BRICS RS-98-11 Frandsen et al.: Lower Bounds for Dynamic Algebraic Problems

BRICS

Basic Research in Computer Science

Lower Bounds for Dynamic Algebraic Problems

Gudmund Skovbjerg Frandsen
Johan P. Hansen
Peter Bro Miltersen

BRICS Report Series RS-98-11

ISSN 0909-0878 May 1998

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/98/11/`

Lower bounds for dynamic algebraic problems

Gudmund Skovbjerg Frandsen*
BRICS†
Department of Computer Science,
University of Aarhus,
DK-8000 Aarhus C,
Denmark.

Johan P. Hansen
Department of Mathematics,
University of Aarhus,
DK-8000 Aarhus C,
Denmark.

Peter Bro Miltersen*
BRICS†
Department of Computer Science,
University of Aarhus,
DK-8000 Aarhus C,
Denmark.

*Supported by the ESPRIT Long Term Research Programme of the EU under project number 20244 (ALCOM-IT).

†Basic Research in Computer Science, Centre of the Danish National Research Foundation

Abstract

We consider dynamic evaluation of algebraic functions (matrix multiplication, determinant, convolution, Fourier transform, etc.) in the model of Reif and Tate; i.e., if $f(x_1, \ldots, x_n) = (y_1, \ldots, y_m)$ is an algebraic problem, we consider serving on-line requests of the form "change input x_i to value v" or "what is the value of output y_i?". We present techniques for showing lower bounds on the worst case time complexity per operation for such problems. The first gives lower bounds in a wide range of rather powerful models (for instance history dependent algebraic computation trees over any infinite subset of a field, the integer RAM, and the generalized real RAM model of Ben-Amram and Galil). Using this technique, we show optimal $\Omega(n)$ bounds for dynamic matrix-vector product, dynamic matrix multiplication and dynamic discriminant and an $\Omega(\sqrt{n})$ lower bound for dynamic polynomial multiplication (convolution), providing a good match with Reif and Tate's $O(\sqrt{n \log n})$ upper bound. We also show linear lower bounds for dynamic determinant, matrix adjoint and matrix inverse and an $\Omega(\sqrt{n})$ lower bound for the elementary symmetric functions. The second technique is the communication complexity technique of Miltersen, Nisan, Safra, and Wigderson which we apply to the setting of dynamic algebraic problems, obtaining similar lower bounds in the word RAM model. The third technique gives lower bounds in the weaker straight line program model. Using this technique, we show an $\Omega((\log n)^2/\log\log n)$ lower bound for dynamic discrete Fourier transform. Technical ingredients of our techniques are the incompressibility technique of Ben-Amram and Galil and the lower bound for depth-two superconcentrators of Radhakrishnan and Ta-Shma. The incompressibility technique is extended to arithmetic computation in arbitrary fields.

1 Introduction

1.1 Setup

Reif and Tate [RT97] considered the following setup of *dynamic algebraic algorithms*. Let $f_1, \ldots, f_m$ be a system of n-variate polynomials over a commutative ring or rational functions over a field. We seek an algorithm, that, when given an initial input vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ to the system, does some preprocessing and then afterwards is able to efficiently handle on-line requests of two forms: "$\texttt{change}_k(v)$: Change x_k to the new value v" and "$\texttt{query}_k$: Return the value of output $f_k(\mathbf{x})$". Several natural concrete examples were given by Reif and Tate, including dynamic polynomial evaluation, dynamic matrix-vector multiplication, dynamic matrix-matrix multiplication, dynamic polynomial multiplication, and dynamic discrete Fourier transform. Reif and Tate provided two general techniques for the design of efficient dynamic algebraic algorithms. They also presented lower bounds and time-space trade-offs for several problems. Apart from Reif and Tate's work, we also meet dynamic algebraic problems in the literature on the PREFIX SUM problem [Fre82, Fre81, Yao85, HF93, FS89, BAG91]; the specific case of $f_i(\mathbf{x}) = \sum_{j=1}^{i} x_i$ for $i = 1, \ldots, n$.

The aim of this paper is to present three techniques for showing lower bounds for dynamic algebraic problems. We use them to show lower bounds on the worst case time complexity per operation for several natural problems where Reif and Tate had no lower bounds or only lower bounds for the time-space trade-off.

1.2 Problems considered

Given a commutative ring R, we look at the following systems of functions.

MATRIX-VECTOR MULTIPLICATION : $R^{n^2+n} \mapsto R^n$. The first n^2 components of the input are interpreted as an $n \times n$ matrix A, the last n components are interpreted as an n-vector $\mathbf{x}$, and $A\mathbf{x}$ is returned.

MATRIX MULTIPLICATION : $R^{2n^2} \mapsto R^{n^2}$. The input is interpreted as two $n \times n$ matrices which are multiplied.

CONVOLUTION : $R^{2n} \mapsto R^{2n}$: The input is interpreted as two n-vectors $\mathbf{x} = (x_0, \ldots, x_{n-1})$ and $\mathbf{y} = (y_0, \ldots, y_{n-1})$, whose convolution is returned. That is, the i'th component of the output is $z_i = \sum_{j+k=i} x_j y_k$.

DETERMINANT : $R^{n^2} \mapsto R$: The input is interpreted as a matrix, whose determinant is returned.

MATRIX ADJOINT : $R^{n^2} \mapsto R^{n^2}$ is the function that maps an $n \times n$ matrix A into the corresponding adjoint matrix given by MATRIX ADJOINT$(A)_{ij} = (-1)^{i+j}\det(A_{ji})$, where A_{ji} denotes the $(n-1)\times(n-1)$ matrix resulting when deleting the j'th row and the i'th column from A.

If k is a field, MATRIX INVERSE : $k^{n^2} \mapsto k^{n^2}$ is the partial function that maps a nonsingular $n \times n$ matrix A into the corresponding inverse matrix A^{-1}. Note that for a nonsingular matrix, MATRIX INVERSE$(A) = \frac{1}{\det A}$MATRIX ADJOINT(A).

DISCRIMINANT : $R^n \mapsto R$: The discriminant of the polynomial for which the n inputs are roots is returned, i.e.

$$\text{DISCRIMINANT}(x_1, \ldots, x_n) = \prod_{i \neq j}(x_i - x_j)$$

SYMMETRIC : $R^n \mapsto R^n$. All n elementary symmetric polynomials of the inputs are computed, i.e., the j'th component of the output is

$$y_j = \sum_{I \subseteq \{1,2,\ldots,n\}, |I|=j} \; \prod_{i \in I} x_i$$

POLYNOMIAL EVALUATION : $R^{n+2} \mapsto R$. A vector $(x, a_0, a_1, \ldots, a_n)$ is mapped to $a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$.

Finally, the following problem is defined for any algebraically closed field k. Let ω be a primitive n'th root of unity k, and let F be the $n \times n$ matrix $F = (\omega^{ij})_{i,j}$. The Discrete Fourier Transform DFT : $k^n \mapsto k^n$, is the map $\mathbf{x} \to F\mathbf{x}$.

1.3 Models of computation

A pivotal issue when considering lower bounds is the model of computation. For dynamic algebraic problems, this issue is quite subtle; models can vary according to *the algebraic domain* (reals, integers, finite fields, etc.), the *atomic operations allowed* (only arithmetic operations or more general operations), and the *possibility of influencing the control flow of the solution* (to what extent is the sequence of atomic operations performed allowed to depend on the previous history of the algorithm). We prove lower bounds in the following models of computation.

The straight line program model. This is the most basic model. Given the problem of dynamic evaluation of a function $f : k^n \mapsto k^m$, we assign a *straight line program* to each of the operations $\texttt{change}_1$, $\texttt{change}_2$, ...,

$\mathtt{change}_n$, $\mathtt{query}_1$, $\mathtt{query}_2$, ..., $\mathtt{query}_m$. The programs corresponding to the change-operations take a single input x and have no output, while the programs corresponding to the query-operations have no input but one output. Each program is a sequence of instructions of the form $y_i \leftarrow y_j \circ y_k$, where $\circ \in \{+, -, *, /\}$, and y_j and y_k are either input variables, memory variables, or constants. We could also assign a program to the initialization operation. However, we find it more convenient to assume that we always initialize to some specific vector (say, $(0, 0, 0, \ldots, 0)$). Then, we just need to assign an initial value to each variable which appears somewhere in one of the programs. Then, the complexity of a solution is the length of the longest program in the solution.

History dependent algebraic computation trees. In the straight line program model, it is not possible for the algorithm to modify the sequence of atomic operations performed. In the history dependent algebraic computation tree model, we allow the algorithm to control the sequence in a strong way. First, instead of assigning straight line programs to operations, we assign algebraic computation trees. As branching nodes, we do not just allow $<$-comparison (which only makes sense for certain fields), instead we allow branching according to *arbitrary* predicates of finite arity. Also, to each operation (such as $\mathtt{change}_{12}$) we assign not one, but *several* (in fact infinitely many) algebraic computation trees: One for each *history*, where a history is every bit of discrete information the system has obtained so far; namely, the sequence of input variables that were changed and output variables that were queried, and the result of every branching test made so far during the execution of the operations performed. When we execute an operation, we find the tree corresponding to the current history and execute that. The complexity of a solution is the depth of its deepest tree.

Random access machine models. A very general way of defining RAM models is outlined by Ben-Amram and Galil [BAG92]. Here, we will only give an informal discussion. A RAM has an infinite number of registers, indexed by the integers. It also has a finite number of CPU-registers with proper names. Each register contains an element of the domain of computation: if we consider computation over the reals, each register contains a real; if we consider computation over the integers, each register contains an integer. In any case, it is convenient if the integers (or at least a sufficiently large subset of the integers) is a subset of the domain of interest; this makes *indirect addressing* possible, an important feature of the RAM. The machine operates on the memory using a finite program containing the following kinds of instructions: direct and indirect reads and writes, conditional jumps and a finite number of atomic computational instructions operating on the CPU-registers. Each instruction is executed at unit cost. When the domain of the registers is the set of integers and the atomic operations are $+, -, *$, we get the *integer RAM*. Another model of interest is the *generalized real RAM* [BAG92]. Here, the registers contain arbitrary reals and as atomic operations we allow any set of functions $\mathbb{R}^c \mapsto \mathbb{R}$ for a constant c, with the property that for some countable closed set $C \subset \mathbb{R}^c$, each function is continuous in $\mathbb{R}^c \setminus C$.

The *word RAM* [FW93, FW94, Hag98] has a somewhat different flavor from the integer RAM and the real RAM. The integer RAM can be considered unreasonably powerful, since it can handle arbitrary integers with unit cost. Then again, the user can give it any sequence of n integers as input and measure the complexity of the computation as a function of n. The word RAM is the result of relaxing the power of both parties, the algorithm and the user. The word RAM does computation on words, i.e. integers in $\{0, 1, \ldots, 2^w - 1\}$ for some parameter w, intuitively determined at compile-time. The RAM has registers indexed by $\{0, 1, \ldots, 2^w - 1\}$; in particular, we assume $w \geq \log n$, so that the input can be given in registers and read. The RAM can operate on words using a number of unit cost operations including addition, subtraction, multiplication, integer division, bitwise Boolean operations, and left and right shifts. The algorithm should be correct for any value of $w \geq \log n$, but n, the number of words in the problem, should be the only variable appearing in the time bound. The word RAM has been extensively studied as a model for sorting and searching. For instance, Andersson *et al* [AHNR95] show that sorting n words can be done in time $O(n \log \log n)$ on a word RAM. The survey of Hagerup [Hag98] gives a good overview of these results. When considered as a model for dynamic algebraic problems, the word RAM is appropriate when the function in question is a constant degree polynomial over the integers. This ensures that when the input is a sequence of single words, i.e. integers in $\{0, 1, \ldots, 2^w - 1\}$, the output can be given in a constant number of words, i.e. we can at least *write* the output with unit cost. For instance, dynamic matrix multiplication makes good sense in the word RAM model while we will not consider dynamic determinant in this model.

1.4 Our results

We present three techniques for proving lower bounds for dynamic algebraic problems. The first technique is very robust. In particular, it holds under a wide range on assumptions about the algebraic domain and the operations allowed, and even if the algorithm is allowed to control the flow of computation in strong ways. The technique is closely related to the *incompressibility* technique of Ben-Amram and Galil [BAG92]. The second technique holds only for the word RAM model (where the first technique fails). It is a modest extension of communication complexity techniques of Miltersen *et al* [MNSW95]. With the first and second technique we show

Theorem 1 *Any solution to dynamic* MATRIX-VECTOR MULTIPLICATION, MATRIX MULTIPLICATION, MATRIX ADJOINT, MATRIX INVERSE, DETERMINANT, POLYNOMIAL EVALUATION *or* DISCRIMINANT *has worst case complexity* $\Omega(n)$ *per operation and any solution to dynamic* CONVOLUTION *or* SYMMETRIC *has worst case complexity* $\Omega(\sqrt{n})$ *per operation, in the following models of computation:*

- *Straight line programs over any fixed finite field (except for* POLYNOMIAL EVALUATION, DISCRIMINANT *and* SYMMETRIC*), with the allowed set of* `change`*-arguments being the field itself.*
- *History dependent algebraic computation trees over any infinite field, with the allowed set of* `change`*-arguments being any infinite subset of the field.*
- *The integer RAM (except for* MATRIX INVERSE*), with the allowed set of* `change`*-arguments being any infinite subset of the integers, and the generalized real RAM, with the allowed set of* `change`*-arguments being the reals.*
- *The word RAM (except for* MATRIX ADJOINT, MATRIX INVERSE, DETERMINANT, DISCRIMINANT, POLYNOMIAL EVALUATION *and* SYMMETRIC*), with the allowed set of* `change`*-arguments being the set of words.*

We should note that the lower bound for dynamic POLYNOMIAL EVALUATION was also proved by Reif and Tate, though not for as wide a range of models as above. Reif and Tate present lower bounds for a number of other problems by reductions from POLYNOMIAL EVALUATION; we can apply the same reductions to get the lower bounds in the wider range of models.

We should also note that for certain models and certain of the above problems, there is an easier way of showing the same lower bound. For instance, we can show a lower bound for dynamic MATRIX-VECTOR MULTIPLICATION over the reals using arithmetic operations as follows: It is well known [Win67, Win70] that $n \times n$ matrices A over the reals exist so that computing $\mathbf{x} \to A\mathbf{x}$ requires $\Omega(n^2)$ arithmetic operations. Now, given an alleged dynamic algorithm for dynamic MATRIX-VECTOR MULTIPLICATION with complexity $o(n)$ per operation, we can initialize the matrix input to this matrix. Then, we can evaluate $A\mathbf{x}$ for any given $\mathbf{x}$ using n `change` and n `query` operations, i.e., a total of $o(n^2)$ arithmetic operations, a contradiction. The same technique was, in fact, used by Reif and Tate to show the lower bounds of their paper (using the fact that explicit hard polynomials exist, rather than the fact that explicit hard matrices exist). However, this argument does not seem to generalize to show, for instance, the linear lower bound for straight line programs over a finite field (where matrices requiring $\Omega(n^2)$ arithmetic operations do not exist [Sav74]), nor to show any lower bound for the generalized real RAM or the word RAM. Also, our technique applies to a wider variety of problems in a uniform way.

Our third technique is more fragile. It only works in the model of history *in*dependent straight line programs. A technical ingredient of the technique is the lower bound for depth-two superconcentrators by Radhakrishnan and Ta-Shma [RTS97]. With the third technique we show

Theorem 2 *Any solution to dynamic* DFT *in the straight line program model over an algebraically closed field of characteristic* 0*, with* `change`*-arguments restricted to any infinite subset of the field, has worst case complexity* $\Omega((\log n)^2/\log\log n)$ *per operation.*

1.5 Optimality (and otherwise) of results

The lower bounds for MATRIX-VECTOR MULTIPLICATION and MATRIX MULTIPLICATION are tight, there are straightforward linear upper bounds. The lower bound for DISCRIMINANT is also tight, there is a linear upper bound for any infinite field (see Theorem 3), and a straightforward constant upper bound for any finite domain in the straight line program model. Interestingly, the linear upper bound does not seem to be implementable in the straight line program model. The lower bound for CONVOLUTION has a fairly good match in the $O(\sqrt{n \log n})$ upper bound of Reif and Tate [RT97] for the same problem. The upper and lower bounds for DETERMINANT, MATRIX ADJOINT, MATRIX INVERSE and SYMMETRIC are not tight, we don't know any solution for DETERMINANT, MATRIX ADJOINT and MATRIX INVERSE better than evaluating queries from scratch, and we don't know any better upper bound for dynamic SYMMETRIC than a (not quite obvious)

$\texttt{change}_i(v)$:	assume $x_i = v_k$ for $[v_k, n_k] \in L$; if $n_k > 1$ then $n_k := n_k - 1$ else $D := D / \prod_{j \neq k}(-1)(v_j - v_k)^2$; $L := L \setminus \{[v_k, 1]\}$; if $v = v_l$ for some $[v_l, n_l] \in L$ then $n_l := n_l + 1$ else $D := D \cdot \prod_j (-1)(v_j - v)^2$; $L := L \cup \{[v, 1]\}$; $x_i := v$;

Figure 1: Computation tree solution for DISCRIMINANT.

$O(n)$ upper bound (see Theorem 4).

Reif and Tate show an $O(\sqrt{n})$ upper bound for dynamic DFT which is valid in the straight line program model. This leaves a rather large gap between upper and lower bounds. Our third technique is inherently unable to show better lower bounds than a constant times $(\log n)^2 / \log \log n$, this quantity being the average number of edges per input/output-vertex in an optimal depth 2 superconcentrator.

Theorem 3 *There is a computation tree solution of complexity $O(n)$ for dynamic evaluation of* DISCRIMINANT. *The solution works over any field.*

Proof. All the current inputs $x_1, \ldots, x_n$ are maintained, and so is the set of their (distinct) values together with the number of occurrences in $L = \{[v_1, n_1], \ldots, [v_{|L|}, n_{|L|}]\}$, i.e. $n_i \geq 1$ and $\sum_i n_i = n$. Finally, we maintain the (nonzero) discriminant of the distinct values: $D = \prod_{i \neq j}(v_i - v_j)$.

With this representation `query` is simple; if all n_i's are 1, we return D, otherwise we return 0. For `change`, we must update D and L, which is easily done in linear time (see Figure 1). ∎

Theorem 4 *There is a straight line program solution of complexity $O(n)$ for* SYMMETRIC. *The solution works over any commutative ring.*

Proof. All the current inputs $x_1, \ldots, x_n$ and corresponding outputs $y_1, \ldots, y_n$ are maintained. This makes the straight-line program for $\texttt{query}_i$ trivial; it needs only return y_i. For the implementation of `change`, we observe that for any i, k, we have that $y_k = x_i z_{k-1,i} + z_{ki}$, where z_{ki} does not depend on x_i, which makes the solution in Figure 2 valid. ∎

```
change_i(v) :  z_0 := 1;
               for k = 1 ... n do
                    z_k := y_k - x_i z_{k-1};
                    y_k := z_k + v z_{k-1};
               x_i := v;
```

Figure 2: Straight line solution for SYMMETRIC.

1.6 Organization of paper

In Section 2, we present our first technique as it applies to the case of history dependent algebraic computation trees and then show how to generalize it to straight line programs over a finite field, the integer RAM, and the generalized real RAM. The lower bounds for the word RAM are presented in Section 3. In Section 4, we present the technique based on superconcentrators and its application to DFT.

2 Incompressibility based lower bounds

Our technique is essentially based on the following *incompressibility* statement: If k is an algebraically closed field, a *rational* map $k^n \mapsto k^{n-1}$ can not be injective. Thus, it is closely related to the technique of Ben-Amram and Galil, who applied incompressibility in various domains to show a gap between the power of random access machines and pointer machines [BAG92].

First, a technical lemma stating a generalization of the above fact. Let k be an algebraically closed field. Recall that an algebraic subset $W \subset k^n$ is an intersection of sets of the form $\{\mathbf{x} \in k^n | p(\mathbf{x}) = 0\}$, where p is a non-trivial multivariate polynomial.

Lemma 5 *Let k be an algebraically closed field. Let W be an algebraic subset of k^m and let $\phi = (f_1/g_1, \ldots, f_n/g_n) : k^m \setminus W \mapsto k^n$ be a rational map where $f_i, g_i \in k[x_1, \ldots, x_m]$ for $i = 1, \ldots, n$. Assume that there exists $\mathbf{y} \in k^n$ such that $\phi^{-1}(\mathbf{y})$ is non-empty and finite. Then $m \leq n$.*

Proof.

1. reduction. We can assume that $\mathbf{y} = (0, \ldots, 0)$.

Otherwise let $\mathbf{y} = (y_1, \ldots, y_n)$ and replace $\left(\frac{f_1}{g_1}, \ldots, \frac{f_n}{g_n}\right)$ with $\left(\frac{f_1}{g_1} - y_1, \ldots, \frac{f_n}{g_n} - y_n\right)$.

2. reduction. We can assume that W is the set of common zeroes of $g_1, \ldots, g_n$.

Otherwise let $\mathbf{x} \in \phi^{-1}(\mathbf{y}) \setminus W$ and choose a polynomial g that vanishes on W with $g(\mathbf{x}) \neq 0$. Consider the rational function

$$\tilde{\phi} = \left(\frac{f_1 g}{g_1 g}, \ldots, \frac{f_n g}{g_n g} \right) : k^m \setminus Z(g) \mapsto k^n$$

where $Z(g)$ is the zeroes of g. As $\mathbf{x} \in \tilde{\phi}^{-1}(\mathbf{y}) \subseteq \phi^{-1}(\mathbf{y})$ it is enough to prove the claim for $\tilde{\phi}$.

3. reduction. We can assume that W is the empty set, and ϕ is a polynomial function.

Otherwise, we assume that $\mathbf{y} = (0, \ldots, 0)$, and that W is the set of common zeroes of $g_1, \ldots, g_n$. Consider the polynomial function

$$\tilde{\phi} = (f_1, \ldots, f_n, x_{m+1} \cdot g_1 \cdot \ldots \cdot g_n - 1) : k^{m+1} \mapsto k^{n+1}$$

The fiber $\tilde{\phi}^{-1}(0, \ldots, 0)$ consists of the tuples $(x_1, \ldots, x_m, x_{m+1})$ such that $\phi(x_1, \ldots, x_m) = (0, \ldots, 0)$ and such that $x_{m+1} = \frac{1}{g_1(x_1,\ldots,x_m) \cdot \ldots \cdot g_n(x_1,\ldots,x_m)}$ which by assumptions on ϕ is non-empty and finite. Therefore it is enough to prove the claim for polynomial functions with $\mathbf{y} = (0, \ldots, 0)$ which follows from Lemma 6 below. ■

Lemma 6 *Let k be an algebraically closed field. Assume that the set of common zeroes of $f_i \in k[x_1, \ldots, x_m]$ for $i = 1, \ldots, n$ is non-empty and finite. Then $m \leq n$.*

Proof. Let X be the set of common zeroes and consider $A(X)$, the coordinate ring of polynomial functions on X. By finiteness of X we conclude that

$$A(X) = \prod_{P \in X} A(P) = \prod_{P \in X} k$$

is a finite dimensional vector space over k. The ideal $\mathcal{M}_{\acute{P}} = \prod_{P \in X \wedge P \neq \acute{P}} A(P)$ is a maximal ideal for all $\acute{P} \in X$.

Let $\mathcal{P}$ be a prime ideal in $A(X)$, then $\mathcal{P} = \mathcal{M}_{\acute{P}}$ for some $\acute{P} \in X$. Otherwise we obtain a contradiction by choosing for each $P \in X$ a $h_P \in \mathcal{M}_P \setminus \mathcal{P}$ and considering $0 = \prod_{P \in X} h_P \notin \mathcal{P}$.

As k is algebraically closed, Hilbert's Nullstellensatz (cf. [Eis95], Theorem 1.6) gives that

$$A(X) = k[x_1, \ldots, x_m]/\mathrm{Rad}(f_1, \ldots, f_n)$$

where $\mathrm{Rad}(f_1, \ldots, f_n)$ is the radical ideal of $(f_1, \ldots, f_n)$.

A minimal prime ideal of $(f_1, \ldots, f_n)$ in $k[x_1, \ldots, x_m]$ is also a minimal prime ideal of $\mathrm{Rad}(f_1, \ldots, f_n)$ and from above a maximal ideal in $k[x_1, \ldots, x_m]$. According to Krull's Principal Ideal Theorem (cf. [Eis95], Theorem 10.2) we have that $m = \dim k[x_1, \ldots, x_m] \leq n$ ■

We shall also need the following version of the well-known "Schwartz-Zippel Lemma".

Lemma 7 *Let k be a field.*

(i) Let $T \subset k$ be finite. If a multivariate polynomial $q \in k[x_1, \ldots, x_n]$ of total degree $\deg q \leq |T|$ *is not the zero-polynomial, then $q(\mathbf{a}) = 0$ for at most a fraction $\frac{\deg q}{|T|}$ of all the n-tuples $\mathbf{a} \in T^n$.*

(ii) Let $S \subset k$ be infinite (implying that k is infinite), and let W be a proper algebraic subset of k^n. Let p be a multivariate polynomial in n variables. If p is identically zero as a function restricted to $S^n \setminus W$, then p is the zero-polynomial.

Proof. The statement of part (i) is adapted from a paper by Schwartz [Sch80]. For part (ii) assume that there is a multivariate polynomial q (that is not the zero-polynomial) such that $W \subseteq \{\mathbf{x} \in k^n | q(\mathbf{x}) = 0\}$. It follows (by part (i)) that if p is not the zero polynomial and if $|T| > \deg p + \deg q$, then there exists $\mathbf{a} \in T^n \setminus W$ such that $p(\mathbf{a}) \neq 0$. ■

Definition 8 *Let k be a field.*

(i) Let B be an arbitrary set. A function $f : k^n \mapsto B$ is quasi-injective *if there is a proper algebraic subset $W \subset k^n$ such that $f^{-1}(f(\mathbf{a}))$ is finite for all $\mathbf{a} \in k^n \setminus W$.*

(ii) Let $f : k^n \mapsto k^m$ be a function. Let $X = \{x_1, \ldots, x_n\}$ be the set of inputs. Let $X_1 \subset X$ of size l. Permute the variables of f so that the variables of X_1 are first, and view f as a function $f : (k^l \times k^{n-l}) \mapsto k^m$. f is said to specialize quasi-injectively (injectively) *to X_1 if the function $F : k^{n-l} \mapsto (k^l \mapsto k^m)$ is quasi-injective (injective), where F maps $\mathbf{a} \in k^{n-l}$ into $f_{\mathbf{a}}$, the function arising from specializing f to the constant vector $\mathbf{a}$ on the input set $X \setminus X_1$.*

Remark. F being quasi-injective means that for almost all $\mathbf{a}$ there are only finitely many $\mathbf{b}$ such that $f_{\mathbf{a}}$ and $f_{\mathbf{b}}$ are identical functions. An example of a function specializing injectively is MATRIX-VECTOR MULTIPLICATION: Different matrices over a field represent different linear maps. Thus, MATRIX-VECTOR MULTIPLICATION specializes injectively to the n variables representing the vector-part of the input.

Theorem 9 *Let k be an algebraically closed field. Let the polynomial function $f : k^n \mapsto k^m$ specialize quasi-injectively to some set X_1 of size l. Then any history dependent algebraic computation tree solution for dynamic evaluation of f has complexity at least $\frac{n-l}{2(l+m)}$.*

Proof. (After permutation of indices) we may assume $X_1 = \{x_1, \ldots, x_l\}$. Let a family of algebraic computation trees solving dynamic evaluation of f be given, and let the max depth of any computation tree representing a `change` or `query` be d.

Consider the specific off-line solution $P = P_1; P_2$ for f that arises from using `change`/`query`-operations in the following order:

$$\begin{aligned} P_1 &: \mathtt{change}_{l+1}(z_1); \cdots ; \mathtt{change}_n(z_{n-l}) \\ P_2 &: \mathtt{change}_1(x_1); \cdots ; \mathtt{change}_l(x_l);\ y_1 := \mathtt{query}_1; \cdots ; y_m := \mathtt{query}_m \end{aligned}$$

¿From the algebraic computation tree $P = P_1; P_2$, we are going to construct a straight line program $Q = Q_1; Q_2$ that computes f when inputs, i.e. arguments $(x_1, \ldots, x_l, z_1, \ldots, z_{n-l})$ to `change`-operations, are restricted to be tuples in $k^n \setminus W$, where W is a proper algebraic subset of k^n. Let L be the number of leaves in the computation tree P. Let $D = 2^{d(n+m)}$, i.e. D is an upper bound on the degree of any polynomial/rational function occurring in any intermediate result in P. Let $T \subset k$ be a finite subset of k satisfying that $|T| > L(D + \deg f)$. Divide the elements of T^n in L classes $C_1, \ldots, C_L$ such that any n-tuple $\mathbf{a} \in C_i$ when given as argument to `change`-operations will make the computation of P follow the path to leaf number i. Clearly, some C_i must have size at least $|T|^n/L$, and without loss of generality assume that $|C_1| \geq |T|^n/L$. Let $Q = Q_1; Q_2$ be the straight-line program arising from the computation path induced by C_1 with all branching tests removed. Then Q computes $\tilde{f} : C_1 \mapsto k^m$, for some rational function $\tilde{f} = (\frac{p_1}{q_1}, \ldots, \frac{p_m}{q_m})$ that is defined on all of C_1 and since none of the q_i's are the zero-polynomial, Q can be extended to be defined on all of k^n except for a proper algebraic subset W defined by $q_1, \ldots, q_m$. Since $\tilde{f}$ is identical to the polynomial function f for the restricted input set C_1, it follows by Lemma 7(i) that Q does compute the polynomial function f whenever no division by zero occurs, i.e. for inputs restricted to $k^n \setminus W$.

We observe that there exists a proper algebraic subset $W_1 \subset k^{n-l}$ such that for all $\mathbf{a} \in k^{n-l} \setminus W_1$, we can find a proper algebraic subset $W_{\mathbf{a}} \subset k^l$ such that the straight-line program $Q = Q_1; Q_2$ will compute $f(\mathbf{x}, \mathbf{a})$ correctly for all $\mathbf{a} \in k^{n-l} \setminus W_1$, and $\mathbf{x} \in k^l \setminus W_{\mathbf{a}}$.

For given $(n-l)$-tuple $\mathbf{a} \in k^{n-l} \setminus W_1$, we may specialize the inputs of Q_1 to $\mathbf{a}$, resulting in program $Q_{\mathbf{a}}$ such that $Q_{\mathbf{a}}; Q_2$ computes the polynomial function $f_{\mathbf{a}}$ restricted to $k^l \setminus W_{\mathbf{a}}$.

Let $V_1 \subseteq V$ denote the set of variables read by the program Q_2. By assumption $|V_1| \leq 2(l+m)d$.

Let $\tilde{V_1}$ denote the values of the variables V_1 after the execution of $Q_{\mathbf{a}}$ but before the execution of Q_2, and let $\tilde{f}$ denote the unique (by Lemma 7(ii)) polynomial function that extends the rational function (from $X_1 = \{x_1, \ldots, x_l\}$ to $Y = \{y_1, \ldots, y_m\}$) computed by program Q_2.

Clearly, $\tilde{V_1}$ is a rational function of $\mathbf{a}$. Let $g : (k^{n-l} \setminus W_1) \mapsto k^{|V_1|}$ denote this function. Similarly, $\tilde{f}$ is a function of $\tilde{V_1}$, since Q_2 does only depend on $\mathbf{a}$ through the intermediate values $\tilde{V_1}$. Let $h : \text{codomain}(g) \mapsto (k^l \mapsto k^m)$ denote this function. We see that $F = h \circ g$. Since F by assumption is quasi-injective, so must also g be quasi-injective and by Lemma 5 this is only possible for $|V_1| \geq n - l$.

Combining the two inequalities for $|V_1|$, we get $n - l \leq |V_1| \leq 2(l+m)d$, i.e. $d \geq \frac{n-l}{2(l+m)}$. ■

Theorem 9 can be used to show lower bounds for a setting where the computation is over an algebraically closed field and arguments to `change`-operations are arbitrary elements thereof. We now give a generalization of Theorem 9 needed to get the lower bounds claimed in Theorem 1, i.e., when the computation is over an arbitrary field and the arguments allowed to `change`-operations an infinite subset thereof. We also need this generalization to get the lower bound for the integer RAM. Note that we can without loss of generality assume that the field is algebraically closed, since, if it is not, we can just consider computation in its algebraic closure.

Theorem 10 *Let k be an algebraically closed field. Let the polynomial function $f : k^n \mapsto k^m$ specialize quasi-injectively to some set X_1 of size l.*

Then, for any infinite subset $S \subseteq k$ it holds that any proposed history dependent algebraic computation tree solution for dynamic evaluation of f

that is correct when arguments to `change`*-operations are restricted to be elements of S must have complexity at least $\frac{n-l}{2(l+m)}$.*

Proof. This is essentially a repetition of the proof of Theorem 9. In the terminology of that proof, one must observe that when constructing the straight-line program Q, we only need to know that the original dynamic solution works properly when arguments to `change`-operations are restricted to some sufficiently large finite subset $T \subset k$. By choosing $T \subset S$ the entire proof of Theorem 9 carries over. ■

2.1 Applications

In this section we show, using Theorem 10, the lower bounds that were claimed for the history dependent algebraic computation tree model in Theorem 1 of the Introduction.

Different matrices over a field represent different linear maps. This means MATRIX-VECTOR MULTIPLICATION specializes injectively to the n variables representing the vector-part of the input and Theorem 10 gives us that any solution to dynamic MATRIX-VECTOR MULTIPLICATION in the history dependent algebraic computation tree model over a field with arguments of `change`-operations restricted to some infinite subset of the field has complexity $\Omega(n)$. Similarly, POLYNOMIAL EVALUATION over an infinite field specializes injectively to its first input, yielding an $\Omega(n)$ lower bound for dynamic POLYNOMIAL EVALUATION. Since matrix-vector multiplication is a specialization of matrix-matrix multiplication, an $\Omega(n)$ lower bound holds for dynamic MATRIX MULTIPLICATION.

We may construct a dynamic solution for matrix multiplication from a dynamic solution for matrix adjoint or matrix inverse using the following fact:

$$\text{MATRIX ADJOINT}\begin{pmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{pmatrix} = \begin{pmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{pmatrix}^{-1} = \begin{pmatrix} I & -A & AB \\ 0 & I & -B \\ 0 & 0 & I \end{pmatrix},$$

where A, B are square matrices of dimension $\frac{n}{3}$ and I is the identity matrix of that dimension. Thus, the $\Omega(n)$ lower bound also holds for MATRIX ADJOINT and MATRIX INVERSE.

We may construct a dynamic solution for matrix adjoint from a dynamic solution for determinant (of the same matrix), when noting that changing the (ij)'th entry in a matrix A by Δ, changes the determinant

MATRIX ADJOINT.$\texttt{change}_{ij}(v)$:	$x_{ij} := v$; DETERMINANT.$\texttt{change}_{ij}(v)$;
MATRIX ADJOINT.$\texttt{query}_{ij}$:	$z :=$ DETERMINANT.$\texttt{query}$; DETERMINANT.$\texttt{change}_{ji}(x_{ji}+1)$; $w :=$ DETERMINANT.$\texttt{query}$; DETERMINANT.$\texttt{change}_{ji}(x_{ji})$; return $(w - z)$;

Figure 3: MATRIX ADJOINT reduces to DETERMINANT.

by $\Delta \cdot (-1)^{i+j} \det A_{ij}$, where A_{ij} is the submatrix arising from deleting the ith row and jth column (Figure 3). Thus, we also have an $\Omega(n)$ lower bound for DETERMINANT.

Next, we show the lower bound for convolution. We can specialize CONVOLUTION to a function $g : k^{n+\sqrt{n}} \mapsto k^{\sqrt{n}}$ by setting $y_{\sqrt{n}} = y_{\sqrt{n}+1} = \cdots = y_n = 0$ and ignoring all outputs but $z_{\sqrt{n}-1}, z_{2\sqrt{n}-1}, \ldots, z_{n-1}$. Now g is computing a matrix vector product:

$$g(\begin{pmatrix} x_{\sqrt{n}-1} & x_{\sqrt{n}-2} & \cdots & x_0 \\ x_{2\sqrt{n}-1} & x_{2\sqrt{n}-2} & \cdots & x_{\sqrt{n}} \\ \vdots & & \ddots & \vdots \\ x_{n-1} & x_{n-2} & \cdots & x_{n-\sqrt{n}} \end{pmatrix}, \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{\sqrt{n}-1} \end{pmatrix}) = \begin{pmatrix} z_{\sqrt{n}-1} \\ z_{2\sqrt{n}-1} \\ \vdots \\ z_{n-1} \end{pmatrix}$$

Hence, we get the $\Omega(\sqrt{n})$ lower bound for CONVOLUTION from the $\Omega(n)$ lower bound for MATRIX-VECTOR MULTIPLICATION.

For the DISCRIMINANT function, we need to apply Theorem 10 again. DISCRIMINANT specializes quasi-injectively to its first input: Let $\text{DISCRIMINANT}_{\mathbf{a}} : k \mapsto k$ denote the function arising from substituting $\mathbf{a} \in k^{n-1}$ for the remaining inputs, i.e. $\text{DISCRIMINANT}_{\mathbf{a}}(x) = D(\mathbf{a})(-1)^{n-1}\prod_{i=2}^{n}(x-a_i)^2$, where D denotes the discriminant function on only $n-1$ roots. Observe that if $\text{DISCRIMINANT}_{\mathbf{a}}$ and $\text{DISCRIMINANT}_{\mathbf{b}}$ are identical functions and $D(\mathbf{a}) \neq 0$, then the coordinates of $\mathbf{a}$ and $\mathbf{b}$ must be identical up to a permutation, and since there is only $(n-1)!$ distinct permutations on $n-1$ elements, then the function $F : k^{n-1} \mapsto (k \mapsto k)$ is quasi-injective, where $F(\mathbf{a}) = \text{DISCRIMINANT}_{\mathbf{a}}$, and by Theorem 10 we have proved an $\Omega(n)$ lower bound for DISCRIMINANT.

For the SYMMETRIC function, we also need to apply Theorem 10 again. Assume, for convenience, that n is a perfect square. Let X_1, Y_1 be the following subsets of inputs and outputs respectively: $X_1 = \{x_1, \ldots, x_{\sqrt{n}}\}$, $Y_1 = \{y_{\sqrt{n}}, y_{2\sqrt{n}}, \ldots, y_n\}$, and let $\pi_{Y_1} : k^n \mapsto k^{\sqrt{n}}$ be the projection that ignores all outputs but those in Y_1. In fact, $\pi_{Y_1} \circ$ SYMMETRIC specializes quasi-injectively to the inputs in X_1. To see this, observe that if $\mathbf{a} \in k^{n-\sqrt{n}}$, $\mathbf{x} \in k^{\sqrt{n}}$, $\mathbf{y} = \text{SYMMETRIC}_{\mathbf{a}}(\mathbf{x})$ and σ_l is the lth elementary symmetric function (of all arities and $\sigma_0(\cdot) = 1$), then $y_{k\sqrt{n}} = \sum_{i=0}^{\sqrt{n}} \sigma_i(\mathbf{x}) \cdot \sigma_{k\sqrt{n}-i}(\mathbf{a})$. Since $\sigma_i(\mathbf{x})$ is a form of degree i, it follows (by Lemma 7) that $y_{k\sqrt{n}}$ as a function of $\mathbf{x} \in k^{\sqrt{n}}$ uniquely determines $\sigma_{k\sqrt{n}-i}(\mathbf{a})$ for $i = 0, \ldots, \sqrt{n}$. Consequently, for $\mathbf{a}, \mathbf{b} \in k^{n-\sqrt{n}}$, we have that $\pi_{Y_1} \circ \text{SYMMETRIC}_{\mathbf{a}} = \pi_{Y_1} \circ \text{SYMMETRIC}_{\mathbf{b}}$ if and only if $\mathbf{a}$ and $\mathbf{b}$ are identical up to a permutation of entries. By Theorem 10, we have an $\Omega(\sqrt{n})$ lower bound for SYMMETRIC.

2.2 Lower bounds for straight line programs over finite fields

In this section, we show our lower bounds for straight line programs over finite fields. We also show certain weak lower bounds when branching is allowed. Note that in a finite domain, we cannot hope for lower bounds in the history dependent algebraic computation tree model, since we may encode the entire input vector as part of the history, yielding a constant upper bound for every problem. The natural model to consider is history *in*dependent computation trees, with the allowed branching instructions being arbitrary predicates on two variables. In this model, Fredman [Fre82] showed a lower bound of $\Omega(\log n/\log\log n)$ for the PREFIX SUM-problem over $\mathbb{F}_2$. By reduction, one gets the same lower bound for MATRIX-VECTOR MULTIPLICATION and the related problems. We get a slightly better $\Omega(\log n)$ lower bound for the latter problems by the following theorem. On the other hand, we don't know any sub-linear upper bound for dynamic MATRIX-VECTOR MULTIPLICATION over a fixed finite field, even if branching is allowed. It is a very interesting open problem to get super-$\Omega(\log n)$ lower bounds for *any* explicit problem over $\mathbb{F}_2$ when branching is allowed.

Theorem 11 *Let $\mathbb{F}$ be a finite field. Let the function $f : \mathbb{F}^n \mapsto \mathbb{F}^m$ specialize injectively to some set X_1 of size l. Then any straight line solution for dynamic evaluation of f over $\mathbb{F}$ has complexity at least $\frac{n-l}{2(l+m)}$. Any history* in*dependent computation tree solution for dynamic evaluation of f over $\mathbb{F}$ has complexity at least* $\log \frac{n-l}{2(l+m)}$.

Proof. The proof of Theorem 9 carries over, with the following adaptations (and simplifications):

First consider the case of straight line programs. We may take $Q = P$, since P is a straight line program that is defined for all possible arguments to `change`-operations. The use of Lemma 5 is replaced by a simple counting argument: when the function $g : \mathbb{F}^{n-l} \mapsto \mathbb{F}^{|V_1|}$ is injective and $\mathbb{F}$ is finite, we have that $|V_1| \geq n - l$ by the pigeon hole principle.

In the case of computation trees, we don't convert the solution to a straight line program. Rather, we let V_1 be the set of variables appearing in the entire tree corresponding to P_2. Then, since the original trees are history independent, $|V_1| \leq 2(l + m)2^d$, yielding the desired lower bound. ∎

2.3 Lower bounds for the integer RAM and the generalized real RAM

We first show how to use Theorem 10 to prove lower bounds in the integer RAM model. Since the integers is a subset of the complex numbers, Theorem 10 implies that the lower bounds holds in the history dependent algebraic computation tree model over the integers (with division disallowed). Now, if an integer RAM solution of a certain complexity exists, we can "fold out" the solution to a solution in the history dependent algebraic computation tree model. Similar unfoldings have been done in several papers, see, for instance, Paul and Simon [PS82]. Unfortunately, in our setting, the unfolded solution may have higher complexity than the original, the problem being *indirect addressing*: An indirect addressing instruction has to be folded out into a chain of branching nodes, the exact number of nodes depending on the number of indirect writes already performed by the system. However, if we inspect the proofs of Theorems 9 and 10, we see that the lower bound holds *even if branching instructions are completely free*, as long as the trees remain finite. Thus, the lower bounds we obtained for polynomial functions apply to the integer RAM as well.

To show the lower bound for the generalized real RAM, we have to replace the use of Lemma 5 with results of Ben-Amram and Galil [BAG92] regarding the incompressibility of real numbers using almost continuous operations.

Let c be a positive integer. Let $\mathcal{F}_c$ be the set of functions $f : \mathbb{R}^c \mapsto \mathbb{R}$, for any $k, m > 0$, such that for some countable, closed set $C \subset \mathbb{R}^c$, f is continuous in $\mathbb{R}^c \setminus C$.

As explained above, we just have to generalize the lower bound to history dependent computation trees with the allowed computational operations being $\mathcal{F}_c$ and the allowed branching instruction being $<$. If the lower bound holds, even if branching is free (as long as the trees remain finite), the lower bound holds for the generalized real RAM.

Let $\mathcal{F}_c^*$ be the closure of $\mathcal{F}_c$ under function composition and aggregation (aggregation combines functions $f_1, f_2, \ldots, f_k : \mathbb{R}^m \mapsto \mathbb{R}$ to a vector valued function $f = (f_1, f_2, \ldots, f_n) : \mathbb{R}^m \mapsto \mathbb{R}^k$).

Fact 12 (Ben-Amram and Galil [BAG92, Theorem 6]) *Let $f \in \mathcal{F}_c^*$. Then there is a non-empty open set O such that f is continuous in O.*

Fact 13 (Ben-Amram and Galil [BAG92, Theorem 10]) *Let $f \in \mathcal{F}_c^*$, $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ with $m < n$. Then f is not injective.*

By a *box* in $\mathbb{R}^n$ we mean a set $I_1 \times I_2 \times \ldots \times I_n$ where I_n is an open interval.

Theorem 14 *Given a polynomial function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ that specialize injectively to a set of variables of size l. Then, any system of history dependent $\mathcal{F}_c$-computation trees solving dynamic evaluation of f has complexity $\Omega(\frac{n-l}{l+m})$.*

Proof. Suppose a solution with complexity d is given. As in the proof of Theorem 9, we let

$$\begin{aligned} P_1 &: \mathtt{change}_{l+1}(z_1); \cdots; \mathtt{change}_n(z_{n-l}) \\ P_2 &: \mathtt{change}_1(x_1); \cdots; \mathtt{change}_l(x_l);\ y_1 := \mathtt{query}_1; \cdots; y_m := \mathtt{query}_m \end{aligned}$$

$P = P_1; P_2$ is now an $\mathcal{F}_c$-computation tree with input variables $x_1, \ldots, x_l$, $z_1, \ldots, z_{n-l}$. The leaves of the tree defines a partition of $\mathbb{R}^n$. We will show that one of the classes of this partition contains an open set. For this, we only have to show that if all elements of some open set S reaches a branching vertex of the tree, we can find an open subset $S' \subseteq S$, so that all elements of S' take the same branch. Without loss of generality, we can assume that the branching vertex branches according to whether $g(x_1, \ldots, x_l, z_1, \ldots, z_{n-l}) > 0$, where g is an $\mathcal{F}_c$-function. Being open, S contains a subset homeomorphic to $\mathbb{R}^n$. This means that Fact 12 applies to functions restricted to S, so we can find a non-empty open subset $T \subseteq S$ so that g is continuous in T. Now, the set $U = \{\mathbf{x} \in T | g(\mathbf{x}) > 0\}$ is open.

If it is empty, we let $S' = T$. If it is non-empty, we let $S' = U$. We have now established that we can find a leaf of the tree whose associated subset of $\mathbb{R}^n$ contains an open set. Let Q be the straight line program we get when we take the path from the root of the tree to this leaf and ignore all branching instructions. Split Q into $Q_1; Q_2$, where Q_1 corresponds to P_1 and Q_2 corresponds to P_2. Let V_1 denote the set of variables read by the program Q_2. By assumption, $|V_1| \leq c(l+m)d$.

$Q_1; Q_2$ computes the same function as $P_1; P_2$ on an open subset S of $\mathbb{R}^n$. If we view S as a subset of $\mathbb{R}^{n-l} \times \mathbb{R}^l$, we can find a box $S_1 \subset \mathbb{R}^{n-l}$ and a box $S_2 \subset \mathbb{R}^l$ so that $S_1 \times S_2 \subseteq S$.

Let $\tilde{V_1}$ denote the values of the variables V_1 after the execution of Q_1 but before the execution of Q_2, and let $\tilde{f}$ denote the function (from $X_1 = \{x_1, \ldots, x_l\} \in S_2$ to $Y = \{y_1, \ldots, y_m\}$) computed by the program Q_2.

Clearly, $\tilde{V_1}$ is a function of $\mathbf{a} = (a_1, a_2, \ldots, a_l) \in S_1$. Denote this function by $g : S_1 \mapsto \mathbb{R}^{|V_1|}$, and observe that $g \in \mathcal{F}_c^*$. Similarly, $\tilde{f}$ is a function of $\tilde{V_1}$, since Q_2 does only depend on $\mathbf{a}$ through the intermediate values $\tilde{V_1}$. Denote this function by $h : \mathbb{R}^{|V_1|} \mapsto (S_2 \mapsto \mathbb{R}^m)$. Note that, by construction, any function $h(y)$ is a polynomial function defined on an open set $S_2 \subset \mathbb{R}^l$. This extends in a unique way to a polynomial function on $\mathbb{R}^l$, so we can view h as a function $h : \mathbb{R}^{|V_1|} \mapsto (\mathbb{R}^l \mapsto \mathbb{R}^m)$. Using that f specializes injectively to X_1, we see that $F = h \circ g$ is injective, and hence also g is injective, We can easily find an injective function g' in $\mathcal{F}_c^*$ mapping $\mathbb{R}^{n-l}$ to S_1, so $g \circ g'$ is an injective map from $\mathbb{R}^{n-l}$ to $\mathbb{R}^{|V_1|}$. By Fact 13, this is only possible if $|V_1| \geq n - l$.

Combining the two inequalities for V_1, we get $n - l \leq |V_1| \leq c(l+m)d$, i.e. $d = \Omega(\frac{n-l}{l+m})$. ■

Using the same reductions as previously, we have: Any generalized real RAM solution for dynamic evaluation of any of the problems MATRIX-VECTOR MULTIPLICATION, MATRIX MULTIPLICATION, MATRIX ADJOINT, MATRIX INVERSE, DETERMINANT and POLYNOMIAL EVALUATION has complexity $\Omega(n)$ per operation. Any solution for dynamic evaluation of CONVOLUTION has complexity $\Omega(\sqrt{n})$ per operation.

To get a lower bound for DISCRIMINANT we consider a weakening of the concept of specializing injectively. The weakening we need is somewhat different from the concept of quasi-injectivity, used in the algebraic case:

Definition 15 *Let $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ be a system of polynomials. Let $X = \{x_1, \ldots, x_n\}$ be the set of inputs. Let $X_1 \subset X$ be of size l. f is said to specialize* weakly *injectively to X_1 if, for some open subset $S \subseteq \mathbb{R}^{n-l}$, the* *function* $F : S \mapsto (\mathbb{R}^l \mapsto \mathbb{R}^m)$ *is injective, where* F *maps* $\mathbf{a} \in S$ *into* $f_{\mathbf{a}}$*, the function arising from specializing* f *to the constant vector* $\mathbf{a}$ *on the input set* $X \setminus X_1$.

It is easy to see that the proof of Theorem 14 goes through with "injectively" replaced with "weakly injectively".

We shall show that DISCRIMINANT specializes weakly injectively to its first variable. Let $X_1 = \{x_1\}$ and $S \subseteq \mathbb{R}^{n-1}$ be the Cartesian product of $n-1$ disjoint intervals. Let $f_{\mathbf{a}} : \mathbb{R} \mapsto \mathbb{R}$ denote the function arising from substituting $\mathbf{a} \in S$ for the inputs in $X \setminus X_1 = \{x_2, \ldots, x_n\}$, i.e.

$$\text{DISCRIMINANT}_{\mathbf{a}}(x) = D(\mathbf{a})(-1)^{n-1} \prod_{i=2}^{n} (x - a_i)^2,$$

where D denotes the discriminant function on only $n-1$ roots. Note that $D(\mathbf{a})$ is non-zero for all $\mathbf{a}$ in S. Observe that if DISCRIMINANT$_{\mathbf{a}}$ and DISCRIMINANT$_{\mathbf{b}}$ are identical functions then the coordinates of $\mathbf{a}$ and $\mathbf{b}$ must be identical up to a permutation, but by construction, this means that they are equal. We have shown: any generalized real RAM solution for dynamic evaluation of DISCRIMINANT has complexity $\Omega(n)$ per operation.

A similar argument shows the $\Omega(\sqrt{n})$ lower bound for SYMMETRIC on the generalized real RAM.

3 Lower bounds for the word RAM

We show a lower bound for dynamic MATRIX-VECTOR MULTIPLICATION on the word RAM. The word size of the RAM is denoted w, i.e. each register contains an integer between 0 and $2^w - 1$ (a word) which is also the range of possible input values. A solution to the problem should work no matter how w and n relate, as long as $w \geq \log n$. This fact is exploited in the lower bound proof.

The technique used is the communication complexity technique of Miltersen *et al* [MNSW95] and the proof is in fact a reduction from a variation of the *span*-problem from that paper. For an exposition of the communication complexity technique and this example in particular, we refer to the book of Kushilevitz and Nisan [KN97].

We present the lower bound proof as a series of reductions. First, assume, to the contrary, that the following holds.

- There is a solution to dynamic MATRIX-VECTOR MULTIPLICATION on the word RAM with worst case time $o(n)$ per operation.

In particular

- There is an algorithm which maintains a representation of an $n \times n$ word matrix A so that matrix entries can be updated in time $t_1 = o(n)$ and, given an n-vector $\mathbf{x}$ of words, we can compute $A\mathbf{x}$ in time $t_2 = o(n^2)$.

Using perfect hashing to compress the representation, as explained, for instance, in [Mil94], we get

- There is a scheme for representing $n \times n$ word matrices so that a matrix can be stored in $s = O(t_1 n^2) = o(n^3)$ words so that, given an n-vector $\mathbf{x}$ of words, we can compute $A\mathbf{x}$ in time $t_2 = o(n^2)$.

Now consider the following communication game G_1 between two players, Alice and Bob. Bob gets an $n \times n$ matrix A of words and Alice gets an n-vector $\mathbf{x}$ of words. The object of the game is for Alice to obtain the value of $A\mathbf{x}$ using as few bits of communication as possible. Bob does not need to obtain this information. Using the communication complexity technique on the scheme above (For instance, Kushilevitz and Nisan, Lemma 9.6, page 116), we arrive at a communication protocol:

- There is a protocol for G_1 where Alice sends $O(t_2 \log s) = o(n^2 \log n)$ bits and Bob sends $O(t_2 w) = o(n^2 w)$ bits.

Note that the above protocol works no matter how w and n relate, as this was the case for the original RAM algorithm.

Given w, let p be the smallest prime between 2^{w-1} and 2^w. Now consider the following communication game G_2: Bob gets n vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ over $\mathbb{F}_p$ (where $\mathbb{F}_p$ is the finite field with p elements), Alice gets a single vector $\mathbf{x}$ over $\mathbb{F}_p$ and they must determine if $\mathbf{x}$ is in the span of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. We can derive a protocol for G_2 from a protocol for G_1 in the following way: Bob picks an $n \times n$ matrix A over $\mathbb{F}_p$ so that the kernel of A is exactly the span of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$. They now identify $\mathbb{F}_p$ with $\{0, \ldots, p-1\}$ in the natural way and run the G_1 protocol on A and $\mathbf{x}$. Alice now knows $A\mathbf{x}$ and can check if it is 0 modulo p and tell Bob if it is. Thus we have:

- There is a protocol for G_2 where Alice sends $o(n^2 \log n)$ bits and Bob sends $o(n^2 w)$ bits.

The following lemma now gives us a contradiction, if we put $w = \Omega(n \log n)$, as we are allowed to do. The same lemma was shown in [MNSW95] for the case $p = 2$. The proof here is an immediate generalization.

Lemma 16 *In any protocol for G_2, either Alice sends $\Omega(nw)$ bits or Bob sends $\Omega(n^2 w)$ bits.*

Proof. For the proof we assume, without loss of generality, that Bob is given exactly $n/2$ linearly independent vectors.

Consider the communication matrix M of G_2, i.e., M has a row for every possible input of Alice (i.e. vectors $\mathbf{x}$) and a column for every possible input of Bob (i.e. sets of vectors V), and $M_{\mathbf{x},V} = 1$ if and only if $\mathbf{x}$ is in the span of V.

A 0-1 matrix M is called (u, v)-rich if at least v columns contain at least u 1-entries. Miltersen *et al* [MNSW95] showed that if a communication problem has a (u, v)-rich communication matrix and a protocol where Alice sends a bits and Bob sends b bits, then M contains a submatrix of dimensions at least $u/2^{a+2} \times v/2^{a+b+2}$ containing only 1-entries.

Using this, it suffices to show

1. M is $(p^{n/2}, p^{n^2/4})$-rich, and
2. M does not contain a 1-monochromatic submatrix of dimensions $p^{n/3} \times p^{n^2/6}$.

For 1, notice that every subspace of $\mathbb{F}_p^n$ of dimension exactly $n/2$ contains exactly $p^{n/2}$ vectors, and that there are more than $p^{n^2/4}$ subspaces of dimension $n/2$. To see this, we count the number of ways of choosing a basis for such a space (i.e., to choose $n/2$ independent vectors). There are $p^n - 1$ possibilities of choosing the first basis element (different from $\vec{0}$), $p^n - 2$ of choosing the second, $p^n - 4$ of choosing the third etc. Also note that each basis is chosen this way $\frac{n}{2}!$ times. Hence the number of bases is $\prod_{i=0}^{n/2-1}(p^n - p^i)/\frac{n}{2}!$. Now, each subspace has a lot of bases. By a similar argument, their number is $\prod_{i=0}^{n/2-1}(p^{n/2} - p^i)/\frac{n}{2}!$. Hence the total number of subspaces is:

$$\frac{\prod_{i=0}^{n/2-1}(p^n - p^i)}{\prod_{i=0}^{n/2-1}(p^{n/2} - p^i)} = \prod_{i=0}^{n/2-1} \frac{p^n - p^i}{p^{n/2} - p^i} \geq \prod_{i=0}^{n/2-1} p^{n/2} = p^{n^2/4}.$$

For 2, consider a 1-rectangle with at least $p^{n/3}$ rows. Note that any $p^{n/3}$ vectors span a subspace of $\mathbb{F}_p^n$ of dimension at least $n/3$ and that, by a similar argument to the one presented above, the number of subspaces of dimension $n/2$ that contain a given subspace of dimension $n/3$ is at most

$$\frac{\prod_{i=0}^{n/6-1}(p^n - p^{n/3+i})}{\prod_{i=0}^{n/6-1}(p^{n/2} - p^{n/3+i})} = \prod_{i=0}^{n/6-1} \frac{p^n - p^{n/3+i}}{p^{n/2} - p^{n/3+i}} \leq \prod_{i=0}^{n/6-1} p^n = p^{n^2/6},$$

as needed. ■

Using the same reductions as previously, we have shown: Any solution to dynamic MATRIX-VECTOR MULTIPLICATION and MATRIX MULTIPLICATION on the word RAM has complexity $\Omega(n)$. Any solution to dynamic CONVOLUTION has complexity $\Omega(\sqrt{n})$.

4 Lower bounds based on superconcentrators

For the purposes of our paper, we shall use the following definition of a superconcentrator of depth 2. The equivalence of this definition to the standard definition is due to Meshulam [Mes84].

Definition 17 *An n-superconcentrator of depth 2 is a graph G with nodes $X \cup V \cup Y$, where X,V and Y are disjoint, $|X| = |Y| = n$, and with edges $E \subseteq (X \times V) \cup (V \times Y)$ such that for any l, for any $X_1 \subseteq X$ and for any $Y_1 \subseteq Y$ with $|X_1| = |Y_1| = l$, we have $|N(X_1) \cap N(Y_1)| \geq l$, where $N(X_1), N(Y_1) \subseteq V$ denote the neighbors to X_1, Y_1.*

Fact 18 (Radhakrishnan and Ta-Shma [RTS97]) *The number of edges in an n-superconcentrator of depth 2 is at least $\Omega(n\frac{\log^2 n}{\log\log n})$.*

Definition 19 *Let k be an algebraically closed field. Let $f : k^n \mapsto k^n$ be a function. Let $X = \{x_1, \ldots, x_n\}$ be the set of inputs, and let $Y = \{y_1, \ldots, y_n\}$ be the set of outputs.*

f is said to be super-injective, when for every l, for every $X_1 \subseteq X$ and for every $Y_1 \subseteq Y$ satisfying that $|X_1| = |Y_1| = l$ there is $\mathbf{a} \in k^{n-l}$ such that $f_{\mathbf{a}} : k^l \mapsto k^l$ is injective, where $f_{\mathbf{a}}$ denotes the function arising from specializing f to the constants $\mathbf{a}$ on the inputs $X \setminus X_1$ and ignoring all outputs in $Y \setminus Y_1$.

Lemma 20 *Let k be an algebraically closed field. Let $f : k^n \mapsto k^n$ be a super-injective polynomial function. From any family of straight line programs for dynamic evaluation of f and of complexity d, one may construct an n-superconcentrator of depth 2 and with at most $3dn$ edges.*

Proof. From the dynamic solution for f, define a graph G as follows. The nodes of G is $X \cup V \cup Y$, where V is the variables used in the dynamic solution for f, i.e. we may assume that $V = \{v_1, \ldots, v_m\}$, where $m \leq 2dn$. The edges of G is $E \subseteq (X \times V) \cup (V \times Y)$ and $(x_i, v) \in E$, if the program for $\mathtt{change}_i$ writes the variable v. Similarly, $(v, y_j) \in E$, if the program for $\mathtt{query}_j$ reads the variable v. Clearly, $|E| \leq 3dn$. We shall argue that G is a superconcentrator.

Let l be given, and let $X_1 \subseteq X$, $Y_1 \subseteq Y$ be given such that $|X_1| = |Y_1| = l$. Let $V_1 = N(X_1) \cap N(Y_1)$. We need to argue that $|V_1| \geq l$. (After permutation of indices) we may assume that $X_1 = \{x_1, \ldots, x_l\}$ and $Y_1 = \{y_1, \ldots, y_l\}$. Use the super-injectivity of f to choose $\mathbf{a} \in k^{n-l}$ such that $f_{\mathbf{a}} : k^l \mapsto k^l$ is injective, where $f_{\mathbf{a}}$ denotes the function arising from specializing the inputs $(x_{l+1}, \ldots, x_n)$ to the constants $\mathbf{a} = (a_1, \ldots, a_{n-l})$ and ignoring all the outputs $(y_{l+1}, \ldots, y_n)$.

¿From the dynamic solution for f, construct an off-line solution $P = P_1; P_2; P_3$ for $f_{\mathbf{a}}$ as follows

$$\begin{aligned} P_1 &: \mathtt{change}_{l+1}(a_1); \cdots; \mathtt{change}_n(a_{n-l}) \\ P_2 &: \mathtt{change}_1(x_1); \cdots; \mathtt{change}_l(x_l) \\ P_3 &: y_1 := \mathtt{query}_1; \cdots; y_l := \mathtt{query}_l \end{aligned}$$

Let $\tilde{X}_1$ denote the values of the input variables X_1 (before the execution of P_2), let $\tilde{V}_1$ denote the values of the variables V_1 after the execution of P_2 but before the execution of P_3, and let $\tilde{Y}_1$ denote the values of the output variables Y_1 (after the execution of P_3). Clearly, $\tilde{V}_1$ is a rational function of $\tilde{X}_1$. Denote this rational function by $g : k^l \mapsto k^{|V_1|}$. Similarly, $\tilde{Y}_1$ is a rational function of $\tilde{V}_1$, since the output does only depend on the input through the intermediate values $\tilde{V}_1$. Denote this rational function by $h : k^{|V_1|} \mapsto k^l$. We see that $f_{\mathbf{a}} = h \circ g$. Since $f_{\mathbf{a}}$ is injective, so must also g be injective, and by Lemma 5 this is only possible if $|V_1| \geq l$. ■

Lemma 20 and Fact 18 together implies the following theorem.

Theorem 21 *Let k be an algebraically closed field. Let $f : k^n \mapsto k^n$ be a super-injective polynomial function. Any family of straight line programs for dynamic evaluation of f has complexity $\Omega(\frac{\log^2 n}{\log \log n})$.*

4.1 Lower bound for discrete Fourier transform

It is obvious that a linear map is super-injective if and only if all minors of the corresponding matrix are non-zero. Thus, by Theorem 21, to show the $\Omega((\log n)^2/\log\log n)$ lower bound for dynamic DFT claimed in Theorem 2 of the introduction, we need to show that this is the case for a large $(n^{\Omega(1)} \times n^{\Omega(1)})$ submatrix of the Fourier transform matrix. The following lemma accomplishes this.

Lemma 22 *Let k be an algebraically closed field of characteristic 0, let $\omega \in k$ be a primitive n'th root of unity, and let $F = (a_{ij})$ be the $n \times n$ discrete Fourier transform matrix with $a_{ij} = \omega^{ij}$.*

Then F contains an $l \times l$ submatrix B for some $l = \Omega(\sqrt[3]{\frac{n}{\log\log n}})$ such that all minors of B are nonzero.

Proof. Let $l = \lfloor \sqrt[3]{\phi(n)} \rfloor$, where $\phi(n)$ denotes the Euler phi function, which is also the number of distinct primitive n'th roots of unity. It is known that $\liminf_{n\to\infty} \frac{\phi(n)\ln\ln n}{n} = e^{-\gamma} \approx 0.56$ (see Hardy and Wright [HW54] page 267, theorem 328), so $l = \Omega(\sqrt[3]{\frac{n}{\log\log n}})$ as required.

Let z be a variable and let $C(z)$ be the $l \times l$ matrix with the ij'th entry being $c_{ij} = z^{ij}$. Let $B = C(\omega)$ and note that B occurs as the $l \times l$ submatrix in the upper left corner of F.

We show that all minors of B are nonzero. Clearly, each minor of $C(z)$ is a polynomial in z with integer coefficients, and we will later show that no minor of $C(z)$ is the zero-polynomial. Therefore, each minor in $C(z)$ is a nonzero polynomial of degree strictly less than $l^3 \leq \phi(n)$ (assuming that $l \geq 2$). This implies that the minors of $B = C(\omega)$ are nonzero. To see this, observe that ω is a root of the nth cyclotomic polynomial which has degree $\phi(n)$ and is irreducible over the field $\mathbb{Q}$ (see Hungerford [Hun74], page 299, Proposition 8.3). Therefore ω is not root of any polynomial with integer coefficients and of degree strictly smaller than $\phi(n)$, as k has characteristic 0.

We now show that no minor in the matrix $C(z)$ is the zero-polynomial. Let an $m \times m$ minor D in $C(z)$ be given by row-indices $i_1 < \cdots < i_m$ and column indices $j_1 < \cdots < j_m$. By Lemma 23, $D = z^{i_1 j_1 + \cdots + i_m j_m} + p(z)$, where $p(z)$ is either the zero-polynomial or has degree strictly less than $i_1 j_1 + \cdots + i_m j_m$. ■

Lemma 23 *Let two sets of m positive integers each be given, namely I containing $i_1 < \cdots < i_m$ and J containing $j_1 < \cdots < j_m$. For any permutation σ of $\{1, \ldots, m\}$, let $S_\sigma = i_1 j_{\sigma(1)} + \cdots + i_m j_{\sigma(m)}$. Then $S_1 > S_\sigma$ for $\sigma \neq 1$, where 1 denotes the identity permutation.*

Proof. Let σ be a permutation on $\{1, \ldots, l\}$ such that $\sigma \neq 1$. We will argue that by changing σ slightly, we can get a new permutation τ (possibly with $\tau = 1$) such that $S_\tau > S_\sigma$, which suffices to prove the lemma.

Since $\sigma \neq 1$, we can find $a < b$ such that $\sigma(a) > \sigma(b)$. Define τ to be identical to σ except that $\tau(a) = \sigma(b)$ and $\tau(b) = \sigma(a)$. This implies that $S_\tau = S_\sigma - i_a j_{\sigma(a)} - i_b j_{\sigma(b)} + i_a j_{\tau(a)} + i_b j_{\tau(b)} = S_\sigma - i_a j_{\sigma(a)} - i_b j_{\sigma(b)} + i_a j_{\sigma(b)} + i_b j_{\sigma(a)} = S_\sigma + (i_b - i_a)(j_{\sigma(a)} - j_{\sigma(b)}) > S_\sigma$. ■

We have established the lower bound claimed in Theorem 2 when the allowed set of `change`-arguments is the entire field. We show how to establish it when arguments are restricted to an infinite subset S. A specific `query`-operation returns a value that is a rational function of the current input values (the function may depend on the sequence of earlier operations). If this function is the correct one when inputs are from S^n, Lemma 7 implies that it will be correct in general, except perhaps on some algebraic subset. This subset could be a problem, since superinjectivity just guarantees that there exists an $\mathbf{a}$ such that $f_{\mathbf{a}}$ is injective, but this particular $\mathbf{a}$ may cause division by zero. Fortunately, in the case of DFT, we observe that any value of $\mathbf{a}$ works. Hence, the lower bound holds with restricted inputs.

References

[AHNR95] A. Andersson, T. Hagerup, S. Nilsson, and R. Raman. Sorting in linear time? In *Proc. Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 427–436, 1995.

[BAG91] Amir M. Ben-Amram and Zvi Galil. Lower bounds for data structure problems on RAMs (extended abstract). In *Proc. 32nd Annual Symposium on Foundations of Computer Science*, pages 622–631, 1991.

[BAG92] Amir M. Ben-Amram and Zvi Galil. On pointers versus addresses. *J. Assoc. Comput. Mach*, 39:617–648, 1992.

[Eis95] David Eisenbud. *Commutative Algebra*, volume 150 of *Graduate Texts in Mathematics*. Springer-Verlag, 1995.

[Fre81] M.L. Fredman. Lower bounds on the complexity of some optimal data structures. *SIAM J. Comput.*, 10:1–10, 1981.

[Fre82] M.L. Fredman. The complexity of maintaining an array and computing its partial sums. *J. Assoc. Comput. Mach.*, 29:250–260, 1982.

[FS89] M.L. Fredman and M.E. Saks. The cell probe complexity of dynamic data structures. In *Proc. Twenty First Annual ACM Symposium on Theory of Computing*, pages 345–354, 1989.

[FW93] M.L. Fredman and D.E. Willard. Surpassing the information-theoretic bound with fusion trees. *J. Comput. System Sci.*, 47:424–436, 1993.

[FW94] M.L. Fredman and D.E. Willard. Trans-dichotomous algorithms for mimimum spanning trees and shortest paths. *J. Comput. System Sci.*, 48:533–551, 1994.

[Hag98] Torben Hagerup. Sorting and searching on the Word RAM. In *Proc. 15th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1373 of *Lecture Notes in Computer Science*, pages 366–398. Springer-Verlag, 1998.

[HF93] H. Hampapuram and M.L. Fredman. Optimal bi-weighted binary trees and the complexity of maintaining partial sums. In *Proc. 34th Annual Symposium on Foundations of Computer Science*, pages 480–485, 1993.

[Hun74] Thomas W. Hungerford. *Algebra*, volume 73 of *Graduate Texts in Mathematics*. Springer-Verlag, 1974.

[HW54] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers. (Third Edition).* Oxford University Press, 1954.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[Mes84] Roy Meshulam. A geometric construction of a superconcentrator of depth 2. *Theoret. Comput. Sci.*, 32:215–219, 1984.

[Mil94] P.B. Miltersen. Lower bounds for Union-Split-Find related problems on random access machines. In *Proc. Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 625–634, 1994.

[MNSW95] Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. In *Proc. Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 103–111, 1995. To appear in *J. Comput. System Sci.*

[PS82] W. Paul and J. Simon. Decision trees and random access machines. In *Logic and Algorithmic*, volume 30 of *Monograph. Enseign. Math.*, pages 331–340. Univ. Genève, 1982.

[RT97] John H. Reif and Stephen R. Tate. On dynamic algorithms for algebraic problems. *J. Algorithms*, 22:347–371, 1997.

[RTS97] Jaikumar Radhakrishnan and Amnon Ta-Shma. Tight bounds for depth-two superconcentrators. In *Proc. 38th Annual Symposium on Foundations of Computer Science*, pages 585–594, 1997.

[Sav74] J.E. Savage. An algorithm for the computation of linear forms. *SIAM J. Comput.*, 3:150–158, 1974.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. Assoc. Comput. Mach.*, 27:701–717, 1980.

[Win67] S. Winograd. On the number of multiplications required to compute certain functions. *Proc. Nat. Acad. Sci. U.S.A.*, 58:1840–1842, 1967.

[Win70] S. Winograd. On the number of multiplications necessary to compute certain functions. *Comm. Pure Appl. Math.*, 23:165–179, 1970.

[Yao85] A.C. Yao. On the complexity of maintaining partial sums. *SIAM J. Comput.*, 14:277–288, 1985.

Recent BRICS Report Series Publications

RS-98-11 Gudmund Skovbjerg Frandsen, Johan P. Hansen, and Peter Bro Miltersen. ***Lower Bounds for Dynamic Algebraic Problems*****. May 1998. 30 pp.**

RS-98-10 Jakob Pagter and Theis Rauhe. ***Optimal Time-Space Trade-Offs for Sorting*****. May 1998. 12 pp.**

RS-98-9 Zhe Yang. ***Encoding Types in ML-like Languages (Preliminary Version)*****. April 1998. 32 pp.**

RS-98-8 P. S. Thiagarajan and Jesper G. Henriksen. ***Distributed Versions of Linear Time Temporal Logic: A Trace Perspective*****. April 1998. 49 pp. To appear in** ***3rd Advanced Course on Petri Nets*****, ACPN '96 Proceedings, LNCS, 1998.**

RS-98-7 Stephen Alstrup, Thore Husfeldt, and Theis Rauhe. ***Marked Ancestor Problems (Preliminary Version)*****. April 1998. 36 pp.**

RS-98-6 Kim Sunesen. ***Further Results on Partial Order Equivalences on Infinite Systems*****. March 1998. 48 pp.**

RS-98-5 Olivier Danvy. ***Formatting Strings in ML*****. March 1998. 3 pp. This report is superseded by the later report BRICS RS-98-12.**

RS-98-4 Mogens Nielsen and Thomas S. Hune. ***Deciding Timed Bisimulation through Open Maps*****. February 1998.**

RS-98-3 Christian N. S. Pedersen, Rune B. Lyngsø, and Jotun Hein. ***Comparison of Coding DNA*****. January 1998. 20 pp. To appear in** ***Combinatorial Pattern Matching: 9th Annual Symposium*****, CPM '98 Proceedings, LNCS, 1998.**

RS-98-2 Olivier Danvy. ***An Extensional Characterization of Lambda-Lifting and Lambda-Dropping*****. January 1998.**

RS-98-1 Olivier Danvy. ***A Simple Solution to Type Specialization (Extended Abstract)*****. January 1998. 7 pp.**

RS-97-53 Olivier Danvy. ***Online Type-Directed Partial Evaluation*****. December 1997. 31 pp. Extended version of an article to appear in** ***Third Fuji International Symposium on Functional and Logic Programming*****, FLOPS '98 Proceedings (Kyoto, Japan, April 2–4, 1998), pages 271–295, World Scientific, 1998.**

RS-97-52 Paola Quaglia. ***On the Finitary Characterization of π-Congruences*****. December 1997. 59 pp.**