Lecture Notes in Artificial Intelligence 1712

Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Harold Boley

A Tight, Practical Integration of Relations and Functions

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Author

Harold Boley
Stanford Medical Informatics
251 Campus Drive, Stanford, CA 94305-5479, USA
E-mail: boley@SMI.Stanford.EDU

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Boley, Harold:
A tight, practical integration of relations and functions / Harold Boley. -
Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris
; Singapore ; Toyko : Springer, 1999
(Lecture notes in computer science ; 1712 : Lecture notes in artificial
intelligence)
ISBN 3-540-66644-3

CR Subject Classification (1998): I.2, F.4, F.3, D.3

ISBN 3-540-66644-3 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN 10705076 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

Preface

As in other fields, in computer science certain objects of study can be synthesized from different basic elements, in different ways, and with different resulting stabilities. In subfields such as artificial intelligence, computational logic, and programming languages various relational and functional ingredients and techniques have been tried for the synthesis of declarative programs. This text considers the notions of relations, as found in logic programming or in relational databases, and of functions, as found in functional programming or in equational languages. We study a declarative integration which is tight, because it takes place right at the level of these notions, and which is still practical because it preserves the advantages of the widely used relational and functional languages PROLOG and LISP. The resulting relational-functional language, RELFUN, will be used here for exemplifying all integration principles.

Part of the unique attraction of computer science stems from the fact that most of its notions permit a multitude of simultaneous perspectives, connecting form and content, theory and practice, etc. Thus, the study of programming involves syntax, semantics, and pragmatics: Syntactically, a program can be specified by grammar formalisms on several levels. Semantically, a program can be characterized as a static entity such as a mathematical model and by dynamic computations such as derivation traces. Pragmatically, a program can be run by an interpreter, in an abstract machine, and as native code of a real computer. In order to obtain insights into new programming paradigms one can start off from either of these ends or from somewhere in the middle. The texts treats most of these perspectives for the RELFUN integration, whose development started at the practical ends, later tailored theoretical ones for them, and now is proceeding in both directions:* In chapter 2.6. the PROLOG-like user syntax is specified by an EBNF grammar. In chapter 3 the semantics is founded equivalently on Herbrand models and on SLD-resolution. In chapter 5 the pragmatics is implemented via the Warren Abstract Machine. For further summaries we refer to the reader's guide at the end of the overview chapter 1 and to the synopses at the beginning of all five chapters.

*This research was supported by the Univ. Kaiserslautern, the SFB 314, and the BMBF under the DFKI Grants ITW 8902 C4 and 413-5839-ITW9304/3.

Here we summarize our contributions to the theory and practice of tight relational-functional integration:

- The relational notions of non-ground terms and (don't-know) non-determinism were integrated with the functional notions of application values and higher-order functions into a minimal kernel system (chapter 1).
- This was extended by 'first-class' finite domains and exclusions (chapter 4), sort hierarchies, 'single-cut' determinism specification (chapter 2), etc.
- Encapsulated partial (non-ground) data structures were transferred to the functional paradigm even for computations with ground I/O (chapter 2).
- The semantics of first-order functions was founded on *the same* model-theoretic level as that of relations (chapter 3).
- Relational-functional transformers, compilers, and abstract machines were developed in LISP (chapter 5).
- Using these, application studies were conducted about declarative programming in engineering domains (chapter 1).
- The reusability of concepts, techniques, and source programs was shown with the languages COLAB (BMBF project ARC-TEC) and DRL (BMBF project VEGA).

My thanks go first to Michael M. Richter for having set standards in combining theory and practice, for discussions concerning Horn-logic and higher-order functional programming, for many suggestions with respect to succinct formulation, for his encouragement during this entire work, as well as for establishing the DFKI department of Intelligent Engineering Systems. And I want to thank all 'IIS' colleagues at the DFKI for their patience in times when I was preoccupied with this work. In particular, in the ARC-TEC knowledge-compilation group, Philipp Hanschke, Knut Hinkelmann, and Manfred Meyer provided input that helped me improve things. More recently, in VEGA, Andreas Abecker, Otto Kühn, and Holger Wache have also given me important feedback on the use and presentation of RELFUN. In particular, in the DRL and RFM groups we have been discussing issues of declarative representation languages and relational-functional machines. Thanks to Panagiotis Tsarchopoulos for joining us for a while. On this occasion I want to acknowledge the software contributions to the defining RELFUN interpreter by the (former) students Simone Andel, Michael Christen, Klaus Elsbernd, Andreas Gilbert, Victoria Hall, Hans-Günther Hein, Michael Herfert, Thomas Labisch, Markus Perling, Ralph Scheubrein, Michael Sintek, Werner Stein, and Stefan Steinacker. Additional special thanks go to Markus Perling and Michael Sintek for their key contributions to the compiler/emulator system and to most of the other RELFUN components as well as for their careful proof-reading of several drafts of this text. Many other people at the Fachbereich Informatik and the DFKI of the University of Kaiserslautern have helped me in important ways, and I want to thank you all.

As this text is a revision of my Habilitationsschrift at the Fachbereich Informatik of the Universität Kaiserslautern, I also want to acknowledge my referees Wolfram-M. Lippe (Institut für Informatik, Universität Münster), Otto Mayer, and Michael M. Richter, as well as the head of my Habilitation committee, Stefan Heinrich. Furthermore, I am thankful to Burckhard Strehl and his colleagues and students from the Fachbereich Mathematik of the Universität Kaiserslautern for their help with our RELFUN-based ontology for the Mathematics International study program. I am also grateful to Andreas Dengel and his collaborators from DFKI's Information Management and Document Analysis lab for providing an environment that permitted the development of Web-oriented RELFUN enhancements such as ONTOFILE. Moreover, my thanks extend to Mark Musen and his collaborators at Stanford Medical Informatics for giving me the opportunity to explore cross-fertilizations of RELFUN and PROTEGE axiom languages. Finally, I appreciate the advice of Ralf Kleinfeld from infoTex Kaiserslautern about setting up the RELFUN Web domain http://www.relfun.org/.

Kaiserslautern and Stanford, July 1999 Harold Boley

Contents