ELSEVIER

Science of Computer Programming 37 (2000) 1–2

Science of Computer Programming

www.elsevier.nl/locate/scico

Foreword

The 4th International Conference on Mathematics of Program Construction was held in June 1998 on the island of Marstrand near Göteborg in Sweden. This special issue of Science of Computer Programming contains a selection of the papers presented at the conference. The conference papers have been extended and improved considerably for this special issue.

The general theme of this series of conferences is the use of crisp, clear mathematics in the discovery and design of algorithms and in the development of corresponding software or hardware. A more extensive description of the field is given by Backhouse [1]. Of course, 'the' mathematics of programming does not exist. Just as we need domain specific languages to implement software for specific domains, we need domain specific mathematics to construct and reason about software for such domains. This need for domain specific mathematics is witnessed by the fact that the conference was followed by three workshops with a more narrow focus: workshops on constructive methods for parallel programming, generic programming, and formal techniques for hardware and hardware-like systems. This special issue also contains a selection of papers presented at these workshops.

The six papers selected from the mathematics of program construction conference cannot show more than a fraction of the field of mathematics of program construction. The topics discussed in this issue are:

- Dijkstra introduces the computation calculus: an algebra that is intended to bridge the gap between abstract programming formalisms and their operational interpretations.
- Gibbons constructs generic accumulations: programs that accumulate information up and down any kind of tree. He shows that the resulting, elegant, definitons satisfy a number of desirable properties.
- Hughes introduces arrows, a combinator library that generalizes monads. He shows how to use this library in the construction of libraries for stream processors and programming active web pages. These libraries would have been difficult or impossible to implement with monads.
- Joshi and Leino give a general condition that specifies a secure program, and show how this condition is used to prove that a program is secure. This approach gives a finer control over security than the known abstract interpretation techiques.
- Von Karger introduces a calculus for reactive systems. He starts with the logical connectives and the axioms of Boolean algebra, and extends this calculus in several ways to obtain calculi such as the sequential calculus for reactive and, especially, real-time systems.

PII: S0167-6423(99)00020-9

- Sørensen presents a framework for proving termination of program transformers. He introduces the notion of an abstract program transformer, and gives a sufficient condition for such a transformer to terminate. Using this framework, he shows that positive supercompilation terminates.

Three papers come from the International Workshop on Constructive Methods for Parallel Programming:

- Jay proposes a preliminary parallel version of his language FISh, whose special trait is the anlaysis of the shape (i.e. topology and size) of data structures. One central requirement for efficient prallelism is the reference to an accurate cost model; Jay sketches one based on shapes.
- Nitsche also uses properties of shape to optimize parallel programs, which are defined by program templates (the so-called skeletons). His requirement is shapeliness: in a mapping, the shape of the result must only be determined by the shape of the arguments, not by their content. He exploits this prerequisite in optimizing the dynamic distribution of data.
- Loulergue, Hains and Foisy propose an extension of the λ-calculus for bulk-synchronous programming (BSP), which they call BSλ, and relate the reduction process in BSλ to the BSP cost model.

Finally, one paper is included from the Workshop on Formal Techniques for Hardware and Hardware-like Systems:

- McMillan presents a methodology for system-level hardware verification based on compositional model checking. The problem of a verifying large system is decomposed into small finite state subgoals, which are discharged by symbolic model checking. The appplication of the methodology, which is supported by a special purpose proof system, is illustrated by examples, including Tomasulo's algorithm for implementing out -of-order execution in instruction set processors.

We hope you enjoy reading these papers.

Johan Jeuring
Christian Lengauer
Mary Sheeran
Department of Computer Science,
P.O. Box 80.089, 3508 TB Utrecht, *The Netherlands*

References

[1] R. Backhouse, Mathematics of Program Construction, Sci. Comput. Programm. 26 (1996) 5–9.