Two-Stage Programming: Compilation and Case Studies

Inauguraldissertation

zur
Erlangung der Würde eines Doktors der Philosophie
vorgelegt der
Philosophisch-Naturwissenschaftlichen Fakultät
der Universität Basel

von

Romeo A. Dumitrescu

aus Drobeta Turnu-Severin, Rumänien

Basel, 2000

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät auf Antrag von

Prof. Dr. Helmar Burkhart, Universität Basel, Fakultätsverantwortlicher

Dr. Edgar F. A. Lederer, Universität Basel, Dissertationsleiter

Prof. Dr. Robert Stärk, ETH Zürich, Korreferent

Basel, den 04.07.2000

Prof. Dr. A. Zuberbühler, Dekan

Research Reports in Computer Science

Band 6

Romeo A. Dumitrescu

Two-Stage Programming: Compilation and Case Studies

Shaker Verlag
Aachen 2000

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Dumitrescu, Romeo A.:
Two-Stage Programming: Compilation and Case Studies /
Romeo A. Dumitrescu. Aachen : Shaker, 2000
(Research Reports in Computer Science ; Bd. 6)
Zugl.: Basel, Univ., Diss., 2000
ISBN 3-8265-7793-0

Printed in Germany.

ISBN 3-8265-7793-0
ISSN 1436-6967

Shaker Verlag GmbH • P.O. BOX 1290 • D-52013 Aachen
Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9
Internet: www.shaker.de • eMail: info@shaker.de

To my wife, my parents, and my grandmother "Maia"

Acknowledgements

I would like to mention here some people who helped me complete this thesis. I apologize to those I have unintentionally omitted.

I thank Prof. Helmar Burkhart for accepting me as a Ph.D. student, giving me the opportunity to work on a topic of my choice, and helping me in critical moments. I also thank Prof. Robert Stärk for accepting to review my thesis.

Special thanks go to Dr. Edgar Lederer, my direct project leader, for being an excellent supervisor who carefully watched my work and invested a lot of time and patience in fruitful discussions which helped me find and chose the best solutions. He was also a very good teammate and teacher from whom I have learned a lot.

I could not have accomplished anything without the love, support, and encouragement of my wife Lavinia. She lovingly accepted my long working days and weekends and gave me real power in the much too little time we spent together. All the time, I have also felt the love of my parents and my grandmother although I was quite far from them.

I am greatly indebted to Lucas Godelmann and Penelope Vounatsou for their sincere friendship, wholeheartedly help, and strong encouragement.

I would like to express my gratitude to Robert Sinclair for his useful hints and advice. He also helped me to set up my new home and get used with a new environment.

Etienne Studer kindly took over a significant part of my assistant duties during the last months when I was writing my thesis.

My colleagues Peter Christen, Niandong Fang, Robert Frank, Guido Häechler, Beat Hörman, Walter Kuhn, Gérald Prétôt, Radu Prodan, Kay Ruchty-Crowley, Christoff Sinn, Gonzalo Travieso, Guoqiang Wei, and Birgit Westermann have created a pleasant work environment. The secretaries Regula Kneubühler-Fricker and Karin Liesenfeld helped me in administrative matters.

Finally, I take this opportunity to thank my high-school teacher Ştefan Bojincă to whom I owe my carrier path. He showed me a computer for the first time and guided my first steps towards studying computer science and research work. I also thank all my professors and assistants from the "Politehnica" University of Timişoara who contributed to my education in computer science, especially Prof. Ioan Jurca who, in addition, offered me the possibility to be his assistant and encouraged me together with Prof. Ştefan Holban and Prof. Vladimir Creţu to pursue Ph.D. studies.

Abstract

Two-Stage Programming (2SP) is a novel, mixed-paradigm approach (functional/imperative) to developing reliable programs based on complete run-time checking of computations with respect to a given specification. A 2SP program consists of a functional specification and an imperative coordination tightly connected to the specification. The coordination maps the specification to an imperative and possibly parallel/distributed program. During run-time, the consistency between the coordination and specification is checked based on their connection. Normal termination of a 2SP program execution implies the correctness of the computed results with respect to the specification, for that execution.

I address in this thesis the impact of 2SP's run-time consistency checks on both program reliability and efficiency.

I describe the basic ideas of the first efficient implementation of 2SP I have developed, and present, based on this implementation, an analysis of two significant case studies (one sequential, the other parallel), which shows that 2SP offers automatic run-time result checking and enhanced debugging support through *early* detection and *precise* location of errors at run-time, for an increase of about one order of magnitude of the program execution time.

I consider that this initial study shows that run-time consistency checks are reasonably efficient, and programming languages can benefit by including a 2SP-like consistency checker mechanism. This is especially useful for ML, well-known as a very robust language, since I show that a 2SP program can be both more reliable and faster than its corresponding ML program.

Keywords Run-time checking, result checking, functional/imperative programming, debugging, coordination language.

Parts of Chapters 2, 3, and 4 are accepted for publication in the *International Journal of Foundations of Computer Science*.

Part of Chapter 3 has been published in the *Proceedings of the* 14$^{\text{th}}$ *ACM Symposium on Applied Computing, 1999*.

This research was supported by the Swiss National Science Foundation by the grants 2100–046932.96/1 and 2000–053690.98/1, and the University of Basel.

Contents

List of Figures