A Non-Recursive Linear O(n) Approach to Symbolic Derivative Calculation

Ryan A. Toner

***Abstract*—**Symbolic differentiation can be seen as a function that maps human legible mathematical functions to their derivative and returning the result in human readable form. The underlying impetus for symbolic differentiation is typically for implementations in Computer Algebra Systems or other pedagogical applications. Moreover, an advantage of taking a symbolic derivative is that the resulting function can be easily evaluated by humans at a point by substituting various values of input. However, symbolic differentiation can be inefficient compared to automatic differentiation, and complicated because of the nuances of human expression input. This approach focuses on solving the ambiguities of symbol parsing and ameliorating the algorithmic time efficiency by eliminating the need for a recursive solution. Instead, the algorithm described by this paper will utilize a Transform-and-Conquer approach by constructing a Postfix (Reverse Polish notation) representation of user-input functions and composing symbolic derivatives as the algorithm converts back to infix notation. This algorithm uses a modified version of Edsger Dijkstra's Shunting-Yard algorithm that is compatible with composite functions. All operator precedence is handled by Shunting-Yard, which was originally described as a method for arithmetic-based parsing.

The Shunting-Yard approach was designed explicitly for operator-precedence parsing to a postfix expression for arithmetic computation; therefore, there were inherent limitations to using the approach for symbolic differentiation. A tokenization algorithm was created to identify monomial expression characteristics (including the negative unary operator) and operator tokens with O(n) (where n is character size) time complexity for string decomposition. Next, these fed a postfix generation algorithm (Shunting-Yard approach) with extended applications to composite functions and removing ambiguities in human input. Afterwards, a postfix-to-infix algorithm procedurally parsed through the order of operations while relying on Stack priority, and forward-accumulating derivatives at each step of the algorithm. The derivatives were computed based upon the specific token types received – identified as three cases: operator, composite, or monomial tokens. Each token type followed the basic rules of differentiation, including sum, difference, product, quotient, chain, and power rules. A separate solution was devised for chain-rule functions that efficiently stored the original input and retained a localized derivative value, meaning that exponent composite functions were solved with a bottom-up approach.

The conclusions of this paper are twofold. First, this approach expanded the notion that the Shunting-Yard algorithm can be used beyond a purely computational approach. Second, it demonstrated that expression parsing and symbolic derivative calculation is solvable in linear time non-recursively. Because the primary operation of the algorithm iterated through each character in the input for tokenization, the general asymptotic worst-case complexity resolves to O(n) time. Furthermore, because the theoretical, but practically impossible, number of tokens is size n, it demonstrates each subsequent algorithm has general-case complexity of O(n).

A significant aspect of investigation is applications into multivariable calculus, whereby partial derivatives are taken with respect to specific variables. The approach described here can be demonstrated to extend to these scenarios with gradient vector generation by adding an intermediate component to handle individual derivatives with respect to each variable.

***Keywords*—**computer algebra, multivariable calculus, shunting-yard algorithm, symbolic differentiation.

Ryan Toner, Computer Science & Mathematics Undergraduate at Fairfield University, Fairfield, CT 06824 USA (phone: 203-609-1777; ryan.toner@student.fairfield.edu).