Acta Informatica (2016) 53:325–326
DOI 10.1007/s00236-016-0259-0

EDITORIAL

Special issue on Rich Models, EU-COST Action IC0901 Rich-Model Toolkit

Bernd Finkbeiner[1] · **Cesar Sanchez**[2]

Published online: 11 February 2016

The main goal of The Rich Model Toolkit initiative was to explore directions and techniques for making automated reasoning (including analysis and synthesis) applicable to a wider range of problems, as well as making them easier to use by researchers, software developers, hardware designers, and information system users and developers. The Rich Model Toolkit was funded by the European Union as COST Action IC0901 between 2009 and 2013.

The action included the participation of researchers from over 20 countries and 50 research groups. The unifying idea of Rich Models is to explore precise mathematical and formal models of key aspects of our infrastructure, developing algorithms, tools, and common standardized formats. The networking activities funded by the action—particularly workshops, visits from young scientists to hosting research groups, and summer schools—aimed to establish connections between different tools, methodologies, and communities.

The objective was to build a unified infrastructure to clearly define Rich Models, to introduce standardized representation formats, and incorporates a number of automated reasoning tools with the ability to establish communication between these formats. During the action benchmarks and competitions for automated reasoning and verification were created, as well as several research results on decision procedures, analysis and synthesis. These activities included competitions on the verification of Numerical Transitions Systems and the participation of researchers from the action in the development of Horn-Clauses as a common representation of several verification problems.

After the end of the action, we invited researchers to submit significant contributions to this special issue of Acta Informatica. After two round of reviews from several experts the following four papers were selected for publication. We thank all reviewers for their commitment and dedication.

The paper "Symbolic Automata for Static Specification Mining" by Hila Peleg, Sharon Shoham, Eran Yahav and Hongseok Yang presents a new automata-based framework for

✉ Cesar Sanchez
cesar.sanchez@imdea.org

1 Universität des Saarlandes, Saarbruecken, Germany

2 IMDEA Software Institute, Madrid, Spain

the static mining of library specifications. Partial specifications are represented as symbolic automata, so techniques and tools for symbolic automata can be used to reason about, for example, partialness and precision of specification. The state of the automata correspond to abstract internal states of the API and transitions correspond to method calls.

The paper "Synthesizing Efficient Systems in Probabilistic Environments" by Christian von Essen, Barbara Jobstmann, David Parker and Rahul Varshneya studies the problem of synthesizing reactive systems that achieve optimal cost-reward. The theory is based on Markov decision processes, and the techniques proposed use game solving to compute optimal strategies for components, composing these strategies into a final strategy for the system. The paper includes an experimental comparison of the different algorithms.

The paper "Verification of Heap Manipulating Programs with Ordered Data by Extended Forest Automata" by Parosh Aziz Abdulla, Lukáš Holík, Bengt Jonsson, Ondřej Lengál, Cong Quy Trinh and, Tomáš Vojnar presents an approach to the verification of heap-manipulating programs based on tree automata. The abstract program heap is represented as forest-graphs, and sets of such forests are then encoded using forest automata. The paper presents an extension of forest automata that enables encoding constraints over data. The paper reports on the empirical evaluation of the technique verifying a number of heap manipulating programs.

The paper "Guiding Craig Interpolation with Domain-specific Abstractions" by Jerome Leroux, Philipp Ruemmer and Pavle Subotic studies a a method for improving the quality of interpolants used in verification of software programs, that is general, does not require invasive changes to the interpolating theorem prover and that is based on a constructive algorithm for exploring the interpolant space. The paper reports on experimental evaluation of the technique, implemented in an existing model-checker.