Taxonomies and Toolkits: Uses for the Mathematics of Program Construction Tutorial Abstract

BRUCE W. WATSON
Faculty of Mathematics and Computing Science
Eindhoven University of Technology
5600 MB, Eindhoven, The Netherlands
E-Mail: `watson@win.tue.nl`
Facsimile: +31 40 436685

Abstract: In this extended abstract, we present a brief description of the tutorial. The primary reference material, a book and some software [1], is available for ftp from the Eindhoven University of Technology.

Keywords: Taxonomy, algorithm derivation, generic programming

In this tutorial, we will consider a mathematical approach to the derivation of families of algorithms solving a particular problem. These families of algorithms are easily translated into practical toolkits of algorithms (C++ class libraries in our case). In the following paragraphs, we briefly describe why we need such toolkits, algorithm derivations, and a slightly non-standard algorithm derivation method.

Generic programming (or template programming in C++) forms one of the cornerstones (along with object-oriented programming) of component programming. Generic algorithm libraries fill the need for standard algorithms such as sorting and string searching algorithms, freeing the programmer from re-implementing such error prone components. The algorithms in a generic library do not make unnecessary assumptions about the nature of the data being operated upon. For example, a string searching library would work well with strings of characters or strings of floating point numbers. The very nature of generic programs make them extremely difficult to debug and test. Exhaustive testing is generally not possible since the number of types (for the template parameter) is usually infinite. These practical difficulties can be alleviated by using the mathematics of program construction to design the program correctly in the first place. In the next paragraph, we will consider the mathematical appeal of taxonomies of algorithms.

There are a number of mature areas of computer science (such as finite automata construction and string pattern matching) for which there are literally dozens of known algorithms. Many of these algorithms remain without rigorous proof, despite the fact that they were developed over twenty years ago. The differing presentation styles, and half-derivations of some of the algorithms makes them particularly difficult to compare to one another, and even more difficult to implement correctly.

Dijkstra's approach to correct program construction is rarely, if ever, used by software engineers working on "real-life" projects. A great deal of time, patience, and practice are required to learn and apply the method. Unfortunately, the strict use of Dijkstra's calculus is still best suited to programming-in-the-small (the development of elegant algorithms solving small problems, such as *greatest-common-divisor*).

We will present a method of constructing taxonomies of algorithms. The method starts with a naïve first algorithm, which is easy to prove correct but is impractical to implement. The taxonomy will be grown as a "family tree", with the naïve algorithm at the root. At each step, a variant of Dijkstra's discipline of programming is used to add either an algorithm detail (some algorithm transformation) or a problem detail (a problem restriction which allows an algorithmic improvement). The details are always added in a correctness-preserving manner, meaning that the correctness proof of any derived algorithm is in the composition of the details. The aim is to arrive at all of the known algorithms and hopefully derive some new ones.

Once such a taxonomy has been developed, it is finally possible to provide implementations (in the form of class libraries) of all of the known algorithms solving a particular problem. The presence of the correctness proofs in the taxonomy has yielded impressive code quality in the class libraries (fewer than five bugs per 10,000 of code in the first release). In the tutorial, we will also discuss the implementation of one such class library.

Additional material: The primary reference for this materal is [1]. That document can be obtained by ftp from `ftp.win.tue.nl` (the Faculty of Mathematics and Computing Science at the Eindhoven University of Technology) as file `/pub/techreports/pi/taxtk.ps.gz`. Attendees of the tutorial are encouraged to obtain the materials after the symposium. Any comments on the taxonomies or the toolkits are welcome; they can be sent to me.

References

[1] Watson, B.W. 1995. *Taxonomies and Toolkits of Regular Language Algorithms.* Faculty of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands.