A New Look at Reverse Polish Notation

Predrag V. Krtolica

Reverse Polish notation was introduced by Polish mathematician Lukasievicz in 1958. By this notation, the arithmetic expression was noted in postfix manner, instead of using the usual infix notation. For example, the infix expression

$$a + b$$

in reverse Polish was noted like

$$ab + .$$

The algorithms for transforming postfix expression to the infix one, and vice versa are well known. However, these algorithms concern only the arithmetic expression with four fundamental binary arithmetic operators. The reason for this lies in the fact that the reverse Polish notation was aimed for computer architectures which had used stack for computing values of the expression. At the machine level all operations are fundamental. Therefore, reverse Polish notation could be useful for many other problems. If one wants to find derivative of the expression symbolically, he should make the tree of the expression. Then, he could make the derivative tree of the original expression tree. Making the expression tree begins with the transformation of the input expression in postfix notation. If we are limited to fundamental arithmetic operations, we are actually limited to polynomial or rational functions. In qualitative analysis of nonlinear differential equations (i.e. dynamic systems governed by these equations) there is a need for processing expression which include some forcing law (usually sine or cosine law). Software which accepts arbitrary differential equation should provide facilities for symbolic derivations of expression which contains binary, unary and function operators.

In this paper, the extension of the reverse Polish notation, as well as the extension of the corresponding algorithms for transforming infix expression to the postfix one and vice versa, are suggested. This extension allows processing the mathematical expressions containing not only fundamental arithmetic operators but also the unary operators and functions. It is shown that this could be extended on n-ary functions.

These improvements could be very useful in symbolic manipulations with expressions which contain not only the four fundamental arithmetic operators. A program using these improved algorithms is also developed.