Math. Struct. in Comp. Science (2008), *vol.* 18, *pp.* 427–429. © 2008 Cambridge University Press
doi:10.1017/S0960129508006737 Printed in the United Kingdom

Rewriting calculi, higher-order reductions and patterns: introduction

HORATIU CIRSTEA[†] and MARIBEL FERNÁNDEZ[‡]

[†]*LORIA and Nancy-Université,*
54506 Vandoeuvre-lès-Nancy Cedex, France
[‡]*King's College London, Department of Computer Science,*
Strand, London WC2R 2LS, UK.

Received 16 July 2007

The integration of first-order and higher-order paradigms has been one of the main challenges in the design of both declarative programming languages and proof environments. It has led to the development of new computation models and new logical frameworks, which have been obtained by enriching first-order rewriting with higher-order capabilities or by adding algebraic features to the λ-calculus.

The Rewriting Calculus was introduced in the late 1990's as a general means of uniformly integrating rewriting and the λ-calculus. This calculus makes all of its components explicit and first class: these components include patterns and pattern-matching (possibly modulo given equational theories), abstraction, application and substitutions. The rewriting calculus is designed and used for logical and semantical purposes. It can be combined with powerful type systems, and has been used for expressing the semantics of both rule-based and object-oriented paradigms. It allows one to express exceptions and imperative features naturally as well as to express elaborated rewriting strategies.

The wealth of new results and the opening up of new, exciting, research directions in the area of pattern-calculi, first and higher-order rewriting, and programming language semantics led to the introduction, beginning in 2004, of dedicated international workshops. The first Rewriting Calculus Workshop took place in Nancy in March 2004, followed by workshops at École Polytechnique in May 2005 and at King's College London in 2006. This special issue on *Rewriting calculi, higher-order reductions and patterns* is devoted to the foundations of the rewriting calculus and of related systems combining pattern-matching and higher-order features. It came out of a call for papers after the second workshop on the Rewriting Calculus, with the aim of including a selection of extended versions of papers from the workshops in addition to opening it up to other authors.

https://doi.org/10.1017/S0960129508006737 Published online by Cambridge University Press

Four articles were selected for this special issue, covering the following topics.

Pure Pattern Type Systems

Pure Pattern Type Systems combine the expressive power of rewriting and the λ-calculus, and are interesting from both computational and logical points of view (thanks to the Curry–Howard isomorphism). The strong normalisation property entails the logical soundness of these systems. The article by Benjamin Wack and Clement Houtmann, *Strong normalisation in two Pure Pattern Type Systems*, provides a positive answer to an open problem: the strong normalisation of the Pure Type Systems in Barendregt's cube generalises to the Pure Pattern Type Systems in two cases: the simply typed system and the dependently typed system.

Imperative rewriting calculus

The article by Luigi Liquori and Bernard Paul Serpette, *iRho: an imperative rewriting calculus*, introduces an imperative version of the Rewriting Calculus of Cirstea and Kirchner, which can be seen as the kernel for an imperative language based on pattern-matching. The paper describes a type system for the calculus, and gives a formal operational semantics that can be used to derive an interpreter in a direct way. The proofs of some of the properties of the calculus, like the decidability of the type checking, are given explicitly in the paper, while some others, like subject reduction and the determinism of the interpreter, have been checked by the proof-assistant Coq.

Lambda-calculus and choice operators

The article by David Sabel and Manfred Schmidt-Schauß, *A call-by-need lambda-calculus with locally bottom-avoiding choice: context lemma and correctness of transformations*, investigates the operational theory of a core higher-order programming language based on the call-by-need lambda-calculus that includes letrec bindings, sequential composition, constructors and case-expressions, and McCarthy's amb (ambiguous choice) operator. The latter introduces a powerful form of non-determinism, which is more general than, for instance, erratic choice or parallel or. The paper defines a strategy of reduction called fair normal order reduction, and a notion of contextual equivalence, which is used to prove the correctness of program transformations. In addition, this article includes a proof of a standardisation theorem for fair normal order reduction.

Computation models and classical sequents

The article by Steffen van Bakel and Pierre Lescanne, *Computation with Classical Sequents*, introduces a continuation-style formal language called $\mathscr{X}$, and shows that a typed sublanguage provides a Curry–Howard isomorphism for a sequent calculus version of implicative classical logic. $\mathscr{X}$ can also be seen as a graphical programming language, since terms in this calculus describe nets by composition of basic elements connected by wires.

https://doi.org/10.1017/S0960129508006737 Published online by Cambridge University Press

The expressive power of the calculus is demonstrated in this paper by showing how several well-known higher-order languages can be easily encoded into $\mathscr{X}$.

Acknowledgements

We would like to thank the authors who contributed to this special issue, and the referees who worked diligently to evaluate and help improve the papers. Very special thanks are due to Giuseppe Longo, the editor-in-chief of Mathematical Structures in Computer Science, for his support and advice with the preparation of this special issue.

https://doi.org/10.1017/S0960129508006737 Published online by Cambridge University Press