Putting "Symbolic" into Symbolic Computation

Stephen M. Watt
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada N2L 3G1

Automated computation using symbols, as opposed to numbers alone, has been contemplated since the early 1800s at the time of Babbage's Analytical Engine. Much later, in the late 1950s, symbols were a fundamental data type in the Lisp programming language. Early software systems for algebraic computation and for machine proof used symbolic expressions in a rather informal way, but were nevertheless able to achieve remarkable results for the time. Since then, increased formalization both in algebra systems and proof systems has supported algorithms of focused scope that are orders of magnitude faster than their predecessors. We are now to some degree victims of our own success – these improved algorithms have channeled research in symbolic computation in the direction of increased rigour at the cost of limited scope.

We examine two approaches to expanding again the scope of symbolic computation. First, we show how certain problem elements, such as polynomial degrees, matrix structure parameters or domain shapes, may be treated soundly as symbolic quantities rather than requiring they be specified by numerical values. Secondly, we explore how different levels of formalization and abstraction may be used together, as would be required in software systems making use of a wide range of captured mathematical knowledge.