On the Size Complexity of
Monotone Formulas

Marc Snir
Computer Science Department
Edinburgh University
Edinburgh, Scotland.

Abstract

"Monotone" formulas, i.e. formulas using positive constants, additions and multiplications are investigated. Lower bounds on the size of monotone formulas representing specific polynomials (permanent, matrix multiplication, symmetric functions) are achieved, using a general, dynamic programming approach. These bounds are tight for the cases investigated. Some generalizations are suggested.

1. Introduction

A long standing goal of the research in the area of computational complexity is to give a precise estimation on the number of operations needed to compute specific arithmetic functions. Whereas for certain problems (as polynomial evaluation) such estimation has been achieved, other cases, for example matrix multiplication, escape analysis, despite continuous efforts.

One approach used to cope with this situation is to restrict the family of allowed algorithms in some sensible way, such that within this family tight bounds can be proved. Thus Miller [Mi], by requiring that computations fulfill some formal "numerical stability" requirements, and Schnorr [Sc], by considering "monotone" computations, i.e. computations using only additions and multiplications (a restriction which implies Miller's one) were able to prove that the straightforward algorithm for the multiplication of matrices which uses \(O(n^3)\) operations is optimal.

A numerical algorithm (producing one output) can be represented by a formula involving constants, variables (inputs) and operations. Different algorithms computing the same function over some fixed field are represented by equivalent formulas, so that given one such formula one can get all the others by applying transformations corresponding to the standard axioms of the ring of polynomials over this field. Restricting the family of allowable algorithms amounts to restricting the set of allowable transformations, i.e. using only a partial system of axioms. Thus, use of monotone computations only is tantamount to prohibition of the use of the cancellation axiom \(x + (-x) = 0\). Under such approach, the problem of finding the shortest restricted algorithm computing a given function is equivalent to the problem of finding the "simplest" formula equivalent to a given one under a restricted set of axioms.

This level of abstraction achieved, it is easy to see that results obtained for
monotone arithmetic computations are in fact valid for other algebras as well: + can be interpreted as minimum and * as addition, yielding the min, + semiring which is often used to formulate and solve optimization problems (see [AHU], p.195), or + can be union and * concatenation, leading to the consideration of star free regular expressions.

Two different ways of achieving this level of generality can be found in [ShS], where bounds are obtained on the depth complexity of formulas, and in [JeS], where the number of multiplications in straight line algorithms is measured.

In this paper we shall be concerned with the size complexity of formulas. In order to simplify the presentation, we shall state our results for monotone arithmetic formulas only. Nevertheless, it should be borne in mind that all results proven in section 3 applies to the above mentioned systems as well.

One of the results is that the shortest monotone formula for the permanent of a $n \times n$ matrix has size $2^{2n-0.25\log^2 n} + O(\log n)$. This result implies a lower bound of $2n - 0.25\log^2 n + O(\log n)$ steps for a parallel monotone computation of the permanent (this bound is tight). The minimal length of a monotone straight line computation of the permanent is $O(n^2 n)$. When subtraction is used without restrictions the upper bounds can be improved to $n^{2n-1}$ for size, $n + O(\log n)$ for depth, and $O((n-1)^2 n)$ for s.l.a. complexity, using the inclusion-exclusion formula of [Ry]. No non-trivial lower bounds are known in this case, although the results of [Va2] and [Va3] lead us to expect for exponential complexity. We remark that in the min, + algebra the permanent function represents the value of a minimal matching in a bipartite graph and is thus related to the "minimal assignment" optimization problem.

Two other problems considered in this paper are multiplication of $n \times n$ matrices (formula size $(2n)^{\log p}$), and evaluation of symmetric functions on $n$ variables (formula size $n^{0.25\log n + O(1)}$ in the worst case). For both problems we are not aware of any significant reduction in formula size complexity when subtraction is used, although subtraction can be used to generate shorter straight line computations of these functions.

2. Main Result

Let $S = \langle R^+, +, \ast \rangle$ be the semiring of positive real numbers with the operations of addition and multiplication; let $X$ be a fixed (infinite) set of variables. We denote by $P$ the semiring $R[X]$ of monotone polynomials over $S$. Each polynomial $p \in P$ has an unique canonical representation $p = \bigoplus_{i=1}^{\infty} r_i m_i$, where $m_i$ are different monomials, and $r_i \in R^+$. We denote by $\text{Mon}(p)$ the set of monomials appearing in this representation, and define $w(p)$, the weight of $p$, to be the cardinality of $\text{Mon}(p)$. The degree of $p$, $d(p)$ is defined as usual.

Polynomials in $P$ are represented by (monotone) formulas: $F$ is a formula if $F = a$, where $a \in R^+ \cup X$, or
\[ F = (F_1 \text{ op } F_2), \text{ where } F_1, F_2 \text{ are formulas, and } \text{ op} \in \{+,*\}. \]

We shall omit subsequently superfluous parentheses when writing down formulas.

To each formula \( F \) we associate a polynomial \( \text{poly}(F) \) in the usual way. The degree of a formula \( F \), \( \text{d}(F) \), the set of its monomials, \( \text{Mon}(F) \), and its weight \( \text{w}(F) \) are defined exactly as \( \text{d}(\text{poly}(F)), \text{Mon}(\text{poly}(F)), \text{w}(\text{poly}(F)) \) respectively.

The formula \( F \) is a subformula of \( G \) if
\[ F = G, \text{ or } \]
\[ G = (G_1 \text{ op } G_2) \text{ and } F \text{ is a subformula of } G_1 \text{ or } G_2. \]

We denote by \( \text{Sub}(G) \) the set of subformulas of \( G \).

A formula is atomic if it consists of an unique constant or variable (i.e. has no operation symbols within it). The size of a formula \( F \), \( s(F) \), is the number of occurrences of variables in \( F \); the height of \( F \), \( h(F) \), is defined inductively by
\[ h(F) = 0 \text{ if } F \text{ is atomic,} \]
\[ h(F_1 \text{ op } F_2) = 1 + \max[h(F_1), h(F_2)]. \]

Clearly, \( h(F) \approx \lg s(F) \) (we denote by \( \lg x \) the logarithm to basis 2 of \( x \)).

Since
\[ \text{Mon}(F_1 + F_2) = \text{Mon}(F_1) \cup \text{Mon}(F_2) \quad (2.1) \]
and
\[ \text{Mon}(F_1 \times F_2) = \{m_1 \times m_2 : m_i \in \text{Mon}(F_i), i = 1,2\}, \quad (2.2) \]
the weight function fulfills
\[ w(a) = 1 \text{ if } a \text{ is a variable, } 0 \text{ if } a \text{ is constant} \quad (2.3) \]
\[ w(F_1 + F_2) \leq w(F_1) + w(F_2); \quad (2.4) \]
and
\[ w(F_1 \times F_2) \leq w(F_1) \times w(F_2). \quad (2.5) \]

For the degree function we have
\[ d(a) = 1 \text{ if } a \text{ is a variable, } 0 \text{ if } a \text{ is constant} \quad (2.6) \]
\[ d(F_1 + F_2) = \max[d(F_1), d(F_2)]; \quad (2.7) \]
and
\[ d(F_1 \times F_2) = d(F_1) + d(F_2). \quad (2.8) \]

A simple, but crucial fact is provided by the next lemma.

**Lemma 2.1:** If \( G \) is a subformula of \( F \) then there exist polynomials \( p,q,p \neq 0 \) such that
\[ \text{poly}(F) = p \cdot \text{poly}(G) + q. \]

In particular
\[ \text{Mon}(F) \supseteq \text{Mon}(G) \times \text{Mon}(p). \quad (2.9) \]

This last relation (which breaks down as soon as cancellation is allowed) puts severe limitations on the structure of formulas representing a given polynomial. We
shall use these limitations (or more exactly a quantitative formulation of them) in order to prove lower bounds on formula size.

Let us introduce the following definitions:

The **complement** of a monomial \( m \) with respect to the polynomial \( p \) is
\[
\text{Comp}(m,p) = \{n : m \cdot n \in \text{Mon}(p)\}.
\]

The **growth function** of a polynomial \( p \) is defined by
\[
\text{Gr}(k,p) = \max_{d(m)=k} \max_{n \in \text{Comp}(m,p)} |\text{Comp}(n,p)|.
\]

(\( \text{Gr}(k,p) = -\infty \) if maximum is taken over an empty set.)

We shall denote by \( \text{Comp}(m,F) \) and \( \text{Gr}(k,F) \) the functions \( \text{Comp}(m,\text{poly}(F)) \) and \( \text{Gr}(k,\text{poly}(F)) \) respectively.

If \( G \) is a subformula of \( F \) then, according to lemma 2.1, \( \text{Mon}(G) \cdot n \subseteq \text{Mon}(F) \) for some monomial \( n \). If \( n \in \text{Mon}(G) \), then \( n \in \text{Comp}(m,F) \). Also, \( \text{Mon}(G) \subseteq \text{Comp}(n,F) \).

It follows, from the definition of the growth function, that \( w(G) \leq \text{Gr}(d(m),F) \). We have shown that

**Lemma 2.2:** If \( G \) is a subformula of \( F \) containing a monomial of degree \( k \), then \( w(G) \leq \text{Gr}(k,F) \).

In particular \( \text{Gr}(k,F) \) is an upper bound on the weight of a subformula of \( F \) of degree \( k \).

It is also easy to check that
\[
\text{Gr}(n,p) = w(p), \text{ if } p \text{ contains a monomial of degree } n, \quad (2.10)
\]
\[
\text{Gr}(0,p) = 1, \text{ provided that no monomial of } p \text{ is a factor of another, and} \quad (2.11)
\]
\[
\text{Gr}(k,p) \text{ is a monotonic non decreasing function of } k. \quad (2.12)
\]

We can now view the task of building a minimal size formula representing a polynomial \( p \) as a dynamic programming problem of building a formula of weight \( w(p) \) and degree \( d(p) \) under the constraints provided by lemma 2.2. and those implicit in 2.3 - 2.8. Using this approach we obtain lower bounds for formula size which depends only on the behaviour of the growth function \( \text{Gr}(\cdot,p) \). We ignore of course by such a quantitative approach, many of the restrictions applying to the original problem, but are still capable in some cases to achieve tight lower bounds.

Let \( F \) be a fixed formula. Define \( w(i,j) \) to be the maximal weight of a subformula of \( F \) of degree \( i \) and size \( j \) \( (w(i,j) = -\infty \) if there is no such subformula). We have the following relations:

(i) \( w(i,1) = 1 \) if \( i = 1 \), \( -\infty \) otherwise; \quad (2.13)
(ii) If \( j > 1 \) then either \( w(i,j) = -\infty \) or there exist numbers \( i_1, i_2, j_1, j_2 \) such that
\[ j_1, j_2 \geq 1, \quad j_1 + j_2 = j, \quad \text{and either} \]
\[ i = \max\{i_1, i_2\} \quad \text{and} \quad w(i, j) \leq w(i_1, j_1) + w(i_2, j_2) \quad \quad (2.14) \]
\text{(addition case), or}
\[ i = i_1 + i_2 \quad \text{and} \quad w(i, j) \leq w(i_1, j_1) \cdot w(i_2, j_2) \quad \quad (2.15) \]
\text{(multiplication case).}

(iii) \[ w(i, j) \leq Gr(i, j) \quad \quad (2.16) \]

Indeed, (i) and (ii) are immediate consequences of relations 2.3 - 2.8, whereas (iii) follows from lemma 2.2.

Define now the function \( W(i, j) \) on the integers inductively by
\[ W(i, 1) = 1 \quad \text{if} \quad i = 1, \quad -\infty \quad \text{otherwise} \]
and for \( j > 1 \)
\[ W(i, j) = \min\{Gr(i, F), \max\{ \max (W(i, j_1) + W(i, j_2)), \]
\[ \max (W(i_1, j), W(i_2, j_2)) \}\} \]

We have
\[ i_1 + i_2 = i, j_1 + j_2 = j \]

**Lemma 2.3:** (i) The function \( W(i, j) \) is monotonic non decreasing in its first variable.

(ii) \[ w(i, j) \leq W(i, j) \quad \text{for all} \quad i, j > 0. \]

**Proof:** Claim (i) follows easily by induction on \( j \), using the fact that the function \( Gr(\cdot, F) \) is monotonic non decreasing; claim (ii) is proved by induction on \( j \):

If \( j = 1 \) then \( w(i, j) = W(i, j) \) from 2.13 and the definition of \( W \);

If \( j > 1 \) then either according to 2.14 \( w(i, j) \leq w(i_1, j_1) + w(i_2, j_2) \),

with \( i = \max\{i_1, i_2\} \quad \text{and} \quad j = j_1 + j_2 \), in which case, using the inductive assertion and (i) we have
\[ w(i, j) \leq W(i_1, j_1) + W(i_2, j_2) \leq W(i, j_1) + W(i_2, j_2) \quad \quad (2.17) \]
or according to 2.15 \( w(i, j) \leq w(i_1, j_1) \cdot w(i_2, j_2) \), with \( i = i_1 + i_2 \), and \( j = j_1 + j_2 \), so that
\[ w(i, j) \leq W(i_1, j_1) \cdot W(i_2, j_2). \quad \quad (2.18) \]

The claim now follows by comparing inequalities 2.16, 2.17, 2.18 to the equations which define the function \( W \).

The value of \( W(i, j) \) is an upper bound on the weight of a subformula of \( F \) of degree \( i \) and size \( j \). In particular we have that \( W(d(F), s(F)) \geq w(F) \), from which it follows that \( s(F) \geq \min\{j: W(d(F), j) \geq w(F)\} \). We have obtained a lower bound on the size of \( F \) which value hinges on the values of \( w(F), d(F) \), and on the behaviour of the function \( W(i, j) \), which in its turn hinges on the behaviour of the growth function \( Gr(\cdot, F) \), which is just the growth function \( Gr(\cdot, p) \) where \( p \) is the polynomial represented by \( F \).
Since \( w(F) = Gr(d(F), F) \) we can rephrase this result as follows:

**Theorem 2.4:** Let \( F \) be a formula representing the polynomial \( p \); let the function \( W(i, j) \) be defined by
\[
W(i, 1) = 1 \quad \text{if } i = 1, \ -\infty \text{ otherwise}
\]
and for \( j > 1 \)
\[
W(i, j) = \min(Gr(i, p), \max_{j_1 + j_2 = j} (W(i, j_1) + W(i, j_2)), \max_{i_1 + i_2 = i} (W(i_1, j_1) \cdot W(i_2, j_2)))
\]
Let the threshold function \( t \) be defined by
\[
t(1) = \min\{j : W(i, j) = Gr(i, p)\}
\]
Then
\[
s(F) \geq t(d(F)).
\]

Thus our strategy for proving lower bounds on the length of any formula representing a given polynomial is to find a closed definition of its growth function, and then a closed definition for the related threshold function. It turns out that in the interesting cases a direct inductive definition can be provided for a function that closely bounds from below the threshold function. Indeed, the dynamic programming problem we consider is interesting only when the boundary constraint provided by the growth function comes into play. This turns out to be the case when this function is submultiplicative, i.e. fulfills
\[
Gr(i+j,F) \geq Gr(i,F) \cdot Gr(j,F) \quad \text{for all } i, j > 0.
\]

When such is the case, the product of two maximal weight formula results in a formula with less than maximal weight, and additions must be intertwined between multiplications.

**Theorem 2.5:** Let \( Gr(\cdot,p) \) be submultiplicative.

Let the functions \( W(i, j) \) and \( t(i) \) be defined as in theorem 2.4, and let the function \( T(i) \) be defined by
\[
T(1) = 1
\]
and for \( k > 1 \)
\[
T(k) = \min_{i+j=k} \left[ T(i) \cdot Gr(i, p) + T(j) \cdot Gr(j, p) \right] / \left[ Gr(i, p) \cdot Gr(j, p) \right]
\]
Then
\[
(i) j \geq T(i) \cdot W(i, j) \quad \text{for all } i, j > 0.
\]
\[
(ii) t(i) \geq T(i) \cdot Gr(i, p) \quad \text{for all } i > 0.
\]

The idea behind the definition of \( T \), and the assertion of the theorem is that an (almost) optimal way of building formulas, according to our dynamic programming model, is to build for each \( k \) a maximal weight formula of degree \( k \) by repeatedly adding to itself a subformula of degree \( k \) with a minimal size to weight ratio.
This subformula is built by multiplying two subformulas of degree $i$ and $j$, with $i + j = k$, of maximal weight (and minimal size to weight ratio). $T(i)$ is the minimal size to weight ratio for a formula of degree $i$. The discrepancy between $t(i)$ and $T(i) \cdot G(i,p)$ is due solely to the impossibility of adding "fractional" formulas, i.e. to the replacing of an integer valued programming problem by a real valued one.

**Proof of theorem 2.5:** (i) is proved by induction on $<i,j>$ (in lexicographic order). It is clearly sufficient to prove the assertion for $j \leq t(i)$. If $i = 1$ then $W(1,j) = j$, and the claim follows. Suppose $i > 1$; if $W(i,j) = -\infty$ the claim is trivial; if $W(i,j) \leq W(i,j_1) + W(i,j_2)$, with $j_1 + j_2 = j$, then, using the inductive assertion

$$j = j_1 + j_2 \geq T(i)W(i,j_1) + T(i)W(i,j_2) \geq T(i)W(i,j);$$

If $W(i,j) \leq W(i,j_1) \cdot W(i_2,j_2)$, where $i_1 + i_2 = i$, $j_1 + j_2 = j$, then

$$T(i) \leq T(i_1)/Gr(i_1,p) + T(i_2)/Gr(i_2,p) \quad \text{by the definition of } T,$$

$$\leq T(i_1)/W(i_2,j_2) + T(i_2)/W(i_1,j_1)$$

$$= \frac{T(i_1)W(i_1,j_1) + T(i_2)W(i_2,j_2)}{W(i_1,j_1) \cdot W(i_2,j_2)} \quad \text{by the inductive assertion}$$

$$\leq \frac{W(i_1,j_1)}{W(i_1,j_1)} \cdot \frac{W(i_2,j_2)}{W(i_2,j_2)}$$

So that

$$j \geq T(i) \cdot W(i,j)$$

(ii) Follows from the relations

$$t(i) \geq T(i) \cdot W(i,t(i)) = T(i) \cdot Gr(i,p).$$

3. Applications

We shall apply now the results of section 2 to specific functions. The first case we consider is the permanent polynomial which is defined on the matrix 

$$(x_{ij}), \quad 1 \leq i, j \leq n$$

by

$$\text{per} = \sum_{\sigma \in S_n} x_{1\sigma(1)} \ldots x_{n\sigma(n)},$$

where $S_n$ is the group of permutations over $\{1, \ldots, n\}$.

**Claim:** The growth function of $\text{per}$ is $Gr(k, \text{per}) = k!$

**Proof:** Let $m$ be a monomial of degree $k$. If $r \in \text{Comp}(m, \text{per})$ then $r$ is a monomial of degree $n - k$ with no row index or column index in common with $m$, and $\text{Comp}(r, \text{per})$ is in fact the set of monomials of the permanent of the $k \times k$ submatrix which row and column indices are those of $m$.

**Theorem 3.1:** If $F$ is a formula representing the permanent polynomial then

$$s(F) \geq 2^{2n - 0.25lg^2 n} + o(1/n)$$

**Proof:** The factorial function is submultiplicative, so that theorem 2.5 applies. In order to prove our theorem it is sufficient to show that the function
\[ L(n) = 2^{2n-0.25} \lg^2 n + o(1/n) \] fulfills the inequalities

\[ L(1) \leq 1 \]
\[ L(i + j) \leq [L(i) + L(j)] \cdot Gr(i + j, \text{per}) / [Gr(i, \text{per}) \cdot Gr(j, \text{per})] \]

Indeed, in such a case \( L(k) / Gr(k, \text{per}) \) is a lower bound to \( T(k) \), and \( L(n) \) a lower bound to \( t(n) \), which in turn, is a lower bound to the size of a formula representing per.

The first inequality can be fulfilled by a suitable choice of the 0 term. It remains to prove that for \( n = i + j \)

\[ L(n) \leq \left( \binom{n}{i} \right) [L(i) + L(j)] . \]

We assume w.l.o.g. that \( i \leq n/2 \). It is easily checked that the function

\[ g(i) = \left( \binom{n}{i} \right) [L(i) + L(n - i)] \]

is unimodal in the interval \( 1 \leq i \leq n/2 \), and the inequality holds at both extremities of the interval. \[ \square \]

The above result is almost optimal. Indeed, using the Laplace method to compute the permanent (see [ShS]), one gets a formula of length \( 2^{2n-0.25} \lg^2 n + o(1/n) \).

Let us turn now to another example - multiplication of matrices.

Define

\[ \text{mat} = \sum_{1 \leq i_1, \ldots, i_p \leq m} x_{i_1 1} \cdot x_{i_2 2} \cdot \ldots \cdot x_{i_p p} \]

The polynomial mat is one of the coefficients of the product of \( p \) \( m \times m \) matrices.

**Theorem 3.2:** If \( F \) is a formula representing the polynomial mat then

\[ s(F) \geq (2m)^{\lg p} \]

**Proof:** The growth function of mat is given by \( Gr(k, \text{mat}) = m^{k-1} \). This function is submultiplicative. So, we use the same method as in the previous proof, and set to prove, for \( n = i + j \), the inequality

\[ (2m)^{\lg n} \leq [(2m)^{\lg i} + (2m)^{\lg j}] m^{n-1} / [m^{i-1} \cdot m^{j-1}] \]

or

\[ (2m)^{\lg n} \leq m[(2m)^{\lg i} + (2m)^{\lg j}] \]

Since the expression on the right is minimal when \( i = j \) it is sufficient to prove that

\[ (2m)^{\lg n} \leq m[(2m)^{\lg (n/2)} + (2m)^{\lg (n/2)}] \]

and both sides of this last inequality are in fact equal. \[ \square \]

The standard binary splitting method yields a formula representing the polynomial mat of length \( (2m)^{\lfloor \lg p \rfloor} \), so that above bound is tight.

Although theorems 2.3 - 2.5 have been stated for the weight function and growth
function as defined in section 2 it should be obvious that these theorems hold true for any pair of functions fulfilling relations 2.3 - 2.5, 2.10 - 2.12, and lemma 2.2. (In fact in 2.3 equality can be replaced by inequality). We shall use this approach to analyze the complexity of the symmetric polynomials, and more particularly of the middle one,

\[ \text{sym} = \sum_{I} \prod_{i \in I} x_i, |I| = n \]

**Theorem 3.3:** If \( F \) is a formula representing the polynomial \( \text{sym} \) then

\[ s(F) \geq n^{0.25} \log n + o(1) \]

**Proof:** Let \( v(G) \) be the number of different variables occurring in the formula \( G \). We use the "weight" function defined by

\[ \tilde{w}(G) = w(G) \cdot 2^{d(G) - v(G)} \]

and the "growth" function defined by

\[ G(k) = \binom{2k}{k} 2^{-k}. \]

(The rationale lying behind these seemingly gratuitous definitions is the wish to "penalize" subformulas where high weight and low degree is achieved at the expense of many variables occurring in them; such subformulas are not subsequently useful.)

We have

\[ v(G_1 + G_2) \leq v(G_1) + v(G_2). \]

Moreover, since the polynomial \( \text{sym} \) is linear in each variable, if \( G_1 \ast G_2 \) is a subformula of \( F \) then

\[ v(G_1 \ast G_2) = v(G_2) + v(G_2). \]

Also, since this polynomial is homogeneous, if \( G_1 + G_2 \) is a subformula of \( F \), then \( d(G_1) = d(G_2) \). Using these facts it is easily verified that the function \( \tilde{w} \) fulfills relations 2.3 - 2.5. Also, the function \( G \) fulfills relations 2.10 - 2.12. It remains to be shown that the "growth" function indeed provides an upper bound on the "weight", i.e. that if \( d(G) = k \) then \( \tilde{w}(G) \leq G(k) \). But if \( G \) is a subformula of \( F \) of degree no greater than \( k \) then \( w(G) \leq \binom{v(G)}{k} 2^{-k} \) with \( v = v(G) \). We are left with the easily checked inequality.

\[ \binom{v}{k} 2^{-v} \leq \binom{2k}{k} 2^{-k} \quad \text{(for } v \geq k). \]

Since the function \( G \) is submultiplicative, we can apply theorem 2.5, and, as before the proof is reduced to the verification of the inequality

\[ n^{0.25} \log n + o(1) \leq \left[ i^{0.25} \log i + j^{0.25} \log j \right] \left( \binom{2n}{i} \cdot \binom{2n}{j} \right) \]

where \( i + j = n \). This inequality is verified using the same argument as in theorem 3.1.

\[ \square \]
Here also using the binary splitting method, a formula for $\text{sym}$ of length $n^{0.5\log n + O(1)}$ can be constructed. This result can be improved to $n^{0.25\log n + O(1)}$ when addition is idempotent (as in the case it is interpreted as union).

4. Conclusion

Tight lower bounds on the size of formulas representing certain functions have been proven, when only addition and multiplication are used. As hinted at in the introduction, these results are valuable for other settings as well. For example, the lower bound proved for the permanent applies as well to the determinant which formal definition is similar: any formula representing the determinant of a matrix of rank $n$, such that no terms cancel when parentheses are opened, has length at least $2^n - 0.25\log^2 n + O(1/n)$. On the other hand, we know that when this limitation is removed, a formula of length $2^{0(\log^2 n)}$ can be built. (Valiant, in [Val], has given an example of a similar gap for a polynomial with positive coefficients.)

From our bounds on formula length, bounds on formula height (i.e. the number of steps in a parallel algorithm) can be directly derived. Indeed, $h(F) \geq \log s(F)$, for any formula $F$. Thus, a parallel "monotone" algorithm for permanent computation takes at least $2^n - 0.25\log^2 n$ steps, an algorithm for the multiplication of $p \times m \times m$ matrices takes $\log p \cdot (\log m + 1)$ steps, an algorithm for the computation of the middle symmetric function on $n$ variables takes $0.25\log^2 n$ steps. (These results were directly proven in [ShS].)

As said before, our results apply to star free regular expressions. New results can be achieved by taking into account the non-commutativity of multiplication (concatenation). This is done by redefining in an obvious way the complement and growth functions. The remaining results of section 2 carry through, without modification. Using this approach we can prove that the minimal size of a regular expression representing the set of all permutations on $n$ symbols is $2^n - 0.25\log^2 n + O(1/n)$; the minimal size of a regular expression representing the set of all paths of length $p + 1$ in an arc-labelled complete graph on $m$ nodes is $(2m)^{1/p}$. These two results match, and can in fact be directly inferred from, the results on the permanent and matrix multiplication. Related results can be found in [EhZ].

Any extension of the results of this paper to less restricted systems (arithmetic with subtraction, monotone Boolean formulas) is likely to be an arduous task, and for the permanent, would be of major significance (see [Va2] and [Va3]).

Acknowledgements

I would like to thank Professor Eli Shamir for his important contribution in the shaping of this work.
References


