# Institute e-Austria Timisoara

## ○ Objectives

## ○ Model Project: Verification

Bruno Buchberger
Research Institute for Symbolic Computation
Johannes Kepler University, Linz, Austria

## ■ Thanks and Congratulation

Thanks and congratulation to the excellent work the Timisoara Working Group of the Institute e-Austria in Timisoara did for setting up the scientific and administrative environment for the institute.

## ■ Institute e-Austria Timisoara: Objectives

### ■ Institute e-Austria Timisoara: A Joint Initiative

Joint initiative of:

- the Research Institute for Symbolic Computation (RISC) at the Johannes Kepler University, and
  the Softwarepark Hagenberg, Austria

- the Department of Computer Science, University of the West and
  the Computer Science Department of the Technical University of Timisoara, Romania.

The project is based on an intensive cooperation, over the past 15 years, between

- the Department of CS, U of the West in Timisoara (Professor Stefan Maruster) and

- the RISC institute, JKU in Linz (Professor Bruno Buchberger)

with Professor Tudor Jebelean, former student both at U of the West in Timisoara and the RISC Institute, a "ambassador" between the two worlds and a permanent source of motivation.

### ■ Pilot Phase: February 2002 - September 2003

Sponsored by

Copyright 2003  B. Buchberger

- the Austrian Ministery of Education, Research, and Culture,
- the Austrian Ministery of Economy and Labor,
- the Romanian Ministery of Research.

## ■ Goal of the Pilot Phase

A feasibility study for creating

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

## ■ Goal of IeAT

Create an environment for IT foundational and applied research and precompetitive product development in Timisoara.

The intellectual power of IeAT: Bases on the joint competence of the faculty and graduates of the Department of Computer Science of the Universities in Timisoara.

The institute should mainly engage in joint R&D IT projects between

- the computer science departments of the two universities in Timisoara,
- the staff of the IeAT,
- and the IT industry in the Timisoara region in Romania and abroad.

## ■ Goal of the IT-Park in Timisoara

Create an environment in Timisoara in which

- IT research institutions, IT educational institutions, and IT companies can cooperate,
- develop intensive synergies,
- develop long-lasting effects for the regional economy in Timisoara
- and also long-lasting technologic and economic benefits for all partners.

## ■ Financing

- IeAT should be initiated and built up by significant seed money from the side of the Austrian Government,

*Copyright 2003 B. Buchberger*

- which should trigger off further investments from other public and private sides in Austria,
  Romania,and other countries and,in particular, the EU.

The results of the pilot phase will be the main criterion and motivation for the Austrian Government for sponsoring the main phase (2004 - 2014) of the project.

In the long run, IeAT and the IT-Park shold be self-sustained.

### ■ Strategical Goals

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 Role of the Model Project

### ■ A Model Project within the Pilot Phase

One particular activity during the pilot phase: a model joint research project between the CS 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 us to organize and carry out joint research projects within IeAT on a large scale.

### ■ The Subject Chosen: Program Verification

This choice is motivated by

- joint research expertise of the partner institutions,

Copyright 2003 B. Buchberger

- the topicality and difficulty of the subject,
- and also by the practical interest of the partner companies.

### ■ The Status of Verification Research

Program (algorithm) verification has a long research tradition but, so far, had relatively little impact on practical software development in the industry. Why?

A notable exception is the use of verification techniques in ckeching the design 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.

Measures for making program verification relevant for the IT-industry?

## ■ Making Program Verification Relevant for the IT-Industry

### ■ Verification and Specification

Exact and formal program verification needs exact and formal program specification.

### ■ An Important Observation re Practicability

In large-scale industrial software, most of the code is relatively "flat" (no loops, no recursion) and only relatively small parts of the code are logically "deep".

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).

- In contrast, for the deep parts, formal program verification *is* an appropriate tool. We definitely advocate the use of formal tools for these parts.
- Observation: In industrial code, the "deep" parts form only a relatively small part of the code.
- The use of techniques of formal exact verification, however, needs specific training of programmers in the formal logic aspects of computer science. A challenge!
- Program verification will only be attractive for industrial applications if a decent percentage of the actual verification task can be automated. i.e. the necessary proofs can be generated by computer software. A big challenge!

### ■ Procedural, Functional and Rewrite Languages

Procedural languages need a translation of the actual verification problem into a proof problem in a logical language like predicate logic.

*Copyright 2003 B. Buchberger*

Thus, there is a unavoidable 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 conveivable to program and to verify in the same language frame.

I advocate the development of rewrite languages in the frame of predicate logic in order to eliminate the gap between the verification language and the programming language.

### ■ Theorema

"Theorema" is a version of predicate logic a part of which is an elegant programming language.

Theorema is a uniform frame for specificantion, 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 wich all the above design objectives are realized.

Theorema is being developed in the Theorema Working Group at RISC (project leader: B. Buchberger).

## ■ Approaches to Program Verification

### ■ At One End of the Spectrum: Verification Condition Generation for Procedural Programs

Given: the specification of a problem. (Example: "B should be a sorted version of A".)

Find: a procedural program that meets the specification.

Method:

- invent the program
- invent the inductive assertions (at each instruction; better: in each "critical path" only one inductive assertion)
- generate the "verification conditions" from the program parts and the inductive assertions
- prove the verification conditions.

Remarks:

- verification condition generation can be easily automated
- proving the verification conditions is a challenge for automation but is possible with current technology

Copyright 2003 B. Buchberger

- finding the verification conditions is a kind of re-invention of the design ideas of the program and, hence, unnatural (some automation is possible by "symbolic execution")

- verification and execution of a program live in two different "worlds" (the world of procedural languages and the world of logic languages).

### ▪ At the Other End of the Spectrum: "Lazy Thinking" for Rewrite Programs

Given: the specification of a problem. (Example: "B should be a sorted version of A".)

Find: a rewrite function that meets the specification.

Method:

- start with a name for the function (Example: "sorted[A]")

- try the verification (using an automated proving system) w.r.t. to a (complete) knowledge base for the concepts involved in the problem specification

- the verification will, of course, fail but, by anlysis of the failures, a structuring of the verification problem into (independent) subproblems will be suggested

- throw in an "algorithm type", i.e. express the unknown function in terms of (unknown) subfunctions (Example: "sorted[A] := m[ sorted[ l[A]], sorted[ r[A]] ]")

- (observation: there are thousands of algorithms but only very few algorithm types)

- try the verification again

- the verification will fail but will yield specifications for the subfunctions (in the example: for m, l, r).

- proceed recursively until we are left with verification problems for known concepts.

Remarks:

- provers which display failing proofs are necessary

- invention of the program goes hand in hand with verification

- invention of programs is actually reduced to the invention of exact specifications (and the application of "algorithm types")

- verification and execution of programs live in the same "worlds" (the world of logic).

## ■ The Goal of the Workshop

Report on various approaches and aspects of program verification.

*Copyright 2003 B. Buchberger*

Report on techniques known in the literature and on our own work.

(We will have an extra internal meeting on a specific verification problem for our partner company.)

Copyright 2003 B. Buchberger