP(f \wedge g) \leq SP(f) + SP(g)$,
- $SP(\neg f) \leq n \cdot SP(\neg f)$, *where n is the arity of f,*
- $mSP(f(g_1, \ldots, g_k)) \leq mSP(f)(mSP(g_1) + \cdots + mSP(g_k))$.

Consequently, if f is (monotone) NC^1-reducible to g and g has a (monotone) span program of polynomial size, f has a (monotone) span program of polynomial size.

PROOF. Suppose that we have two span programs $(\vec{a}\,\mathbf{A})$ and $(\vec{b}\,\mathbf{B})$ ($\vec{a}$ and $\vec{b}$ are their first columns). Then the span program

$$\begin{pmatrix} \vec{a} & \mathbf{A} & \mathbf{0} \\ \vec{b} & \mathbf{0} & \mathbf{B} \end{pmatrix}$$

[1] It was pointed out by the referee that similar constructions are known for at least the first three properties, but were not published before.

with the same labels computes their disjunction. The span program

$$\begin{pmatrix} \vec{a} & \vec{0} & \mathbf{A} & \mathbf{0} \\ \vec{0} & \vec{b} & \mathbf{0} & \mathbf{B} \end{pmatrix}$$

with the same labels computes their conjunction, if the target vector is $(1, 1, 0, \ldots, 0)$ instead of $(1, 0, \ldots, 0)$; the target vector can be changed by a linear transformation of the matrix.

Now we construct a span program which computes $\neg f$ given a span program $\mathbf{A}$ for f. Let $\vec{u}$ be an input for $\mathbf{A}$. Consider the matrix

$$\overline{\mathbf{A}(\vec{u})} = \begin{pmatrix} \vec{e} & \mathbf{A}(\vec{u})^\top \end{pmatrix}$$

By the linear programming duality, $\mathbf{A}$ does not accept $\vec{u}$ iff $(1, 0, \ldots, 0)$ is in the span of the rows of $\overline{\mathbf{A}(\vec{u})}$. It remains to construct a span program $\mathbf{B}$ such that $\mathbf{B}(\vec{u})$ behaves similarly as $\overline{\mathbf{A}(\vec{u})}$. Consider

$$\mathbf{B} = \begin{pmatrix} \vec{0} & \mathbf{I} \\ \vec{e} & \mathbf{A}^\top \end{pmatrix} \tag{1}$$

where the jth row is labelled by the negation of the label of jth row in $\mathbf{A}$, the remaining rows (corresponding to the rows of $\mathbf{A}^\top$) are labelled by 1. If the jth row in $\mathbf{A}$ is labelled by a literal which is assigned 0, the $j$the row of $\mathbf{B}$ is labelled by a literal which is assigned 1, and hence this basis vector can be used to cancel any number appearing in this column, which has the same effect as deleting this column in $\overline{\mathbf{A}(\vec{u})}$. Thus $\mathbf{B}(\vec{u})$ behaves equivalently to $\overline{\mathbf{A}(\vec{u})}$ and $\mathbf{B}$ computes the negation of the function computed by $\mathbf{A}$. The size of $\mathbf{B}$ is at most 1 larger than the number of rows of $\mathbf{A}$, which can be bounded by n times the size of $\mathbf{A}$.

Suppose that $f, g_1 \ldots g_k$ are computed by monotone span programs $\mathbf{A}, \mathbf{B}_1 \ldots \mathbf{B}_k$, and suppose that jth row of A is labelled by x_{i_j}. Then we claim that $f(g_1, \ldots, g_k)$ is computed by the span program

$$\begin{pmatrix} \mathbf{A} & \mathbf{I}_1 & \mathbf{I}_2 & \cdots \\ \mathbf{0} & \mathbf{B}_{i_1} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_{i_2} & \\ \vdots & & & \ddots \end{pmatrix}$$

where the rows corresponding to $\mathbf{A}$ are labelled all by 1 and the other rows are labelled as in the corresponding $\mathbf{B}_i$; $\mathbf{I}_j$ is the matrix which has a single 1 in the jth row and 1st column, all other entries are 0. If this span program accepts, it means that $(1, 0, \ldots, 0)$ is in a span of some rows of $\mathbf{A}$; moreover if jth row is used, the vector $(1, 0, \ldots, 0)$ from $\mathbf{I}_j$ must cancel with some other rows, which is possible only if $\mathbf{B}_{i_j}$ accepts. The other implication is easy as well.

To conclude that span programs of polynomial size are closed under NC^1-reducibility, note that in the circuits giving the reduction we can assume that NOT gates are either at the leaves or at the output of a gate for g; in that case we can substitute the span program for $\neg g$. $\square$

As we shall see in Section 3.3, over finite fields $GF(p)$, dependency and span programs are equivalent, and hence have the same closure properties. For monotone dependency programs and dependency programs over general field we can only prove that they are closed under disjunction. We know that monotone dependency programs are not closed under conjunction, due to Theorem 3.1, where we proved an exponential lower bound for a function which is a conjunction of polynomially many functions with constant size monotone dependency programs.

LEMMA 3.5. *For arbitrary field K and all boolean functions f and g, g_1, …, g_m,*

- $mDP(f \vee g) \leq mDP(f) + mDP(g)$, *and*
- $DP(f \vee g) \leq DP(f) + DP(g)$.

PROOF. Suppose we have two dependency programs **A** and **B**. Then the dependency program

$$\begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix},$$

with the rows labelled as in A and B, computes their disjunction. This reduction preserves monotonicity. □

3.3. Dependency programs vs. span programs. In this section we compare the power of dependency and span programs. An easy reduction shows that span programs are at least as strong as dependency programs. In the monotone case, by Theorem 3.1 it follows that there is an exponential gap, as the function for which we proved an exponential lower bound for monotone dependency programs has linear-size span programs by Theorem 3.4. On the other hand, we prove that over finite fields $GF(p)$, non-monotone dependency programs are as strong as span programs.

LEMMA 3.6. *For an arbitrary field K and an arbitrary boolean function f, $mSP(f) \leq n \cdot (mDP(f))^2 + 1$ and $SP(f) \leq n \cdot (DP(f))^2 + 1$.*

PROOF. Let **A** be a dependency program, let k be the number of its rows. By Section 3.2 we may assume that $k \leq n \cdot DP(f)$. Then the function f is computed by the span program

$$\begin{pmatrix} \vec{e}_1 & \mathbf{A} & \mathbf{0} & \cdots & \mathbf{0} \\ \vec{e}_2 & \mathbf{0} & \mathbf{A} & & \mathbf{0} \\ \vdots & \vdots & & \ddots & \vdots \\ \vec{e}_k & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A} \end{pmatrix}$$

with all rows labelled as in **A**, where $\vec{e}_i$ denotes the ith vector of the standard basis. Monotonicity is preserved, and the size is as claimed. If an input is accepted by the dependency program, let i be an index of a row which appears in the non-trivial linear combination which sums to zero; then the same linear combination of the rows in the ith block is a non-zero multiple of the target vector. Conversely, if the target vector is a linear combination of the rows of the span program, take any block i containing a row with non-zero coefficient; the linear combination of the rows in this block gives a non-trivial linear dependency among the rows of **A**. □

THEOREM 3.7. *For any prime p and boolean function f, $DP_p(f) \leq (SP_p(f))^{O(1)}$.*

PROOF. In [**13**] it is observed that the size of a span program over $GF(p)$ for f is polynomially related to the size of a branching program counting modulo p computing f. (This equivalence is based on the results of [**7**] which proves that counting branching programs can perform rank computations, based on the results of [**5, 16**].)

Branching program is a directed acyclic graph with the edges labelled by literals or 1. A counting branching program accepts on a given assignment, if the number of accepting paths from the source to the sink is divisible by p; a path is accepting if all its edges are labelled by a true literal or 1.

We need to show that any function computed by a counting branching program can be computed by a dependency program with only polynomially larger size. We can represent a branching program by an adjacency matrix $\mathbf{A}$ with entries 1, x_i, and $1-x_i$, according to the labelling of the edges. The matrix is upper triangular, assuming that the vertices are ordered topologically from source to sink. Let $\mathbf{B}$ be the adjacency matrix modified in the following way: put 1 in all diagonal entries and remove the first column and the last row. In [**7**] it is shown that the number of accepting paths is equal to the determinant of $\mathbf{B}$, up to a possible sign change. Hence it is sufficient to construct a dependency program that decides if the rows of $\mathbf{B}$ are linearly dependent (it will compute the negation of the function, but this does not matter since span programs are closed under negation).

Suppose that $\mathbf{B}$ has size $k \times k$. We construct a dependency program with $k(k+1)$ columns computing 1 iff the rows of $\mathbf{B}$ are dependent. Consider a row $(b_{j1},\dots,b_{jk})$; note that all the entries are literals, or constants 0 or 1. We replace this row by the matrix

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \vec{0}^\top & \vec{0}^\top & \cdots & \vec{0}^\top & \vec{1}^\top & \vec{0}^\top & \cdots & \vec{0}^\top \end{pmatrix}$$

where $\mathbf{I}$ is $k \times k$ identity matrix, and the vector $\vec{1}^\top$ is in the $(j+1)$st block of the matrix, i.e. in a unique position for each row of $\mathbf{B}$. The last row is labelled by 1, for $j \le k$ the jth row is labelled by c_j and the $(j+k)$th row is labelled by $1-c_j$. Thus all rows are labelled by a literal or 0 or 1 (naturally, in the final stage we remove all rows labelled by constants, as in the proof that the dependency programs are closed under restrictions). Since the position of the vector $\vec{1}^\top$ is unique for each row of $\mathbf{B}$, in any linear dependence the ones in these columns must cancel within the rows corresponding to the same row of $\mathbf{B}$. It is easy to verify that the only vectors that can be obtained as a linear combination of the rows corresponding to the same rows of $\mathbf{B}$ and have zeroes in these positions necessarily have in the first block some multiple of the row of $\mathbf{B}$ $(b_{j1},\dots,b_{jk})$. It follows that the dependency program computes 1 iff the rows of $\mathbf{B}$ are linearly dependent (over the same field $GF(p)$). □

3.4. Polynomial programs. Polynomial programs can also be represented by a matrix, where each row is a vector of the coefficients of a polynomial at all the monomials of degree at most d. In this representation their size exactly corresponds to the size of span programs. It follows that each polynomial program of degree 1 is a span program, as in the degree 1 polynomial refutation only a constant polynomial can be multiplied, and this is clearly irrelevant.

LEMMA 3.8. *For an arbitrary field K and an arbitrary boolean function f, $mSP(f) = mPP_1(f)$ and $SP(f) = PP_1(f)$.*

We prove that polynomial programs over arbitrary finite field of any degree $d \ge 2$ are equivalent to boolean circuits, and hence are probably significantly stronger than span programs. Monotone polynomial programs of degree 2 simulate both monotone span programs and monotone boolean circuits. In [**1**] it has been shown that there are functions which can be computed by polynomial size monotone span programs, but need at least $n^{\log n}$ size monotone circuits. No polynomial simulation of monotone circuits by monotone span programs is known and it is very unlikely that there is one. Thus monotone polynomial programs seem to be stronger than both previously considered models of monotone computations.

THEOREM 3.9. *Polynomial programs over an arbitrary finite field of an arbitrary constant degree $d \geq 2$ and boolean circuits mutually simulate each other. Monotone polynomial programs over any finite field simulate monotone circuits.*

PROOF. The simulation of polynomial programs by boolean circuits follows from the results of [**9**] which give a polynomial time procedure for deciding whether there exists a degree d refutation of given polynomials in polynomial calculus.

Let a circuit C be given; suppose C uses only $\vee, \wedge$ and literals. We construct a polynomial program of degree 2 which simulates C. Choose one variable for each vertex of the circuit. In particular we take some variable y_i for the input node x_i and z_i for the input node $\neg x_i$. Let g be a gate (i.e. non-input node) which is the conjunction, resp. disjunction, of the gates h and k; let u, v, w be the variables corresponding to g, h, k. Then we assign to g the polynomial $u + vw - v - w$, resp. $u - vw$. Let v_0 be the variable assigned to the output gate. The polynomial program consists of all polynomials assigned to the gates labelled by 1, the polynomial $1 - v_0$ labelled by 1, the polynomials y_i labelled by x_i, and the polynomials z_i labelled by $1 - x_i$.

It is easy to prove by induction that, for a given input, one can derive by a polynomial calculus derivation of degree 2 all polynomials v where v is a variable corresponding to a gate which computes 1 in the circuit. Hence, if the circuit outputs 1, v_0 is derivable and thus also 1 is derivable.

Now suppose that C outputs 0 for a given input. We shall show that there is a satisfying assignment to all equations chosen by the input, hence the constant polynomial 1 is not derivable. The assignment is given simply by putting $v = 0$, if the circuit computes 1 on the corresponding gate and $v = 1$ otherwise.

It is clear that the above simulation gives a monotone polynomial program, if the circuit is monotone. □

Note that for a general field we can always replace a polynomial program of degree d by a polynomial program of degree 2 with only a small increase in the size, by using more variables for the intermediate results.

4. Interpolation

4.1. Interpolation for Nullstellensatz and polynomial calculus. Clegg, Edmonds and Impagliazzo [**9**] showed that over a finite field it is possible to decide in polynomial time if a given set of polynomials has a polynomial refutation of a constant degree d. As a corollary they derive the following interpolation theorem.

THEOREM 4.1 ([**9**]). *Let a finite field be fixed. Suppose a set of polynomial equations of the form $p_i(\vec{x}, \vec{y}) = 0, q_j(\vec{x}, \vec{z}) = 0$ with $\vec{y}$ and $\vec{z}$ disjoint sets of variables, has a polynomial refutation of a constant degree. Then there exists a polynomial size circuit C with the following property. For a given input $\vec{u} \in \{0,1\}^n$, if C outputs 1 then $p_i(\vec{u}, \vec{y}) = 0$ are not satisfiable, and if C outputs 0 then $q_j(\vec{u}, \vec{z}) = 0$ are not satisfiable.*

PROOF. For a given input $\vec{u}$ test, using the polynomial time algorithm from [**9**] whether there exists a refutation of the given degree d of $p_i(\vec{u},\vec{y}) = 0$; output 1 if there is such a refutation and 0 otherwise. If there is no such refutation then $q_j(\vec{u},\vec{z}) = 0$ is not satisfiable, since otherwise we could get a refutation of $p_i(\vec{u},\vec{y}) = 0$ from the refutation of $p_i(\vec{x},\vec{y}) = 0, q_j(\vec{x},\vec{z}) = 0$ by substituting $\vec{u}$. $\square$

The same theorem obviously holds also for NS refutations, since it is a weaker system than polynomial refutations. We prove some refinements of this corollary which may be used for proving lower bounds on NS and possibly also polynomial refutations.

Let us call a polynomial $p(\vec{x},\vec{y})$ *monotone* in $\vec{x}$ if it can be represented in the form $p'(\vec{x})p''(\vec{y})$ where $p'(\vec{x})$ is a monomial.

THEOREM 4.2. *Suppose a set of polynomial equations of the form $p_s(\vec{x},\vec{y}) = 0, q_t(\vec{x},\vec{z}) = 0$ has a NS refutation of a constant degree d. Then there exists a span program $\mathbf{C}$ with the property that if $\mathbf{C}$ accepts $\vec{u} \in \{0,1\}^n$ then $p_s(\vec{x},\vec{y}) = 0$ are not satisfiable, and if C does not accept, then $q_t(\vec{x},\vec{z}) = 0$ is not satisfiable. The size of $\mathbf{C}$ is polynomial in the number of variables $\vec{x},\vec{y}$. Moreover, if all $p_s(\vec{x},\vec{y})$ are monotone in $\vec{x}$, then $\mathbf{C}$ is monotone.*

PROOF. (I) First we consider a special case where all polynomials $p_s(\vec{x},\vec{y})$ can be written in the form $x_i p'(\vec{y}) + p''(\vec{y})$ with x_i a single variable or just contain only variables $\vec{y}$. (In fact this is the usual form in applications, so the general case is considered here just for sake of having a nicer theorem.) We assume that the equations $u^2 - u = 0$ for all variables are present, so we have also polynomials of the form $x_i^2 - x_i = 0$. We can think of them as belonging to $q_t(\vec{x},\vec{z}) = 0$.

Suppose that

$$x_i p'_{i,j}(\vec{y}) + p''_{i,j}(\vec{y}), \quad j = 1,\dots,k_i,$$

are all the polynomials containing x_i.

The columns of the span program $\mathbf{C}$ will correspond to all monomials in $\vec{y}$ of degree $\leq d$, hence its size is polynomial in the number of variables, as d is constant.

The rows of $\mathbf{C}$ will be polynomials, or more precisely, the vectors of their coefficients, of the following three types. First, all polynomials of degree at most d which can be put in the form

$$r(\vec{y})(p'_{i,j}(\vec{y}) + p''_{i,j}(\vec{y})), \quad j = 1,\dots,k_i;$$

labelled by x_i. Second, we include and all polynomials of degree $\leq d$ which can be put in the form

$$r(\vec{y})p''_{i,j}(\vec{y}), \quad j = 1,\dots,k_i$$

labelled by $1 - x_i$. Third, we include all polynomials of degree at most d of the form

$$r(\vec{y})p(\vec{y})$$

where $p(\vec{y})$ is an initial polynomial which depends only on $\vec{y}$, and label them by 1.

It is easy to see that this span program tests whether for a given input $\vec{u}$, there is a NS refutation of $p_s(\vec{u},\vec{y}) = 0$ of degree d.

If we have moreover the condition that the polynomials $p_s(\vec{x},\vec{y}) = 0$ are monotone in variables $\vec{x}$, then the polynomials $p''_{i,k}(\vec{y})$ are 0. Hence all the polynomials labelled by $1 - x_i$ are 0, which means that the span program is monotone.

(II) Now we shall reduce a more general case to the case considered in (I). In the general case we modify the initial polynomials in the following way. For each variable x_i in it introduce a new variable w_i. Then replace all polynomials $p_s(\vec{x},\vec{y})$ by $p_s(\vec{w},\vec{y})$ and add the polynomials $x_i - w_i$ for all variables x_i. The new set of equation is satisfiable iff the original set of equations is, and the new polynomials have the required form. Also the original equations are derivable from the new ones.

Now consider the monotone case, i.e., suppose $p_s(\vec{x},\vec{y})$ has the form $x_{j_1} \dots x_{j_k} p(\vec{y})$. If $k \geq 2$, we replace this polynomial by k polynomials

$$x_{j_1}(p(\vec{y}) + w_2 + \ldots + w_k), \quad x_{j_2} w_2, \quad \ldots, \quad x_k w_k,$$

where $w_2, \ldots, w_k$ are new variables. It is easily seen that the old system of equations in variables $\vec{x}$, $\vec{y}$ is derivable from the new using a degree at most one larger than the degree of the original polynomials. It is also clear that the new set of equation is satisfiable iff the original set of equations is. □

The same considerations as above can be used to derive a version of Theorem 4.2 for polynomial refutations and polynomial programs. (The construction is even simpler, as we do not need to add all the multiples of the polynomials.) We obtain the following theorem. Note that for the finite fields the non-monotone part is equivalent to Theorem 4.1, because polynomial programs are equivalent to boolean circuits by Theorem 3.9.

THEOREM 4.3. *Suppose a set of polynomial equations of the form* $p_s(\vec{x},\vec{y}) = 0, q_t(\vec{x},\vec{z}) = 0$ *has a polynomial refutation of a constant degree* d. *Then there exists a polynomial program* C *with the property that if* C *accepts* $\vec{u} \in \{0,1\}^n$ *then* $p_s(\vec{x},\vec{y}) = 0$ *are not satisfiable, and if* C *does not accept, then* $q_t(\vec{x},\vec{z}) = 0$ *is not satisfiable. The size of* C *is polynomial in the number of variables* $\vec{x},\vec{y}$. *Moreover, if all* $p_s(\vec{x},\vec{y})$ *are monotone in* $\vec{x}$, *then* C *is monotone.*

4.2. A characterization of NS interpolants. Here we prove a converse to Theorem 4.2, which shows that we cannot reduce the class of interpolating functions any further.

THEOREM 4.4. *Let* $\mathbf{A}$ *be a span program of size* m *for inputs of size* n. *Then there are equations* $p_s(\vec{x},\vec{y}) = 0, q_t(\vec{x},\vec{z}) = 0$ *of degree 2 with* $O(nm)$ *variables which have a NS refutation proof of degree 3 and whose unique interpolant is the boolean function computed by* A.

PROOF. Let $\mathbf{A}$ be a matrix with m columns and M rows. By Section 3.2 we may assume that $M \leq nm$. Let l_j be the label of the jth row of $\mathbf{A}$, i.e. x_i or $1 - x_i$. Recall that by the construction in Theorem 3.4 equation (1) the negation of the function computed by $\mathbf{A}$ is computed by

$$\mathbf{B} = \begin{pmatrix} \vec{0} & \mathbf{I} \\ \vec{e} & \mathbf{A}^\top \end{pmatrix} \tag{2}$$

with the jth row labelled by $1 - l_j$, for $1 \leq j \leq m$, and the remaining rows labelled by one.

We shall write down equations $p_s(\vec{x},\vec{y}) = 0$ which describe $\mathbf{A}$ and $q_t(\vec{x},\vec{z}) = 0$ which describe $\mathbf{B}$. This means that $p_s(\vec{u},\vec{y})$, resp. $q_t(\vec{u},\vec{z})$ will not be satisfiable iff $\mathbf{A}$ resp. $\mathbf{B}$ accepts the input $\vec{u}$. Since, for every $\vec{u}$, either $\mathbf{A}$ or $\mathbf{B}$ accepts $\vec{u}$, the equations together will not be satisfiable. We shall show that in fact they have degree three NS refutation. It is also clear that the unique interpolant of such equations is the boolean function computed by $\mathbf{A}$.

In order to describe $\mathbf{A}$ by polynomial equations we index the columns of $\mathbf{A}$ by the constant 1 and variables $y_2, \ldots, y_m$. The row vectors correspond to linear polynomials in $\vec{y}$, in particular the target vector corresponds to the constant polynomial 1. As the considered polynomials are linear, the polynomial 1 can be expressed from them iff it is a linear combination (with coefficients from the field) of them. So the system $p_s(\vec{x}, \vec{y})$ contains polynomials

$$(a_{k,1} + \sum_{k=2}^{m} a_{k,j} y_j) l_k, \quad k = 1, \ldots, M \tag{3}$$

For B we index columns of B by $1, z_1, \ldots, z_M$. The system $q_t(\vec{x}, \vec{z})$ consists of polynomials

$$z_j(1 - l_j), \quad j = 1, \ldots, M, \tag{4}$$

$$1 + \sum_{k=1}^{M} a_{k,1} z_k, \tag{5}$$

$$\sum_{k=1}^{M} a_{k,j} z_k, \quad j = 2, \ldots, m. \tag{6}$$

Now we describe a NS refutation of degree 3. First multiply polynomials (3) by z_k, so that we get

$$(a_{k,1} + \sum_{j=2}^{m} a_{k,j} y_j) l_k z_k, \quad k = 1, \ldots, M \tag{7}$$

The equation $z_k(1 - l_k) = 0$ can be rewritten as $z_k = l_k z_k$. Hence by adding the appropriate multiples of polynomials (4) to the polynomials (7) we get

$$(a_{k,1} + \sum_{j=2}^{m} a_{k,j} y_j) z_k, \quad k = 1, \ldots, M \tag{8}$$

We sum all these equations to get

$$\sum_{k=1}^{M} a_{k,1} z_k + \sum_{j=2}^{m} \sum_{k=1}^{M} a_{k,j} y_j z_k. \tag{9}$$

Finally, we subtract the polynomials (5) and (6) multiplied respectively by 1, y_2, $\ldots$, y_m and get 1. □

4.3. An application. We shall show an application of Theorem 4.2 based on recent result [**1**]. In [**1**] a lower bound of $n^{\Omega(\log n / \log\log n)}$ is proved for the size of monotone span programs computing an explicitly defined boolean function. The form of the result allows us to deduce an $\Omega(\log n / \log\log n)$ lower bound on the degree of a NS refutation of an explicit set of polynomials of constant degree. This corollary is not interesting *per se*, since the system of equations is quite complicated and the lower bound is small compared to lower bounds from [**8**]. The importance of it is in showing that there is another context in which an interpolation theorem can be used to prove a lower bound. This gives us some hope to prove lower bounds in this way for other systems, in particular for systems like polynomial calculus where we lack any lower bounds.

We shall start with the description of the result of [**1**]. Let $\Gamma \subseteq V_1 \times V_2$ be a bipartite graph, $|V_1| = |V_2| = n$, let $s \leq n$. We define two sets of subsets of vertices:

$$\mathcal{A} = \{X \subseteq V_1 \cup V_2 \mid \exists A \subseteq V_1,\ |A| = s,\ A \cup \Gamma(A) \subseteq X)\},$$

$$\mathcal{B} = \{X \subseteq V_1 \cup V_2 \mid \exists T \subseteq V_1,\ |T| \leq s,\ X \subseteq \bigcup_{B \subseteq V_1,\ |B|=s,\ B \cap T \neq \emptyset} (B \setminus T) \cup \Gamma(B)\}.$$

Here we denote by $\Gamma(C)$ the vertices which are connected by an edge to *all* vertices in C. While $\mathcal{A}$ is clearly an NP definition, it is not so obvious for $\mathcal{B}$. However, we can describe it as follows:

$$\begin{aligned}\mathcal{B} \quad = \quad & \{X \subseteq V_1 \cup V_2 \mid \exists T \subseteq V_1,\ |T| \leq s,\ X \cap T = \emptyset \text{ and} \\ & \forall j \in X \cap V_2\ \exists B_j \subseteq V_1,\ |B_j| = s,\ B_j \cap T \neq \emptyset,\ j \in \Gamma(B_j))\}.\end{aligned}$$

In [**1**] graphs Γ have been constructed such that for a suitable s ($s = \Theta(\log n/\log\log n)$) the sets $\mathcal{A}$ and $\mathcal{B}$ are disjoint and any monotone span program which accepts (the characteristic vectors of) sets of $\mathcal{A}$ and rejects sets of $\mathcal{B}$ has size $\Omega(\log n/\log\log n)$.

In order to be able to apply Theorem 4.2 we have to define these sets using polynomials of small degree and such that the polynomials defining $\mathcal{B}$ are monotone in variables $\vec{x}$ – the common variables of the two sets of polynomials which code the subsets X. The first thing can be done easily for any NP sets. By Cook's theorem we can express such predicates using 3-CNF's. Then each disjunction can be easily stated as an equation of degree ≤ 3. To prove the second condition first rewrite the definition of $\mathcal{B}$ using the predicate $j \in X$, instead of set operations and the inclusion relation. In the 3-CNF representation a variable x_j will represent the truth of $j \in X$. As the relation $j \in X$ occurs in the definition only negatively, so will the variables x_j in the 3-CNF. Thus the polynomials representing the disjunctions will be monotone in $\vec{x}$.

5. Different fields

Very little is known about the relative power of the algebraic models of computation if we change the underlying field. It is easy to see that a span or dependency program over a field with p^l elements can be written as a program over $GF(p)$ larger by a factor of l; this is implicit in [**13**]. Nothing is known if we change the characteristic of the field.

For characteristic 0, the natural question is whether a span program over reals can be converted into a span program over the rationals, and whether the length of numbers in a span program over rationals can be polynomially bounded. If both of these problems are resolved positively, it would show that functions computed by span programs over $\mathbf{R}$ can be computed by boolean circuits that are only polynomially larger.

The most natural approach is to replace the current coefficients of all vectors by (small) rational ones, so that all the linear dependencies are exactly preserved (for example by "moving" all the irrational points to rational ones carefully, so that the dependencies are preserved). In such a case the matroid represented by the matrix would not change. We show that this approach cannot work, as there exist matroids that are representable over $\mathbf{R}$ but not over $\mathbf{Q}$, and also matroids that are representable over $\mathbf{Q}$, but only with doubly exponential coefficients. This result is based on the technique used to prove that the matroid of all vectors in the space K^3 has sufficient information to recover the whole field, see e.g. [**23**].

LEMMA 5.1. *Suppose that we are given real numbers $x_1 = 1, x_2, \ldots, x_n$ and polynomials $p_1, \ldots, p_m$ of n unknowns with integral coefficients such that the equations $p_j(\vec{x}) = 0$ are all satisfied. Then there exists a matroid M of elements $u_1, \ldots, u_n, \ldots, u_N$ such that for any representation of M over $\mathbf{R}$ the numbers $\tilde{x}_i = \alpha_i/\alpha_1$, where $\alpha_i = (u_i^\top u_{N-1})/(u_i^\top u_N)$, satisfy the equations $\tilde{x}_1 = 1$ and $p_j(\tilde{x}_1, \ldots, \tilde{x}_n) = 0$ for all $j \leq m$.*

PROOF. The idea is to use the usual geometric construction of sums and products in $\mathbf{R}^2$, namely the facts that $x_i + x_j = x_k$ iff the lines $(1,0),(0,x_i)$ and $(1,x_j),(0,x_k)$ are parallel, and $x_i x_j = x_k$ iff the lines $(1,0),(0,x_i)$ and $(x_j,0),(0,x_k)$ are parallel. To be able to speak about parallel lines in terms of linear dependencies, we use the projective plane represented by the vectors in $\mathbf{R}^3$. The point (a,b) is represented by $(a,b,1)$, up to scalar multiplication, and the lines are parallel if they both contain the same vector $(c,d,0)$ (corresponding to their point in the infinity). If we take the matroid of all the auxiliary points, any representation has to preserve the identities.

Due to the integrality of the coefficients we may assume that all the equations have form either $x_i + x_j = x_k$ or $x_i x_j = x_k$ (we may have to add some numbers representing the intermediate values; also we have to use the fact that $x_1 = 1$). Now consider the matroid M in $\mathbf{R}^3$ consisting of all the vectors $u_i = (0, x_i, 1)$, $v_i = (x_i, 0, 1)$, $w_i = (1, x_i, 1)$, $z_i = (-1, x_i, 0)$, and the three basis vectors. We claim that the matroid M satisfies the condition in the lemma.

Suppose that the matroid M is represented in some vector space over the reals. The basis vectors have to be represented by three linearly independent vectors, and all other vectors have to be represented by vectors linearly dependent on them. Hence we may assume that the vector space is $\mathbf{R}^3$, moreover, we may change the basis so that the basis vectors are represented by basis vectors. We may assume that any vector with non-zero last coordinate is represented with 1 in that coordinate: Multiplying a vector by a non-zero scalar does not change the linear dependencies; moreover, the values of α_i and hence $\tilde{x}_i$ are not affected by this change, since the last two vectors are already fixed to be the last two basis vectors.

After these transformations, the vectors u_i are represented by $(0, \alpha_i, 1)$. In M, all the triples u_i, v_i, z_1 are linearly dependent. It follows that there exists a non-zero scalar β such that all vectors v_i are represented by $(\beta\alpha_i, 0, 1)$. Also, since both triples $(1,0,0), u_i, w_i$ and $(0,1,0), v_1, w_i$ are linearly dependent, w_i must be represented by $(\beta\alpha_1, \alpha_i, 1)$.

Now we prove that the numbers $\tilde{x}_i$ satisfy all the equations. First suppose that the equation $x_i + x_j = x_k$ is present. Then both triples v_1, u_i, z_i and w_j, u_k, z_i are linearly dependent. It follows that $\alpha_i = \alpha_k - \alpha_j$ and hence $\tilde{x}_i + \tilde{x}_j = \tilde{x}_k$. Now suppose that the equation $x_i x_j = x_k$ is present. Then the triples v_1, u_i, z_i and v_j, u_k, z_i are both linearly dependent. It follows that $\alpha_i/\alpha_1 = \alpha_k/\alpha_j$ and hence $\tilde{x}_i \tilde{x}_j = \tilde{x}_k$. □

THEOREM 5.2. *There exists a matroid represented over $\mathbf{R}$ but not representable over $\mathbf{Q}$.*

PROOF. Use Lemma 5.1 for $x_2 = \sqrt{2}$ and the equation $x_2^2 = 2x_1$. It follows that if the matroid is represented over $\mathbf{Q}$, the equation has a solution in $\mathbf{Q}$, which is a contradiction. (We use the fact that any vectors with rational coefficients are linearly dependent over **R** are also linearly dependent over **Q**.) □

THEOREM 5.3. *There exists a matroid of $O(n)$ points which can be represented over* **Q** *only so that the ratio of some coordinates is at least* 2^{2^n}.

PROOF. Use Lemma 5.1 for $x_{i+1} = 2^{2^i}$ and equations equations $x_2 = 2x_1$, $x_{i+1} = x_i^2$ for $2 \leq i < n$. □

6. Open problems

Lower bounds for monotone polynomial programs.
The main open problem suggested by this paper is to prove lower bounds for monotone polynomial programs over finite fields. If such a bound is proved for a suitable function, a lower bound for polynomial calculus would follow by the interpolation theorem (Theorem 4.3).

Lower bounds for non-monotone span programs.
Proving a superpolynomial lower bound on non-monotone span programs would provide a lower bound on branching programs, and hence it would be a major achievement. Our results show that it could be beneficial to consider dependency programs instead of span programs, as their structure is simpler.

Better lower bounds for monotone span programs.
The best lower bound for monotone span programs is $n^{\Omega(\log n/\log\log n)}$ [**1**]; no exponential bound is known. Furthermore, even though we have this superpolynomial lower bound for monotone span programs, it would be interesting to have such a bound for a function computable by non-monotone span programs of polynomial size, to separate these two models.

Separation of programs over different fields.
All lower bounds for the monotone dependency or span programs known to us work for an arbitrary field. It would be interesting to have some technique which would distinguish the fields, similarly as in the results for bounded depth circuits with MOD_m gates [**22**]. Thus we ask to prove separation between $mDP_p(f)$ and $mDP_q(f)$ (or $mSP_p(f)$ and $mSP_q(f)$) for some explicit function f.

Power of span programs over the reals.
It is still open whether functions computed by polynomial size span programs over **R**, or even **Q**, can be computed by polynomial size circuits. The problem is that the span program can contain huge constants, whose size is not included in the size of the span program. (If, e.g., the span programs are uniform, the size of constants is bounded and the function is in P.)

Acknowledgements. We are grateful to Sam Buss for useful discussions and to the anonymous referee for the comments. The last open problem was communicated to us by Eric Allender and Avi Wigderson.

References

[1] L. Babai, A. Gál, and A. Wigderson. Superpolynomial lower bounds for monotone span programs. DIMACS technical report 96-37, submitted to Combinatorica, 1996.

[2] L. Babai, A. Gál, J. Kollár, L. Rónyai, T. Szabó, and A. Wigderson. Extremal bipartite graphs and superpolynomial lower bounds for monotone span programs. In *Proc. of the 28th Ann. ACM Symp. on Theory of Computing*, pages 603–611. ACM, 1996.

[3] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, and P. Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. *Proc. London Math. Soc.*, 73:1–26, 1996.

[4] A. Beimel, A. Gál, and M. Paterson. Lower bounds for monotone span programs. In *Proc. of the 36th Ann. IEEE Symp. on Foundations of Computer Sci.*, pages 674–681. IEEE, 1995.

[5] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Inf. Process. Lett.*, 18:147–150, 1984.

[6] M. Bonet, T. Pitassi, and R. Raz. Lower bounds for cutting planes proofs with small coefficients. In *Proc. of the 27th Ann. ACM Symp. on Theory of Computing*, pages 575–584. ACM, 1995.

[7] G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and importance of Logspace-MOD-classes. *Mathematical Systems Theory*, 25:223–237, 1992.

[8] S. Buss, R. Impagliazzo, J. Krajíček, P. Pudlák, A. A. Razborov, and J. Sgall. On constant-depth Frege systems with a modular counting connective. To appear in Computational Complexity.

[9] M. Clegg, J. Edmonds, and R. Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proc. of the 28th Ann. ACM Symp. on Theory of Computing*, pages 174–183. ACM, 1996.

[10] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[11] S. A. Cook and A. Haken. An exponential lower bound for the size of monotone real circuits. Manuscript, 1995.

[12] A. Haken. The intractability of resolution. *Theoretical Comput. Sci.*, 39:297–308, 1985.

[13] M. Karchmer and A. Wigderson. On span programs. In *Proc. of the 8th Structure in Complexity Theory*, pages 102–111. IEEE, 1993.

[14] J. Krajíček. Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic. To appear in J. of Symbolic Logic.

[15] J. Krajíček, P. Pudlák, and A. Woods. Exponential lower bount to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures and Algorithms*, 7:15–39, 1995.

[16] K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. In *Proc. of the 18th Ann. ACM Symp. on Theory of Computing*, pages 338–339. ACM, 1986.

[17] T. Pitassi, P. Beame, and R. Impagliazzo. Exponential lower bounds for the pigenhole principle. *Comput. Complexity*, 3:97–140, 1993.

[18] P. Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. To appear in J. of Symbolic Logic.

[19] P. Pudlák and V. Rödl. A combinatorial approach to complexity. *Combinatorica*, 12(2):221–226, 1992.

[20] A. A. Razborov. Applications of matrix methods for the theory of lower bounds in computational complexity. *Combinatorica*, 10:91–93, 1990.

[21] A. A. Razborov. Lower bounds for the polynomial calculus. Manuscript, 1996.

[22] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. of the 19th Ann. ACM Symp. on Theory of Computing*, pages 77–82. ACM, 1987.

[23] D. J. A. Welsh. *Matroid Theory.* Academic Press, 1976.

Mathematical Institute, AV ČR, Žitná 25, 115 67 Praha 1, Czech Republic
E-mail address: pudlak@math.cas.cz

Mathematical Institute, AV ČR, Žitná 25, 115 67 Praha 1, Czech Republic, and Dept. of Applied Mathematics, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic
E-mail address: sgall@math.cas.cz