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

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

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.