Math.Comput.Sci. (2011) 5:357–358
DOI 10.1007/s11786-011-0105-2

**Mathematics in Computer Science**

# Foreword

**Georgios Fainekos · Eric Goubault · Sylvie Putot · Stefan Ratschan**

Published online: 4 December 2011


## 1 Overview and Scientific Context of this Special Issue

For us, the subject of numerical software verification is the application of logical and mathematical techniques to reasoning about numerical aspects of software. Which numerical aspects can software have? On the one hand, the core numerical aspect of software is its usage of numerical data types, especially in the computation with floating-point numbers. On the other hand, models (i.e., abstract mathematical descriptions of a given system) play a more and more important role in the development of software, and those models often have essential numerical aspects. Those numerical aspects may model the software itself (e.g., control engineers often design controllers based on ordinary differential equations, but then implement those controllers in digital software) or its physical environment (e.g., timing constraints, energy consumption, physical laws governing the environment in a control loop).

The first four papers of this volume belong to the first category, and the final two papers to the second category.

The first two papers deal with the verification of the accuracy of computations achieved by numerical programs that use finite precision date-types (floating-point numbers) to approximate computations in real numbers. Both papers propose software tools to assess this accuracy. The diversity in the approaches indicates the difficulty of the problem tackled.

The paper by Graillat and co-authors relies on stochastic arithmetic: the tool runs automatically, but can only give some probabilistic estimation of the accuracy of the result. The paper by Boldo and Marché presents a tool

G. Fainekos
Arizona State University, Tempe, USA
e-mail: fainekos@asu.edu

E. Goubault · S. Putot
CEA LIST, Gif-sur-Yvette Cedex, France
e-mail: Eric.Goubault@cea.fr

S. Putot
e-mail: Sylvie.PUTOT@cea.fr

S. Ratschan (✉)
Institute of Computer Science,
Academy of Sciences of the Czech Republic,
Prague, Czech Republic
e-mail: stefan.ratschan@cs.cas.cz

chain that gives a formal proof of the accuracy: although some emphasis has been put on the mechanisation of this proof, its use requires some expertise by the user. But the results are guaranteed.

An area where numerical computation plays a central role is high-performance scientific computing. In this context, programs usually undergo various transformations with the purpose of optimization and parallelization. It is important to know whether the outcome of such transformations is equivalent to the original program.

The two papers by Siegel and Zirkel push the boundaries along this line of research. Their first paper presents a tool (Toolkit for Accurate Scientific Software—TASS) for verifying that a user-provided program that has undergone such transformations, is equivalent to the initial version of this program. One important feature of TASS is that it can handle parallel programs that have been built using the Message Passing Interface (MPI). The second paper by the same authors presents in detail a library of numerical programs in C. Each benchmark problem in the library includes a sequential version of the numerical program and two parallel versions of the same code—one that has known bugs and another which is a correct implementation. Such a library of benchmark problems can hopefully facilitate research on the verification of scientific computation software.

A basic data type used by models in software development is the notion of a real function (i.e., a function whose domain/codomain ranges over the real numbers): for example, certain quantities may evolve in time, and hence, for modeling them, one uses functions in time. Classical numerical mathematics introduces round-off and discretization errors when computing with functions, and then analyzes algorithms manually to study the propagation of those errors. The resulting analysis may depend on unknown quantities (condition numbers), and hence, if not applied carefully, may have unexpected and—in certain contexts—even catastrophic consequences.

An alternative approach is to enclose the correct computation result into sets, and to ensure that the set computed in every step of an algorithm always includes the correct result. The paper by Collins and co-authors takes this alternative approach, and develops a general framework and a library for correctly over-approximating computation with real functions.

The topic of the final paper of this volume is the verification of hybrid automata. Such hybrid automata are an important formalism for modeling software in some physical environment. Here, the numerical aspect is formed by ordinary differential equations or inequalities, that can be used for both modeling controllers that will—in their final form—be implemented in software, or the continuous behavior of some physical environment. While most verification problems for hybrid automata are undecidable, the paper identifies various classes of verification problems that are decidable in polynomial, nondeterministic-polynomial, or exponential time.

## 2 Organizational Context

This journal volume represents a follow-up special issue to the Third International Workshop on Numerical Software Verification (NSV-3) that took place as part of the Federated Logic Conference FLoC in 2010 in Edinburgh. The predecessor instances of this workshop took place in St. Francisco (part of CPSWeek, 2009), and Princeton (held with CAV 2008), and a further instance as part of CAV 2011 in Snowbird, Utah. The first NSV workshop resulted in a special issue of the journal Formal Methods in System Design (volume 35, number 3), while the second one has an accompanying special issue titled “Verification of Cyber-Physical Software Systems” that is currently in press, and will appear in the ACM Transactions of Embedded Computing Systems.