Editorial

Special Issue Dedicated to the Fifth International Symposium on Functional and Logic Programming (FLOPS 2001)

FLOPS is a forum for research on all issues concerning functional programming and logic programming. In particular, it aims to stimulate the cross-fertilization as well as integration of the two paradigms. The symposium takes place about every 1.5 years in Japan. Previous FLOPS meetings were held in Fuji-Susono (1995), Shonan (1996), Kyoto (1998), and Tsukuba (1999).

This special issue contains four papers selected from those presented at the symposium. All papers have been resubmitted after the symposium to a new reviewing process so as to meet the standards of JFLP. Each paper has been carefully reviewed by at least two referees.

The first article by Elvira Albert, Michael Hanus, and German Vidal describes an online partial evaluator for a concurrent functional logic programming language. It has been integrated into an implementation of the language Curry and corresponding experimental results are provided. Besides the implementation, the main contribution of the paper is the extension of the underlying calculus to a practically useful language. In particular, features like case lifting, external functions, constraints, guarded expressions, higher-order functions, and or-nodes are considered.

The next paper by Pierre Deransart and Jan-Georg Smaus studies in detail the prescriptive typing of logic programs. Prescriptive means that only well-typed programs are given a semantics. This is opposed to descriptive typing which builds a finite but usually approximate description of the success-set of a program. An important result of a prescriptive type theory is the so-called subject reduction theorem, which says that computation preserves well-typedness. The authors completely revisit the prescriptive type theory of logic programming and the so-called head condition in a semantic framework called derivation trees.

The paper by Aart Middeldorp, Taro Suzuki, and Mohamed Hamada investigates functional logic programs where functions are defined by conditional rewrite rules which may also contain extra variables in conditions. The operational semantics is based on the novel calculus LCNC, which is applicable to a wide range of programs (even to programs without constructor discipline). For this operational semantics, the authors provide three interesting results: 1) for confluent rewrite systems, LCNC with the leftmost selection strategy for equations in goals is complete w.r.t. normalizable solutions, 2) LCNC is complete whenever basic conditional narrowing is complete, 3) LCNC is complete for terminating and level-confluent rewrite systems with extra variables.

Finally, the article by Masahiko Sato, Takafumi Sakurai, and Yukiyoshi Kameyama presents a simply typed lambda calculus that takes both contexts and environments as first class values. In this calculus, holes can be naturally represented by ordinary variables and hole filling can be represented by function application, with the help of a new abstraction mechanism introduced for managing packing and unpacking of those terms which are to be filled in the holes. The new calculus enjoys a number of good properties, such as subject reduction, confluence, and strong normalizability.

We thank all the people who contributed to this special issue. In particular, we wish to thank all the referees for their careful reviews and the authors for preparing and submitting extended versions of their conference papers.

Münster and Tokyo, 1 March 2002

Herbert Kuchen
Kazunori Ueda