Symbolic Computation in Software Science: Foreword from the Editor

The series of workshops on Symbolic Computation in Software Science has been initiated by the research groups of Bruno Buchberger at RISC, Johannes Kepler University of Linz, Tetsuo Ida at the University of Tsukuba, and Masahiko Sato at Kyoto University. As the name suggests, the focus of the series is on applications of symbolic computation techniques in various areas of software science. The first workshop has been organized in the Castle of Hagenberg in July 2008. Following an open call afterwards, we received 15 submissions for this issue, five of which have been selected after rigorous reviewing.

The accepted papers address theoretical and practical aspects of software science using symbolic computation techniques. The topics include a special technique for declarative programming, enhanced pattern matching for rule-based programming, rewriting-based approach to behavioral specification of systems, verification of safety properties for infinite-state systems, and languages with variable binding and α-equivalence.

The tutorial by *Sergio Antoy* presents narrowing from a programmer's viewpoint. Narrowing is a symbolic computation technique, used in declarative programming, to rewrite or evaluate expressions that contain variables. It involves solving sets of equations, possibly with user-defined abstract data types. There have been many research papers that demonstrate the use of this technique. There have been surveys that discuss theoretical or implementation issues of narrowing. But there have been no tutorials so far on the use of narrowing for programming tasks. Antoy's article fills this gap. It explains, with carefully chosen examples, when, why, and how to use narrowing in programming. The presentation is largely informal and comprehensible for non-specialists.

Another symbolic technique that plays an important role in programming is pattern matching. In their article, *Horatiu Cirstea, Claude Kirchner, Radu Kopetz,* and *Pierre-Etienne Moreau* extend patterns with complement symbols, which permits base searching criteria on both positive and negative conditions. Such extended patterns are called anti-patterns. They have been used in the rule-based language Tom that is designed to add pattern matching capabilities to imperative languages like C or Java. In the paper, syntactic and AU (associative with unit) anti-pattern matching problems, algorithms to solve them, and their relationships with equational pattern matching are studied in detail.

In the article by *Masaki Nakamura, Kazuhiro Ogata,* and *Kokichi Futatsugi*, the notion of reducibility of operation symbols in term rewriting systems is introduced and its application to algebraic specifications is investigated. An operation symbol is reducible if any term containing the symbol is reducible. Its restriction, which allows to reduce terms by the context-sensitive rewrite relation instead of the standard rewrite relation, is useful in behavioral specifications. In particular, the article shows that, using such a restriction, one can obtain a sufficient condition for behavioral coherence in behavioral specifications,

and a sufficient condition for a behavioral specification to be an observational transition system/CafeOBJ specification.

Thomas Genet and *Vlad Rusu* address the problem of verification of safety properties for infinite-state systems modeled by term rewriting systems. The approach is based on the tree automata completion techniques. Given a term language defined by a tree automaton $\mathcal{A}$ and a term rewriting system $\mathcal{R}$, the tree automata completion algorithm produces another tree automaton $\mathcal{A}^*_{\mathcal{R}}$ that, usually, is an overapproximation of the set of terms reachable by $\mathcal{R}$ from the language recognized by $\mathcal{A}$. To verify safety properties, one then checks whether a bad state occurs in the overapproximation. In earlier approaches, defining approximation functions requires one to know how the completion technique works, which makes it quite a complex task for the user. The article deals with this problem, proposing a new tree automata completion algorithm based on the use of equations for approximation. The users who are not experts in tree automata formalism will find this new technique appealing because the approximation rules can be defined by equations, without referring to the structure of tree automata.

The article by *Masahiko Sato* and *Randy Pollack* presents a framework for treating languages with variable binding formally, in the form of distinction between external and internal syntax, and illustrates it using the λ-calculus. The external syntax is intended to be used by humans, while the internal syntax, where bound variables are represented by canonical names, can be easily implemented and is amenable to inductive reasoning. The paper gives an inductive characterization of the set of "well-behaved" canonical internal representatives of α-equivalence classes. By a semantic mapping, two external terms that are α-equivalent are mapped to the same internal term. Substitutions enjoy nice properties with respect to the mapping.

I would like to thank all the authors who submitted their papers to this issue. I would also like to thank all the reviewers who devoted their time and effort to evaluate the papers and to help authors to improve their contributions. And finally, I would like to thank Hoon Hong, the Editor-in-Chief of the Journal of Symbolic Computation, who suggested to organize this special issue and who has been very helpful throughout the editorial process.

Temur Kutsia
Research Institute for Symbolic Computation,
Johannes Kepler University,
Altenbergerstrasse 69,
A-4040 Linz,
Austria
E-mail address: kutsia@risc.uni-linz.ac.at

22 January 2010