# THE PARALLEL EVALUATION OF ARITHMETIC EXPRESSIONS IN LOGARITHMIC TIME

Richard P. Brent,
Computer Centre,
Australian National University,
Canberra, A.C.T. 2600, Australia.

## 1. INTRODUCTION

This paper gives a survey of some results on the time required to evaluate arithmetic expressions with several processors computing simultaneously. Proofs are omitted if they have appeared elsewhere, but the proofs of several new results are included.

Our fundamental assumption is that several processors which can independently perform the basic arithmetic operations (addition, subtraction, multiplication, and sometimes division) in unit time are available. The time required for accessing data, storing results, and communicating between processors is ignored. (For many plausible machine organizations, considering this overhead would increase our upper bounds by a small constant factor.)

We consider well-formed arithmetic expressions with distinct atoms (or indeterminates), i.e. expressions $E$ such that one of the following holds:
1. $E = x_i$ for some atom $x_i$ ;
2. $E = (L)\theta(R)$ for $\theta =$ "+", "-", "*" or "/", and well-formed expressions $L$ and $R$ which depend on disjoint sets of atoms; or
3. $E = \theta(L)$ for $\theta =$ "+" or "-" and a well-formed expression $L$ .

Redundant pairs of parentheses may be inserted or removed at will. If values in a field (or ring if division is excluded) are assigned to the atoms, then the values of the expressions may be found in the usual way. The context should make clear whether $E(x_1, \ldots, x_n)$ denotes an expression with atoms $x_1, \ldots, x_n$ or the value of an expression

for certain values $x_1, \ldots, x_n$ of its atoms.

In Sections 2 to 5 we assume that all expressions have distinct atoms, so expressions like

$$a + x(b + x(c + x))$$

and

$$x^{100}$$

are excluded. This restriction is not serious, for our results give upper bounds on the time required to evaluate the more general expressions

$$a + x_1(b + x_2(c + x_3))$$

and

$$x_1 x_2 \ldots x_{100}$$

respectively.

Similarly, in Sections 2 to 5, expressions involving constants are not allowed. Thus, the results do not apply to expressions like

$$a + 1/(b + 1/(c + 1/d)) ,$$

though they do apply to

$$a + u_1/(b + u_2/(c + u_3/d)) .$$

These restrictions are relaxed in Section 6, where we give some results for common special classes of expressions, such as polynomials, continued fractions, etc.

Expressions containing the subtraction operation can easily be transformed into equivalent expressions with addition, multiplication, division and (at most) some unary subtractions acting on atoms, e.g.

$$a - (b + c/(d - e) - f) =$$

$$a + ((-b) + c/((-d) + e) + f) .$$

Since the unary subtractions may be performed (in parallel) at the start of the evaluation of an expression, there is no real loss of generality in restricting our attention to expressions containing the operations of addition ("+"), multiplication ("*"), and division ("/") only.

In Section 2 we consider arithmetic expressions with $n$ distinct atoms, any level of parenthesis nesting, and operations "+" and "*". Corollary 2.1 shows that such expressions may be evaluated in time $4\lceil \log_2 n \rceil$ using $n-1$ processors. In Section 3 this result is extended to expressions which contain the operation "/". Corollary 3.1 shows that such expressions may be evaluated in time $4\lceil \log_2 n \rceil$ with $3(n-1)$ processors.

In Section 4 we deduce some results on the time required to evaluate expressions with $n$ distinct atoms if a fixed number $p \geq 1$ of processors is available. From Theorems 4.1 and 4.2, expressions without division can be evaluated in time $4\lceil \log_2 n \rceil + 2(n-1)/p$, and expressions with division can be evaluated in time $4\lceil \log_2 n \rceil + 10(n-1)/p$. These results are within constant factors of the best possible.

The results of Sections 2 to 4 hold for exact arithmetic over any commutative field (or ring if division is excluded). In Section 5 we show that the algorithm of Section 2 for evaluating expressions without division is numerically stable (in a backward sense) if approximate arithmetic over the real field is used instead of exact arithmetic. It is an open question whether the same applies to the algorithm of Section 3 for evaluating expressions including the division operation.

## 2. EXPRESSIONS WITHOUT DIVISION

In this section we consider expressions with $n$ distinct atoms and operations of addition ("+") and multiplication ("*") over a commutative ring. First we need some definitions and a simple lemma.

Definitions

If $x$ is a real number then $\lceil x \rceil$ denotes the integer such that $x \leq \lceil x \rceil < x + 1$.

If $E$ is an arithmetic expression then $|E|$ denotes the number of atoms (relabelled if necessary to become distinct) in $E$. If $T$ is a parse tree for $E$, then $|T| = |E|$ is the number of terminal nodes of $T$. If $|T| > 1$ we write $T = L \hat{} R$, where $L$ and $R$ are the maximal proper subtrees of $T$. A subexpression of $E$ is the expression corresponding to a subtree (not necessarily proper) of a parse tree for $E$.

Lemma 2.1 (Brent [73])

If $1 < m \leq n$ and $T$ is a binary tree with $|T| = n$, then there is a subtree $X_1 = L_1 \hat{} R_1$ of $T$ such that $|X_1| \geq m$, $|L_1| < m$, and $|R_1| < m$. Also, if $x$ is one of the terminal nodes of $T$, there is a subtree $X_2 = L_2 \hat{} R_2$ of $T$ such that $|X_2| \geq m$ and either

1. $x$ is a terminal node of $L_2$ and $|L_2| < m$, or
2. $x$ is a terminal node of $R_2$ and $|R_2| < m$.

Theorem 2.1

Let $E$ be an arithmetic expression with $n$ distinct atoms and operations "+" and "*" over a commutative ring with identity. Suppose $P(n) = n-1$ processors capable of performing "+" and "*" in unit time are available. Let

$$k = \left\{ \begin{matrix} n+2 & \text{if} & n \leq 3, \\ \lceil 4 \log_2 (n-1) \rceil & \text{if} & n \geq 4, \end{matrix} \right\}$$

$$Q_1(n) = 2(n-1),$$

and

$$Q_2(n) = \max(0, 2(n-2)).$$

Then $E$ can be evaluated in time $k-3$ with at most

$Q_1(n)$ operations. Also, if $x$ is any atom of $E$, then $E = Ax + B$, where $A$ and $B$ are expressions which do not involve $x$ and which can be evaluated simultaneously in time $k$ with at most $Q_2(n)$ operations. (Here $A$ may be identically 1 and $B$ may be identically 0.)

Proof

The definition of $k$ gives $k \geq n+2$ for $n \leq 7$. We can evaluate $E$ with one processor in $n-1$ steps, and $A$ and $B$ with two processors in $n-2$ steps and $2(n-2)$ operations $(n \geq 2)$. (A "step" is one unit of time.) Hence, the result holds for $n \leq 7$, so we may assume that $n = N \geq 8$ (so $k \geq 12$). As inductive hypothesis we assume that the result holds for $n < N$.

Applying Lemma 2.1 with $m = \lceil (7n+5)/12 \rceil$ to a parse tree for $E$, we see that there is a subexpression $X_1$ of $E$ such that $X_1 = L_1 \theta_1 R_1$ (where $\theta_1$ is "+" or "*"), $|X_1| \geq (7n+5)/12$, $|L_1| < (7n+5)/12$, and $|R_1| < (7n+5)/12$. Let $E_1$ be the expression formed by replacing $X_1$ by an atom in $E$. From the definition of $k$,

$$n \leq 2^{k/4} + 1 ,$$

and it is easy to verify that $5/12 < 2^{-5/4}$, so

$$|E_1| = n + 1 - |X_1| \leq 5(n-1)/12 + 1 < 2^{(k-5)/4} + 1 .$$

Thus

$$\lceil 4 \log_2(|E_1| - 1) \rceil \leq k-5 ,$$

so the inductive hypothesis (applied to $E_1$) gives

$$E = A_1 X_1 + B_1 ,$$

where $A_1$ and $B_1$ can be evaluated in time $k-5$ with $P(|E_1|)$ processors and $Q_2(|E_1|)$ operations.

Since $7/12<2^{-3/4}$ , we have

$$|L_1| < (7n+5)/12 < 2^{(k-3)/4} + 1 ,$$

and similarly for $|R_1|$ . Thus, by the inductive hypothesis, $L_1$ and $R_1$ can be evaluated in $(k-3)-3 = k-6$ steps with $P(|L_1|) + P(|R_1|)$ processors and $Q_1(|L_1|) + Q_1(|R_1|)$ operations . Thus, $X_1$ can be evaluated in $k-5$ steps, and $E$ in $k-3$ steps. The number of processors required is at most

$$1 + P(|E_1|) + P(|L_1|) + P(|R_1|) = n-1 .$$

If $X_1 \neq E$ , the number of operations required is at most

$$3 + Q_2(|E_1|) + Q_1(|L_1|) + Q_1(|R_1|)$$
$$= 2n - 3 < 2(n-1) .$$

If $X_1 = E$ then $E = L_1\theta_1 R_1$ can be evaluated with at most

$$1 + Q_1(|L_1|) + Q_1(|R_1|) = 2n - 3 < 2(n-1)$$

operations. Thus, $E$ can be evaluated in $k-3$ steps with $P(n)$ processors and $Q_1(n)$ operations, so the first half of the proof is complete.

Let $x$ be an atom of $E$ . Applying the second half of Lemma 2.1 with $m = \lceil (n+1)/2 \rceil$ to a parse tree for $E$ , we see that there is a subexpression $X_2$ of $E$ such that $X_2 = L_2\theta_2 R_2$ , $|X_2| \geq (n+1)/2$ , and either $x$ is an atom of $L_2$ and $|L_2| \leq n/2$ , or $x$ is an atom of $R_2$ and $|R_2| \leq n/2$ . Without loss of generality we may assume that $x$ is an atom of $L_2$ .

Let $E_2$ be the expression formed by replacing $X_2$ by an atom in $E$ . Thus

$$|E_2| = n + 1 - |X_2| \leq (n+1)/2 \leq 2^{(k-4)/4} + 1 ,$$

so the inductive hypothesis (applied to $E_2$) gives

$$E = A_2X_2 + B_2 ,$$

where $A_2$ and $B_2$ can be evaluated in time $k-4$ with $P(|E_2|)$ processors and $Q_2(|E_2|)$ operations. Similarly,

$$L_2 = A_3x + B_3 ,$$

where $A_3$ and $B_3$ can be evaluated in time $k-4$ with $P(|L_2|)$ processors and $Q_2(|L_2|)$ operations. Since $|R_2| \leq n-1$, the inductive hypothesis also shows that $R_2$ can be evaluated in $k-3$ steps with $P(|R_2|)$ processors and $Q_1(|R_2|)$ operations.

From $X_2 = L_2\theta_2R_2$ and the above expressions for $E$ and $L_2$, we find that $E = Ax + B$, where the form of $A$ and $B$ depends on $\theta_2$. There are two possibilities:

Case 1: $\theta_2 =$ "+"

$$A = A_2A_3$$

and

$$B = A_2(B_3 + R_2) + B_2 .$$

From the above, $A$ and $B$ can be evaluated in $k$ steps with

$$1 + P(|E_2|) + P(|L_2|) + P(|R_2|) = n - 1$$

processors. If $|E_2| > 1$ and $|L_2| > 1$ then $A$ and $B$ can be evaluated with

$$4 + Q_2(|E_2|) + Q_2(|L_2|) + Q_1(|R_2|) = 2(n-2)$$

operations. If $|E_2| = 1$ (so $E = X_2$) or $|L_2| = 1$ (so $L_2 = x$) or both, the forms of $A$ and $B$ simplify, and it is easy to show that they can be evaluated with $2(n-2)$ operations.

Case 2: $\theta_2 = "*"$

$$A = (A_2 R_2) A_3$$

and

$$B = (A_2 R_2) B_3 + B_2 .$$

As in Case 1, the inductive hypothesis shows that $A$ and $B$ can be evaluated in $k$ steps with $n-1$ processors and $2(n-2)$ operations.

The theorem now follows by induction on $N$.

Since the statement of Theorem 2.1 is rather cumbersome (though necessary so that the result may be proved by induction), we give the immediate:

Corollary 2.1

Let $E$ be an arithmetic expression with operations "+" and "*" over a commutative ring. Then $E$ can be evaluated in time $4 \log_2 |E|$ with $|E| - 1$ processors.

The constant "4" of Corollary 2.1 can be reduced at the expense of increasing the number of processors:

Theorem 2.2 (Brent, Kuck and Maruyama [73])

Let $n \geq 2$ and $E$ be as in Theorem 2.1. Then $E$ can be evaluated in time $\log_\lambda(\alpha n + \beta)$ if sufficiently many processors are available, where

$$\lambda = 1.3247179572...$$

is the real positive root of $z^3 = 1 + z$,

$$\alpha < 0.596 ,$$

and

$$\beta < 0.167 .$$

Since $\log_\lambda 2 \approx 2.4649 < 4$, the time given by Theorem 2.2 is less than that given by Corollary 2.1, but the number of processors required is $O(n^{1.71...})$

instead of $0(n)$ . (Here $1.71... = \log_\lambda((1+\sqrt{5})/2)$.) Hence, Corollary 2.1 is more useful than Theorem 2.2 for deducing results which apply when a fixed number of processors is available.

For earlier (and weaker) results on expressions without division, see Baer and Bovet [68] and Miranker [71].

## 3. EXPRESSIONS WITH DIVISION

Theorem 2.1 can be generalized to cover expressions with division. The result is:

Theorem 3.1 (Brent [73])

Let $E$ be an arithmetic expression with $n$ distinct atoms and operations "+", "*" and "/" over a commutative field. Suppose that sufficiently many processors capable of performing "+" and "*" (but not necessarily "/") in unit time are available. Let $P_1(n) = 3(n-1)$ , $P_2(n) = \max(0, 3n-8)$ , $Q_1(n) = 10(n-1)$ , $Q_2(n) = \max(0, 10n-29)$ , and

$$k = \begin{cases} n+1 & \text{if } n \leq 2 , \\ \lceil 4\log_2(n-1) \rceil & \text{if } n \geq 3 . \end{cases}$$

Then $E = F/G$ , where $F$ and $G$ are expressions which can be evaluated simultaneously in time $k-2$ with $P_1(n)$ processors and $Q_1(n)$ operations. Also, if $x$ is any atom of $E$ , then

$$E = (Ax + B)/(Cx + D) ,$$

where $A$ , $B$ , $C$ and $D$ are expressions which do not contain $x$ and which can be evaluated simultaneously in time $k$ with $P_2(n)$ processors and $Q_2(n)$ operations. (Note that some of $A, \ldots , G$ may be identically $0$ or $1$ .)

The proof of Theorem 3.1 follows much the same lines as that of Theorem 2.1, and is given in Brent [73]. Corresponding to Corollary 2.1, we have:

Corollary 3.1

Let $E$ be an arithmetic expression with operations "+", "*" and "/" over a commutative field. Suppose that a sufficient number of processors is available and at least one processor can perform a division in unit time. Then $E$ can be evaluated in time $4\log_2|E|$ with $3(|E| - 1)$ processors.

The constants "3" and "4" of Corollary 3.1 can possibly be reduced by more refined arguments, but no generalization of Theorem 2.2 to cover expressions with division is known.

## 4. PARALLEL EVALUATION USING $p$ PROCESSORS

Theorems 2.1 and 3.1 apply if $O(|E|)$ processors are available, but in practice the number of processors is likely to be bounded. In this section we suppose that $p \geq 1$ processors are available, and use the following lemma to deduce some interesting results from Theorems 2.1 and 3.1.

Lemma 4.1 (Brent [73])

If expressions $E_1, \ldots, E_m$ can be evaluated simultaneously in time $t$ with $q$ operations and sufficiently many processors which perform arithmetic operations in unit time, then $E_1, \ldots, E_m$ can be evaluated in time $t + (q-t)/p$ with $p$ such processors.

Proof

Suppose that $s_i$ operations are performed at step $i$, for $i = 1, 2, \ldots, t$. Thus $\sum_{i=1}^{t} s_i = q$. Using $p$ processors, we can simulate step $i$ in $\lceil s_i/p \rceil$ steps. Hence, $E_1, \ldots, E_m$ can be evaluated with $p$ processors in time

$$\sum_{i=1}^{t} \lceil s_i/p \rceil \leq (1 - 1/p)t + \sum_{i=1}^{t} s_i/p$$

$$= t + (q-t)/p .$$

Theorem 4.1

Let $E$ be an arithmetic expression with operations "+" and "*" over a commutative ring. If $p$ processors which can perform "+" and "*" in unit time are available, then $E$ can be evaluated in time $4\log_2 n + 2(n-1)/p$, where $n = |E|$.

Proof

Suppose $n \geq 4$, for otherwise the result is trivial. From Theorem 2.1 and Lemma 4.1, $E$ can be evaluated in time

$$(1 - 1/p)(\lceil 4\log_2 (n-1) \rceil - 3) + 2(n-1)/p$$

$$< 4\log_2 n + 2(n-1)/p .$$

Similarly, from Theorem 3.1 and Lemma 4.1 we deduce:

Theorem 4.2 (Brent [73])

Let $E$ be an arithmetic expression with operations "+", "*" and "/" over a commutative field. If $p$ processors which can perform these operations in unit time are available, then $E$ can be evaluated in time $4\log_2 n + 10(n-1)/p$, where $n = |E|$.

Since only one division is performed, Theorem 4.2 is easily modified if a division takes longer than an addition or multiplication.

Theorems 4.1 and 4.2 are within a constant factor (6 and 14 respectively) of the best possible, for the evaluation of $x_1 + x_2 + \ldots + x_n$ requires time at least $\max(\log_2 n , \frac{n-1}{p})$.

## 5. NUMERICAL STABILITY

We now consider evaluating expressions over the real field using (approximate) floating-point arithmetic. Following Wilkinson [63], we assume that the approximate arithmetic operations satisfy

$$fl(a*b) = ab\delta_1$$

and

$$fl(a + b) = a\delta_2 + b\delta_3 ,$$

where

$$1 - \varepsilon \lesssim \delta_i \lesssim (1 - \varepsilon)^{-1} \quad \text{for} \quad i = 1, 2 \text{ and } 3 ,$$

and $\varepsilon<1$ is a positive constant (the "machine precision"). We also assume that $fl(a*1) = a$ exactly (or that such trivial multiplications are omitted).

For our purposes it is sufficient to say that an algorithm for evaluating the expression $E(x_1, \ldots , x_n)$ is numerically stable if the computed value $\tilde{E}$ satisfies

$$\tilde{E} = E(\alpha_1 x_1, \ldots , \alpha_n x_n)$$

for some numbers $\alpha_i$ satisfying

$$(1 - \varepsilon)^{\phi(n)} \lesssim \alpha_i \lesssim (1 - \varepsilon)^{-\phi(n)}$$

where $\phi(n)$ is a function independent of $x_1, \ldots , x_n$. In other words, the computed result $\tilde{E}$ would be obtained if exact arithmetic were used after applying "small" relative perturbations (of order $\varepsilon\phi(n)$) to the arguments $x_1, \ldots , x_n$.

The main result of this section is that the algorithm implied by the proof of Theorem 2.1 is numerically stable. In fact, from Theorem 5.1, we can take

$$\phi(n) = 4\log_2 n$$

in the above. (A similar result holds for approximate arithmetic over the complex field.) First we need two lemmas.

Lemma 5.1

Let $E(x_1, \ldots , x_n)$ be an expression with $n$ distinct atoms $x_1, \ldots , x_n$ and operations "+" and "*" over the real field. If $0<r\lesssim 1$ and $r\lesssim\alpha\lesssim r^{-1}$ then, for given values of $x_1, \ldots , x_n$, there are

numbers $\alpha_i$ , satisfying $r \leq \alpha_i \leq r^{-1}$ , such that

$$\alpha E(x_1, \ldots, x_n) = E(\alpha_1 x_1, \ldots, \alpha_n x_n) .$$

Proof

If $n=1$ then $E = x_1$ , so $\alpha_1 = \alpha$ satisfies the lemma. Hence, suppose that $n = N \geq 2$ . As inductive hypothesis, suppose that the result holds for $n<N$ .

Since $n \geq 2$ , we have $E = L\theta R$ , where $\theta$ = "+" or "*" and the expressions $L$ and $R$ depend on disjoint subsets of $\{x_1, \ldots, x_n\}$ . Since

$$\alpha E(x_1, \ldots, x_n) = \left\{ \begin{array}{ll} \alpha L + \alpha R & \text{if } \theta = \text{"+"} , \\ \alpha L * R & \text{if } \theta = \text{"*"} , \end{array} \right\}$$

the result follows by induction.

Lemma 5.2

Let $E = E(x_1, \ldots, x_n) = A(x_2, \ldots, x_n) * x_1 + B(x_2, \ldots, x_n)$ be an expression with $n$ distinct atoms and operations "+" and "*" over the real field. (The assumption of distinct atoms is essential here.) If $0<r \leq 1$ , $0<s \leq 1$ , $r \leq \alpha \leq r^{-1}$ , and $s \leq \beta \leq s^{-1}$ , then for given values of $x_1, \ldots, x_n$ there are numbers $\gamma_i$ , satisfying $rs \leq \gamma_i \leq (rs)^{-1}$ for $i = 2, \ldots, n$ , such that

$$\beta B(x_2, \ldots, x_n) = B(\gamma_2 x_2, \ldots, \gamma_n x_n)$$

and either

1. $A(x_2, \ldots, x_n)$ is identically 1 (which we write as $A \equiv 1$), or
2. $\alpha A(x_2, \ldots, x_n) = A(\gamma_2 x_2, \ldots, \gamma_n x_n)$ .

Proof

If $A \equiv 1$ the result follows from Lemma 5.1, so assume that $A \not\equiv 1$ . If $n=1$ then $E = x_1$ and $A \equiv 1$

contrary to our assumption, so $n \geq 2$. Hence, suppose $n = N \geq 2$, and as inductive hypothesis suppose that the result holds for $n < N$.

Since $n \geq 2$, we have $E = L \theta R$, where $\theta$ = "+" or "*", and without loss of generality $L$ depends on $x_1$. Thus (by Lemma 1 of Brent, Kuck and Maruyama [73]),

$$L = A_1 x_1 + B_1$$

for some expressions $A_1$ and $B_1$ which are independent of $x_1$. There are two cases to consider.

Case 1: $\theta$ = "+"

$$E = A_1 x_1 + B_1 + R,$$

so $A = A_1$ and $B = B_1 + R$. Thus $\alpha A = \alpha A_1$ and $\beta B = \beta B_1 + \beta R$. Now $A_1 \not\equiv 1$ as $A \not\equiv 1$, and the atoms of $L$ and $R$ are disjoint, so the result follows from the inductive hypothesis (applied to $A_1$ and $B_1$) and Lemma 5.1 (applied to $R$).

Case 2: $\theta$ = "*"

$$E = A_1 R x_1 + B_1 R,$$

so $A = A_1 R$ and $B = B_1 R$. Thus $\alpha A = A_1 * (\alpha R)$ and $\beta B = (\beta/\alpha) B_1 * (\alpha R)$. The atoms of $L$ and $R$ are disjoint, so the result follows from the inductive hypothesis (applied to $A_1$ and $B_1$ with $r$, $s$, $\alpha$ and $\beta$ replaced by 1, $rs$, 1 and $\beta/\alpha$ respectively) and Lemma 5.1 (applied to $R$). Thus, the lemma follows by induction on $N$.

Theorem 5.1

Let $n$, $k$, $E(x_1, \ldots, x_n)$, $A$, $B$, and $x = x_1$ be as in Theorem 2.1. If approximate floating-point arithmetic with machine precision $\varepsilon$ over the real field is used, then $E$, $A$ and $B$ can be evaluated in the time (and with the number of processors and operations) given by Theorem 2.1 so that the computed values $\tilde{E}$, $\tilde{A}$ and $\tilde{B}$ satisfy

$$\tilde{E} = E(\alpha_1 x_1, \ldots, \alpha_n x_n) ,$$

$$\tilde{A} = A(\beta_2 x_2, \ldots, \beta_n x_n) ,$$

and

$$\tilde{B} = B(\beta_2 x_2, \ldots, \beta_n x_n) ,$$

where

$$(1 - \varepsilon)^{k-3} \leq \alpha_i \leq (1 - \varepsilon)^{3-k} \quad \text{for} \quad i = 1, \ldots, n$$

and

$$(1 - \varepsilon)^{k} \leq \beta_j \leq (1 - \varepsilon)^{-k} \quad \text{for} \quad j = 2, \ldots, n .$$

Proof

The result is easily deduced from Lemmas 5.1 and 5.2 if $n \leq 7$, so suppose that $n = N \geq 8$. As inductive hypothesis, suppose that the result holds for $n < N$.

It is convenient to make the convention that all $\delta_i$ below satisfy

$$1 - \varepsilon \leq \delta_i \leq (1 - \varepsilon)^{-1} .$$

From the proof of Theorem 2.1,

$$\tilde{E} = fl(\tilde{A}_1 * (\tilde{L}_1 \theta_1 \tilde{R}_1) + \tilde{B}_1) ,$$

where $A_1$ is as in the proof of Theorem 2.1, and $\tilde{A}_1$ is the computed value of $A_1$, etc. Thus

$$\tilde{E} = \tilde{A}_1((\delta_1^3 \tilde{L}_1) \theta_1 (\delta_2^3 \tilde{R}_1)) + \delta_3 \tilde{B}_1 .$$

Since the atoms on which $L_1$, $R_1$ and $A_1$ (or $B_1$) depend are disjoint, and

$$|L_1| \leq 2^{(k-3)/4} + 1$$

etc. (see the proof of Theorem 2.1), the result for $\tilde{E}$ follows from the inductive hypothesis and Lemmas 5.1 and 5.2.

Similarly, from the proof of Theorem 2.1,

$$\tilde{A} = \begin{cases} fl(\tilde{A}_2\tilde{A}_3) & \text{if } \theta_2 = "+" , \\ fl((\tilde{A}_2\tilde{R}_2)\tilde{A}_3) & \text{if } \theta_2 = "*" , \end{cases}$$

and

$$\tilde{B} = \begin{cases} fl(\tilde{A}_2(\tilde{B}_3 + \tilde{R}_2) + \tilde{B}_2) & \text{if } \theta_2 = "+" , \\ fl((\tilde{A}_2\tilde{R}_2)\tilde{B}_3 + \tilde{B}_2) & \text{if } \theta_2 = "*" . \end{cases}$$

As usual, there are two cases to consider.

Case 1: $\theta_2 = "+"$

$$\tilde{A} = \begin{cases} \tilde{A}_2(\delta_4\tilde{A}_3) & \text{if } A_3 \not\equiv 1 , \\ \tilde{A}_2 & \text{if } A_3 \equiv 1 , \end{cases}$$

and

$$\tilde{B} = \tilde{A}_2(\delta_5^3\tilde{B}_3 + \delta_6^3\tilde{R}_2) + \delta_7\tilde{B}_2 .$$

From the proof of Theorem 2.1,

$$|A_3x_1 + B_3| \leq 2^{(k-4)/4} + 1 ,$$

so the result for $\tilde{A}$ and $\tilde{B}$ follows from the inductive hypothesis and Lemmas 5.1 and 5.2.

Case 2: $\theta_2 = "*"$

$$\tilde{A} = \begin{cases} \tilde{A}_2(\delta_8\tilde{R}_2)(\delta_9\tilde{A}_3) & \text{if } A_3 \not\equiv 1 , \\ \tilde{A}_2(\delta_8\tilde{R}_2) & \text{if } A_3 \equiv 1 , \end{cases}$$

and

$$\tilde{B} = \tilde{A}_2(\delta_8\tilde{R}_2)(\delta_{10}^2\tilde{B}_3) + \delta_{11}\tilde{B}_2 .$$

Again, the result for $\tilde{A}$ and $\tilde{B}$ follows from the inductive hypothesis and Lemmas 5.1 and 5.2. Hence, the theorem follows by induction on $N$.

## 6. SPECIAL CLASSES OF EXPRESSIONS

Although the results of Sections 2 to 4 are within constant factors of the best possible, it is

worthwhile to try to reduce the constants as much as possible in special cases of practical importance. In this section we briefly summarize some results for various special classes of arithmetic expressions.

Dorn [62], Estrin [60], Muraoka [71] and others have considered parallel polynomial evaluation. The sharpest results are those of Maruyama [73a] and (independently) Munro and Paterson [71], who have shown that polynomials of degree $n$ can be evaluated in time $\log_2 n + 0((\log_2 n)^{\frac{1}{2}})$ if sufficiently many processors are available. Munro and Paterson [71] have also shown that time $2n/p + \log_2 p + 0(1)$ is sufficient if $p \leq n/\log_2 n$ processors are available. By a result of Winograd [70], this is very close to the best possible. If preconditioning is allowed some improvement is possible (see Rabin and Winograd [71], Belaga [61], and Motzkin [55]).

A polynomial may be written in the form $a_0 + x(a_1 + x(a_2 + \ldots (a_{n-1} + a_n x)\ldots))$. Brent [70] has shown that the slightly more general expression $a_0 + x_1(a_1 + x_2(a_2 + \ldots (a_{n-1} + a_n x_n)\ldots))$ over any commutative ring (e.g. the ring of real numbers or the Boolean ring) may be evaluated in time $\log_2 n + 0((\log_2 n)^{\frac{1}{2}})$ if sufficiently many processors are available.

Kuck and Maruyama [73] have shown that continued fractions $b_0 + a_1/(b_1 + a_2/(\ldots(b_{n-1} + a_n/b_n)\ldots))$ can be evaluated in time $2\log_2 n + 0(1)$ if sufficiently many (in fact $0(n)$) processors are available. (As in the algorithm of Section 3, the evaluation requires only one division.) The result also applies to "continued parenthesis" expressions of the form $a_0\theta_0(a_1\theta_1(a_2\theta_2(\ldots(a_{n-1}\theta_{n-1}a_n)\ldots)))$, where each $\theta_i$ = "+", "-", "*", or "/". Results for a limited number of processors may be deduced.

Various other classes of expressions, such as expressions with a limited depth of parenthesis

nesting or a limited number of division operations, have been considered by Kuck [73], Kuck and Maruyama [73], Kuck and Muraoka [73], Maruyama [73b], Muraoka [71], and others. For additional references, see the bibliography in Miranker [71]. It would be worthwhile to combine the techniques used by various authors to produce a good algorithm for compiling arithmetic expressions for subsequent execution on a machine with $p$ processors.

Finally we note that little attention has been paid to the numerical stability (or instability) of parallel numerical algorithms. Sometimes the requirements of parallelism and stability seem to conflict, but this is not always so, as the results of Section 5 show. The question of stability remains open for many of the algorithms mentioned above. For those algorithms which are fast but unstable, the existence of stable algorithms which are as fast (or nearly as fast) is also unsettled.

## POSTSCRIPT

The constant "4" in Theorem 2.1 and Corollary 2.1 can be reduced to "3" at the expense of increasing the number of operations from $2(n-1)$ to $5(n-1)/2$. The proof is similar to that of Theorem 2.1: it may be shown by induction on $n > 1$ that $E$, $A$ and $B$ can be evaluated in times $3\lfloor \log_2 n \rfloor - 1$, $3\lfloor \log_2 (n-1) \rfloor$ and $3\lfloor \log_2 (n-1) \rfloor + 1$ respectively.

Our results may be extended to expressions over noncommutative rings and fields. This extension is described in the paper "The parallel evaluation of matrix expressions" (to appear) by K. Maruyama.

## ACKNOWLEDGEMENT

The suggestions of David Kuck and Kiyoshi Maruyama were very useful.

## REFERENCES

Baer and Bovet [68]: Baer, J.L. and Bovet, D.P., "Compilation of arithmetic expressions for parallel computations", Proc. IFIP Congress (1968), 340-346.

Belaga [61]: Belaga, E.G., "On computing polynomials in one variable with initial conditioning

of the coefficients", Problemi Kibernetiki 5 (1961), 7-15.

Brent [70]: Brent, R.P., "On the addition of binary numbers", IEEE Trans. Comp. C-19 (1970), 758-759.

Brent [73]: Brent, R.P., "The parallel evaluation of general arithmetic expressions", submitted to J. ACM.

Brent, Kuck and Maruyama [73]: Brent, R.P., Kuck, D.J. and Maruyama, K.M., "The parallel evaluation of arithmetic expressions without division", to appear in IEEE Trans. Comp. C-22 (May 1973).

Dorn [62]: Dorn, W.S., "Generalizations of Horner's rule for polynomial evaluation", IBM Jour. Res. & Dev. (1962), 239-245.

Estrin [60]: Estrin, G., "Organization of computer systems - The fixed plus variable structure computer", Proc. Western Joint Computer Conference (1960), 33-40.

Kuck [73]: Kuck, D.J., "Evaluating arithmetic expressions of n atoms and k divisions in $\alpha(\log_2 n + 2\log_2 k) + c$ steps", to appear.

Kuck and Maruyama [73]: Kuck, D.J. and Maruyama, K., "The parallel evaluation of arithmetic expressions of special forms", submitted for publication.

Kuck and Muraoka [73]: Kuck, D.J. and Muraoka, Y., "Bounds on the parallel evaluation of arithmetic expressions using associativity and commutativity", to appear.

Maruyama [73a]: Maruyama, K.M., "On the parallel evaluation of polynomials", IEEE Trans. Comp. C-22 (1973), 2-5.

Maruyama [73b]: Maruyama, K.M., "Upper bounds on the time required to evaluate arithmetic expressions", submitted for publication.

Miranker [71]: Miranker, W.L., "A survey of parallelism in numerical analysis", SIAM Review 13 (1971), 524-547.

Motzkin [55]: Motzkin, T.S., "Evaluation of polynomials and evaluation of rational functions", Bull. Amer. Math. Soc. 61 (1955), 163.

Munro and Paterson [71]: Munro, I. and Paterson, M., "Optimal algorithms for parallel polynomial evaluation", Report RC 3497, IBM Research Center, Yorktown Heights (1971).

Muraoka [71]: Muraoka, Y., "Parallelism exposure and exploitation in programs", Report 424, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign (1971).

Rabin and Winograd [71]: Rabin, M.O. and Winograd, S., "Fast evaluation of polynomials by rational preparation", Report RC 3645, IBM Research Center, Yorktown Heights (1971).

Wilkinson [63]: Wilkinson, J.H., "Rounding errors in algebraic processes", HMSO, London (1963).

Winograd [70]: Winograd, S., "On the number of multiplications necessary to compute certain functions", Comm. Pure and Appl. Math. 23 (1970), 165-179.