Automatic Programming Using Axiomatic Language

Walter W. Wilson & Yu Lei, The Univ. of Texas at Arlington

This grand challenge seeks to improve programmer productivity and software reliability.

We assert that the ideal programming language should be a specification language – you tell the computer what to do without telling it how to do it. Specification definitions should be smaller, more readable, and more reusable than procedural code constrained by efficiency. Thus a specification language should give the programmer greater productivity. Specifications should also be easier to validate as correct. The challenge is to automatically generate efficient programs from specifications by a provably-correct process – the "automatic programming" problem.

We assert that the ideal programming language should be a metalanguage – able to define within itself other language features and paradigms which can then be used for specification without one leaving the original language. We want a "universal" language that can incorporate the specification advantages of any other language.

But an ideal language should also be minimal – as small and simple as possible without sacrificing expressiveness. We see elegance in reducing language to its most fundamental core. Nothing would be built-in that can be defined. Such a minimal language must also be highly extensible so that those features that are not built-in can be easily defined and used as if they had been built-in.

This grand challenge proposes use of a logic programming language called "axiomatic language" (http://www.axiomaticlanguage.org/) for this automatic programming effort. Axiomatic language may be the perfect language for this problem. Being a formal language it avoids the issue of natural language understanding that was a difficulty of earlier automatic programming efforts. Axiomatic language is a pure specification language that defines the external behavior of a program without having anything to say about its internal processing. Thus this automatic programming problem *must* be solved for this language to be considered implemented. The metalanguage capability of axiomatic language would give programmers flexibility and expressiveness in specification. Thus axiomatic language may stand a greater chance of mainstream use than other formal languages. But this metalanguage capability adds further to the automatic programming challenge.

Axiomatic language is also extremely minimal. Its simplicity and purity would make it a good candidate for formal methods and proof. Thus a guarantee of correctness of a generated program with respect to its specification may be achievable. It may also be possible to prove assertions about specifications to validate their correctness.

One can argue from computability that no finite amount of knowledge can automatically transform all possible specifications to efficient algorithms. Instead, the best we can hope for would be a system that can transform most "typical" specifications and then rely on an expert to add new knowledge whenever novel specifications are encountered. This is the goal of this grand challenge.