MOBILE COMPUTATION WITH FUNCTIONS

Advances in Information Security

Sushil Jajodia
Consulting editor

Center for Secure Information Systems
George Mason University, Fairfax, VA 22030-4444
email: jajodia@gmu.edu

The goals of Kluwer International Series on ADVANCES IN INFORMATION SECURITY are, one, to establish the state of the art of, and set the course for future research in information security and, two, to serve as a central reference source for advanced and timely topics in information security research and development. The scope of this series includes all aspects of computer and network security and related areas such as fault tolerance and software assurance.

ADVANCES IN INFORMATION SECURITY aims to publish thorough and cohesive overviews of specific topics in information security, as well as works that are larger in scope or that contain more detailed background information than can be accommodated in shorter survey articles. The series also serves as a forum for topics that may not have reached a level of maturity to warrant a comprehensive textbook treatment.

Researchers as well as developers are encouraged to contact Professor Sushil Jajodia with ideas for books under this series.

Additional titles in the series:

TRUSTED RECOVERY AND DEFENSIVE INFORMATION WARFARE by Peng Liu and Sushil Jajodia, ISBN: 0-7923-7572-6
RECENT ADVANCES IN RSA CRYPTOGRAPHY by Stefan Katzenbeisser, ISBN: 0-7923-7438-X
E-COMMERCE SECURITY AND PRIVACY by Anup K. Ghosh, ISBN: 0-7923-7399-5
INFORMATION HIDING: Steganography and Watermarking-Attacks and Countermeasures by Neil F. Johnson, Zoran Duric, and Sushil Jajodia ISBN: 0-7923-7204-2

Additional information about this series can be obtained from
www.wkap.nl/series.htm/ADIS

MOBILE COMPUTATION WITH FUNCTIONS

by

ZELİHA DİLSUN KIRLI
MIT Laboratory for Computer Science, U.S.A.

SPRINGER SCIENCE+BUSINESS MEDIA, LLC

ISBN 978-1-4613-5348-5 ISBN 978-1-4615-1007-9 (eBook)
DOI 10.1007/978-1-4615-1007-9

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.

Originally published by Kluwer Academic Publishers in 2002
Softcover reprint of the hardcover 1st edition 2002

Printed on acid-free paper.

Contents

Preface

The practice of computing has reached a stage where computers are seen as parts of a global computing platform. The possibility of exploiting resources on a global scale has given rise to a new paradigm – the mobile computation paradigm – for computation in large-scale distributed networks. Languages which enable the mobility of code over the network are becoming widely used for building distributed applications.

This work explores distributed computation with languages which adopt functions as the main programming abstraction and support code mobility through the mobility of functions between remote sites. It aims to highlight the benefits of using languages of this family in dealing with the challenges of mobile computation. The possibility of exploiting existing static analysis techniques suggests that having functions at the core of a mobile code language is a particularly apt choice.

A range of problems which have impact on the safety, security and performance of systems are discussed here. It is shown that types extended with effects and other annotations can capture a significant amount of information about the dynamic behaviour of mobile functions and offer solutions to the problems under investigation.

The book presents a survey of the languages Concurrent ML, Facile and PLAN which remain loyal to the principles of the functional language ML and hence inherit its strengths in the context of concurrent and distributed computation. The languages which are defined in the subsequent chapters have their roots in these languages.

Two chapters focus on using types to statically predict whether functions are used locally or may become mobile at runtime. Types are exploited for distributed call-tracking to estimate which functions are invoked at which sites in the system. Compilers for mobile code languages would benefit from such estimates in dealing with the heterogeneity of the network nodes, in providing static profiling tools and in estimating the resource-consumption of programs.

Two chapters are devoted to the use of types in controlling the flow of values in a system where users have different trust levels. The confinement of values within a specified mobility region is the subject of one of these. The other focuses on systems where values are classified with respect to their confidentiality level. The sources of undesirable flows of information are identified and a solution based on noninterference is proposed.

ZELİHA DİLSUN KIRLI

To my parents
İnciser and Orhan Kırlı

Acknowledgments

This book is based on the work I did towards my PhD degree at the University of Edinburgh, within the Laboratory for Foundations of Computer Science. I owe the most special thanks to Stephen Gilmore who supervised me throughout my postgraduate studies. He not only introduced me to interesting research topics and guided me in an inspiring way but also encouraged me to publish my work in this form. I am indebted to him for reading every piece I wrote with scrutiny. His insightful comments led to significant improvements both in the content and style of my work. It is a rare chance to work with someone whose advice is always so helpful and so thoughtfully communicated; I am very fortunate.

Jane Hillston made herself available whenever I needed additional feedback. Her ideas and criticisms were very useful in organizing the material presented in this book. I also would like to thank both Jane Hillston and Stephen Gilmore for their invaluable moral support. Thanks to their excellent academic guidance and friendliness, I felt that I was in safe hands since my arrival in Edinburgh.

I would like to thank all members of the Laboratory for Foundations of Computer Science for creating a research environment of highest quality. I would like to acknowledge Martin Hofmann and David Aspinall in particular for making it possible for me to collaborate with them on an interesting project. It was always a pleasure to talk to Don Sannella, Ian Stark and Chris Walton. Their interest in my work was a great source of motivation. I appreciate the time Bonnie Webber spared for me as my mentor. I also would like to thank Matías Menni and Sibylle Fröschle for their nice company.

Gordon Plotkin and Peter Sewell reviewed a full draft of my PhD thesis which forms the core of this book. I am grateful to them for the thoroughness of their review and the many helpful comments they offered.

It was Lance Wobus who drove me into the project of publishing this book upon the suggestion of Stephen Gilmore. He was enthusiastic and supportive. Susan Lagerstrom-Fife and Sharon Palleschi followed the progress of the

manuscript through to the final production. I thank them all for their patience and help.

At various stages in my PhD research, The British Council, Türk Eğitim Vakfı, the University of Edinburgh and EPSRC provided financial support for which I feel indebted. The very final corrections to the manuscript was completed during my position as a postdoctoral research associate at the Laboratory for Computer Science, MIT. I am grateful to Nancy Lynch for making this possible.

I would like to thank all of my friends for their warm support. They did not allow me to feel lonely although I was a long away from home. Murat Kaynar was wonderful; he made sure that I could work with peace of mind and that I remained optimistic and cheerful. It is a privilege to have big-hearted and understanding parents like mine. I have always drawn a lot of strength from their love and support. They deserve a very, very big thank you.

Introduction

1. Mobile computation with functions

The recent developments in telecommunications technology have made it possible to envisage a global computing platform in which computers interact easily and share a wide range of resources. Computers are no longer viewed as largely self-contained computing devices which use local resources and occasionally communicate with each other. The traditional assumptions about computation in distributed systems and desirable features for programming languages are being revised to allow for better use of the global infrastructure. A consequence of this has been the emergence of *the mobile computation paradigm* along with its supporting technologies. The key characteristic of this paradigm is to give programmers control over the mobility of code or active computations across the network by providing appropriate language features. Therefore, a typical mobile computation language is expected to facilitate the expression and execution of mobile code-containing entities. The dynamism and flexibility offered by this form of computation, however, brings about a set of problems, the most challenging of which are relevant to safety and security.

Opinions are diverse as to the primary concerns of languages for mobile computation. We argue that a sound formal foundation is of the greatest significance. By a formal foundation we mean a collective body of work which describes the computational model of the language at a suitable level of abstraction and enables rigorous or even formal reasoning about programs. Such a foundation would preclude ambiguities about the meaning of programs while also enabling the formulation and proof of certain properties including safety and security related ones.

Functional languages are known for their well-understood computational models and their amenability to formal reasoning. They also have strong expressive power due to higher-order features. Functions can flow from one program point to another as first-class values. These facts suggest that the kind

of mobile computation language we put forward can be obtained by adopting a functional core and extending it with features which are in keeping with the principles of functional computation. In such a language functions can represent mobile code-containing entities and formal systems for reasoning about functional programs can be further exploited to reason about the behaviour of mobile code.

In general, this book contributes simple but inspiring ideas to the research in formal models of mobile computation and program analysis . In particular, novel applications of type and effect based analysis and suggestions for future directions are presented.

2. Type and effect based static analysis

Conventionally type systems for functional languages have been used to ensure that programs cannot corrupt the runtime representation of data values so that further execution of the program is not faithful to the language semantics. This property is known as *type safety* in the literature. Effect systems were initially proposed as a solution to the problems encountered in preserving type safety and polymorphism while integrating functional and imperative features. The basic idea was to enhance the type systems so that the expressions were associated with their observable side-effects as well as types and to use this information in making judgements with respect to safety. Some authors have further explored the use of type and effect systems for memory management and safe integration of concurrent and functional features.

The exploitation of type and effect systems need not be confined to the enforcement of type safety. Annotated with effects and other kinds of information, types can capture a significant amount of static information about a program's potential dynamic behaviour. The general methodology of type and effect systems then consists of devising a semantics for the language, expressing a program analysis by means of types and effects and showing the semantic correctness of this analysis. In other words, the type system extracts the overall behaviour of the program as a first step and as a later step one can devise various analyses to reason about it in a sound way. These analyses may be put to use in various areas such as compiler optimizations, cost profiling and safety and security. The literature includes examples of such analyses devised prior to the emergence of the mobile computation paradigm. This work introduces new analyses motivated by the characteristics of mobile computation.

A slightly different approach to exploiting type and effect systems can be to determine the properties which are desirable for all programs and design the type and effect system so that those programs which violate these properties are rejected by the system. This is closer in spirit to the earlier exploitations of type and effect systems for enforcing type safety. In the context of mobile computation, enforcing type safety alone is not sufficient to address many of the

safety and security concerns. Just as the languages are revisited to examine their position with respect to the new paradigm of mobile computation, type and effect systems need to be revisited to adapt their methodology to the requirements of the context of mobile computation. The work presented in this book can be considered as a step in this direction.

Enforcing safety and security properties by type systems is an active research area where the significance of secure flow of information is emphasized. Most of the existing work is in the framework of computational models different to the one considered here. In this respect, we contribute to the area of type-based approaches to security by presenting type and effect systems which incorporate a machinery for tracing the flow of values in a distributed setting where functions are the essential elements of computation.

3. Overview of the book

Introduction describes the characteristics of mobile computation and functional computation. It argues that integrating these two paradigms can offer solutions to the problems which have proved to be challenging in the context of mobile computation. The useful role which can be played by type and effect systems is discussed.

Chapter 1 gives an overview of the process calculi which provide formal models of distributed and mobile computation. This is followed by a closer look at the programming languages Concurrent ML, Facile and PLAN (Programming Language for Active Networks). These languages point to a consistent effort to benefit from the fundamental ideas behind ML in designing and implementing languages for concurrent and distributed computation.

Chapter 2 focuses on a language similar to Facile where values of all types, including functions and communication channels, can be transmitted between remote sites. The problem investigated in this chapter is the static estimation of functions and channels which may become mobile at run-time. A static analysis such as the one considered in this chapter would be a useful asset for compilers in dealing with the heterogeneity of the network nodes, detecting the locality of certain values and providing static profiling tools.

Chapter 3 focuses on the language PLAN. The form of support for code mobility in PLAN is different from that of Facile. It is based on a remote evaluation facility for functions. The design of PLAN has been influenced by the need to meet the strong safety and security requirements of active networks; especially by the need to protect against denial of service. The subject of this chapter is distributed call-tracking by means of a type and effect system in the framework of a PLAN-like language. It is argued that for an applicative language distributed call-tracking can provide the basis for static estimation of resource consumption.

Chapter 4 shifts the focus back to a language which resembles Facile. Some distributed systems are characterized by their heterogeneity in terms of the nature of the computing devices, security requirements of the information flowing in the system and the trust level of the users. Programmers who provide code for such systems would find it useful to have a language mechanism which enables them to confine the flow of certain values to a particular part of the system – a mobility region. This chapter discusses how a static type system can be used to enforce confinement in a specified mobility region.

Chapter 5 revisits the language of Chapter 2 and introduces a variant of it where the values of the language are classified with respect to their confidentiality level. As in Chapter 4, it is assumed that users which interact with the system may not be equally trustworthy. The sources of undesirable information flows are identified and a secure information flow property based on noninterference is introduced. Programs which are accepted by the proposed type and effect system for the language are shown to enjoy this property.

Chapter 6 includes a summary of the book which clarifies contributions made to the research areas of functional and mobile code languages, annotated type and effect systems and the language-based approach to security.