MIT/LCS/TM-46

COMBINING DIMENSIONALITY AND RATE OF GROWTH ARGUMENTS FOR ESTABLISHING LOWER BOUNDS ON THE NUMBER OF MULTIPLICATIONS

Zvi M. Kedem

June 1974

TM-46

Combining Dimensionality and Rate of Growth Arguments for Establishing Lower Bounds on the Number of Multiplications *

Zvi M. Kedem

June 1974

This research was supported by the National Science Foundation in part under grant number P4P3488-00 and in part under grant number GJ34671.

* A preliminary version of this report appeared in Proceedings of the Sixth Annual Symposium on the Theory of Computing (1974).

ABSTRACT

A new method for establishing lower bounds on the number of multiplications and divisions required to compute rational functions is described. The method is based on combining two known methods, dimensionality and rate of growth. The method is applied to several problems and new lower bounds are obtained.

key words and phrases: algebraic operations, analysis of algorithms, computational complexity, dimensionality, lower bounds, multiplications, optimality, polynomials, rate of growth, rational functions.

CR categories: 5.12, 5.25.

Combining Dimensionality and Rate of Growth Arguments for Establishing Lower Bounds on the Number of Multiplications

1. Introduction

In this paper we describe a new method for establishing lower bounds on the number of multiplications and divisions required to compute rational functions. If F is a field and $a_1,\ldots,a_n$ are algebraically independent (indeterminates) over F then, as usual, $F(a_1,\ldots,a_n)$ is the field of the rational functions of $a_1,\ldots,a_n$ over F. For various F and $H \subset F$ we shall consider algorithms over $(F(a_1,\ldots,a_n), H\cup\{a_1,\ldots,a_n\})$, namely algorithms computing elements of $F(a_1,\ldots,a_n)$ by applying chains of rational operations to elements of $H\cup\{a_1,\ldots,a_n\}$.

The problem of establishing lower bounds on the number of operations was tackled first using dimensionality arguments. Namely, describing the method informally, it was proved that algorithms computing certain sets of rational functions had to use at least one multiplication or division to introduce one or several out of many of the indeterminates appearing in the functions being computed. Using these arguments no lower bound greater than the number of the indeterminates appearing in the functions can be established. These methods were used first for polynomials by Ostrowski [4] and Pan [5] and later by Winograd [9] and [10] for sets of rational functions which are linear in some of the indeterminates (note that multivariate polynomials are linear in their coefficients). Finally, Strassen [7] used these arguments for sets of arbitrary rational functions. The methods were mostly based on dimensionality arguments in linear algebra. A typical result, due to Pan, states that at least n multiplications and divisions are required to compute $\sum_{i=1}^{n} a_i x^i$ over a suitable domain.

Another approach, due to Strassen [8], used algebraic geometry to establish rate of growth arguments. This approach allowed Strassen to prove among other things that the computation of the set $\{\sum_{i=1}^{n} a_i^j \mid 1 \leq j \leq n\}$ requires at least $n \log_2 \frac{n}{e}$ multiplications and divisions.

For the case of the computation of <u>bilinear forms</u> special methods for establishing lower bounds were developed by, among others, Hopcroft and Musinski [3] and Brockett and Dobkin [2].

Our method will allow us, in certain cases, to combine dimensionality and rate of growth arguments, so as to find lower bounds which could not be obtained using the previous methods. Among other results we obtain lower bounds on the number of multiplications required to compute $a_1x, a_2x^2, \ldots, a_nx^n$ or the polynomial $\sum_{i=1}^{n} a_i x^{\alpha(i)}$ for $0 < \alpha(1) < \ldots < \alpha(n)$.

2. Previous results and other preliminaries

Definition 1. Let E be a field and let $D \subset E$. An algorithm of length N over (E,D) is a sequence of instructions A of length N, each instruction of one of the following types:

(1) A [i]: = input d;

(2) A [i] : = A [j] ω A[k]

where $i,j,k \in \{1,\ldots,N\}$, $d \in D$, $\omega \in \{+,-,\times,:\}$ and for i,j,k appearing in (2) $j,k < i$. We shall sometimes say in case (2) that instruction A [i] (with a label i) refers to instructions A [j] and A [k] .

With every A a partial function Eval (A) : $(1,\ldots,N) \rightarrow E$ is associated and defined inductively by

If A[i]: = input d; then Eval (A) [i] ≙ d.

If A[i]: = A[j] ω A [k] ; then if Eval (A)[j], Eval (A) [k] , Eval (A) [j] ω Eval (A) [k] ; are defined, then Eval (A) [i] ≙ Eval (A) [j] ω Eval (A) [k].

In other cases Eval (A) [i] is not defined. From here on we shall consider only algorithms A for which Eval (A) is total, as there will be no interest in discussing the more general case .

For a subsequence B of A, we define

Comp (B) ≙ {Eval (A) [i] | A [i] ⊂ B }. We shall also say that B computes $\{e_1,\ldots,e_m\}$ (respectively e) if

$\{e_1, \ldots, e_m\} \subset$ Comp ($\mathcal{B}$) (respectively $e \in$ Comp ($\mathcal{B}$).

The definitions above are based on those of Winograd [9].

Example 1. To clarify the notions which have been introduced above we give a very simple example. Let Q denote, as usual, the field of rationals and let x be an indeterminate over Q. Then the following algorithm over $(Q(x), Q \cup \{x\})$ computes $1 + x + x^2 + x^3 = \frac{x^4-1}{x-1}$

i	A		Eval (A)
1	A [1]: =	input 2;	2
2	A [2]: =	A [1] : A [1];	1
3	A [3]: =	input x;	x
4	A [4]: =	A [3] × A [3];	x^2
5	A [5]: =	A [4] × A [4];	x^4
6	A [6]: =	A [5] − A [2];	x^4-1
7	A [7]: =	A [3] − A [2];	$x-1$
8	A [8]: =	A [6] : A [7];	$\frac{x^4-1}{x-1}$

Let F be an infinite field and let G be an infinite subfield of F. Let $a_1, \ldots, a_n$ be indeterminates over F. For some sets H such that $G \subset H \subset F$ we shall be interested in studying algorithms over $(F(a_1, \ldots, a_n), H \cup \{a_1, \ldots, a_n\})$. It will be useful to have a "pool" of parameters. To this end we shall have an infinite sequence $(c_i | 0 \leq i \leq \infty)$ of indeterminates over F which does not contain any elements from $(a_i | 1 \leq i \leq n)$. Let G_c, H_c, F_c denote $G(c_0, c_1, \ldots)$, $H(c_0, c_1, \ldots)$ and $F(c_0, c_1, \ldots)$ respectively. Then any algorithm over $(F(a_1, \ldots, a_n), H \cup \{a_1, \ldots, a_n\})$ can be considered to be over $(F_c(a_1, \ldots, a_n), H \cup \{a_1, \ldots, a_n, c_0, c_1 \ldots\})$ The purpose of introducing G and $c_0, c_1, \ldots$ will become clear later. To simplify the notation, we shall write * for either × or :. Furthermore, we shall sometimes write $\underline{a}$ for either $\{a_1, \ldots, a_n\}$ or $a_1, \ldots, a_n$, $\underline{c}$ for either $\{c_0, c_1, \ldots\}$ or $c_0, c_1, \ldots$ etc.

Our purpose is to be able to combine two distinct methods for establishing lower bounds. To this end we first divide the MDs (multiplications and divisions) into several classes.

Let $F \subset E$ and $G \subset D$. Then $E \cap F_c \neq \emptyset$ and $G_c \cap D \neq \emptyset$. We shall characterize MDs in algorithms over (E,D) according to whether the operands belong to G_c , F_c , $E-F_c$ etc.

Definition 2 A MD $A[i] := A[i] * A[k]$ in an algorithm over (E,D) where $F \subset E$, $G \subset D$ will be

(1) strongly counted (a SCMD) if

when $* \equiv \times$ then $\underline{\text{Eval}}(A)[j] \in E - F_c$ and $\underline{\text{Eval}}(A)(k) \in E - G_c$ or $\underline{\text{Eval}}(A)[j] \in E - G_c$ and $\underline{\text{Eval}}(A)[k] \in E - F_c$;

when $* \equiv :$ then $\underline{\text{Eval}}(A)[j] \in E - F_c$ and $\underline{\text{Eval}}(A)[k] \in E - G_c$ or $\underline{\text{Eval}}(A)[j] \in E - \{0\}$ and $\underline{\text{Eval}}(A)[k] \in E - F_c$;

(2) weakly counted (a WCMD) if

when $* \equiv \times$ then $\underline{\text{Eval}}(A)[j]$, $\underline{\text{Eval}}(A)[k] \in F_c - G_c$;

when $* \equiv :$ then $\underline{\text{Eval}}(A)[j] \in F_c - \{0\}$ and $\underline{\text{Eval}}(A)[k] \in F_c - G_c$;

(3) not counted (a NCMD) if

when $* \equiv \times$ then $\underline{\text{Eval}}(A)[j] \in G_c$ or $\underline{\text{Eval}}(A)[k] \in G_c$;

when $* \equiv :$ then $\underline{\text{Eval}}(A)[k] \in G_c$.

We shall sometimes refer informally to $\underline{\text{Eval}}(A)[j] * \underline{\text{Eval}}(A)[k]$ instead of $A[i] := A[j] * A[k]$ and say that $\sigma_1 * \sigma_2$ for $\sigma_1, \sigma_2 \in E$ is a SCMD etc.

Example 2. Set $F = \mathbb{R}$ (the field of reals), $G = Q$, $H = Q \cup \{\sqrt[4]{2}\}$. Then in the following algorithm over $(F(a_1), H \cup \{a_1\})$.

i	A	Eval (A)
1	A [1]: = input 4.2;	4.2
2	A [2]: = input $\sqrt[4]{2}$	$\sqrt[4]{2}$
3	A [3]: = input a_1;	a_1
4	A [4]: = A [1] × A [3];	$4.2a_1$
5	A [5]: = A [2] × A [2];	$\sqrt[2]{2}$
6	A [6]: = A [1]: A [3];	$4.2/a_1$

A [4] is a NCMD, A [5] is a WCMD and A [6] is a SCMD.

Every MD is in exactly one of the three classes. As every MD can be simulated by SCMDs and ASs (additions and subtractions) there is no way of establishing nontrivial lower bounds on the number of WCMDs or NCMDs appearing in arbitrary algorithms computing certain rational functions. We would obviously prefer to give lower bounds on the total number of MDs, but we do not have good methods for this. Instead, we shall give lower bounds on the number of CMDs (counted MDs) which comprise both SCMDs and WCMDs.

We start by citing a well known characterization of of algorithms without SCMDs [9].

Lemma 1. Every element computed by an algorithm A over $(F_c(\underline{a}), H \cup \{\underline{c},\underline{a}\}$ which has no SCMDs is of the form

$$f + \sum_{i=1}^{n} g_i a_i,\ g_i \in G_c,\ f \in F_c \qquad (1)$$

Proof. By induction on the length of A.

[0] Trivial

[N+1] We only remark that if

$(f^1 + \Sigma g_i^1\ a_i) * (f^2 + \Sigma g_i^2\ a_i)$ is not a SCMD,

then the result of this MD is of the form (1). □

In the interest of making the exposition as clear as possible some of our proofs will not be completely formalized. For example, in some proofs, instead of using a formal induction on the length of algorithms we shall use more informal **arguments**.

We quote now (using our formulation) the theorem which established the dimensionality arguments [7].

Theorem 1 (Strassen). Let A be an algorithm over $(F(a_1,\ldots,a_n)\ H \cup \{a_1,\ldots,a_n\})$ computing $\Psi(\underline{a}) \equiv \{\psi_i(\underline{a}) | 1 \leq i \leq t\}$ using $m \leq n$ SCMDs. Then there exists a (Zariski)-dense subset S of F^n, such that for every $\underline{s} \in S$ there is an $n \times (n-m)$ matrix Γ over G of rank $n-m$, such that

$\Psi'(\underline{b}) \triangleq \{\psi_i(\ \Gamma\underline{b} + \underline{s}) \mid 1 \le i \le t\}$ can be computed without SCMDs over $(F(b_1,\ldots,b_{n-m}),\ H\ \cup\{b_1,\ldots,b_{n-m}\})$ where $\underline{b} = (b_1,\ldots,b_{n-m})$ is a sequence of indeterminates over **F**. □

Although Strassen did not use our classification of MDs, his proof can be used for the above theorem.

Remark 1. Similarly to Lemma 1, it can be shown that the elements of Ψ' are of the form

$$f + \sum_{i=1}^{n-m} g_i\ b_i,\ g_i \in G\ ,\quad f \in \mathbf{F}\ . \qquad \square$$

To show how Theorem 1 is used we present a slightly strengthened version of Pan's result.

Corollary 1. Let **x** be an indeterminate over **G** and let **F** $= G(x)$. Then every algorithm over $(G(x\ ,\underline{a}),\ G\cup\{x,\underline{a}\})$ which computes $\sum_{i=1}^{n} a_i x^{\alpha(i)}$ for $0 < \alpha(1) < \ldots < \alpha(n)$ requires at least **n** SCMDs.

Proof. Assume that there exists A satisfying the assumptions of Theorem 1 which has only n-1 SCMDs. Then there exist $s_1,\ldots,s_n \in G(x)$ and $g_1,\ldots,g_n \in G$ not all of them zero, such that $\sum_{i=1}^{n} (g_i\ b_1 + s_i)\ x^{\alpha(i)} = \beta(x)\ b_1 + \alpha(x)$ can be computed without SCMDs. But, on the other hand, it is impossible by Remark 1 as $\beta(x) \in G(x) - G$. □

For a set Ψ and an algorithm A computing Ψ we shall denote by $\mu_D(\Psi)$ and $\mu_D(A)$ respectively the lower bound on the number of SCMDs which can be obtained by an application of Strassen's theorem. Similarly, we shall denote by $\mu(\Psi)$ and $\mu(A)$ the number of CMDs required to compute Ψ by an algorithm A.

In many cases it is simpler to use Winograd's theorem [9] which can also be obtained as a corollary to Strassen's theorem.

Corollary 2 (Winograd). Let A be an algorithm over $(F(a_1,\ldots,a_n), H \cup \{a_1,\ldots,a_n\})$ computing $\Phi \underline{a} + \phi$ where Φ is $t \times n$ matrix and ϕ a t-vector of elements in F. Furthermore, assume that there are m columns in Φ such that no nontrivial linear combination of them over G is in G^t (obviously it is in F^t). Then A has at least $\mu_D(A) = m$ SCMDs. □

3. Main results

We shall now describe briefly and informally the main idea behind our method. We start with an algorithm A over $(F(\underline{a}), H \cup \underline{a})$ which computes $\Psi(\underline{a})$. This algorithm has $\mu(A)$ CMDs (we want to find a lower bound on $\mu(A)$), at least μ_D of them SCMDs. We transform this A into an algorithm B over $(F_c, H \cup \underline{c})$. This algorithm computes certain $\underline{A}$ and $\Psi(\underline{A})$ and is obtained from A by (among other operations) a substitution of $\underline{A}$ for $\underline{a}$ which reduces at least μ_D SCMDs into NCMDs. Thus, we shall have $\mu(B) \leq \mu(A) - \mu_D(A)$. We can now use rate of growth arguments to give a lower bound, say $\mu_G(B)$, on the number of CMDs in B. Thus, $\mu(B) \geq \mu_G(B)$ and $\mu(A) \geq \mu_D(A) + \mu_G(B)$. We shall show that for some $\Psi(\underline{a})$ there are nontrivial lower bounds on $\mu_G(B)$ which do not depend on a particular B. Thus, we can establish a lower bound on $\mu(\Psi)$.

The following is our main theorem:

Theorem 2. Let A be an algorithm over $(F(a_1,\ldots,a_n),$ $H \cup \{a_1,\ldots,a_n\})$ computing $\Psi(\underline{a}) = \{\psi_j(\underline{a}) \mid 1 \leq j \leq t\}$. Then there exists an algorithm B over $(F(c_1,\ldots,c_n), H \cup \{c_1,\ldots,c_n\})$ computing $\underline{u} = (u_i(c_1,\ldots,c_{i-1}) \mid 1 \leq i \leq n)$, $\underline{v} = (v_i(c_1,\ldots,c_{i-1}) \mid 1 \leq i \leq n)$, $\underline{A} = (A_i \mid 1 \leq i \leq n)$ and $\Psi(\tilde{\underline{A}}) = \{\psi_j(\tilde{\underline{A}}) \mid 1 \leq j \leq t\}$ such that

(1) $\mu(B) \leq \mu(A) - \mu_D(A)$,

(2) Comp (B) $\subset G_c(H \cup \{a,\ldots,a_n\})$ (the set of rational functions in elements of $H \cup \{a_1,\ldots,a_n\}$ over G_c),

(3) $\bar{A}_i = u_i + c_i v_i + \sum_{j=i+1}^{n} w_i^j (u_j + c_j v_j)$,

(4) $u_i, v_i \in F(c_1,\ldots,c_{i-1})$, $w_i^j \in G(c_1,\ldots,c_{j-2})$,

(5) $\bar{\underline{A}}$ is a permutation of $\underline{A}$

(6) If there were no SCDs (strongly counted divisions) in A, then $v_i = 1$, $i = 1,\ldots,n$,

(7) If there were no CDs (counted divisions) in A, then Comp (B) $\in G_c[H \cup \{a_1,\ldots,a_n\}]$ (the set of polymomials in elements of $H \cup \{a_1,\ldots,a_n\}$ over G_c).

Before proving the theorem we shall prove some results which will be useful in its proof.

If A is an algorithm over $(F_c(\underline{a}), H \cup \{\underline{a},\underline{c}\})$ without SCMDs, then by Lemma 1 every element computed by A is of the form (1). We shall show that it is possible to "reorganize" A so as to obtain B without increasing the number of CMDs and which computes all the f's appearing in elements of the form (1) and then all the other elements computed by A. To state the result formally:

Lemma 2. Let A be an algorithm over $(F_c(\underline{a}), H \cup \{\underline{c}, \underline{a}\})$ without SCMDs. Then there exists an algorithm $B \equiv B_1 B_2$ (concatenation of two sequences of instructions B_1 and B_2) over $(F_c(\underline{a}), H \cup \{\underline{c}, \underline{a}\})$ such that

(1) $\mu(A) = \mu(B)$,

(2) Comp (A) $\subset$ Comp (B),

(3) Comp $(B_1) \in F_c$,

(4) $\{f \mid f + \Sigma q_i a_i \in$ Comp(A)$\} \subset$ Comp (B_1).

Proof. We note first that by Lemma 1 every element in Comp (A) has the form (1). We shall first construct B_1, by induction on the length of A.

[0] B_1 is empty.

[N+1] Let A' consist of the first N instructions of A and let B_1' satisfy the lemma for it. We shall now define B_1. There are several cases to be considered.

(1) A[N+1]: = input d;

If $d \in F_c$ then add this instruction (changing the label if necessary) to B_1' obtaining B_1; otherwise, set $B_1 = B_1'$.

(2) A[N+1]: = A[j] ω A[k];

then

Eval (A) [j] = $f^j + \Sigma q_i^j a_i$,

Eval (A) [k] = $f^k + \Sigma q_i^k a_i$,

and $f^j, f^k \in$ Comp (B_1').

There are several subcases to be considered:

(a) $\omega \in \{+,-\}$. In this case add the instruction computing $f^j \omega f^k$ to B_1' obtaining B_1,

(b) $\omega \in \{\times,:\}$ and $g_1^j = \dots = g_n^i = g_1^k = \dots = g_n^k = 0$. Then add or A [n+1] to B_1' obtaining B_1.

(c) $\omega \in \{\times,:\}$ and $\{g_1^j,\dots,g_n^j,\ g_1^k,\dots,g_n^k\} \neq \{0\}$. Then either $g_1^j = \dots = g_n^j = 0$ or $g_1^k = \dots = g_n^k = 0$. If, say $g_1^k = \dots = g_n^k = 0$ then $f^j \in G_c$ (otherwise this would be a SCMD) and add the instruction computing $f^j \omega f^k$ to B_1' obtaining B_1.

We saw that in all the three subcases we added the instruction $f^j \omega f^k$ to B_1', but we treated them separately so the reader could convince himself that no CMDs were added.

Now it is possible to add instructions, which are not CMDs to B_1 which compute all the elements of <u>Eval</u> (A) - <u>Eval</u> (B_1) thus defining $B \equiv B_1 B_2$. □

<u>Corollary 3</u>. Let $B \equiv B_1 B_2$ satisfy the conditions of Lemma 2 and let $f \in$ <u>Comp</u> (B_1), $g_0, g_1, \dots, g_n \in G_c$. Then it is possible to construct $B' \equiv B_1\ B_{1.5}\ B_2$ over $(F_c(\underline{a}), H \cup \{\underline{c},\underline{a}\})$ such that

(1) $g_0 f + \Sigma g_i a_i \in$ Comp ($B_{1.5}$),

(2) $\mu(B') = \mu(B)$.

Proof. B' is obtained from B by (informally) inserting between B_1 and B_2 instructions which are not CMDs and which compute $g_0 f + \Sigma g_i a_i$ using f which was computed by B_1. $\square$

Proof of Theorem 2. Instead of proving the theorem formally by induction on (say) length of A we shall show how to transform every A into an appropriate B satisfying the theorem.

We consider the first SCMD A[i]:=A[j]*A[k]. Let $\sigma_1 \triangleq$ Eval (A)[j], $\sigma_2 \triangleq$ Eval (A)[k]. Then at least one of σ_1 and σ_2, denote it by σ, has the form $f + \Sigma g_i a_i$ where

(1) $f \in G(H)$,

(2) $g_1, \ldots, g_n \in G$,

(3) $\{g_1, \ldots, g_n\} \neq \{0\}$.

If both operands have this form and satisfy condition (3) let $\sigma \triangleq \sigma_2$. By (possibly) permuting the vector $\underline{a}$ and using the same symbol, $\underline{a}$, to denote the permuted vector, we may assume that $g_1 \neq 0$. We shall consider two cases:

(1) If $\sigma_1 * \sigma_2 \equiv \sigma_1 \times \sigma_2$ or $\sigma \equiv \sigma_2$ then consider the equation $f + \Sigma g_i a_i = g_1 c_1$ and define $v_1 \triangleq 1$.

(2) If $\sigma_1 * \sigma_2 \equiv \sigma_1 : \sigma_2$ and $\sigma \equiv \sigma_1$ then consider the equation $f + \Sigma g_i a_i = g_1 v_1 c_1$ and define $v_1 \triangleq \sigma_2$ (we know that $\sigma_2 \in F - \{0\}$).

Solving the appropriate equation for a_1 and naming the solution A_1 we have $A_1 = u_1 + c_1 v_1 + \sum_{2}^{n} g_2^1 a_i$ where $u_1 = -f/g_1$ $g_i^1 = -g_i/g_1$.

Denote by $\tilde{A}$ the algorithm obtained from A by taking all the instruction up to, and not including, the first SCMD, This algorithm obviously has no SCMDs and therefore using Lemma 2 and Corollary 3 we may transform $\tilde{A}$ into $B_1 B_{1.5} B_2$ where B_1 and B_2 satisfy the conditions of Lemma 2 and $B_{1.5}$ computes u_1, v_1 and A_1 . Thus we may assume that A is already in the form $B_1 B_{1.5} B_2$, and A is an algorithm over $(F(c_1,\underline{a})$, $F \cup \{c_1,\underline{a}\})$.

We construct now A_1 from A by (informally) substituting A_1 for a_1 . More formally, we drop instructions with <u>input</u> a_1 and each instruction which refered to one of them will refer to the instruction which computed A_1 . Thus A_1 is an algorithm over $(F(c_1,a_2,\ldots,a_n)$, $H \cup \{c_1,a_2,\ldots,a_n\})$. We note first that <u>Eval</u> (A_1) is total, namely there are no attempts in A_1 to divide by zero. To show this it is enough to show that if $\psi(a_1,a_2,\ldots,a_n) \in F(a_1,a_2,\ldots,a_n) - \{0\}$ then $\psi(A_1,a_2,\ldots,a_n) \in F(c_1,a_2,\ldots,a_n\} - \{0\}$. But this follows immediately because:

(1) if $\psi(a_1,a_2,\ldots,a_n)$ is not a function of a_1 then $\psi(a_1,a_2,\ldots,a_n) = \psi(A_1,a_2,\ldots,a_n)$,

(2) if $\psi(a_1,a_2,\ldots,a_n)$ is a nontrivial function of a_1 then $\psi(A_1,a_2,\ldots,a_n)$ is a nontrivial function of c_1 .

We also see that $\mu(A_1) \leq \mu(A) - 1$. First we note that the instruction in A_1 which corresponds in an obvious way to the first SCMD in A is a NCMD. Indeed there are two cases corresponding to the two equations above:

(1) in this case the SCMD is transformed into $\sigma_1 * g_1 c_1$ or $g_1 c_1 \times \sigma_2$,

(2) in this case the SCMD is transformed into $g_1 c_1$ which can always be computed without CMDS.

In addition we note that any operation in A which was not a CMD is transformed into an operation in A_1 which is not a CMD. Thus the number of CMDs has been reduced by at least one. To show this we only remark that if $\psi(a_1, a_2, \ldots, a_n) \in G$ then $\psi(A_1, a_2, \ldots, a_n) \in G$.

Finally, we summarize for the reader that A_1 computes u_1, v_1, A_1, $\Psi(A_1, a_2, \ldots, a_n)$ and $\mu(A_1) \leq \mu(A) - 1$.

Let us now consider the first SCMD in A_1 . Once again we consider $\sigma_1 * \sigma_2$ and at least one of σ_1 and σ_2 , denote it by σ , has the form $f + \sum_2^n g_i a_i$ where

(1) $f \in G(H \cup \{c_1\})$,

(2) $g_2, \ldots, g_n \in G(c_1)$,

(3) $\{g_2, \ldots, g_n\} \neq \{0\}$.

If both operands satisfy these conditions, pick the second one. Similarly to above we consider an appropriate equation which would reduce this SCMD into an operation which is not a CMD and construct an algorithm A_2 over

$$(F(c_1,c_2,a_3,\ldots,a_n),\ H \cup \{c_1,c_2,a_3,\ldots,a_n\})$$

which computes

$$u_1,v_1,A_1 = u_1 + c_1v_1 + \sum_{2}^{n} g_i^1 a_i \ ,$$

$$u_2,v_2,A_2 = u_2 + c_2v_2 + \sum_{3}^{n} g_i^2 a_i$$

$$\text{and} \quad \Psi(A_1,A_2,a_3,\ldots,a_n)$$

where $u_2,v_2 \in F(c_1)$ (actually $u_2,v_2 \in G(H \cup \{c_1\})$), $g_i^2 \in G(c_1)$ and $\mu(A_2) \leq \mu(A_1) - 1 \leq \mu(A) - 2$.

We proceed in this manner $m \leq n$ times, until there are no SCMDs left. Then A_m computes $u_1,\ldots,u_m$, $v_1,\ldots,v_m$, $A_1,\ldots,A_m$ and $\Psi(A_1,\ldots,A_m\ ,\ a_{m+1},\ldots,a_n)$ over $(F(c_1,\ldots,c_m\ ,\ a_{m+1},\ldots,a_n)\ ,\ H \ \ \{c_1,\ldots,c_m\ ,\ a_{m+1},\ldots,a_n\})$. $A_1,\ldots,A_m$ can be written in terms of $a_{m+1},\ldots,a_n$ as following:

$$A_i = s_i + \sum_{j=1}^{n-m} \gamma_{i,j}\ a_{m+j} \qquad (2)$$

where $s_i \in F(c_1,\ldots,c_m)$, $\gamma_{i,j} \in G(c_1,\ldots,c_m)$. Setting now $b_1 \triangleq a_{m+1},\ldots,b_{n-m} \triangleq a_n$ we may say that A_m is an algorithm over $(F_c(b_1,\ldots,b_{n-m})\ ,\ H_c \cup \{b_1,\ldots,b_{n-m}\})$ which computes $\Psi'(\underline{b}) \triangleq \Psi(\underline{s} + \Gamma\underline{b})$ where $\Gamma = (\gamma_{i,j})$ is defined by

$$\gamma_{i,j} = \begin{cases} \gamma_{i,j} & \text{from equation (2)}; \quad 1 \le i \le m \\ 1 & ; \quad m < i \le n \ , \ j=i \\ 0 & ; \quad m < i \le n \ , \ j \ne i \end{cases}$$

Γ is an $n \times (n-m)$ matrix of rank $n-m$ and from here using Strassen's result it is possible to show that $m = \mu_D$. Thus $\mu(A_m) \le \mu(A) - \mu_D(A)$.

Now we set

$$u_i = 0 \quad v_i = 1 \qquad i = m+1,\dots,n$$

$$g_j^i = 0 \quad i = m+1,\dots,n \qquad j = i+1,\dots,n$$

$$A_i = c_i \qquad i = m+1,\dots,n \ .$$

Thus we have the following system of equations

$$\tilde{A}_i = u_i + c_i v_i + \sum_{j=i+1}^{n} g_j^i \tilde{A}_j \qquad i=1,\dots,n$$

where $u_i, v_i \in F(c_1,\dots,c_{i-1})$, $g_j^i \in G(c_1,\dots,c_{i-1})$ and where we introduce $\tilde{A}_i$ instead of A_i as we are actually dealing with a permutation of A_i . This system being an upper diagonal can be very easily solved so as to give the solution claimed by the theorem. The last algorithm constructed was A_m and it was over

$$(F(c_1,\dots,c_n,a_{m+1},\dots,a_n),\ H \cup \{c_1,\dots,c_n,a_{m+1},\dots,a_n\})$$

We substitute in it c_i for a_i , $j = m+1,\dots,n$ obtaining the required algorithm B (we can of course assume that it computed $u_{m+1},\dots,u_n,v_{m+1},\dots,v_n,\ \tilde{A}_{m+1},\dots,\tilde{A}_n$). Finally it also follows easily that if $\psi(\underline{a}) \in F(\underline{a}) - \{0\}$ then $\psi(\underline{A}) \in F_c - \{0\}$ and there are no attempts in B to divide by zero. □

Corollary 4. Under the assumptions of the theorem there exists B over $(F_c, H \cup \underline{c})$ computing $\underline{u} = (u_i | 1 \leq i \leq n)$, $\underline{v} = (v_i | 1 \leq i \leq n)$, $\underline{A} = (A_i | 1 \leq i \leq n)$ and $\Psi(\underline{A}) = \{\psi_j(\underline{A}) | 1 \leq j \leq t\}$ such that

(1) $\mu(B) \leq \mu(A) - \mu_D(A)$,

(2) Comp $(B) \in G_c(H \cup \{a,\ldots,a_n\})$,

(3) $A_i = u_i + c_{\iota(i)} v_i + \sum_{j=i+1}^{m} w_i^j (u_j + c_{\iota(j)} v_j)$,

(4) $u_i, v_i \in F(c_{\iota(1)},\ldots,c_{\iota(i-1)})$, $w_i^j \in G(c_{\iota(i)},\ldots,c_{\iota(i-1)})$,

(5) $\iota:\{1,\ldots,n\} \to \{1,\ldots,n\}$ is a permutation,

(6) if there were no SCDs in A then $v_i = 1$, $i=1,\ldots,n$,

(7) if there were no CDs in A then Comp $(B) \in G_c[H \cup \{a_1,\ldots,a_n\}]$.

Proof. Follows from Theorem 2 by giving the permutation of A explicitly. □

Corollary 5. Under the assumptions of Theorem 2 we may assume that there exist $\underline{\alpha} = (\alpha_i | 1 \leq i \leq n)$, $\underline{\beta} = (\beta_i | 1 \leq i \leq n)$ s.t. B computes $\underline{\alpha}$, $\underline{\beta}$, $\underline{A}$ and $\Psi(\underline{A})$ and

$$A_{\iota(k)} = \alpha_{\iota(k)} + \gamma_{\iota(k)} \, c_{\iota(\lambda(k))} \, \beta_{\iota(\lambda(k))} \quad 1 \leq k \leq n$$

where

$$\alpha_{\iota(k)} \in F(c_{\iota(1)},\dots,c_{\iota(\lambda(k)-1)}) ,$$

$$\beta_{\iota(\lambda(k))} \in F(c_{\iota(1)},\dots,c_{\iota(\lambda(k)-1)}) - \{0\} ,$$

$$\gamma_{\iota(k)} \in G(c_{\iota(1)},\dots,c_{\iota(\lambda(k)-2)}) - \{0\} ,$$

$$k \leq \lambda(k) \leq n$$

and

(1) if there were no SCDs in A then

$\beta_{\iota(\lambda(k))} = 1$,

(2) if there were no CDs in A then

$\underline{\text{Comp}}\ (B) \in G_c[H \cup \{a_1 ,\dots, a_n\}]$.

Proof. We use the form for $\underline{A}$ derived in Corollary 4 and discuss two cases

(1) $\{w_k^{k+1},\dots,w_k^n\} \neq 0$. Let

$\lambda(k) \triangleq \max\{j \mid w_k^j \neq 0\}$ and set

$$\alpha_{\iota(k)} \triangleq u_k + c_{\iota(k)}v_k + \sum_{j=k+1}^{\lambda(k)-1} w_k^j\,(u_j + c_{\iota(j)}v_j) + w_k^{\lambda(k)}u_{\iota(k)}$$

$$\beta_{\iota(\lambda(k))} \triangleq v_{\lambda(k)}$$

$$\gamma_{\iota(k)} \triangleq w_k^{\lambda(k)} ,$$

(2) $\{w_k^{k+1},\dots,w_k^n\} = 0$. Let $\lambda(k) \triangleq k$ and set

$\alpha_{i(k)} \in F(c_{i(1)}, \ldots, c_{i(\lambda(k)-1)})$,

$\beta_{i(\lambda(k))} \in F(c_{i(1)}, \ldots, c_{i(\lambda(k)-1)}) - \{0\}$,

$\gamma_{i(k)} \in G(c_{i(1)}, \ldots, c_{i(\lambda(k)-2)}) - \{0\}$,

$k \le \lambda(k) \le n$

and

(1) if there were no SCDs in A then

$\beta_{i(\lambda(k))} = 1$,

(2) if there were no CDs in A then

$\underline{\text{Comp}}(B) \in G_s[E \cup \{a_1, \ldots, a_n\}]$.

Proof. We use the form for $\underline{A}$ derived in Corollary 4 and discuss two cases

(1) $\{w_k^{k+1}, \ldots, w_k^n\} \neq 0$. Let

$\lambda(k) \triangleq \max\{j \mid w_k^j \neq 0\}$ and set

$$\alpha_{i(k)} \triangleq u_k + c_{i(k)} v_k + \sum_{j=k+1}^{\lambda(k)-1} w_k^j (u_j + c_{i(j)} v_j) + w_k^{\lambda(k)} \alpha_{i(k)}$$

$$\beta_{i(\lambda(k))} \triangleq v_{\lambda(k)} ;$$

$$\gamma_{i(k)} \triangleq w_k^{\lambda(k)} .$$

(2) $\{w_k^{k+1}, \ldots, w_k^n\} = 0$. Let $\lambda(k) \triangleq k$ and set

$$\alpha_{i(k)} \triangleq u_k ,$$

$$\beta_{i(\lambda(k))} \triangleq v_k$$

$$\gamma_{i(k)} \triangleq 1. \quad \square$$

4. Applications

From now on we assume that $F = G(x)$, $H = G \cup \{x\}$ where x is an indeterminate over G . In this case we shall also write sometimes explicity

$$A_{\iota(k)}(x, c_{\iota(1)},\ldots,c_{\iota(\lambda(k))}) = \alpha_{\iota(k)}(x,c_{\iota(1)},\ldots,c_{\iota(\lambda(k)-1)})$$

$$+ \gamma_{\iota(k)}(c_{\iota(1)},\ldots,c_{\iota(\lambda(k)-2)})c_{\iota(\lambda(k))}\ \beta(x,c_{\iota(1)},\ldots,c_{\iota(\lambda(k)-1)})$$

Furthermore we shall discuss only the case $\Psi(\underline{a}) = \Phi\underline{a} + \phi$ where Φ is a $t\times n$ matrix and ϕ a t-vector of elements in F (compare with Corollary 2). In order to be able to deal more uniformly with this case we set $a_0 \mathrel{\hat{=}} c_0$ and write $\Phi\underline{a} + \phi c_0 = \Phi\underline{a} + \phi a_0$ (thus, say multiplications by elements from $G(a_0) = G(c_0)$ won't be CMDs). Defining now a $t\times(n+1)$ matrix $\hat{\Phi}$ by

$$\hat{\Phi}_{i,j} = \begin{cases} \phi_i & ; \quad j=0 \\ \Phi_{i,j} & ; \quad \text{otherwise} \end{cases}$$

we may say that $\Psi(\underline{a}) \equiv \hat{\Phi}\underline{\hat{a}}$ where $\underline{\hat{a}} = (a_0,a_1,\ldots,a_n)$ If we set $A_0 = c_0$ all our results up to this point follow. We could have been more formal and replaced everywhere F by $F(a_0)$ etc., but it would serve no useful purpose.

For an element p of $G_c[x]$ (the set of polynomials in x over elements of G_c) we shall denote by $\partial(p)$ the degree (in x) of p .

<u>Definition 3</u>. Let $\hat{\Phi} = (\hat{\Phi}_{i,j})$ $i=1,\ldots,t;\ j=0,\ldots,n$ be a matrix in $G[x]$. Then define $Sp(\hat{\Phi})$, the sparsness index of $\hat{\Phi}$, as following

$$Sp(\hat{\Phi}) = \max_{1\le i\le t} \{\max \{\partial(\Phi_{i,\iota(n)}) - \partial(\Phi_{i,\iota(n-1)}) ,$$

$$\partial(\Phi_{i,\iota(n-1)}) - \partial(\Phi_{i,\iota(n-2)}), \ldots, \partial(\Phi_{i,\iota(1)})$$

$$-\partial(\Phi_{i,\iota(0)}) , \partial(\Phi_{i,\iota(0)}) \,|\, \iota{:}\{0,1,\ldots,n\} \to \{0,1,\ldots,n\}$$

is a permutation such that

$$\partial(\Phi_{i,\iota(n)}) \ge \partial(\Phi_{i,\iota(n-1)}) \ge \cdots \ge \partial(\Phi_{i,\iota(0)}\}$$

<u>Example 3</u>. Let $t=2$, $n=3$ and

$$\Phi = \begin{pmatrix} x^3+1 & x^4 & x \\ 1 & x^5 & x \end{pmatrix} \qquad \phi = \begin{pmatrix} x^3 \\ x^2 \end{pmatrix}$$

then

$$\hat{\Phi} = \begin{pmatrix} x^3 & x^3+1 & x^4 & x \\ x^2 & 1 & x^5 & x \end{pmatrix}$$

If the first row is rearranged in the order of descending degrees then it becomes x^4, x^3, x^3+1, x . Similarly for the second row: x^5, x^2, x, 1 . Thus $Sp(\hat{\Phi}) = \partial(x^5)-\partial(x^2)=3$.

<u>Theorem 3</u>. Let A compute $\hat{\Phi}\hat{\underline{a}}$ over $(G(x,\hat{\underline{a}})$, $G \cup \{x,\hat{\underline{a}}\})$. Without CDs (thus necessarily $\hat{\Phi} \subset G[x]$) . Then $\mu(A) \ge \mu_D(\hat{\Phi}\hat{\underline{a}}) + \lceil \log_2 Sp(\hat{\Phi}) \rceil$ provided that either

(1) All $\hat{\Phi}_{i,j}$ are monomials of the form $\prod_{k-1}^{n} a_k^{\alpha(k)}$ or,

(2) For every i the degrees of $\hat{\Phi}_{i,0},\ldots,\hat{\Phi}_{i,n}$ are distinct.

Lemma 3. Under the assumptions of Theorem 3 the rational functions $\underline{\hat{A}}$ are polynomials in x which do not vanish for $x=0$.

Proof. Let $0 \leq k \leq n$. Then $A_{\iota(k)}$ is a polynomial in x of the form

$$\alpha_{\iota(k)}(x,c_{\iota(1)},\ldots,c_{\iota(\lambda(k)-1)})$$

$$+\gamma_{\iota(k)}(c_{\iota(1)},\ldots,c_{\iota(\lambda(k)-2)})c_{\iota(\lambda(k)}\ .$$

If we substitute $x=0$ into $A_{\iota(k)}$ we obtain a nontrivial polynomial in $c_{\iota(\lambda(k))}$. □

Lemma 4. Let B compute $p \in G_c[x]$ over $(G_c(x),G_c \cup \{x\})$ without CDs. Then $\mu(B) \geq \lceil \log_2 \partial(p) \rceil$

Proof. Well known. □

Proof of Theorem 3. By Corollary 4 there exists B which computes $\underline{\hat{A}}$, $\hat{\Phi}\underline{\hat{A}}$,($\underline{\hat{A}} = (A_0,\ldots,A_n)$), s.t. $\mu(B) \leq \mu(A)-\mu_D(A)$, and it is enough to show that at least one of the $(n+1)+t$ polynomials $\underline{\hat{A}}$, $\hat{\Phi}\underline{\hat{A}}$ has the degree which is at least $Sp(\hat{\Phi})$. W.l.o.g. we may assume that $\partial(\hat{\Phi}_{i,o}) \leq \partial(\hat{\Phi}_{i,1}) \leq \ldots \leq \partial(\hat{\Phi}_{i,n})$ and $Sp(\hat{\Phi}) = \max\{\partial(\hat{\Phi}_{i,n})-\partial(\hat{\Phi}_{i,n-1}),\ldots,$

$\partial(\hat{\Phi}_{i,1}) - \partial(\hat{\Phi}_{i,0})$, $\partial(\hat{\Phi}_{i,0})\}$. Let $0 \leq k \leq n$ be such that $Sp(\hat{\Phi}) = \partial(\hat{\Phi}_{i,k}) - \partial(\hat{\Phi}_{i,k-1})$ (for completeness: $\partial(\hat{\Phi}_{i,-1}) \triangleq 0$). We can write

$$q \triangleq \sum_{j=0}^{n} A_j \hat{\Phi}_{i,j} = q_1 + q_2 \quad \text{where}$$

$$q_1 = \sum_{j=0}^{k-1} A_j \hat{\Phi}_{i,j} \quad , \quad q_2 = \sum_{j=k}^{n} A_j \hat{\Phi}_{i,j}$$

and we shall show that at least one of the polynomials (in x) $q, A_0, A_1, \ldots, A_n$ has the degree at least $Sp(\hat{\Phi})$. There are two cases:

(1) $\partial(q) \geq Sp(\hat{\Phi})$ and the result follows.

(2) $\partial(q) < Sp(\hat{\Phi})$. As by Lemma 3

$$\partial(q_2) \geq \partial(\hat{\Phi}_{i,k}) \geq \partial(\hat{\Phi}_{i,k}) - \partial(\hat{\Phi}_{i,k-1}) \geq Sp(\hat{\Phi})$$

It follows that $\partial(q_1) \geq \partial(\hat{\Phi}_{i,k})$. Thus for some j , $0 \leq j \leq k-1$,

$$\partial(\hat{A}_j) \geq \partial(q_2) - \partial(\hat{\Phi}_{i,j}) \geq \partial(\hat{\Phi}_{i,k}) - \partial(\hat{\Phi}_{i,k-1}) \geq Sp(\hat{\Phi})$$

and the result follows. □

<u>Corollary 5</u>. If under the assumption of Theorem 3 A computes $\sum_{i=1}^{n} a_i x^{\alpha(i)}$ for $0 < \alpha(1) < \ldots < \alpha(n)$ then $\mu(A) \geq n + \lceil \log_2 \max\{\alpha(n) - \alpha(n-1), \ldots, \alpha(2) - \alpha(1), \alpha(1)\} \rceil$.

<u>Proof</u>. Immediate as by Corollary 1 $\mu_D(A) = n$. □

Let $p(x) \in G(x)$. Then for every integer ε and a sufficiently large integer N there is a unique sequence $(\pi_j | \varepsilon \leq j \leq N) \subset \mathbf{G}$ such that

$$p(x) = \sum_{\varepsilon}^{N} \pi_j x^j + r(x)$$

where $\pi_j \in G$ and $r(x) = r_1(x)/r_2(x)$ where r_1, r_2 are polynomials satisfying $\partial(r_1) - \partial(r_2) < \varepsilon$.

<u>Definition 4</u>. Let $\underline{\alpha} = (\alpha(1), \ldots, \alpha(m))$ be an increasing sequence of integers, and let $\underline{p} = (p_1(x), \ldots, p_m(x))$ be a sequence of elements in $G(x)$. Then $\underline{p}$ is <u>α-normal</u> if and only if the determinant $\det(\pi^i_{\alpha(j)})$ $i=1,\ldots,m$, $j=1,\ldots,m$ is not zero where $p_i = \sum_{j=\alpha(1)}^{N} \pi^i_j x^j + r_{1,i}/r_{2,i}$ like above.

For a sequence $\underline{\alpha}$ like above we define $\mu(\underline{\alpha}) \triangleq \min\{$ CMDs required to compute a sequence $(q_1, \ldots, q_m) \subset G(x)$ $| (q_1, \ldots, q_m)$ is $\underline{\alpha}$-normal$\}$.

<u>Theorem 4</u>. Let A be an algorithm without SCDs over $(G(x,\underline{a})$, $G \cup \{x,\underline{a}\})$ computing the sequence $a_1 p_1(x), \ldots,$ $a_n p_n(x)$ where $p_1(x), \ldots, p_n(x)$ is a sequence of rational functions having an $\underline{\alpha}$-normal subsequence for some $\underline{\alpha} =$ $(\alpha(1), \ldots, \alpha(m))$. Then $\mu(A) \geq \mu_D(A) + \mu(\underline{\alpha})$.

<u>Proof</u>. After applying Corollary 4 (we can assume that $\iota(k) = k$) we have an algorithm B computing (among others) the sequence $(A_k p_k \mid 1 \leq k \leq n)$ where

$$A_k p_k = [\alpha_k(x, c_1, \ldots, c_{\lambda(k)-1})$$

$$+ \gamma_k(c_1, \ldots, c_{\lambda(k)-2})\, c_{\lambda(k)}]\, p_k(x) \qquad \text{for } 1 \leq k \leq n.$$

W.l.o.g. $(p_1, \ldots, p_m)$ was $\underline{\alpha}$-normal. We can expand

$$A_i p_i = \sum_{j=\alpha(1)}^{\overline{N}} \overline{\pi}_j^i(\underline{c}) x^j + \overline{r}_{1,i}(x)/\overline{r}_{2,i}(x)$$

where $\overline{\pi}_j^i \in G(\underline{c})$, $\overline{r}_{1,i}$, $\overline{r}_{2,i} \in G(x, \underline{c})$, $\partial(\overline{r}_{1,i}) - \partial(\overline{r}_{2,i}) < \alpha(1)$. Thus $\det(\overline{\pi}_{\alpha(j)}^i)$ is a rational function of $c_1, \ldots, c_n$ and it will suffice to show that it is a nontrivial function. From the above it follows that

$$\overline{\pi}_j^i = \pi_j^i[c_{\lambda(i)}\gamma_i(c_1, \ldots, c_{\lambda(i)-2}) + \zeta_{i,j}(c_1, \ldots, c_{\lambda(1)-1})]$$

$$= \pi_j^i\, c_{\lambda(i)}\gamma_i(c_1, \ldots, c_{\lambda(i)-2}) + \pi_j^i \zeta_{i,j}(c_1, \ldots, c_{\lambda(1)-1})$$

Using the well known formula concerning expansion of a determinant the elements of which are sums of pairs we have

$$\det(\overline{\pi}_j^i) = [\prod_{i=1}^{n} c_{\lambda(i)}\gamma_i(c_1, \ldots, c_{\lambda(i)-2})] \det(\pi_j^i)$$

$$+ \eta(c_1, \ldots, c_{\lambda(n)})$$

and from here by induction on n and analyzing η it is possible to show that indeed $\det(\overline{\pi}_j^i) \neq 0$. Thus $(A_k p_k \mid 1 \leq k \leq m)$ is $\underline{\alpha}$-normal and the result follows. □

A slightly weaker version of the following result was stated by Shaw and Traub [6] and used by them to prove the optimality of one of their algorithms.

<u>Corollary 6</u>. Let A compute $(a_1x_1,\ldots,a_nx_n)$ over $(G(x,\underline{a})$, $G \cup \{x,\underline{a}\})$ under the restrictions of Theorem 4. Then $\mu(A) \geq 2n-1$.

<u>Proof</u>. As $(a_1x,\ldots,a_nx^n) = \Phi\underline{a}$ where $\Phi_{i,j} = \delta_{i,j}x^j$ then by Corollary 2 $\mu_D(A) = n$. The subsequence $(x^2,\ldots,x^n)$ of length $n-1$ of $(x,x^2,\ldots,x^n)$ is $(2,\ldots,n)$ normal and therefore B computes a $(2,\ldots,n)$-normal sequence $A_2x^2,\ldots,A_nx^n$. We shall show that if B computes a $(2,\ldots,n)$-normal sequence, say $(p_1,\ldots,p_{n-1})$ then $\mu(B) \geq n-1$ and the result will follow. Assume that only $\ell < n-1$ CMDs appear in B . Then if $\gamma_1,\ldots,\gamma_\ell \in G_c(x)$ are the results of these CMDs then

$$p_i = \sum_{j=1}^{\ell} g_{i,j}\,\gamma_j + \delta_j \qquad i=1,\ldots,n-1$$

where $g_{i,j} \in G_c$ and $\delta_j \in G_c[x]$ and are linear in x (δ_j were computed without CMDs). Dropping terms linear in x we have

$$\bar{p}_i = \sum_{j=1}^{\ell} \bar{g}_{i,j}\,\bar{\gamma}_j \qquad i=1,\ldots,n-1$$

Now $(\bar{p}_1,\ldots,\bar{p}_{n-1})$ is $(2,\ldots,n)$ – normal and therefore linearly independent. Thus it cannot be spanned by a linear combination of $\ell < n-1$ elements $\bar{\gamma}_1,\ldots,\bar{\gamma}_\ell$. Thus $\mu(B) \geq n-1$ and the result follows. □

As the next application of our method we sketch briefly how we can prove Brodin's result [1] that Horner's rule is uniquely optimal.

Theorem 5. (Borodin) Horner's rule it the only algorithm which uses n MDs and $n-1$ ASs (additions and subtractions) to compute $\sum_{i=1}^{n} a_i x^i$ over $(G(x,\underline{a}), G \cup \{x,\underline{a}\})$, and thus is the uniquely optimal algorithm computing $\sum_{i=1}^{n} a_i x^i$.

Proof. (sketch). It is known that n SCMDs and $n-1$ ASs are required. If we apply our method of substitution to Horner's rule (which can be described informally as

$$\sum_{i=1}^{n} a_i x^i = (\ldots(a_n x + a_{n-1})x + \ldots + a_1) x \quad)$$

we note that $\iota(1) = n, \ldots, \iota(n) = 1$, B computes
$\sum_{i=1}^{n} A_i(x) x^i = -(c_2 x + c_1) x + (-c_3 x + c_2) x^2 + \ldots + (-c_n x + c_{n-1}) x^{n-1} + c_n x^n = c_1 x$ and $A_n = c_n$,
$A_i = -c_{i+1} x + c_i \quad i=1,\ldots,n-1$.

Let now A be an arbitrary algorithm computing $\Sigma a_i x^i$ which has exactly n CMDs (all of them necessarily SCMDs). Thus $\mu(B) \leq \mu(A) - n = n-n = 0$ and therefore $\Sigma A_i(x) x^i$, $A_1(x), \ldots, A_n(x)$ are necessarily linear in x. In addition to this using the facts that A had no WCMDs or NCMDs and exactly $n-1$ ASs it is relatively easy to show the claimed result. □

A special case of the following "decomposition" theorem was first proved by Winograd [11].

Theorem 6. Let A over $(F(\underline{a}), H \cup \underline{a})$ compute $\Psi_1(\underline{a}) \cup \Psi_2$ where $\Psi_1(\underline{a}) \subset F(\underline{a})$ and $\Psi_2 \subset F$. Let $\mu(\Psi_2)$ be (as usual) the minimum number of CMDs required to compute Ψ_2 over $(F(\underline{a}), H \cup \underline{a})$. Then $\mu(A) \geq \mu_D(A) + \mu(\Psi_2) = \mu_D(\Psi_1(\underline{a})) + \mu(\Psi_2)$.

Proof. Using Theorem 2 we obtain B computing $\underline{A}$, $\Psi_1(\underline{A}) \cup \Psi_2$. Thus $\mu(\Psi_2) \leq \mu(B) \leq \mu(A) - \mu_D(A)$ and the theorem follows. □

We now give an example without making the assumptions appearing at the beginning of this section.

Example 4. Let A be an algorithm over $(\mathbb{R}(\underline{a}), \mathbb{Q} \cup \{\rho, \underline{a}\})$ computing $\rho^m \sum_{i=1}^{n} a_i^2$ without CDs where $\rho = 2^{1/M}$ and $M > (m+1)2^n$. We shall show that $\mu(A) \geq n + \lceil \log_2 m \rceil$. Assume that $\mu(A) < n + \lceil \log_2 m \rceil < \log_2 M$. Then (compare with Lemma 4) it is possible to conclude that we may think of ρ as if it were an indeterminate over $\mathbb{Q}$. Now, using essentially the same method which Strassen [7] used to prove that an algorithm computing $\sum_{i=1}^{n} a_i^2$ over $(\mathbb{R}(\underline{a}), \mathbb{R} \cup \underline{a})$ requires at least n MDs, we can show that $\mu_D(A) = n$. The corresponding B computes $\rho^m \Sigma A_i^2$ which is a polynomial in ρ over $\mathbb{Q}_c$. While proving Theorem 2 we noted that if $\psi(\underline{a}) \in F(\underline{a}) - \{0\}$ then $\psi(\underline{A}) \in F_c - \{0\}$. Thus $\sum_{i=1}^{n} A_i^2$ is a nontrivial polynomial in ρ and $\rho^m \Sigma A_i^2$ is a polynomial in ρ of degree m at least. Thus $\mu(B) \geq \lceil \log_2 m \rceil$ and the result follows.

Theorem 6. Let A over $(F(\underline{a}), H \cup \underline{a})$ compute $\Psi_1(\underline{a}) \cup \Psi_2$ where $\Psi_1(\underline{a}) \subset I(\underline{a})$ and $\Psi_2 \subset F$. Let $\mu(\Psi_2)$ be (as usual) the minimum number of CMDs required to compute Ψ_2 over $(F(\underline{a}), H \cup \underline{a})$. Then

$$\mu(A) \geq \mu_D(A) + \mu(\Psi_2) = \mu_D(\Psi_1(\underline{a})) + \mu(\Psi_2).$$

Proof. Using Theorem 2 we obtain B computing $\underline{A}$, $\Psi_1(\underline{A}) \cup \Psi_2$. Thus $\mu(\Psi_2) \leq \mu(B) \leq \mu(A) - \mu_D(A)$ and the theorem follows. □

We now give an example without making the assumptions appearing at the beginning of this section.

Example 4. Let A be an algorithm over $(R(\underline{a}), Q \cup \{\rho, \underline{a}\})$ computing $\rho^m \sum_{i=1}^{n} a_i^2$ without CDs where $\rho = 2^{1/M}$ and $M > (m+1)2^n$. We shall show that $\mu(A) \geq n + \lceil \log_2 m \rceil$.
Assume that $\mu(A) < n + \lceil \log_2 m \rceil < \log_2 M$. Then (compare with Lemma 4) it is possible to conclude that we may think of ρ as if it were an indeterminate over Q. Now, using essentially the same method which Strassen [7] used to prove that an algorithm computing $\sum_{i=1}^{n} a_i^2$ over $(R(\underline{a}), R \cup \underline{a})$ requires at least n MDs, we can show that $\mu_D(A) = n$. The corresponding B computes $\rho^m \Sigma A_i^2$ which is a polynomial in ρ over Q_c. While proving Theorem 2 we noted that if $\psi(\underline{a}) \in F(\underline{a}) - \{0\}$ then $\psi(\underline{A}) \in F_c - \{0\}$. Thus $\sum_{i=1}^{n} A_i^2$ is a nontrivial polynomial in ρ and $\rho^m \Sigma A_i^2$ is a polynomial in ρ of degree m at least. Thus $\mu(B) \geq \lceil \log_2 m \rceil$ and the result follows.

5. Conclusions

We have described a new method for establishing lower bounds on the number of multiplications and divisions. Although most of the applications were given in a rather restricted setting (linear functions of indeterminates were considered) the basic theorem was formulated in a gneeral framework. Actually some common but unnecessary assumptions were made (e.g. it was not necessary to assume that $a_1, \ldots, a_n$ were indeterminates, and weaker assumptions would have been sufficient). As the rate of growth arguments do not handle divisions as easily as multiplications, our results were also deficient in this manner. It seems to us that at least some of our results can be strengthed by studying $\underline{A}$ and $\Psi(\underline{A})$ more carefully. It seems that the method, also very useful in some cases, is inherently limited to proving lower bounds of the form $0(n)$. We would like to mention also that it seems that inherently the same method can be used to handle additions and subtractions. Unfortunately, rate of growth arguments for additions and subtractions are almost nonexistent.

5. Conclusions

We have described a new method for establishing lower bounds on the number of multiplications and divisions. Although most of the applications were given in a rather restricted setting (linear functions of indeterminates were considered) the basic theorem was formulated in a general framework. Actually some common but unnecessary assumptions were made (e.g. it was not necessary to assume that $a_1, \ldots, a_n$ were indeterminates, and weaker assumptions would have been sufficient). As the rate of growth arguments do not handle divisions as easily as multiplications, our results were also deficient in this manner. It seems to us that at least some of our results can be strengthed by studying $\underline{A}$ and $V(\underline{A})$ more carefully. It seems that the method, also very useful in some cases, is inherently limited to proving lower bounds of the form $0(n)$. We would like to mention also that it seems that inherently the same method can be used to handle additions and subtractions. Unfortunately, rate of growth arguments for additions and subtractions are almost nonexistent.

ACKNOWLEDGMENT

The author wishes to thank Dr. S. Winograd for his encouragement and many helpful suggestions.

ACKNOWLEDGMENT

The author wishes to thank Dr. S. Winograd for his encouragement and many helpful suggestions.

REFERENCES

1. Borodin A. Horner's rule is uniquely optimal, "Theory of machines and computations" ed. by Z. Kohavi and A. Paz, Academic Press, New York (1971), 45-58.

2. Brockett R.W. and Dobkin, D., On the optimal evaluation of a set of bilinear forms, Proc. of Fifth Annual Symposium on Theory of Computing, (1973), 88-95.

3. Hopcroft J. and Musinski J., Duality applied to the complexity of matrix multiplication and other bilinear forms, SIAM J. Comp. 2 (Sept. 1973), 159-173.

4. Ostrowski A.M., On two problems in abstract algebra connected with Horner's rule, "Studies in Mathematics and Mechanics", Academic Press, New York (1954), 40-48.

5. Pan V.Ya., Methods of computing values of polynomials, Russian Math. Surveys 21 (1966), 105-136.

6. Shaw M. and Traub J.F. On the number of multiplications for the evaluation of a polynomial and some of its derivatives, J. ACM 21 (Jan. 1974), 161-167.

7. Strassen V. Evaluation of rational functions, "Complexity of Computer computations" ed. by R.E. Miller and J.W. Thatcher, Plenum Press, New York (1972), 1-10.

8. Strassen V. Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten, Nu. Math. 20 (1973), 238-251.

9. Winograd S. On the number of multiplications required to compute certain functions, Proc. Nat. Acad. Sci. U.S.A. 58 (1967), 1840-1842.

10. Winograd S. On the number of multiplications necessary to compute certain functions, Comm. Pure Appl. Math. 23 (1970), 165-179.

11. Winograd S., On the parallel evaluation of certain arithmetic expressions, RC 4808, IBM Thomas J. Watson Research Center (April 15, 1974), 1-35.

BIBLIOGRAPHIC DATA SHEET	1. Report No. NSF-OCA-GJ34671 - TM -46	2.	3. Recipient's Accession No.
4. Title and Subtitle: Combining Dimensionality and Rate of Growth Arguments for Establishing Lower Bounds on the Number of Multiplications			5. Report Date: Issued June 1974
			6.
7. Author(s): Zvi M. Kedem			8. Performing Organization Rept. No. MAC TM- 46
9. Performing Organization Name and Address: PROJECT MAC; MASSACHUSETTS INSTITUTE OF TECHNOLOGY: 545 Technology Square, Cambridge, Massachusetts 02139			10. Project/Task/Work Unit No.
			11. Contract/Grant No. GJ34671
12. Sponsoring Organization Name and Address: Associate Program Director, Office of Computing Activities, National Science Foundation, Washington, D. C. 20550			13. Type of Report & Period Covered: Interim Scientific Report
			14.

15. Supplementary Notes

A preliminary version of this report appeared in the Proceedings of the Sixth Annual Symposium on the Theory of Computing, 1974.

16. Abstracts

A new method for establishing lower bounds on the number of multiplications and divisions required to compute rational functions is described. The method is based on combining two known methods, dimensionality and rate of growth. The method is applied to several problems and new lower bounds are obtained.

17. Key Words and Document Analysis. 17a. Descriptors

Algebraic operations
Analysis of algorithms
Computational complexity
Dimensionality
Lower bounds
Multiplications
Optimality
Polynomials
Rate of growth
Rational functions

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Approved for Public Release; Distribution Unlimited	UNCLASSIFIED	41
	20. Security Class (This Page) UNCLASSIFIED	22. Price