Lecture Notes in Computer Science 1158

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Stefano Berardi Mario Coppo (Eds.)

Types for Proofs and Programs

International Workshop, TYPES '95
Torino, Italy, June 5-8, 1995
Selected Papers

Springer

Volume Editors

Stefano Berardi
Mario Coppo
Università degli Studi di Torino, Dipartimento di Informatica
Corso Svizzera 185, I-10149 Torino, Italy
E-mail: {berardi/coppo}@di.unito.it

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Types for proofs and programs : selected papers / International Workshop TYPES '95, Torino, Italy, June 5 - 8, 1995. Stefano Berardi ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Budapest ; Hong Kong ; London ; Milan ; Paris ; Santa Clara ; Singapore ; Tokyo : Springer, 1996
(Lecture notes in computer science ; Vol. 1158)
ISBN 3-540-61780-9
NE: Berardi, Stefano [Hrsg.]; International Workshop TYPES <3, 1995, Torino>; GT

CR Subject Classification (1991): F.4.1, F.3.1, D.3.3, I.2.3

ISSN 0302-9743
ISBN 3-540-61780-9 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN 10549802 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

Preface

This book is a selection of papers presented at the third annual workshop held under the auspices of the ESPRIT Basic Research Action 6453 *Types for Proofs and Programs.* It took place in Torino, Italy, from the 5th to the 8th of June 1995. Eighty people attended the workshop.

We thank the European Community for the funding which made the workshop possible. We thank Franco Barbanera, Luca Boerio, and Ferruccio Damiani, who took care of the local arrangements. Finally, we thank the following researchers who acted as referees: P. Audebaud, T. Altenkirch, F. Barbanera, G. Barthe, U. Berger, I. Beylin, T. Coquand, C. Cornes, P. Curmin, M. Dezani, P. Dybjer, H. Geuvers, E. Giménez, U. Herbelin, M. Hofmann, F. Honsell, Z. Luo, L. Magnusson, V. Padovani, H. Persson, C. Paulin-Mohring, I. Polack, A. Ranta, M. Ruys, A. Saibi, T. Schreiber, H. Schwichtenberg, K. Slind, J. Smith, M. Stefanova, Y. Takayama, T. Tammet, D. Terrasse, J. von Plato.

This volume is a follow-up to *Types for Proofs and Programs '93*, LNCS 806, edited by H.Barendregt and T.Nipkow and *Types for Proofs and Programs '94*, LNCS 996, edited by P. Dybjer, B. Nordstrom, and J. Smith. *Types for Proofs and Programs* is a continuation of ESPRIT Basic Research Action 3245 *Logical Frameworks: Design, Implementation and Experiments.* Papers from the annual workshops of these projects are collected in the books *Logical Frameworks* and *Logical Environments.* Both volumes were edited by G. Huet and G. Plotkin and published by Cambridge University Press.

Torino July 1996

Stefano Berardi and Mario Coppo

Contents

Introduction

The papers in these proceedings focus on various aspects of the development of computer-aided systems for formal reasoning using logical frameworks based on type theory. The most important applications we are interested in are the mechanization of mathematics and the realization of powerful tools for real software development. A logical framework provides a formalism in which a large class of theories can be represented. This is important since experience has shown that different aspects of mathematics and computer science are better represented using different theories. Moreover an implementation of the framework provides a proof system for each of the theories represented in it.

Type theory is a formalism in which theorems and proofs, specifications and programs can be represented in a uniform way. In particular, a type can be understood both as a proposition and as a specification, and a term having that type can be seen both as a proof of that proposition and as a program meeting that specification. A characteristic feature of type theory is that it supports constructive reasoning, a kind of reasoning frequently used in computer science, but very little developed so far inside computer-aided systems. The logical frameworks based on type theory which have been designed and tested during the TYPES project are Alf, Coq, and LEGO. They follow the same leading ideas but differ in the type theory on which they are based and in some choices regarding implementation. A related logical framework is Isabelle, which is based on higher order logic.

We expect these tools will soon help researchers in mathematics and computer science in developing software and checking its correctness, as systems like Mathematica or Maple already do in the more restricted field of symbolic computing. As compared to these latter, computer-aided systems for formal reasoning are less advanced at present, but are intended to be of broader use since they will help in any situation where logical or mathematical reasoning is required.

The papers in these proceedings deal with the three main aspects of the project: foundations of type theory and logical frameworks, implementation, and applications.

In the group of foundational papers, Barthe and Ruys and Barendregt develop an equational description of a proof checking algorithm, which is a basic tool in a logical framework. Barthe studies the possibility of including the notions of inheritance and overloading in type theory. Berger and Schwichtenberg give an example of extraction of a program from a classical proof. Cederquist and Negri describe the formalization in type theory of a central result of mathematical analysis. Coquand and Smith explain how to derive a typical logical result inside a constructive formalism. Curmin introduces an algorithm for the extraction of a program from a constructive proof. Stefanova and Geuvers introduce a new class of models for the calculus of construction, the formalism upon which the

Coq system is based. Hofmann and Padovani solve two interesting open problems in type theory. Von Plato, finally, presents a methodological reflexion on the translation of mathematical concepts and results in our constructive setting.

Another group of papers deals more closely with implementation aspects. Cornes and Terrasse describe a possible implementation of inductive reasoning in Coq. Magnusson describes an implementation of a proof-checking algorithm in Alf. Smith and Tammet investigate the theoretical background of a complex proof search algorithm. Ranta's paper is a (mostly theoretical) study of an algorithm for translating a symbolic proof in English.

The remaining papers are on the side of the applications. Under this heading we classify both large-scale testing, like examples of development of theoretical computer science within our systems, and industrial applications. Beylin and Dybjer develop in Alf a basic result in category theory. Dybjer also formalizes in Alf a part of the type theory on which Alf itself is based. Giménez reports on an industrial application: a formal correctness proof for the alternating bit protocol developed in Coq. Honsell and Miculan encode dynamic logic in Coq. Finally, Paulin-Mohring develops a correctness proof for a multiplier circuit in Coq using streams.