The Objective of the Institute e-Austria and the Role of the Scientific Project Practical and Theoretical Aspects of Program Verification

Bruno Buchberger

Research Institute for Symbolic Computation
Johannes Kepler University, Linz, Austria
buchberger@risc.uni-linz.ac.at

The Institute e-Austria Timisoara is a joint initiative of

- the Research Institute for Symbolic Computation (RISC) and the Softwarepark Hagenberg, Austria, and
- the Department of Computer Science, University of the West, and the Computer Science Department of the Technical University of Timisoara, Romania.

The pilot phase of the initiative (February 2002 - September 2003) is sponsored by the Austrian Ministry of Education, Research, and Culture, the Austrian Ministry of Economy and Labor, and the Romanian Ministry of Research.

The goal of the pilot phase is a feasibility study for creating

- a technology transfer institute (Institute e-Austria Timisoara, IeAT) and
- an IT-Park in Timisoara centered around IeAT.

The goal of the Institute e-Austria Timisoara is to create an environment for IT foundational and applied research and precompetitive product development in Timisoara. The intellectual power of this institute should basically build upon the joint competence of the faculty and graduates of the Department of Computer Science, University of the West and the Computer Science Department of the Technical University of Timisoara. The institute should mainly engage in joint R&D IT projects between the computer science departments of the two universities in Timisoara, IeAT, and the IT industry in the Timisoara region, in Romania, and abroad.

The goal of the IT-Park in Timisoara is to create an environment in Timisoara in which IT research institutions, IT educational institutions, and IT companies can cooperate and develop intensive synergies and long-lasting effects for the regional economy in Timisoara and also long-lasting technologic and economic benefits for all partners. In a later stage, the IT-Park may evolve into a technology park that encompasses also other technologies. However, at the beginning, we recommend to focus on only one subject, namely IT.

IeAT should be initiated and built up by investing significant seed money from the side of the Austrian Government, which should trigger off further investments from other public and private sides in Austria, Romania, and other countries and, in particular, the EU. The success of the feasibility study will be the major criterion for the Austrian Government to invest seed money into the IeAT.

The overall political, scientific, technologic, and economic goals of the project are:

- create opportunities for the IT experts (professors and graduates) of the Timisoara region to stay in Timisoara and to contribute to the future technologic, economic and social welfare of their home country,

- create an international environment for high-quality research and development for the benefit of all partners,
- establish a firm and dynamic link between academia and economy in the Timisoara region,
- create, directly and indirectly, a strong economic impulse for the city of Timisoara and the Timisoara region,
- create a frame for scientific and economic interaction and cooperation between the Timisoara region and the region of Upper Austria, Austria, and other EU countries,
- and, thereby, establish a model for long-lasting and sustaining measures for a harmonic and beneficial integration of Europe.

The project is based on an intensive cooperation between the Department of Computer Science, University of the West in Timisoara (Professor Stefan Maruster) and the Research Institute for Symbolic Computation, Johannes Kepler University in Linz (Professor Bruno Buchberger) over the past 15 years.

One particular activity during the pilot phase of the project is a model joint research project between the computer science departments in Timisoara, the RISC institute in Austria, and a couple of selected companies in Timisoara and Austria. In this research project, we want to build up and try out the organizational structures that will enable to organize and carry out joint research projects within IeAT on a large scale. As a subject for the model project we chose "Practical and theoretical aspects of program verification". This choice is motivated by joint research expertise of the partner institutions and also by the practical interest of the partner companies.

Program (algorithm) verification has a long research tradition but, so far, had relatively little impact on practical software development in the industry. A notable exception is the use of verification techniques in the test of computer processors. However, the design and implementation of reliable software still is an important issue and any progress in this area will be of utmost importance for the future development of IT.

In this talk we will give a personal assessment and overview on current verification techniques and their practical impact on industrial software and hardware development. We will put particular focus on the analysis of the reasons why program verification does not yet have the industrial impact it deserves and on measures that must be taken in order to make program verification fruitful for the practical purposes of the industry.

For this

- we analyze the verification problem for large-scale industrial software as opposed to small-scale algorithms and
- we distinguish program verification for procedural languages as opposed to functional and rewrite language.

Our main observation w.r.t. to program verification for large-scale industrial software is the following:

- Exact and formal program verification needs exact and formal program specification. Now, in large-scale industrial software, most of the code is relatively "flat" and only relatively small parts of the code are logically "deep". Flat code often does not contain loops or recursion but consists of long sequences of assignments and case distinction. For the verification of flat code, formal program verification is *not* the appropriate means because the formal specification of such a code is nothing else than a reformulation of the code itself (or vice versa) and, thus, is equally error-prone as the formulation of the actual program code. In fact, testing is the appropriate means for generating high-quality flat code. In contrast, the logically deep parts of the code are characterized by (nested) loops or recursions. These parts constitute the "tricky" or interesting part of the code. For these parts, formal program verification *is* an appropriate tool. We definitely advocate the use of formal tools for these parts.

- The use of techniques of formal exact verification, however, needs specific training of programmers in the formal logic aspects of computer science. This is a major challenge for the future of computer science.
- Formal program verification will only be attractive for industrial applications if a decent percentage of the actual verification task (the mathematical proof of the verification conditions) can be automated, i.e. the necessary proofs can be generated by computer software. This is another major challenge for the future of computer science.

The distinction between procedural and functional / rewrite languages is important because formal verification tools for procedural languages need a translation of the actual verification problem into a proof problem in a logical language like predicate logic. Thus, there is a natural gap between the language in which programs are written and executed and the language in which the verification is carried out. In contrast, for functional languages and, in particular, rewrite languages it is conceivable to program and to verify in the same language frame. Personally, I advocate the development o rewrite languages in the frame of predicate logic in order to eliminate the gap between the verification language and the programming language. In the talk we will report on one such language frame, "Theorema", which in essence is a version of predicate logic a part of which is an elegant programming language. Theorema is being developed in the Theorema Working Group at RISC. In addition to being a uniform frame for both programming and verification, Theorema also provides various provers that generate the proofs of mathematical propositions (in particular the verification conditions for programs) completely automatically. Theorema is currently one of the very few languages in which all the above design objectives are realized.

Finally, in the talk, we will also compare and assess various approaches to design program specifications, programs, verification conditions and proofs for the verification conditions in an integral process.