Several algorithms that attempt to reduce the tree height of a parse tree of an arithmetic expression by using associativity and commutivity have been proposed [1] - [3]. Baer and Bovey [2] conjectured that their algorithm obtained a minimal-height tree. Later Beatty [2] proved this conjecture. Without distributivity, the upper bound on the tree height is: 1 + 2d + \log_2 n \text{ where } n \text{ is the number of operands and } d \text{ is the depth of parenthesis nesting [6].}

The selective use of distribution reduces the upper bound on the tree height to \( 4 \log (n-1) + \log_2 n \) [7]. Several algorithms which use distribution have been proposed [7] - [9]. Using the algorithms of Beatty [2] and Brent [7] we can show that the use of associativity and commutivity adds \( O(n) \) steps to the parsing process, while the use of associativity, commutivity, and distributivity adds \( O(n \log_2 n) \) steps. Both algorithms works on parse trees produced by ordinary compilers and the times quoted are in addition to the ordinary parsing time.

We shall assume that an arithmetic expression contains \( N \) tokens (identifiers, constants, and operators). In calculating the number of operations required to parse an expression we shall count each instance of an arithmetic operation, logical operation, push on stack, pop stack, and store as one operation.

The results are summarized in the following theorem:

**Theorem [10]** Let \( E \) be any arithmetic expression with \( N \) tokens. The additional time to parse \( E \) is at most:

1. \( 13N + 2 \) by Beatty's algorithm
2. \( 3N \log_2 N \) by Brent's algorithm

**References**

10. R.A. Towle, Control and Data Dependence for Program Transformations, Dept. of Computer Science, University of Illinois at Urbana-Champaign, UIUCDCS-R-76-788, (March, 1976), 113 pp.

This work was sponsored by the National Science Foundation through grant # NSR-73-0780 A02 while the author was at the University of Illinois.