LNCS 2937

Bernhard Steffen
Giorgio Levi (Eds.)

Verification, Model Checking, and Abstract Interpretation

5th International Conference, VMCAI 2004
Venice, Italy, January 2004
Proceedings

Springer

Bernhard Steffen Giorgio Levi (Eds.)

Verification, Model Checking, and Abstract Interpretation

5th International Conference, VMCAI 2004
Venice, Italy, January 11-13, 2004
Proceedings

Springer

Volume Editors

Bernhard Steffen
Universität Dortmund, LS V
Baroper Str. 301, 44221 Dortmund, Germany
E-mail: steffen@cs.uni-dortmund.de

Giorgio Levi
Università di Pisa, Dipartimento di Informatica
Via Buonarroti, 2, 56100 Pisa, Italy
E-mail: levi@di.unipi.it

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): F.3.1-2, D.3.1, D.2.4

ISSN 0302-9743
ISBN 3-540-20803-8 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag is a part of Springer Science+Business Media

springeronline.com

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 10975695 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 2937
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Lecture Notes in Computer Science

For information about Vols. 1–2837
please contact your bookseller or Springer-Verlag

Vol. 2838: N. Lavrač, D. Gamberger, L. Todorovski, H. Blockeel (Eds.), Knowledge Discovery in Databases: PKDD 2003. Proceedings, 2003. XVI, 508 pages. 2003. (Subseries LNAI).

Vol. 2839: A. Marshall, N. Agoulmine (Eds.), Management of Multimedia Networks and Services. Proceedings, 2003. XIV, 532 pages. 2003.

Vol. 2840: J. Dongarra, D. Laforenza, S. Orlando (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. Proceedings, 2003. XVIII, 693 pages. 2003.

Vol. 2841: C. Blundo, C. Laneve (Eds.), Theoretical Computer Science. Proceedings, 2003. XI, 397 pages. 2003.

Vol. 2842: R. Gavaldà, K.P. Jantke, E. Takimoto (Eds.), Algorithmic Learning Theory. Proceedings, 2003. XI, 313 pages. 2003. (Subseries LNAI).

Vol. 2843: G. Grieser, Y. Tanaka, A. Yamamoto (Eds.), Discovery Science. Proceedings, 2003. XII, 504 pages. 2003. (Subseries LNAI).

Vol. 2844: J.A. Jorge, N.J. Nunes, J.F. e Cunha (Eds.), Interactive Systems. Proceedings, 2003. XIII, 429 pages. 2003.

Vol. 2846: J. Zhou, M. Yung, Y. Han (Eds.), Applied Cryptography and Network Security. Proceedings, 2003. XI, 436 pages. 2003.

Vol. 2847: R. de Lemos, T.S. Weber, J.B. Camargo Jr. (Eds.), Dependable Computing. Proceedings, 2003. XIV, 371 pages. 2003.

Vol. 2848: F.E. Fich (Ed.), Distributed Computing. Proceedings, 2003. X, 367 pages. 2003.

Vol. 2849: N. García, J.M. Martínez, L. Salgado (Eds.), Visual Content Processing and Representation. Proceedings, 2003. XII, 352 pages. 2003.

Vol. 2850: M.Y. Vardi, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. Proceedings, 2003. XIII, 437 pages. 2003. (Subseries LNAI)

Vol. 2851: C. Boyd, W. Mao (Eds.), Information Security. Proceedings, 2003. XI, 443 pages. 2003.

Vol. 2852: F.S. de Boer, M.M. Bonsangue, S. Graf, W.-P. de Roever (Eds.), Formal Methods for Components and Objects. Proceedings, 2003. VIII, 509 pages. 2003.

Vol. 2853: M. Jeckle, L.-J. Zhang (Eds.), Web Services – ICWS-Europe 2003. Proceedings, 2003. VIII, 227 pages. 2003.

Vol. 2854: J. Hoffmann, Utilizing Problem Structure in Planning. XIII, 251 pages. 2003. (Subseries LNAI)

Vol. 2855: R. Alur, I. Lee (Eds.), Embedded Software. Proceedings, 2003. X, 373 pages. 2003.

Vol. 2856: M. Smirnov, E. Biersack, C. Blondia, O. Bonaventure, O. Casals, G. Karlsson, George Pavlou, B. Quoitin, J. Roberts, I. Stavrakakis, B. Stiller, P. Trimintzios, P. Van Mieghem (Eds.), Quality of Future Internet Services. IX, 293 pages. 2003.

Vol. 2857: M.A. Nascimento, E.S. de Moura, A.L. Oliveira (Eds.), String Processing and Information Retrieval. Proceedings, 2003. XI, 379 pages. 2003.

Vol. 2858: A. Veidenbaum, K. Joe, H. Amano, H. Aiso (Eds.), High Performance Computing. Proceedings, 2003. XV, 566 pages. 2003.

Vol. 2859: B. Apolloni, M. Marinaro, R. Tagliaferri (Eds.), Neural Nets. Proceedings, 2003. X, 376 pages. 2003.

Vol. 2860: D. Geist, E. Tronci (Eds.), Correct Hardware Design and Verification Methods. Proceedings, 2003. XII, 426 pages. 2003.

Vol. 2861: C. Bliek, C. Jermann, A. Neumaier (Eds.), Global Optimization and Constraint Satisfaction. Proceedings, 2002. XII, 239 pages. 2003.

Vol. 2862: D. Feitelson, L. Rudolph, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing. Proceedings, 2003. VII, 269 pages. 2003.

Vol. 2863: P. Stevens, J. Whittle, G. Booch (Eds.), «UML» 2003 – The Unified Modeling Language. Proceedings, 2003. XIV, 415 pages. 2003.

Vol. 2864: A.K. Dey, A. Schmidt, J.F. McCarthy (Eds.), UbiComp 2003: Ubiquitous Computing. Proceedings, 2003. XVII, 368 pages. 2003.

Vol. 2865: S. Pierre, M. Barbeau, E. Kranakis (Eds.), Ad-Hoc, Mobile, and Wireless Networks. Proceedings, 2003. X, 293 pages. 2003.

Vol. 2866: J. Akiyama, M. Kano (Eds.), Discrete and Computational Geometry. Proceedings, 2002. VIII, 285 pages. 2003.

Vol. 2867: M. Brunner, A. Keller (Eds.), Self-Managing Distributed Systems. Proceedings, 2003. XIII, 274 pages. 2003.

Vol. 2868: P. Perner, R. Brause, H.-G. Holzhütter (Eds.), Medical Data Analysis. Proceedings, 2003. VIII, 127 pages. 2003.

Vol. 2869: A. Yazici, C. Şener (Eds.), Computer and Information Sciences – ISCIS 2003. Proceedings, 2003. XIX, 1110 pages. 2003.

Vol. 2870: D. Fensel, K. Sycara, J. Mylopoulos (Eds.), The Semantic Web - ISWC 2003. Proceedings, 2003. XV, 931 pages. 2003.

Vol. 2871: N. Zhong, Z.W. Raś, S. Tsumoto, E. Suzuki (Eds.), Foundations of Intelligent Systems. Proceedings, 2003. XV, 697 pages. 2003. (Subseries LNAI)

Vol. 2873: J. Lawry, J. Shanahan, A. Ralescu (Eds.), Modelling with Words. XIII, 229 pages. 2003. (Subseries LNAI)

Vol. 2874: C. Priami (Ed.), Global Computing. Proceedings, 2003. XIX, 255 pages. 2003.

Vol. 2875: E. Aarts, R. Collier, E. van Loenen, B. de Ruyter (Eds.), Ambient Intelligence. Proceedings, 2003. XI, 432 pages. 2003.

Vol. 2876: M. Schroeder, G. Wagner (Eds.), Rules and Rule Markup Languages for the Semantic Web. Proceedings, 2003. VII, 173 pages. 2003.

Vol. 2877: T. Böhme, G. Heyer, H. Unger (Eds.), Innovative Internet Community Systems. Proceedings, 2003. VIII, 263 pages. 2003.

Vol. 2878: R.E. Ellis, T.M. Peters (Eds.), Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003. Part I. Proceedings, 2003. XXXIII, 819 pages. 2003.

Vol. 2879: R.E. Ellis, T.M. Peters (Eds.), Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003. Part II. Proceedings, 2003. XXXIV, 1003 pages. 2003.

Vol. 2880: H.L. Bodlaender (Ed.), Graph-Theoretic Concepts in Computer Science. Proceedings, 2003. XI, 386 pages. 2003.

Vol. 2881: E. Horlait, T. Magedanz, R.H. Glitho (Eds.), Mobile Agents for Telecommunication Applications. Proceedings, 2003. IX, 297 pages. 2003.

Vol. 2882: D. Veit, Matchmaking in Electronic Markets. XV, 180 pages. 2003. (Subseries LNAI)

Vol. 2883: J. Schaeffer, M. Müller, Y. Björnsson (Eds.), Computers and Games. Proceedings, 2002. XI, 431 pages. 2003.

Vol. 2884: E. Najm, U. Nestmann, P. Stevens (Eds.), Formal Methods for Open Object-Based Distributed Systems. Proceedings, 2003. X, 293 pages. 2003.

Vol. 2885: J.S. Dong, J. Woodcock (Eds.), Formal Methods and Software Engineering. Proceedings, 2003. XI, 683 pages. 2003.

Vol. 2886: I. Nyström, G. Sanniti di Baja, S. Svensson (Eds.), Discrete Geometry for Computer Imagery. Proceedings, 2003. XII, 556 pages. 2003.

Vol. 2887: T. Johansson (Ed.), Fast Software Encryption. Proceedings, 2003. IX, 397 pages. 2003.

Vol. 2888: R. Meersman, Zahir Tari, D.C. Schmidt et al. (Eds.), On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. Proceedings, 2003. XXI, 1546 pages. 2003.

Vol. 2889: Robert Meersman, Zahir Tari et al. (Eds.), On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops. Proceedings, 2003. XXI, 1096 pages. 2003.

Vol. 2890: M. Broy, A.V. Zamulin (Eds.), Perspectives of System Informatics. Proceedings, 2003. XV, 572 pages. 2003.

Vol. 2891: J. Lee, M. Barley (Eds.), Intelligent Agents and Multi-Agent Systems. Proceedings, 2003. X, 215 pages. 2003. (Subseries LNAI)

Vol. 2892: F. Dau, The Logic System of Concept Graphs with Negation. XI, 213 pages. 2003. (Subseries LNAI)

Vol. 2893: J.-B. Stefani, I. Demeure, D. Hagimont (Eds.), Distributed Applications and Interoperable Systems. Proceedings, 2003. XIII, 311 pages. 2003.

Vol. 2894: C.S. Laih (Ed.), Advances in Cryptology - ASIACRYPT 2003. Proceedings, 2003. XIII, 543 pages. 2003.

Vol. 2895: A. Ohori (Ed.), Programming Languages and Systems. Proceedings, 2003. XIII, 427 pages. 2003.

Vol. 2896: V.A. Saraswat (Ed.), Advances in Computing Science – ASIAN 2003. Proceedings, 2003. VIII, 305 pages. 2003.

Vol. 2897: O. Balet, G. Subsol, P. Torguet (Eds.), Virtual Storytelling. Proceedings, 2003. XI, 240 pages. 2003.

Vol. 2898: K.G. Paterson (Ed.), Cryptography and Coding. Proceedings, 2003. IX, 385 pages. 2003.

Vol. 2899: G. Ventre, R. Canonico (Eds.), Interactive Multimedia on Next Generation Networks. Proceedings, 2003. XIV, 420 pages. 2003.

Vol. 2901: F. Bry, N. Henze, J. Maluszyński (Eds.), Principles and Practice of Semantic Web Reasoning. Proceedings, 2003. X, 209 pages. 2003.

Vol. 2902: F. Moura Pires, S. Abreu (Eds.), Progress in Artificial Intelligence. Proceedings, 2003. XV, 504 pages. 2003. (Subseries LNAI).

Vol. 2903: T.D. Gedeon, L.C.C. Fung (Eds.), AI 2003: Advances in Artificial Intelligence. Proceedings, 2003. XVI, 1075 pages. 2003. (Subseries LNAI).

Vol. 2904: T. Johansson, S. Maitra (Eds.), Progress in Cryptology – INDOCRYPT 2003. Proceedings, 2003. XI, 431 pages. 2003.

Vol. 2905: A. Sanfeliu, J. Ruiz-Shulcloper (Eds.), Progress in Pattern Recognition, Speech and Image Analysis. Proceedings, 2003. XVII, 693 pages. 2003.

Vol. 2906: T. Ibaraki, N. Katoh, H. Ono (Eds.), Algorithms and Computation. Proceedings, 2003. XVII, 748 pages. 2003.

Vol. 2910: M.E. Orlowska, S. Weerawarana, M.P. Papazoglou, J. Yang (Eds.), Service-Oriented Computing – ICSOC 2003. Proceedings, 2003. XIV, 576 pages. 2003.

Vol. 2911: T.M.T. Sembok, H.B. Zaman, H. Chen, S.R. Urs, S.H.Myaeng (Eds.), Digital Libraries: Technology and Management of Indigenous Knowledge for Global Access. Proceedings, 2003. XX, 703 pages. 2003.

Vol. 2913: T.M. Pinkston, V.K. Prasanna (Eds.), High Performance Computing – HiPC 2003. Proceedings, 2003. XX, 512 pages. 2003.

Vol. 2914: P.K. Pandya, J. Radhakrishnan (Eds.), FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science. Proceedings, 2003. XIII, 446 pages. 2003.

Vol. 2916: C. Palamidessi (Ed.), Logic Programming. Proceedings, 2003. XII, 520 pages. 2003.

Vol. 2918: S.R. Das, S.K. Das (Eds.), Distributed Computing – IWDC 2003. Proceedings, 2003. XIV, 394 pages. 2003.

Vol. 2922: F. Dignum (Ed.), Advances in Agent Communication. Proceedings, 2003. X, 403 pages. 2004. (Subseries LNAI).

Vol. 2923: V. Lifschitz, I. Niemelä (Eds.), Logic Programming and Nonmonotonic Reasoning. Proceedings, 2004. IX, 365 pages. 2004. (Subseries LNAI).

Vol. 2927: D. Hales, B. Edmonds, E. Norling, J. Rouchier (Eds.), Multi-Agent-Based Simulation III. Proceedings, 2003. X, 209 pages. 2003. (Subseries LNAI).

Vol. 2929: H. de Swart, E. Orlowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments. Proceedings. VII, 273 pages. 2003.

Vol. 2932: P. Van Emde Boas, J. Pokorný, M. Bieliková, J. Štuller (Eds.), SOFSEM 2004: Theory and Practice of Computer Science. Proceedings, 2004. XIII, 385 pages. 2004.

Vol. 2937: B. Steffen, G. Levi (Eds.), Verification, Model Checking, and Abstract Interpretation. Proceedings, 2004. XI, 325 pages. 2004.

Preface

This volume contains the proceedings of the 5th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2004), held in Venice, January 11–13, 2004, in conjunction with POPL 2004, the 31st Annual Symposium on Principles of Programming Languages, January 14–16, 2004. The purpose of VMCAI is to provide a forum for researchers from three communities—verification, model checking, and abstract interpretation—which will facilitate interaction, cross-fertilization, and the advance of hybrid methods that combine the three areas. With the growing need for formal tools to reason about complex, infinite-state, and embedded systems, such hybrid methods are bound to be of great importance.

Topics covered by VMCAI include program verification, static analysis techniques, model checking, program certification, type systems, abstract domains, debugging techniques, compiler optimization, embedded systems, and formal analysis of security protocols.

This year's meeting follows the four previous events in Port Jefferson (1997), Pisa (1998), Venice (2002), LNCS 2294 and New York (2003), LNCS 2575. In particular, we thank VMCAI 2003's sponsor, the Courant Institute at New York University, for allowing us to apply a monetary surplus from the 2003 meeting to this one.

The program committee selected 22 papers out of 68 on the basis of three reviews. The principal criteria were relevance and quality. The program of VMCAI 2004 included, in addition to the research papers,

- a keynote speech by David Harel (Weizmann Institute, Israel) on *A Grand Challenge for Computing: Full Reactive Modeling of a Multicellular Animal*,
- an invited talk by Dawson Engler (Stanford University, USA) on *Static Analysis Versus Software Model Checking for Bug Finding*,
- an invited talk by Mooly Sagiv (Tel Aviv University, Israel) called *On the Expressive Power of Canonical Abstraction*, and
- a tutorial by Joshua D. Guttman (Mitre, USA) on *Security, Protocols, and Trust.*

We would like to thank the Program Committee members and the reviewers, without whose dedicated effort the conference would not have been possible. Our thanks go also to the Steering Committee members for helpful advice, to Agostino Cortesi, the Local Arrangements Chair, who also handled the conference's Web site, and to David Schmidt, whose expertise and support was invaluable for the budgeting. Special thanks are due to Martin Karusseit for installing, managing, and taking care of the METAFrame Online Conference Service, and to Claudia Herbers, who, together with Alfred Hofmann and his team at Springer-Verlag, collected the final versions and prepared the proceedings.

Special thanks are due to the institution that helped sponsor this event, the Department of Computer Science of Ca' Foscari University, and to the professional organizations that support the event: VMCAI 2004 is held in cooperation with ACM and is sponsored by EAPLS.

January 2004 Bernhard Steffen

Steering Committee

Agostino Cortesi (Italy)
E. Allen Emerson (USA)
Giorgio Levi (Italy)
Andreas Podelski (Germany)
Thomas W. Reps (USA)
David A. Schmidt (USA)
Lenore Zuck (USA)

Program Committee

Chairs: Giorgio Levi (University of Pisa)
Bernhard Steffen (Dortmund University)

Ralph Back (Åbo Akademi University, Finland)
Agostino Cortesi (Università Ca' Foscari di Venezia, Italy)
Radhia Cousot (CNRS and École Polytechnique, France)
Susanne Graf (VERIMAG Grenoble, France)
Radu Grosu (SUNY at Stony Brook, USA)
Orna Grumberg (Technion, Israel)
Gerhard Holzmann (Bell Laboratories, USA)
Yassine Lakhnech (Université Joseph Fourier, France)
Jim Larus (Microsoft Reseach, USA)
Markus Müller-Olm (FernUniversität in Hagen, Germany)
Hanne Riis Nielson (Technical University of Denmark, Denmark)
David A. Schmidt (Kansas State University, USA)
Lenore Zuck (New York University, USA)

Reviewers

Rajeev Alur
Roberto Bagnara
Ittai Balaban
Rudolf Berghammer
Chiara Bodei
Victor Bos
Dragan Bosnacki
Marius Bozga
Liana Bozga
Chiara Braghin
Roberto Bruni
Glenn Bruns
Sagar Chaki
Patrick Cousot
Silvia Crafa
Pierpaolo Degano
Benet Devereux
Agostino Dovier
Christian Ene
Javier Esparza
Jérôme Feret
Jean-Claude Fernandez
Gianluigi Ferrari
Riccardo Focardi
Martin Fränzle
John Gallagher
Roberto Giacobazzi
Arie Gurfinkel
Rene Rydhof Hansen
Jonathan Herzog
Patricia Hill
Frank Huch
Radu Iosif
Romain Janvier
Salvatore La Torre
Flavio Lerda
Francesca Levi
Flaminia Luccio
Jens Knoop
Daniel Kroening
Damiano Macedonio
Monika Maidl
Oded Maler
Damien Massé
Laurent Mauborgne
Fred Mesnard
Antoine Miné
Jean-François Monin
David Monniaux
Laurent Mounier
Kedar Namjoshi
Flemming Nielson
Sinha Nishant
Iulian Ober
Joël Ouaknine
Carla Piazza
Amir Pnueli
Cory Plock
Shaz Qadeer
Sriram Rajamani
Xavier Rival
Sabina Rossi
Grigore Rosu
Oliver Rüthing
Nicoletta Sabadini
Ursula Scheben
Axel Simon
Eli Singerman
Francesca Scozzari
Margaret H. Smith
Muralidhar Talupur
Simone Tini
Tayssir Touili
Stavros Tripakis
Enrico Tronci
Helmut Veith
Andreas Wolf
Ben Worrell
James Worrell
Aleksandr Zaks

Table of Contents

Formal Methods II

Software Checking

Invited Talk

Software Checking

Liveness and Completeness

Formal Methods III

Key Note

Security, Protocols, and Trust*

Joshua D. Guttman

guttman@mitre.org
http://www.ccs.neu.edu/home/guttman

Information security has benefited from mathematically cogent modeling and analysis, which can assure the absence of specific kinds of attacks. Information security provides the right sorts of problems: Correctness conditions may be subtle, but they have definite mathematical content. Systems may be complex, but the essential reasons for failures are already present in simple components. Thus, rigorous methods lead to clear improvements.

In this tutorial, we focus on one problem area, namely cryptographic protocols. Cryptographic protocols are often wrong, and we will start by studying how to break them. Most protocol failures arise from *unintended services* contained in the protocols themselves. An unintended service is an aspect of the protocol that requires legitimate principals unwittingly to provide an attacker with information that helps the attacker defeat the protocol. We describe a systematic way to discover unintended services and to piece them together into attacks.

Turning to the complementary problem of proving that there are no attacks on a particular protocol, we use the same insights to develop three basic patterns for protocol verification. These patterns concern the way that fresh, randomly chosen values ("nonces") are transmitted and later received back in cryptographically altered forms. We explain how these patterns, the *authentication tests*, are used to achieve authentication and to guarantee recency. They serve as a design method as well as a verification method.

In themselves, however, these methods do not explain the commitments that a principal makes by specific protocol actions, nor the trust one principal must have in another in order to be willing to continue a protocol run. In the last part of the tutorial, we describe how to combine protocol analysis with a *trust management logic* in order to formalize the trust consequences of executing protocols for electronic commerce and access control.

* Supported by the United States National Security Agency and the MITRE-Sponsored Research Program.

B. Steffen and G. Levi (Eds.): VMCAI 2004, LNCS 2937, p. 1, 2004.

Security Types Preserving Compilation*
(Extended Abstract)

Gilles Barthe[1], Amitabh Basu[2**], and Tamara Rezk[1]

[1] INRIA Sophia-Antipolis, France {Gilles.Barthe,Tamara.Rezk}@sophia.inria.fr
[2] IIT Delhi, India csu00099@cse.iitd.ernet.in

Abstract. Initiating from the seminal work of Volpano and Smith, there has been ample evidence that type systems may be used to enforce confidentiality of programs through non-interference. However, most type systems operate on high-level languages and calculi, and "low-level languages have not received much attention in studies of secure information flow" (Sabelfeld and Myers, [16]). Further, security type systems for low-level languages should appropriately relate to their counterparts for high-level languages; however, we are not aware of any study of type-preserving compilers for type systems for information flow.
In answer to these questions, we introduce a security type system for a low-level language featuring jumps and calls, and show that the type system enforces termination-insensitive non-interference. Then, we introduce a compiler from a high-level imperative programming language to our low-level language, and show that the compiler preserves security types.

1 Introduction

Type systems are popular artefacts to enforce safety properties in the context of mobile and embedded code. While such safety properties fail short of providing appropriate guarantees with respect to security policies to which mobile and embedded code must adhere, recent work has demonstrated that type systems are adequate to enforce statically security policies. These works generally focus on confidentiality and in particular on non-interference [7], which ensures confidentiality through the absence of information leakage. Initiating from the seminal work of Volpano, Smith and Irvine [20], type systems for non-interference have been thoroughly studied in the literature, see e.g. [16] for a survey. However, most works focus on high-level calculi, including λ-calculus, see e.g. [8], π-calculus, see e.g. [9], and ς-calculus [3], or high-level programming languages, including Java [2,12] and ML [15].

In contrast, relatively little is known about non-interference for low-level languages, in particular because their lack of structure renders control flow more intricate; in fact existing works, see e.g. [4,5], use model-checking and abstract

* Work partially supported by IST Projects Profundis and Verificard.
** This work was performed while the author was visiting INRIA Sophia-Antipolis.

B. Steffen and G. Levi (Eds.): VMCAI 2004, LNCS 2937, pp. 2–15, 2004.

interpretation techniques to detect illegal information flows, but do not provide proofs of non-interference for programs that are accepted by their analysis. Thus the first part of this paper is devoted to the definition of a security type system for a low-level language with jumps and calls, and a proof that the type system enforces termination-insensitive non-interference.

Of course, security type systems for low-level languages should appropriately relate to their counterparts for high-level languages. Indeed, one would expect that compilation preserves security typing. Thus the second part of the paper is devoted to a case study in compilation with security types: we define a high-level imperative language with procedures, and a compiler to the low-level language studied in the first part of the paper. Further, we endorse the language with a type system that guarantees termination-insensitive non-interference, and show that compilation function preserves typing. The proof that compilation preserves typing proceeds by induction on the structure of derivations, and can be viewed as a procedure to compute, from a certificate of well-typing at the source program, another certificate of well-typing for the compiled program. It is thus very close in spirit to a certifying compiler [13].

Contents. The remaining of the paper is organized as follows. In Section 2 we define an assembly language that shall serve as the compiler target, endorse it with a security type system, and prove that the type system enforces termination-insensitive non-interference. In Section 3, we introduce a high-level imperative language with procedures and its associated type system. Further, we introduce a compiler that we show to preserve security typing; we also discuss how type-preserving compilation can be used to lift non-interference to the high-level language. We conclude in Section 4, with related work and directions for further research.

2 Assembly Language

2.1 Syntax and Operational Semantics

The assembly language is a small language with jumps and procedures. A *program* P is a set of *procedures* with a distinguished, main, procedure; we let P_f be the procedure associated to an identifier f in P. Each procedure P_f consists of an array of instructions; we let $P_f[i]$ be the i-th instruction in P_f. The set Instr of instructions and the set Prog_c of compiled programs are defined in Figure 1. We often denote programs by $P_c :: [f := i^\star]^\star$. Given a program P, we let $\mathcal{PP}$ be its set of *programs points*, i.e. the set of pairs $\langle f, i\rangle$ with $f \in \mathcal{F}$, where $\mathcal{F}$ is a set of procedure names, and $i \in dom(P_f)$. Further, we assume programs to satisfy the usual well-formedness conditions, such as code containment: for every program point $\langle f, i\rangle$, $P_f[i] = \mathsf{if}\ j \ \Rightarrow\ j \in dom(P_f)$, etc.

The operational semantics is given as a transition relation between states. In our setting, values are integers, i.e. $\mathcal{V} = \mathbb{Z}$ and states are triples of the form $\langle \mathsf{cs}, \rho, s\rangle$ where $\mathsf{cs} \in \mathcal{PP}^\star$ is a *call string* whose length is bounded by some