ELSEVIER

Theoretical Computer Science 190 (1998) 1–2

Theoretical Computer Science

Editorial

This special issue contains extended versions of four papers presented at the 21st Colloquium on Trees in Algebra and Programming CAAP '96, which was held on 22–26 April 1996, in Linköping (Sweden).

The CAAP colloquium series was originally devoted to the algebraic and combinatorial properties of trees and their roles in various fields of computer science. Its scope has extended to several other structures, and now covers algebraic, logical, and combinatorial properties of discrete structures, and their application, in particular to syntax and semantics of programming languages and concurrent systems, and to logic and formal verification.

Papers selected for this special issue address the topics of term rewriting systems and interaction nets, logics for verification of concurrent systems, semantics of deterministic programs in nondeterministic environments, behavioural semantics for CCS-like languages.

Term rewriting systems and interaction nets have in common to provide a framework for specification and programming, based in the first case on a traditional syntax of oriented equations, and in the second case on a graphical syntax closer to implementations. The aim of the paper *Interaction Nets and Term Rewriting Systems* is to bridge the gap between the two formalisms. The authors, Maribel Fernández and Ian Mackie, provide a translation from interaction nets to term rewriting systems, in order to transfer results from one paradigm to the other. They derive results concerning termination and modularity of interaction nets. An interesting research direction, opened by this paper, is conversely to carry over the semantics of interaction nets to the term rewriting area.

Computation tree logic is used in specification and verification of concurrent systems. An extension is the extended computation tree logic, for which Roope Kaivola proposes a sound and complete axiomatisation, in *Axiomatising Extended Computation Tree Logic*. The completeness proof is based on a transformation of formulae to a specific normal form obtained through a deterministic finite automaton on infinite strings, and illustrates how insights and techniques from automata theory can be used to show completeness of a logical axiomatisation. This approach may help to solve the still open problem of a complete axiomatisation for computation tree logic.

Computer programs can often be decomposed into a computational component, describing what kind of computations are carried out, and a descriptive component that models the environment. In *Computing in Unpredictable Environments: Semantics, Reduction Strategies, and Program Transformations*, Björn Lisper proposes to

PII S0304-3975(97)00081-9

formalize systems where deterministic computations take place in environments which may behave non-deterministically, by unions of abstract reduction systems. A normal form semantics, handling infinite fair computations, is proposed, reduction strategies for systems with computational and descriptive components are explored, and program transformations preserving the semantics are studied. Supporting the design of lazy recursive languages with process primitives is a possible further application of this formalism.

The notion of behavioural semantics for CCS-like languages has been formalized through the concept of bisimulation. To capture this notion inside a categorical framework, spans of open maps in category theory have been proposed to define an abstract equivalence, called $\mathscr{P}$-bisimilarity, on the category of models of computations, where $\mathscr{P}$ is a sub-category of observations. An issue left open was to find conditions under which algebraic constructs viewed as functors preserve $\mathscr{P}$-bisimilarity. The paper by Allan Cheng and Mogens Nielsen, entitled *Open Maps, Behavioural Equivalences, and Congruences*, answers this question by introducing the notion of $\mathscr{P}$-factorizable functor. For the category of labelled transition systems and functors representing CCS-like operators, the paper shows which conditions $\mathscr{P}$ must satisfy in order for these functors to be $\mathscr{P}$-factorizable. In the future, other interesting behavioural equivalences, such as weak bisimulation, are to be explored.

As guest editor, I would like to express my thanks to the authors for contributing to this special issue with high quality papers, and to the referees who participate in the selection and the improvement of these papers. I am grateful to Maurice Nivat for the opportunity of this special issue that, I hope, is reflecting part of interests of the Theoretical Computer Science community.

Hélène Kirchner
Guest Editor