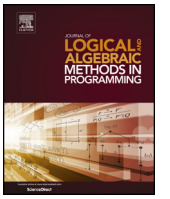
Journal of Logical and Algebraic Methods in Programming 85 (2016) 653–654

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp



# Automated verification of programs and Web systems

This special issue of the Journal of Logical and Algebraic Methods in Programming (JLAMP) is organised around the themes of the 10th International Workshop on Automated Specification and Verification of Web Systems (WWV) and the 2nd International Workshop on Verification and Program Transformation (VPT), both of which took place during the Vienna Summer of Logic on July 17th and 18th, 2014.

The WWV workshop series aims to provide an interdisciplinary forum to facilitate the cross-fertilisation and the advancement of hybrid methods that exploit concepts and tools drawn from rule-based programming, formal methods, software engineering, and Web-oriented research. The VPT workshop series aims to bring together researchers working in the fields of program verification and program transformation to raise the awareness and stimulate the development of novel and challenging verification methods and techniques.

Recent research in automated verification and program transformation has shown a great potential for beneficial interactions. On the one hand, the methods, techniques and tools developed in the field of program transformation have been successfully applied for the verification of programs, (Web) systems and protocols specified by programs. Examples are partial evaluation, partial deduction, fold/unfold transformations, supercompilation and distillation, in particular for the verification of infinite-state and parameterised systems. On the other hand, verification by model checking, abstract interpretation, automated and interactive theorem proving, and SAT/SMT solving have been used to strengthen and optimise program transformation techniques. Moreover, the formal certification of program transformation tools, such as automated refactoring tools and compilers, has recently attracted considerable interest and posed major challenges. Finally, since many Web systems have become interactive, completely automated, Web-based applications to be used in areas like e-health, e-business, e-learning, and e-government, their increased complexity has made their design and implementation a challenging task. Hence the use of systematic, formal approaches for their specification and verification allows to address the problems of this specific domain by means of automated and effective techniques and tools.

An open call for papers led to the submission of nine papers. After extensive reviewing, during which each paper was reviewed by at least three experts, the following six papers were accepted.

The article *Implementing type systems for the IDE with Xsemantics* by Lorenzo Bettini showcases Xsemantics, which is an Eclipse-based tool designed to ease the implementation of type systems for general-purpose programming languages and domain-specific languages developed within the Xtext Integrated Development Environment (IDE). The syntactic aspects, including parsing and syntax highlighting, are handled by Xtext. The abilities of Xsemantics are described by showing partial implementations of typing rules, auxiliary functions, and reduction rules for both Featherweight Java and a simple typed $\lambda$-calculus, as well as several additional features including extensions, caching, and IDE support.

The article *A descriptive type foundation for RDF Schema* by Ross Horne, Gabriel Ciobanu, and Vladimiro Sassone presents a type system to address the classification of linked data, in particular RDF Schema, which is a collection of rules for reasoning about type, subtype, and property relationships for RDF data (triples of subject, predicate, and object, using URIs). In contrast to standard prescriptive type systems, which guarantee absence of runtime type errors, their descriptive type system enables the detection of simple data-modelling slips, making it more suitable for aspects of RDF Schema concerning simple data types like integers and strings. The type system deals with type mismatches interactively: when a mismatch is detected, it presents refinements that would solve the problem (of which the default RDFS behaviour is only one of several possibilities) leaving it to the programmer to choose the most intuitive one.

The article *Debugging Maude programs via runtime assertion checking and trace slicing* by María Alpuente, Demis Ballis, Francisco Frechina, and Julia Sapiña presents a language for defining assertions in Maude to allow for the debugging of Maude programs by runtime assertion checking. These assertions can be either about states, called system assertions, or on reductions, called functional assertions. Once an assertion has been detected to fail, trace slicing is used to provide information on the causes that may have led to the violation, thus providing assistance for locating the error. Generalization is used to identify the variables of interest to observe in the slicing. The resulting assertion-based, dynamic trace analysis methodology has been implemented in a prototypical tool, which is presented as well.

http://dx.doi.org/10.1016/j.jlamp.2016.06.005
2352-2208/© 2016 Published by Elsevier Inc.

The article *Declarative layout constraints for testing Web applications* by Sylvain Hallé, Nicolas Bergeron, Francis Guérin, Gabriel Le Breton, and Oussama Beroual addresses layout-related 'bugs' in Web applications. Real-world Web sites and applications are surveyed to identify the different types of bugs which may occur. An automated testing tool called Cornipickle is presented, based on a declarative language to specify the required properties of a Web application. These properties, expressed in a first-order extension of linear temporal logic, must be observable from the perspective of the client (i.e. when viewing the Website via a browser). These properties can be verified in a semi-automatic way, in the sense that a user (or robot) views and interacts with a Webpage while the system analyses the stream of HTML documents ('trace') that results from taking a snapshot of the Webpage after each relevant user-triggered event. Cornipickle requires no browser-specific plugins, all information is gathered on the client side, and all verification is carried out on the server side.

The article *Realizable temporal logics for Web service choreography* by Ramaswamy Ramanujama and Shamimuddin Sheerazuddin proposes to model Web service choreographies using temporal logics, which ensure realisability of those choreographies by design. A choreography describes from a global viewpoint how a set of communicating services exchange messages among each other, i.e. their execution order. Realising such descriptions means verifying whether there exists a distributed system that implements the same conversation as described by a given choreography. The article focusses on distributed systems with an asynchronous communication semantics over message buffers. Relying on the idea of partial orders to specify realisable choreographies, two restricted (local) temporal logics, called p-LTL and q-LTL, are introduced for specifying Web service choreographies, together with techniques for synthesising service implementations in terms of systems of communicating automata and message-passing automata, respectively.

The article *Ping-pong protocols as prefix grammars: Modelling and verification via program transformation* by Antonina Nepeivoda presents a novel approach to the verification of a class of cryptographic protocols (so-called multi-party ping-pong protocols). It is based on modelling the protocols by prefix grammars in such a way that the verification problem is reduced to deciding whether or not a word from a particular set of 'private' words is derivable in the grammar by unfolding its generated traces. The analysis of prefix grammars requires encoding them by a functional program, after which supercompilation, a generic program transformation technique, is applied. Furthermore, a set of possible attack models is built explicitly. Finally, experiments with a real supercompiler are reported.

We thank all authors for their contributions and the numerous referees for their thorough and exhaustive reviewing. We are grateful to Rocco De Nicola, JLAMP's editor-in-chief, Luca Aceto, the editor responsible for special issues, and Alberto Lluch Lafuente, the editorial manager, for their support in the creation of this special issue. Finally, we are grateful to the people at Elsevier for their support, assistance, and—above all—patience during the editorial process.

Maurice H. ter Beek
*ISTI–CNR, Pisa, Italy*
*E-mail address:* maurice.terbeek@isti.cnr.it

Alexei Lisitsa
*University of Liverpool, UK*
*E-mail address:* a.lisitsa@liverpool.ac.uk

Andrei P. Nemytykh
*Russian Academy of Sciences, Russia*
*E-mail address:* nemytykh@math.botik.ru

António Ravara
*Universidade Nova de Lisboa, Portugal*
*E-mail address:* aravara@fct.unl.pt

Available online 27 June 2016