Alfonso Miola (Ed.)

Design and Implementation of Symbolic Computation Systems

International Symposium, DISCO '93
Gmunden, Austria, September 15-17, 1993
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Volume Editor

Alfonso Miola
Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"
Via Salaria, 113, I-00198 Roma, Italia

CR Subject Classification (1991): D.1, D.2.1, D.2.10, D.3, I.1, I.2.2-3, I.2.5, I.3.5-6

6330

ISBN 3-540-57235-X Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-57235-X Springer-Verlag New York Berlin Heidelberg

Typesetting: Camera-ready by author
Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
45/3140-543210 - Printed on acid-free paper

Foreword

This volume contains the proceedings of the Third International Symposium on Design and Implementation of Symbolic Computation Systems, DISCO '93.

The growing importance of systems for symbolic computation has essentially influenced the decision to organize the DISCO conference series. DISCO '93 takes place in Gmunden, Austria, September 15 - 17, 1993, as an international event in the field, organized and sponsored by the Research Institute for Symbolic Computation (University J. Kepler, Linz, Austria) and by the Dipartimento di Informatica e Sistemistica (University "La Sapienza", Roma, Italy).

DISCO '93 focuses mainly on the most innovative methodological and technological aspects of hardware and software system design and implementation for symbolic and algebraic computation, automated reasoning, geometric modeling and computation, and automatic programming.

The international research communities have recognized the relevance of the proposed objectives and topics which are generally not well covered in other conferences in the areas of symbolic and algebraic computation.

DISCO '93 includes papers on theory, languages, software environments, architectures and in particular, papers on the design and the development of significant running systems.

The general objective of DISCO '93 is to present an up-to-date view of the field, while encouraging the scientific exchange among academic, industrial and user communities on the development of systems for symbolic computation. Therefore it is devoted to researchers, developers and users from academia, scientific institutions, and industry who are interested in the most recent advances and trends in the field of symbolic computation.

The Program Chairman received 56 submissions for DISCO '93 and organized the reviewing process in cooperation with the Program Committee. Each paper was sent to two Program Committee members and then carefully reviewed by at least three independent referees, including Program Committee members. The Program Committee met on April 13 to 14, 1993 at the Dipartimento Informatica e Sistemistica. Università di Roma "La Sapienza" (Italy), to reach the final decision on acceptance of the submitted papers. The resulting DISCO '93 Scientific Program corresponds well to the initial objectives.

Among the submissions, 22 papers were selected as full contributions for presentation at the conference, as well as in this volume, under classified sections. Six further papers were selected as contributions for a presentation at the conference, concerning work in progress or running systems relevant to the themes of the symposium. These papers are included in a separate section of the present volume.

All my personal appreciation goes, in particular to Franz Lichtenberger, the Symposium Chairman, and to both the Program Committee and the Organizing Committee members for their indefatigable and valuable cooperation.

On behalf of the Program Committee, I would like to thank the authors of the submitted papers for their significant response to our Call for Papers, the invited speakers for having agreed to make their outstanding contributions to DISCO '93, and the referees for their cooperation in timely and precisely reviewing the papers.

Roma, July 1993 Alfonso Miola

Symposium Officers

General Chairman

F. Lichtenberger (Austria)

Program Committee

J. Fitch (UK), C. M. Hoffmann (USA), H. Hong (Austria), C. Kirchner (France), A. Kreczmar (Poland), A. Miola (Chairman, Italy), M. Monagan (Switzerland), E. G. Omodeo (Italy), F. Pfenning (USA), M. Wirsing (Germany)

Organizing Committee

- Research Insitute for Symbolic Computation, Johannes Kepler University, Austria
- Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Roma, Italy

List of Referees

L. C. Aiello
M. P. Bonacina
A. Bossi
A. Bouhoula
M. Bronstein
R. Bündgen
H. J. Bürckert
D. Cantone
O. Caprotti
M. Casini Schaerf
P. Ciancarini
B. Ciciani
G. Cioni
F. D'Amore
R. De Nicola
J. Derzinger
J. Despeyroux
P. Di Blasio
F. Donini
D. Dranidis
S. Gastinger
W. Gehrke
E. Giovannetti
M. Grabowsky
D. Gruntz
R. Hennicker
C. Hintermeyer
T. Jebelean
B. Kacewicz
F. Kluźniak
F. Kroeger
M. Lenzerini
C. Limongelli
V. Manca
L. Mandel
A. Marchetti Spaccamela
V. Marian
M. Mehlich
T. Mora
A. Muech
A. Neubacher
F. Nickl
C. Palamidessi
F. Parisi Presicce
A. Pettorossi
A. Pierantonio
F. Pirri
A. Policriti
M. Proietti
B. Reus
G. Rossi
M. Rusinovitch
P. Santas
M. Schaerf
W. Schreiner
K. Siegl
A. Skowron
R. Stabl
T. Streicher
K. Sutner
M. Temperini
M. Turski
L. Unycryn
S. Valentini
I. Walukiewicz
W. Windsteiger
J. Winkowski
P. Zimmermann

Contents

Theoretical Aspects

Algorithm Implementation

Programming with Types

Automated Reasoning

Software Systems

System Description

Mathematica: A System for Doing Mathematics by Computer?

Bruno Buchberger

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University
A-4040 Linz, Austria
Tel. ++43 (7236) 3231-41
FAX ++43 (7236) 3231-30
buchberg@risc.uni-linz.ac.at

Abstract

The title of my talk coincides with the title of Stephen Wolfram's book on his Mathematica system except that in the title of my talk there is a question mark. The content of my talk is my personal answer to this question.

We start from analyzing what it means to do mathematics and introduce what we call the "creativity spirale in mathematics": "Doing mathematics", in my view, is iterating the cycle "observation – proof – programming – application" .

Our main point is that Mathematica supports well three passes of this spirale, namely "observation – programming – application" and it helps a little in some simple forms of proof. However, Mathematica does not yet seem to be the right setting for proving in the broad sense of the word as understood by the mathematics community. We give some arguments for this and develop some ideas how a future system for doing mathematics might look like.

Proving the Correctness of Algebraic Implementations by the ISAR System

Bernhard Bauer *, Rolf Hennicker**

* Institut für Informatik, Technische Universität München,
Arcisstr. 16, D-8000 München 2,
E-mail: bauer@informatik.tu-muenchen.de
** Institut für Informatik, Ludwig-Maximilians-Universität München,
Leopoldstr. 11b, D-8000 München 40,
E-mail: hennicke@informatik.uni-muenchen.de

Abstract. We present an interactive system, called ISAR, which provides an environment for correctness proofs of algebraic implementation steps. The correctness notion of implementation is based on behavioural semantics and the underlying proof procedure of the system is based on the principle of context induction (which is a particular instance of structural induction). The input of the ISAR system is an implementation step consisting of an abstract specification to be implemented, a concrete specification used as a basis for the implementation and an implementation construction. If all steps of the (interactive) proof procedure are performed the system has proved the correctness of the implementation step.

1 Introduction

Much work has been done in the field of algebraic specifications to provide formal concepts for the development of correct programs from given specifications. However, in order to be useful in practice, a formal theory of correct program development is not sufficient: Formal implementation notions should be supplied by appropriate proof methods and, even more important, by tools providing mechanical support for proving the correctness of implementation steps.

In this paper an interactive system for algebraic implementation proofs, called ISAR, is presented which sets out from the observational view of software development: The basic assumption is that a software product is a correct implementation if it satisfies the desired input/output behaviour, independently from the internal properties of a program which may not satisfy a given specification. This covers well known practical examples like the implementation of sets by lists (since lists do not satisfy the characteristic set equations but lists have the same behaviour as sets if only membership tests $x \in S$ are observable) or the familiar implementation of stacks by arrays with pointer (since arrays with pointer do not satisfy the usual stack equation $pop(push(x, s)) = s$ but they have the same behaviour as stacks if only the top elements of stacks are observed).

A formalization of this intuitive idea is presented in [Hennicker 90, 92] where an implementation relation for specifications is defined based on behavioural semantics

in the sense of [Nivela, Orejas 88], [Reichel 85]. In particular, in [Hennicker 90, 92] a proof theoretic characterization and a proof method, called *context induction*, is presented for proving behavioural implementation relations. The characterization of implementations says that a specification *SP1* is a *behavioural implementation* of a specification *SP* if and only if for all *observable contexts c* (over the signature of *SP*) and for all axioms t = r of *SP* the "observable" equations c[t] = c[r] are deducible from the axioms of the implementation *SP1* (for all ground instantiations over the signature of *SP*).

It is the basic idea of the ISAR system to prove this condition by context induction, i.e. by structural induction on the set of observable contexts, in order to show that SP1 is an implementation of SP. The underlying algorithm of the ISAR system providing a procedure for context induction proofs was developed in [Hennicker 92].

Usually implementations of an abstract specification are constructed on top of existing (concrete) specifications of standard data structures like lists, arrays, trees etc. In order to document the construction of the implementation, the input of the ISAR system is an *implementation step* which consists of three parts: an abstract specification SP-A to be implemented, a concrete specification SP-C used as a basis for the implementation and a construction of the implementation. Such constructions are represented by appropriate enrichments and/or renamings performed on top of SP-C. An implementation step is called *correct* if the application of the implementation construction to SP-C yields a behavioural implementation of SP-A.

In order to prove the correctness of an implementation step the ISAR system first normalizes all specification expressions. Then the *context induction prover*, the kernel of the system, is called for proving the implementation relation for the normalized specifications. Thereby all contexts and all proof obligations to be considered for the implementation proof are automatically generated. For the proof of equations the system is connected to the TIP system which is a narrowing-based inductive theorem prover (cf. [Fraus, Hußmann 91]). All steps of an implementation proof can be guided by appropriate interaction with the user. In particular, as usual when dealing with induction proofs, it is often necessary to find an appropriate generalization of the actual induction assertion if a nesting of context induction (implemented by a recursive call of the context induction prover) is performed. For that purpose the ISAR system generates automatically a set of particular contexts each context representing a generalization of the actual induction assertion. Then the user may select an appropriate context representing an assertion which is general enough for achieving successful termination of the proof algorithm.

As we will show by an example for the construction of generalized induction assertions it may be necessary to define additional function symbols which generalize (some) functions of the abstract specification. (For instance for the proof of the array pointer implementation of stacks a generalization of the *pop* operation by an operation *iterated_pop: nat, stack* $\rightarrow$ *stack* is used where *iterated_pop(n, s)* performs *n* pop

operations on a stack *s*.) Such function generalizations can be added as "hints" to an implementation step. Hints cannot be generated automatically. In this case the intuition of the system user is needed.

2 Basic Concepts

In this section we summarize the theoretical foundations of the ISAR system. In particular the notions of behavioural specifications and behavioural implementations are defined. Most definitions and results of this section can be found in [Hennicker 90] or (slightly revised) in [Hennicker 92].

2.1 Algebraic Preliminaries

First, we briefly review the basic notions of algebraic specifications which will be used in the following (for more details see e.g. [Ehrig, Mahr 85]). A (many sorted) *signature* Σ is a pair (S, F) where S is a set of *sorts* and F is a set of *function symbols* (also called *functions* for short). To every function symbol $f \in F$ a functionality $s_1,\ldots, s_n \rightarrow s$ with $s_1,\ldots, s_n \in S$ is associated. If $n = 0$ then f is called *constant* of sort s. A *signature morphism* $\rho: \Sigma \rightarrow \Sigma'$ between two signatures $\Sigma = (S, F)$ and $\Sigma' = (S', F')$ is a pair $(\rho_{sorts}, \rho_{functs})$ of mappings $\rho_{sorts}: S \rightarrow S'$, $\rho_{functs}: F \rightarrow F'$ such that for all $f \in F$ with functionality $s_1, \ldots, s_n \rightarrow s$, $\rho_{functs}(f)$ has functionality $\rho_{sorts}(s_1), \ldots, \rho_{sorts}(s_n) \rightarrow \rho_{sorts}(s)$. A signature $\Sigma' = (S', F')$ is called *subsignature* of Σ (written $\Sigma' \subseteq \Sigma$) if $S' \subseteq S$ and $F' \subseteq F$.
The *term algebra* $W_\Sigma(X)$ of all Σ-*terms* with variables of X (where $X = (X_s)_{s \in S}$ is an S-sorted family of sets of variables) is defined as usual. In particular, for any sort $s \in S$, $W_\Sigma(X)_s$ denotes the set of *terms* of sort s. If $X = \emptyset$ then $W_\Sigma(\emptyset)$ is abbreviated by W_Σ and W_Σ is called *ground term algebra*. We assume that any signature $\Sigma = (S, F)$ is *inhabited*, i.e. for each sort $s \in S$ there exists a ground term $t \in (W_\Sigma)_S$. A *substitution* $\sigma: X \rightarrow W_\Sigma(X)$ is a family of mappings $(\sigma_s: X_s \rightarrow W_\Sigma(X)_s)_{s \in S}$. For any term $t \in W_\Sigma(X)$, the *instantiation* $\sigma(t) =_{def} t[\sigma(x_1)/x_1, \ldots, \sigma(x_n)/x_n]$ is defined by replacing all variables $x_1, \ldots, x_n$ occurring in t by the terms $\sigma(x_1), \ldots, \sigma(x_n)$. A substitution $\sigma: X \rightarrow W_\Sigma$ is called *ground substitution*.

2.2 Behavioural Specifications

The syntax of behavioural specifications is defined similarly to [Nivela, Orejas 88] and [Reichel 85] where a distinguished set of sorts of a specification is declared as observable:
A *behavioural specification* $SP = (\Sigma, Obs, E)$ consists of a signature $\Sigma = (S, F)$, a subset $Obs \subseteq S$ of *observable sorts* and a set E of *axioms*. Any behavioural specification is assumed to contain the observable sort *bool*, two constants *true, false:* $\rightarrow$ *bool* (denoting the truth values) and the axiom *true* $\neq$ *false*. The axioms of E\ {true $\neq$ false} are equations $t = r$ with terms $t, r \in W_\Sigma(X)$.