

# Parallel Generation of Postfix and Tree Forms

ELIEZER DEKEL and SARTAJ SAHNI

University of Minnesota Twin Cities

---

Efficient parallel algorithms to obtain the postfix and tree forms of an infix arithmetic expression are developed. The shared memory model of parallel computing is used.

Categories and Subject Descriptors: D.3.4 [Programming Languages]: Processors—*parsing*

General Terms: Algorithms

Additional Key Words and Phrases: Arithmetic expressions, postfix, infix, tree form, parallel computing, complexity

---

## 1. INTRODUCTION

The parallel parsing and evaluation of arithmetic expressions has been the focus of research for many years. References [1, 2, 10, 11, 13] are some of the important papers written on the parallel evaluation of arithmetic expressions. The most significant result here is due to Brent [1]. He has shown [1] that arithmetic expressions containing  $n$  operands,  $n \geq 1$ ; operators (+, \*, and /); and parentheses can be evaluated in  $4 \log_2 n + 10(n - 1)/p$  time when  $p$  processors are available. Parallel parsing of arithmetic expressions has been considered by Fischer [5], Krohn [8], Lipkie [12], and Schell [16] (among others). Fischer's work is restricted to vector (or pipelined) computers. While Krohn's work was intended primarily for pipelined computers (specifically for the CDC STAR-100), the ideas contained in [8] can be extended to parallel multiprocessor computers. Krohn, however, does not consider the asymptotic performance that could be obtained from his parallel parsing algorithm. Lipkie [12] and Schell [16] explicitly consider parsing on parallel multiprocessor computers. Lipkie [12] provides some grammar rules for parallel parsing but does not develop a formal algorithm. Schell [16] is a thorough study of parallel techniques for several of the phases normally encountered in compiling (scanning, syntax analysis, parsing, error recovery, etc.). Schell develops a parallel LR parser. The complexity of this parser is, however, quadratic

---

This research was supported in part by the Office of Naval Research under contract N00014-80-C-0650.

Authors' present addresses: E. Dekel, Mathematical Sciences Program, University of Texas at Dallas, Richardson, TX 75080; S. Sahni, Department of Computer Science, University of Minnesota Twin Cities, 136 Lind Hall, 207 Church Street S.E., Minneapolis, MN 55455.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1983 ACM 0164-0925/83/0700-0300 \$00.75

ACM Transactions on Programming Languages and Systems, Vol. 5, No. 3, July 1983, Pages 300-317.

in the input size (under some constraints, he shows that its complexity becomes linear). Schell also discusses the applicability of his techniques to precedence grammars.

In this paper, we study the following problems:

- (1) parallel generation of the postfix form;
- (2) parallel generation of the binary tree form.

In both cases, we start with the infix form of the expression. Further, we assume that the input infix expression is syntactically correct. The reader unfamiliar with the postfix and tree forms of an expression is referred to [6].

The study of the two problems cited above is motivated by the following considerations:

(1) We could conceivably build a special-purpose infix-to-postfix chip that could be used like a peripheral on a very high-speed number cruncher. The use of this parallel translator chip would speed compilation of programs.

(2) Most code optimizers for single-processor machines start with the tree form of an expression. Hence, a high-speed special-purpose chip that performs the translation from infix to tree form could be used in the context of (1).

(3) If the program is to be executed on a parallel machine, it can also be compiled on that machine using a parallel compiler. Such a compiler will need to be able to translate in parallel, from the infix form to a more usable form. The postfix and tree forms are two such forms. In fact, the parallel evaluation methods suggested in [1, 2, 10, 11, 13] all begin with the tree form of the arithmetic expression.

(4) While the length of individual arithmetic expressions in typical programs is small [7], Kuck [9] has shown that optimizing compilers for parallel machines can generate very long expressions even when the input program contains only short expressions. Furthermore, it is possible to view the entire program as a single expression and obtain its postfix form.

The model of parallel computation that we use here is commonly referred to as the shared memory model (SMM). This has the following characteristics:

(1) There are  $p$  processing elements (PEs) or processors. These are indexed  $0, 1, \dots, p - 1$ , and an individual PE may be referenced as  $PE(i)$ . Each PE is capable of performing the standard arithmetic and logical operations. In addition, each PE knows its index.

(2) There is a common memory that is shared by all the PEs. All  $p$  PEs can read and write into this memory in the same time instance. If two PEs attempt to read the same word of memory simultaneously, a *read conflict* occurs. Similarly, if two PEs attempt to write simultaneously into the same word of memory, a *write conflict* occurs. In this paper, we assume that read and write conflicts are prohibited.

(3) The PEs are synchronized and operate under the control of a single instruction stream.

(4) An enable/disable mask can be used to select a subset of the PEs that are to perform an instruction. Only the enabled PEs will perform the instruction.

