The Ideal of Verified Software

Tony Hoare

This talk is intended as an introduction to this afternoon's discussion on Grand Challenge problems for the ACL2 community. I want to invite you to address the possibility of an even more broadly based challenge, directed towards the scientific ideal of verified software. This is a challenge that will bring together the talents of theorists, tool-builders, and experimental scientists from around the world. It will provide opportunity for collaboration and scientific competition between many research communities, over a period of fifteen years or more. And it has a testable goal—the production and automatic verification of (say) a million of lines of useful code, drawn from many areas of computer application.

In the current state of the art, I regard our challenge as a scientific one, like the challenge of the human genome. Engineering challenges are also very important, but they differ from scientific challenges in their goals, their scope, their timescales and other constraints. A typical engineering challenge is Please find the bugs in this particular program before its delivery date (say next Tuesday). Scientists often engage themselves in such short-term challenges, and find it rewarding to earn the gratitude of their clients and customers. But Science itself is something different. It is motivated primarily by curiosity: it seeks the answers to basic questions, and aims at ideals that no engineer can afford to pursue. A typical scientific challenge seeks answers to questions How and why does software work. And how can we exploit this knowledge to make programs that are correct? And it is this kind of challenge that I would like to focus on.

The ACL2 project and its fore-runners have made an enormous contribution to meeting challenges of both science and engineering. As a scientific project, it is securely based on the mathematically formalised theories of functions, arithmetic, algebra and logic. It has embodied these theories in a comprehensive toolset to make them accessible and exploitable by the writer of application programs. And its tools have proved their maturity by convincing application to the challenge of engineering, recently in particular by routine use in the detection of bugs in programs that simulate computer hardware designs.

Can we now broaden and prolong this success story by repeating it on the challenge of correctness of general software as well? Can we supply a single coherent verification toolset that will be applicable to a wide range of typical components and programs in use at the present day? Embedded systems, distributed services, desktop applications, collection libraries, program generators, compilers, operating system kernels, ... Do we have the sound and general theories underlying the current technology of concurrent programming, pointer manipulation, inheritance, concurrency,... when used separately or in combination? Who will take these theories as the basis of toolsets which make their benefits available to the engineer? And how can we assemble convincing evidence of the solution of the problems listed above?

We can give positive answers to many of these questions, for example, recalling the old CLInc STACK project in the eighties. There are good ideas emerging from other research communities committed to formalisation of concepts from other programming languages. If as scientists we really seek the ideal of program correctness, I suggest that our progress might be accelerated by an appropriate degree of collaboration between many research communities who contribute their own theories, tools, and experiments. Tool-builders are already making progress on inter-working of tools which have complementary merits. Theorists are already working on unification of theories that could underlie the design of a next generation of more tightly integrated tools. Experimentalists are beginning to discuss the assembly of a corpus of experimental material—specifications, programs, documentation, design histories, test cases, etc. A publicly accessible repository of this material can be used to stimulate and judge scientific competition between rival toolsets. It will be the responsibility of the experimentalists to chart the yearly progress of the state of the art, and in the end to demonstrate beyond scientific doubt the applicability of the results of the project to the programs of the real world. As in other branches of science, experiment is paramount.

Here is an example of a start that has already been made on the experimental side. Jim Woodcock has suggested a challenge problem—an electronic purse implemented on a smart card—for which an abstract specification is already available. The challenge is to refine the specification to a design, with a mechanical check of all the theorems needed to assure design correctness. Three or four teams from round the world have taken up the challenge with their own tools, some of them dating back more than ten years. Although

ACL2 '06 Seattle, Washington USA

there is a long way to go, these teams have provided evidence that their tools are more cost-effective than previous manual proofs. All the material generated is being preserved, so that it can be exploited in future researches. The challenge still stands for other users and builders of verification tools to join the project and to demonstrate the current state of their art.

I hope that my remarks will stimulate and inspire your discussions later this afternoon on Grand Challenge problems for ACL2. And not just for ACL2, but for the whole computing research community. I would like all of you to give your support for the project I have described, and sincerely wish it well. But some of you, I hope, will make a stronger and more personal commitment to devote your own research to the progress of this broader project, maybe even participating in some of the early pilot studies like the electronic purse that I described above. I am sure that a sample of programs already proved in ACL2 will be a very welcome addition to the repository of challenge problems for experiment using other proof tools. The history and achievements of the participants in the ACL2 project undoubtedly qualify your community to serve as fore-runners and leaders of the even more ambitious project which I have dreamed about.