Computing and Informatics, Vol. 35, 2016, 761–763

SPECIAL SECTION ON FUNCTIONAL PROGRAMMING PARADIGM AND ITS APPLICATIONS

Wojciech Turek, Roman Dębski, Aleksander Byrski
Marek Kisiel-Dorohinicki

AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
e-mail: {wojciech.turek, rdebski, olekb, doroh}@agh.edu.pl

Current trends in hardware development provide powerful computing platforms, which support large number of independent processing units. Implementation of highly concurrent systems, which can efficiently use this type of hardware, poses a significant challenge for the practitioners. Parallel execution both on manycore architectures and on distributed hardware rises issues related to proper synchronization between threads and processes, development of robust communication protocols, and maintaining scalability and resilience of systems, to name a few.

Many mechanisms covering the above-mentioned issues are easily available for the developers following functional paradigm of programming. Its inherent capabilities, such as referential transparency, lazy evaluation and absence of side-effects in pure functional languages make them perfect as a means for implementation of concurrent algorithms and distributed systems. Choosing such languages as Erlang, Scala/Akka, Haskell, F# or Clojure simplifies implementation of concurrent systems, making utilization of modern hardware platforms far more efficient.

Recent years have brought fast development of the paradigm and the languages that use it, changing the approach to design and implementation of large scale and massively concurrent systems. Thus in this special section, we focus on selected issues connected with functional paradigm utilization and development, in order to present the promising, novel solutions to the researchers interested in functional programming.

The first part of the section shows programming language development issues, showing the idea of automatic parallelization based on static program analysis with refactoring support in the context of Erlang (Tamás Koszik, Melinda Tóth, István Bozó, Zoltán Horváth: Static Analysis for Divide-and-Conquer Pattern Discovery). Next, the construction of algorithmic skeletons (parallel programming patterns) is discussed, showing an Erlang-based computing system as a use case (Adam D. Barwell, Christopher Brown, Kevin Hammond, Wojciech Turek, Aleksander Byrski: Using Program Shaping and Algorithmic Skeletons to Parallelise an Evolutionary Multi-Agent System in Erlang). Finally, metaprogramming issues are addressed by the authors trying to employ Haskell into flexible and reliable generation of C preprocessor macros (Bodizsár Németh, Máté Karácsony, Zoltán Kelemen, Máté Tejfel: Defining C Preprocessor Macro Libraries with Functional Programs).

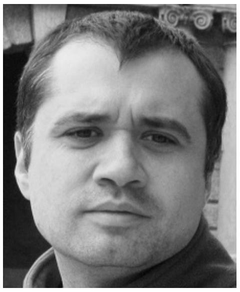
The following two papers are focused on reactive programming. The first one presents the development of the Reactive Streams API in order to fit better for mobile applications (Przemysław Dadel, Krzystof Zieliǹski: Evolution of Reactive Streams API for Context-Aware Mobile Applications), while the second one examines the challenges and advantages of using an actor framework for the programming and execution of scientific workflows (Bartosz Balis, Krzysztof Borowski: Using an Actor Framework for Scientific Computing: Opportunities and Challenges).

The authors of the last paper show how to make easier-to-read the property-based testing, by transforming the properties and the test models into semi-natural language representation, allowing not only skilled programmers to interfere with the complex and demanding process of testing (Laura M. Castro, Pablo Lamela, S. Thompson: Making Property-Based Testing Easier to Read for Humans).

These valuable research results are only the examples of the great potential and the increasing interest given to functional programming paradigm nowadays. The further development of functional languages and their applications will definitely lead to novel solutions in various areas of computer science, especially in the context of building large-scale systems.

Wojciech Turek received his Ph.D. in 2010 from the AGH University of Science and Technology in Cracow. He works as Assistant Professor at the Department of Computer Science of AGH-UST. His research focuses on agent-based systems, multi-robot systems and functional programming.

Roman Dębski holds his M.Sc. in computer science (AGH University of Science and Technology) and in mechanical engineering and Ph.D. in computational mechanics (both received from the Cracow University of Technology). He has over 14 years of experience in IT and currently works as Assistant Professor at the AGH University of Science and Technology. His current research focuses on modelling and simulation.



Aleksander Byrski received his Ph.D. in 2007 and D.Sc. (habilitation) in 2013 from the AGH University of Science and Technology in Cracow. He works as Assistant Professor at the Department of Computer Science of AGH-UST. His research focuses on multi-agent systems, biologically-inspired computing and other soft computing methods.

Marek Kisiel-Dorohinicki received his Ph.D. in 2001 and D.Sc. (habilitation) in 2013 from the AGH University of Science and Technology in Cracow. He works as Assistant Professor at the Department of Computer Science of AGH-UST. His research focuses on intelligent software systems, particularly using agent technology and evolutionary algorithms, but also other soft computing techniques.