The Parallel Evaluation of General Arithmetic Expressions

RICHARD P. BRENT

Australian National University, Canberra, Australia

ABSTRACT. It is shown that arithmetic expressions with $n \geq 1$ variables and constants; operations of addition, multiplication, and division; and any depth of parenthesis nesting can be evaluated in time $4 \log_2 n + 10(n - 1)/p$ using $p \geq 1$ processors which can independently perform arithmetic operations in unit time. This bound is within a constant factor of the best possible. A sharper result is given for expressions without the division operation, and the question of numerical stability is discussed.

KEY WORDS AND PHRASES: arithmetic expressions, compilation of arithmetic expressions, computational complexity, general arithmetic expressions, numerical stability, parallel computation, code optimization

CR CATEGORIES. 4.12, 5.11, 5.25

1. *Introduction*

The question of how quickly arithmetic expressions can be evaluated on a computer with several independent arithmetic processors is of theoretical and practical interest. In this paper we determine the answer to within a constant multiplicative factor (see Corollary 2 in Section 4). All our proofs are constructive, and reasonably efficient algorithms for compiling expressions for subsequent execution on a parallel computer may be derived from our proofs. These algorithms compare favorably with those given in [1, 2].

We assume that a number of processors are available and that each can perform an arithmetic operation (addition, multiplication, and sometimes division) in unit time. The time required for accessing data, storing results, communicating between processors, etc., is ignored. Also, the effect of rounding errors is neglected, except in Section 5. The results hold for exact arithmetic with expressions over any commutative field.

Several special cases have been considered previously. For example, Maruyama [14] and Munro and Paterson [19] have shown that polynomials of degree n can be evaluated in time $\log_2 n + O((\log_2 n)^{\frac{1}{2}})$ if sufficiently many processors are available, and Brent [3] has shown that this is true for expressions of the form $a_0 + x_1(a_1 + x_2(a_2 + \cdots (a_{n-1} + a_n x_n) \cdots))$. Baer and Bovet [1] and Muraoka [20] considered expressions with n distinct variables and operations of addition and multiplication over a commutative ring. It has recently been shown in [5] that such expressions can be evaluated in time $2.465 \log_2 n$ if sufficiently many processors are available. (For results that apply if a fixed number of processors is available, see Section 5.) Kuck and Maruyama [12] have shown that continued fractions of the form $b_0 + a_1/(b_1 + a_2/(\cdots (b_{n-1} + a_n/b_n) \cdots))$ can be evaluated in time $2 \log_2 n + O(1)$. Kuck [10], Maruyama [15], and Muraoka [20] have considered expressions with a limited depth of parenthesis nesting and/or a limited number of divisions. See also [6, 8, 9, 13, 18] and the references given there.

Author's address: Computer Centre, Australian National University, P.O. Box 4, Canberra, A.C.T. 2600, Australia.

Our results (Corollary 1 and Theorem 2) show that parallelism may be used to speed up the evaluation of large arithmetic expressions. Knuth [7] has shown that most expressions which occur in real FORTRAN programs have only a small number of operands. Nevertheless, our results (or the method used to obtain them) may ultimately be of practical value, for Kuck [11] has shown that an optimizing compiler for a parallel machine might generate large expressions when compiling programs like those studied by Knuth [7].

In this paper we assume commutativity, but Maruyama [16] has recently extended some of our results to expressions over noncommutative rings (e.g. rings of matrices).

2. *Notation and Assumptions*

We consider well-formed arithmetic expressions with the operations addition ("+"), multiplication ("*"), and division ("/"); any level of parenthesis nesting; and distinct indeterminates (or "atoms") $x_1, x_2, \cdots$ over a commutative field. We neglect the subtraction operation because expressions containing it can easily be transformed into equivalent expressions with "+", "*", "/" and (at most) some unary subtractions acting on atoms, e.g. $a - (b + c/(d - e) - f) = a + ((-b) + c/((-d) + e) + f)$.

The restriction to expressions with distinct atoms means that we do not consider expressions such as $a + x(b + x(c + x))$, $a + 1/(b + 1/(c + 1/d))$, and x^{100}. However, our results give upper bounds on the time required to evaluate such expressions, because they apply to the more general expressions $a + x_1(b + x_2(c + x_3))$, $a + u_1/(b + u_2/(c + u_3/d))$, and $x_1x_2 \cdots x_{100}$ respectively. For further discussion and examples, see [5].

If E is an arithmetic expression then $|E|$ denotes the number of atoms (relabeled 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 $T = \hat{L}R$, 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.

If r is a real number then $\lceil r \rceil$ denotes the integer satisfying $r \leq \lceil r \rceil < r + 1$.

3. *Main Theorem*

Theorem 1 states slightly more than we use subsequently, but the statement is necessary so that the result may be proved by induction. The most interesting consequences of the theorem are stated in Corollaries 1 and 2 (Section 4).

We first state, without proof, a trivial but useful lemma.

LEMMA 1. *If $1 < m \leq n$ and T is a binary tree with $|T| = n$, then there is a subtree $X_1 = \hat{L_1}R_1$ of T such that $|X| \geq m$, $|L_1| < m$, and $|R_1| < m$. Also, if x is one of the terminal nodes of T, there is a subtree $X_2 = \hat{L_2}R_2$ 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 1. *Let E be any 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 $P_1(n) = 3(n - 1)$, $P_2(n) = max(0, 3n - 4)$, $Q_1(n) = max(0, 10n - 19)$, $Q_2(n) = max(0, 10n - 29)$, 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 (1) and (2) below hold:

(1) $E = F/G$, 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.

(2) *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 $A, \cdots, G$ may be identically 0 or 1.)*

PROOF. By inspection, the result holds for $n \leq 4$, so we assume that $n = N \geq 5$ (so $k \geq 8$). The proof is by induction on N. As inductive hypothesis we assume that parts (1) and (2) of the theorem hold for $n < N$.

We shall show that part (1) holds with $n = N$. Applying Lemma 1 with $m = \lceil (n + 1)/2 \rceil$ to a parse tree for E, we see that there is a subexpression $X_1 = L_1\theta_1R_1$ of E such that $| X_1 | \geq (n + 1)/2$, $| L_1 | \leq n/2$, $| R_1 | \leq n/2$, and $\theta_1 =$ "+", "*", or "/".

From the definition of k, $n \leq 2^{k/4} + 1$; so $| L_1 | \leq n/2 < 2^{(k-4)/4} + 1$, and similarly for R_1. Thus, by part (1) of the inductive hypothesis, $L_1 = F_1/G_1$ and $R_1 = F_2/G_2$, where F_1, G_1, F_2, and G_2 can be evaluated simultaneously in time $(k - 4) - 2 = k - 6$ with $P_1(| L_1 |) + P_1(| R_1 |)$ processors and $Q_1(| L_1 |) + Q_1(| R_1 |)$ operations.

Now $X_1 = L_1\theta_1R_1 = (F_1/G_1)\theta_1(F_2/G_2) = F_3/G_3$, where

$$F_3 = \begin{cases} F_1G_2 + F_2G_1 & \text{if } \theta_1 = \text{"+"}, \\ F_1F_2 & \text{if } \theta_1 = \text{"*"}, \\ F_1G_2 & \text{if } \theta_1 = \text{"/"}, \end{cases} \quad \text{and} \quad G_3 = \begin{cases} G_1G_2 & \text{if } \theta_1 = \text{"+" or "*"}, \\ G_1F_2 & \text{if } \theta_1 = \text{"/"}. \end{cases}$$

Hence F_3 and G_3 can be evaluated in time $k - 4$.

Let E_1 be the expression formed by replacing X_1 by an atom in E. Since $| E_1 | = n + 1 - | X_1 | \leq (n + 1)/2 \leq 2^{(k-4)/4} + 1$, part (2) of the inductive hypothesis (applied to E_1) gives $E = (A_1X_1 + B_1)/(C_1X_1 + D_1)$, where A_1, B_1, C_1, and D_1 can be evaluated simultaneously in time $k - 4$ with $P_2(| E_1 |)$ processors and $Q_2(| E_1 |)$ operations. Since $X_1 = F_3/G_3$, it follows that $E = F/G$, where $F = A_1F_3 + B_1G_3$ and $G = C_1F_3 + D_1G_3$ can be evaluated in time $k - 2$.

Consider the number of processors required to compute F and G as above. In the first $k - 6$ steps we compute F_1, G_1, F_2, G_2 and start computing A_1, B_1, C_1, and D_1, using $P_1(| L_1 |) + P_1(| R_1 |) + P_2(| E_1 |)$ processors. From time $k - 6$ to $k - 4$ we compute F_3 and G_3 and finish computing A_1, B_1, C_1, and D_1, using $2 + P_2(| E_1 |)$ processors. Finally, from time $k - 4$ to $k - 2$ we compute F and G, using four processors. Thus, the number of processors required is

$$\begin{aligned} &\max [P_1(| L_1 |) + P_1(| R_1 |) + P_2(| E_1 |), 2 + P_2(| E_1 |), 4] \\ &\quad = \max [3(| L_1 | + | R_1 | + | E_1 |) - 10, 3(| L_1 | + | R_1 |) - 6, 3 | E_1 | - 2, 4] \\ &\quad < 3(n - 1) = P_1(n), \end{aligned}$$

as $| L_1 | + | R_1 | + | E_1 | = n + 1$, $| L_1 | + | R_1 | \leq n$, $| E_1 | \leq (n + 1)/2$, and $n > 2$.

Now consider the number of operations required to compute F and G as above. Since $3 \leq (n + 1)/2 \leq | X_1 | = | L_1 | + | R_1 |$, the definition of Q_1 gives $Q_1(| L_1 |) + Q_1(| R_1 | \leq 10(| L_1 | + | R_1 |) - 29$. Thus, the number of operations is at most

$$\begin{aligned} &10 + Q_1(| L_1 |) + Q_1(| R_1 |) + Q_2(| E_1 |) \\ &\qquad \leq \max [10(| L_1 | + | R_1 | + | E_1 |) - 48, 10(| L_1 | + | R_1 |) - 19] \\ &\qquad \leq 10n - 19 = Q_1(n), \end{aligned}$$

so part (1) holds with $n = N$.

To complete the proof, we must show that part (2) holds with $n = N$. Let x be an atom of E. Applying the second half of Lemma 1 with $m = \lceil (n + 1)/2 \rceil$ to a parse tree for E, we see that there is a subexpression $X_2 = L_2\theta_2R_2$ of E such that $| X_2 | \geq (n + 1)/2$, $\theta_2 =$ "+", "*", or "/", and either x is an atom of L_2 and $| L_2 | \leq n/2$, or x is an atom of R_2 and $| R_2 | \leq n/2$. We shall suppose that x is an atom of L_2. (The proof is similar if x is an atom of R_2.)

Let E_2 be the expression formed by replacing X_2 by an atom in E. Thus $|E_2| = n + 1 - |X_2| \leq (n+1)/2 \leq 2^{(k-4)/4} + 1$, and part (2) of the inductive hypothesis (applied to E_2) gives $E = (A_2X_2 + B_2)/(C_2X_2 + D_2)$, where A_2, B_2, C_2, and D_2 can be evaluated simultaneously in time $k - 4$ with $P_2(|E_2|)$ processors and $Q_2(|E_2|)$ operations.

Similarly, $L_2 = (A_3x + B_3)/(C_3x + D_3)$, where A_3, B_3, C_3, and D_3 can be evaluated in time $k - 4$ with $P_2(|L_2|)$ processors and $Q_2(|L_2|)$ operations. Also, since $|R_2| \leq n - 1$, part (1) of the inductive hypothesis shows that $R_2 = F_4/G_4$, where F_4 and G_4 can be evaluated in time $k - 2$ with $P_1(|R_2|)$ processors and $Q_1(|R_2|)$ operations.

From $X_2 = L_2\theta_2R_2$ and the above expressions for E, L_2, and R_2, we find that $E = (Ax + B)/(Cx + D)$, where

$$A = \begin{cases} (A_2C_3)F_4 + (A_2A_3 + B_2C_3)G_4 & \text{if } \theta_2 = \text{"+"}, \\ (A_2A_3)F_4 + (B_2C_3)G_4 & \text{if } \theta_2 = \text{"*"}, \\ (A_2A_3)G_4 + (B_2C_3)F_4 & \text{if } \theta_2 = \text{"/"}, \end{cases}$$

and B, C, and D are given by similar expressions. Thus A, B, C, and D can be evaluated in time k.

The number of processors required to compute $A, \cdots, D$ simultaneously in time k is at most

$$\begin{aligned} &\max [P_2(|E_2|) + P_2(|L_2|) + P_1(|R_2|), 8 + P_1(|R_2|)] \\ &\quad = \max [3(|E_2| + |L_2| + |R_2|) - 11, 3(|L_2| + |R_2|) - 7, \\ &\qquad\qquad 3(|E_2| + |R_2|) - 7, 3|R_2| + 5]. \end{aligned}$$

Since $|E_2| + |L_2| + |R_2| = n + 1$, $|L_2| + |R_2| \leq n$, $|E_2| + |R_2| \leq n$, and $n > 1$, the number of processors required is at most $3n - 4 = P_2(n)$ provided $3|R_2| + 5 \leq 3n - 4$, i.e. provided $|R_2| \leq n - 3$. If $|R_2| = n - 2$ or $n - 1$, the expressions for A, B, C, and D simplify, and a straightforward examination of cases shows that $P_2(n)$ processors suffice.

Similarly, if $|E_2| > 2$ and $|L_2| > 2$, the number of operations required is at most $28 + Q_2(|E_2|) + Q_2(|L_2|) + Q_1(|R_2|) \leq 10n - 30 < Q_2(n)$. If $|E_2| \leq 2$ or $|L_2| \leq 2$ or both, the expressions for A, B, C, and D simplify, and $Q_2(n)$ operations suffice. This completes the proof of part (2), so the theorem follows by induction on N.

4. *Consequences of Theorem 1*

We need the following lemma, which is of some independent interest.

LEMMA 2. *If a computation C can be performed in time t with q operations and sufficiently many processors which perform arithmetic operations in unit time, then C can be performed in time $t + (q - t)/p$ with p such processors.*

PROOF. Suppose that s_i operations are performed at step i, for $i = 1, 2, \cdots, t$. Thus $\sum_{i=1}^{t} s_i = q$. Using p processors, we can simulate step i in time $\lceil s_i/p \rceil$. Hence, the computation C can be performed with p processors in time

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

COROLLARY 1. *Let E be as in Theorem 1 and suppose that p processors which can perform addition, multiplication, and division in unit time are available. Then E can be evaluated in time $4 \log_2 n + 10(n - 1)/p$.*

PROOF. Suppose that $n \geq 3$, for otherwise the result is trivial. By Theorem 1, $E = F/G$, where F and G can be evaluated in time $\lceil 4 \log_2(n-1) \rceil - 2 < 4 \log_2 n - 1$ with less than $10(n - 1)$ operations. Applying Lemma 2 with $t = \lceil 4 \log_2(n-1) \rceil - 2$ and $q = 10(n - 1)$, we see that F and G can be evaluated in time $4 \log_2 n - 1 + 10(n - 1)/p$ with p processors. Finally, $E = F/G$ can be evaluated in one more unit of time. (Note that only one division is performed, so the result is easily modified if a division takes longer than an addition or multiplication.)

COROLLARY 2. *Let $\tau(n, p)$ be the maximum time required to evaluate arithmetic expressions with n atoms, using p processors which can perform arithmetic operations in unit time. Let $\phi(n, p) = max(log_2 n, (n-1)/p)$. Then, for all $n \geq 1$ and $p \geq 1$, $\phi(n, p) \leq \tau(n, p) \leq 14\phi(n, p)$.*

PROOF. Consider the expression $x_1 + x_2 + \cdots + x_n$. By a fan-in argument, its evaluation requires time at least $\log_2 n$. Also, at least $n - 1$ operations must be performed, so p processors require time at least $(n-1)/p$. Hence, the lower bound on $\tau(n, p)$ is established. The upper bound follows from Corollary 1.

5. *Concluding Remarks*

Corollary 2 establishes the complexity of parallel evaluation of general arithmetic expressions to within a constant factor. The constant 14 can doubtless be reduced by more refined arguments, and the lower bound for $\tau(n, p)$ can be improved slightly (see [5]).

The proof of Theorem 1 simplifies, and the constants can be reduced, if division is excluded. Corresponding to Corollary 1 we have the following, which is slightly weaker than Theorems 1 and 2 of [5] if $p \gg n$, but much stronger if p is of order n or less.

THEOREM 2. *Let E be any arithmetic expression with n (distinct) atoms and 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 $4\log_2 n + 2(n-1)/p$.*

A proof of Theorem 2 is given in [4], where we also show that, for real expressions and approximate arithmetic, the evaluation of E in the time given by Theorem 2 is numerically stable (in the sense that the computed result can be obtained by making small relative changes in the values assigned to the atoms and then performing exact arithmetic). Unfortunately, this result does not extend to expressions with division, and examples found by a program of Miller [17] show that the algorithm implied by the proof of Theorem 1 is not always numerically stable. Hence, it is an open question whether general arithmetic expressions can be evaluated stably in the time given by Corollary 1.

Acknowledgments. David Kuck and Kiyoshi Maruyama made several stimulating suggestions, without which this paper might not have been written. Webb Miller kindly verified the numerical instability mentioned above, and a referee's comments were useful in clarifying the proof of Theorem 1.

REFERENCES

1. BAER, J. L., AND BOVET, D. P. Compilation of arithmetic expressions for parallel computations. Proc. IFIP Congr. 1968, North-Holland Pub. Co., Amsterdam, pp. 340–346.
2. BEATTY, J. C. An axiomatic approach to code optimization for expressions. *J. ACM 19*, 4 (Oct. 1972), 613–640.
3. BRENT, R. P. On the addition of binary numbers. *IEEE Trans. Comp. C-19* (Aug. 1970), 758–759.
4. BRENT, R. P. The parallel evaluation of arithmetic expressions in logarithmic time. Proc. Symposium on Complexity of Sequential and Parallel Numerical Algorithms (Carnegie-Mellon U., Pittsburgh, Pa., May 1973), Academic Press, New York, 1973, pp. 83–102.
5. BRENT, R. P., KUCK, D. J., AND MARUYAMA, K. M. The parallel evaluation of arithmetic expressions without division. *IEEE Trans. Comput. C-22* (May 1973), 532–534.
6. HOBBS, L. C. (Ed.) Parallel processor systems, technologies and applications. Spartan Books, New York, 1970.
7. KNUTH, D. E. An empirical study of FORTRAN programs. *Software 1* (April 1971), 105–133.
8. KOGGE, P. M. Parallel algorithms for the efficient solution of recurrence problems; the numerical stability of parallel algorithms for solving recurrence problems; and minimal parallelism in the solution of recurrence problems. Stanford Electronics Lab. Reps. 43–45, Sept. 1972.
9. KOGGE, P. M., AND STONE, H. S. A parallel algorithm for the efficient solution of a general class of recurrence equations. Rep. CS-72-298, Comput. Sci. Dep., Stanford U., Stanford, Calif., March 1972.
10. KUCK, D. J. Evaluating arithmetic expressions of n atoms and k divisions in $\alpha(\log_2 n + 2\log_2 k) + c$ steps. Manuscript, March 1973.

11. KUCK, D. J. Parallelism in ordinary programs. Proc. Symposium on Complexity of Sequential and Parallel Numerical Algorithms (Carnegie-Mellon U., Pittsburgh, Pa., May 1973), Academic Press, New York, 1973, pp. 17–47.
12. KUCK, D. J., AND MARUYAMA, K. M. The parallel evaluation of arithmetic expressions of special forms. Rep. RC 4276, IBM Res. Center, Yorktown Heights, N.Y., March 1973.
13. KUCK, D. J., MURAOKA, Y., AND CHEN, S. On the number of operations simultaneously executable in FORTRAN-like programs and their resulting speedup. *IEEE Trans. Comput. C-21* (Dec. 1972), 1293–1310.
14. MARUYAMA, K. M. On the parallel evaluation of polynomials. *IEEE Trans. Comput. C-22* (Jan. 1973), 2–5.
15. MARUYAMA, K. M. Upper bounds on the time required to evaluate arithmetic expressions. Rep. RC 4260, IBM Res. Center, Yorktown Heights, N. Y., March 1973.
16. MARUYAMA, K. M. The parallel evaluation of matrix expressions. Rep. RC 4380, IBM Res. Center, Yorktown Heights, N. Y., June 1973.
17. MILLER, W. C. Numerical heuristics in computer-aided roundoff analysis. Rep. RC 4332, IBM Res. Center, Yorktown Heights, N. Y., May 1973.
18. MIRANKER, W. L. A survey of parallelism in numerical analysis. *SIAM Rev. 13* (Oct. 1971), 524–547.
19. MUNRO, I., AND PATERSON, M. Optimal algorithms for parallel polynomial evaluation. Proc. IEEE 12th Annual Symposium on Switching and Automata Theory (Oct. 1971), pp. 132–139.
20. MURAOKA, Y. Parallelism exposure and exploitation in programs. Rep. 424, Dep. of Comput. Sci., U. of Illinois at Urbana-Champaign, Feb. 1971.

RECEIVED MARCH 1973; REVISED JUNE 1973