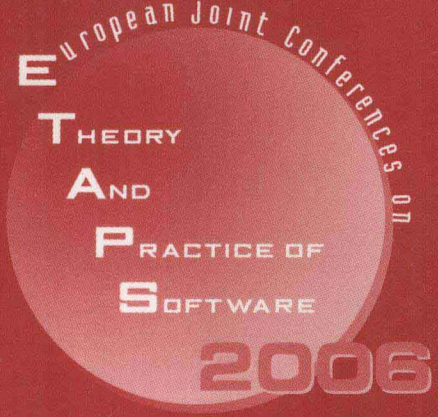
Alan Mycroft
Andreas Zeller (Eds.)

LNCS 3923

# Compiler Construction

## 15th International Conference, CC 2006
## Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006
## Vienna, Austria, March 2006, Proceedings




Springer

Alan Mycroft Andreas Zeller (Eds.)

# Compiler Construction

15th International Conference, CC 2006
Held as Part of the Joint European Conferences
on Theory and Practice of Software, ETAPS 2006
Vienna, Austria, March 30-31, 2006
Proceedings

Springer

Volume Editors

Alan Mycroft
Cambridge University
Cambridge, UK
E-mail: am@cl.cam.ac.uk

Andreas Zeller
Saarland University
Saarbrücken, Germany
E-mail: zeller@cs.uni-sb.de

Library of Congress Control Number: 2006922081

CR Subject Classification (1998): D.3.4, D.3.1, F.4.2, D.2.6, F.3, I.2.2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-540-33050-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-33050-9 Springer Berlin Heidelberg New York



Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11688839 06/3142 5 4 3 2 1 0

# Lecture Notes in Computer Science 3923

*Commenced Publication in 1973*
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

# Foreword

ETAPS 2006 was the ninth instance of the European Joint Conferences on Theory and Practice of Software. ETAPS is an annual federated conference that was established in 1998 by combining a number of existing and new conferences. This year it comprised five conferences (CC, ESOP, FASE, FOSSACS, TACAS), 18 satellite workshops (ACCAT, AVIS, CMCS, COCV, DCC, EAAI, FESCA, FRCSS, GT-VMT, LDTA, MBT, QAPL, SC, SLAP, SPIN, TERMGRAPH, WITS and WRLA), two tutorials, and seven invited lectures (not including those that were specific to the satellite events). We received over 550 submissions to the five conferences this year, giving an overall acceptance rate of 23%, with acceptance rates below 30% for each conference. Congratulations to all the authors who made it to the final programme! I hope that most of the other authors still found a way of participating in this exciting event and I hope you will continue submitting.

The events that comprise ETAPS address various aspects of the system development process, including specification, design, implementation, analysis and improvement. The languages, methodologies and tools which support these activities are all well within its scope. Different blends of theory and practice are represented, with an inclination towards theory with a practical motivation on the one hand and soundly based practice on the other. Many of the issues involved in software design apply to systems in general, including hardware systems, and the emphasis on software is not intended to be exclusive.

ETAPS is a loose confederation in which each event retains its own identity, with a separate programme committee and proceedings. Its format is open-ended, allowing it to grow and evolve as time goes by. Contributed talks and system demonstrations are in synchronised parallel sessions, with invited lectures in plenary sessions. Two of the invited lectures are reserved for "unifying" talks on topics of interest to the whole range of ETAPS attendees. The aim of cramming all this activity into a single one-week meeting is to create a strong magnet for academic and industrial researchers working on topics within its scope, giving them the opportunity to learn about research in related areas, and thereby to foster new and existing links between work in areas that were formerly addressed in separate meetings.

ETAPS 2006 was organized by the Vienna University of Technology, in cooperation with

- European Association for Theoretical Computer Science (EATCS);
- European Association for Programming Languages and Systems (EAPLS);
- European Association of Software Science and Technology (EASST);
- Institute for Computer Languages, Vienna;
- Austrian Computing Society;
- The *Bürgermeister der Bundeshauptstadt Wien*;

- Vienna Convention Bureau;
- Intel.

The organizing team comprised:

| | |
|---|---|
| Chair: | Jens Knoop |
| Local Arrangements: | Anton Ertl |
| Publicity: | Joost-Pieter Katoen |
| Satellite Events: | Andreas Krall |
| Industrial Liaison: | Eva Kühn |
| Liaison with City of Vienna: | Ulrich Neumerkel |
| Tutorials Chair, Website: | Franz Puntigam |
| Website: | Fabian Schmied |
| Local Organization, Workshops Proceedings: | Markus Schordan |

Overall planning for ETAPS conferences is the responsibility of its Steering Committee, whose current membership is:

Perdita Stevens (Edinburgh, Chair), Luca Aceto (Aalborg and Reykjavík), Rastislav Bodík (Berkeley), Maura Cerioli (Genova), Matt Dwyer (Nebraska), Hartmut Ehrig (Berlin), José Fiadeiro (Leicester), Marie-Claude Gaudel (Paris), Roberto Gorrieri (Bologna), Reiko Heckel (Leicester), Michael Huth (London), Joost-Pieter Katoen (Aachen), Paul Klint (Amsterdam), Jens Knoop (Vienna), Shriram Krishnamurthi (Brown), Kim Larsen (Aalborg), Tiziana Margaria (Göttingen), Ugo Montanari (Pisa), Rocco de Nicola (Florence), Hanne Riis Nielson (Copenhagen), Jens Palsberg (UCLA), Mooly Sagiv (Tel-Aviv), João Saraiva (Minho), Don Sannella (Edinburgh), Vladimiro Sassone (Southampton), Helmut Seidl (Munich), Peter Sestoft (Copenhagen), Andreas Zeller (Saarbrücken).

I would like to express my sincere gratitude to all of these people and organizations, the programme committee chairs and PC members of the ETAPS conferences, the organizers of the satellite events, the speakers themselves, the many reviewers, and Springer-Verlag for agreeing to publish the ETAPS proceedings. Finally, I would like to thank the organizing chair of ETAPS 2006, Jens Knoop, for arranging for us to have ETAPS in the most beautiful city of Vienna.

Edinburgh, January 2006

Perdita Stevens
ETAPS Steering Committee Chair

# Preface

The Program Committee is pleased to present the proceedings of the 15th International Conference on Compiler Construction (CC 2006) which was held on March 30 and 31 in Vienna, Austria, as part of the Joint European Conference on Theory and Practice of Software (ETAPS 2006).

Traditionally, CC had been a forum for research on compiler construction. Starting last year, CC has expanded its remit to include a broader spectrum of programming tools, from analysis tools to compilers to virtual machines to debuggers. The submissions we received again reflected the new scope of the conference.

The Program Committee received 71 submissions. From these, 17 research papers and 3 tool demonstrations were selected, giving an overall acceptance rate of 28%.

The Program Committee included 16 members representing 9 countries on 3 continents. Each member reviewed roughly 16 papers and each paper received at least three reviews. In all, 45 external reviewers participated in the review process. Committee members were allowed to submit papers; these would be screened by four reviewers. The Program Committee met on December 5 in London for a one-day meeting. All but three of the members attended the meeting.

Many people contributed to the success of this conference. First of all, we would like to thank the authors for all the care they put into their submissions. Our gratitude also goes to the Program Committee members and external reviewers for their substantive and insightful reviews. Intel generously funded parts of the Program Committee meeting. Special thanks go to Jay McCarthy for maintaining the Continue Conference Server.

CC 2006 was made possible by the ETAPS Steering Committee, in particular by the hard work of Jens Knoop in the role of ETAPS 2006 Organizing Committee Chair, and by that of Anton Ertl in taking care of the local arrangements. We would also like to thank Reinhard Wilhelm and Ras Bodik, recent CC chairs, for on-going helpful discussions about CC's future direction. Finally, we are grateful to George Necula for accepting the invitation to give a keynote talk.

January 2006

Alan Mycroft and Andreas Zeller
CC 2006 Program Chairs

# Conference Organization

## Program Chairs

| | |
|---|---|
| Alan Mycroft | University of Cambridge, UK |
| Andreas Zeller | Saarland University, Germany |

## Program Committee

| | |
|---|---|
| Radhia Cousot | CNRS, France |
| Koen De Bosschere | Ghent University, Belgium |
| Arie van Deursen | CWI, Netherlands |
| Michael Ernst | Massachusetts Institute of Technology, USA |
| Sergei Gorlatch | University of Münster, Germany |
| Chris Hankin | Imperial College, UK |
| Jens Knoop | TU Vienna, Austria |
| Shriram Krishnamurthi | Brown University, Rhode Island, USA |
| K. Rustan M. Leino | Microsoft Research, Washington, USA |
| Oege de Moor | Oxford University, UK |
| Greg Morrisett | Harvard University, Massachusetts, USA |
| Morten Rhiger | Roskilde University, Denmark |
| Barbara Ryder | Rutgers University, New Jersey, USA |
| Frank Tip | IBM Research, New York, USA |
| Des Watson | University of Sussex, UK |
| Kwangkeun Yi | Seoul National University, Korea |

## Additional Reviewers

| | | |
|---|---|---|
| Martin Alt | Erik D'Hollander | Martin Griebl |
| Gerco Ballintijn | Bruno De Bus | Christoph Herrmann |
| Anne Benoit | John Dias | Oleg Kiselyov |
| Jim Benham | Julian Dolby | Taeke Kooiker |
| Kristof Beyls | Jan Dünnweber | Christoph Kessler |
| Magiel Bruntink | Bruno Dufour | Andreas Krall |
| Dries Buytaert | Rob Economopoulos | Jens Krinke |
| Dominique Chanet | Anton Ertl | Yossi Lev |
| Ophelia Chesley | Chen Fu | Jonas Maebe |
| Jamieson M. Cobleigh | Robert Fuhrer | Guillaume Marceau |
| Bas Cornelissen | Andy Georges | Thomas J. Marlowe |

Jay McCarthy
Ali Mesbah
Nick Mitchell
Francois Pottier
Xiaoxia Ren
Oliver Rüthing
Markus Schordan
Jeffrey Siskind
Ludo Van Put
Mandana Vaziri
Weilei Zhang
Lenore Zuck

# Lecture Notes in Computer Science

For information about Vols. 1–3806

please contact your bookseller or Springer

Vol. 3923: A. Mycroft, A. Zeller (Eds.), Compiler Construction. XIII, 277 pages. 2006.

Vol. 3903: K. Chen, R. Deng, X. Lai, J. Zhou (Eds.), Information Security Practice and Experience. XIV, 392 pages. 2006.

Vol. 3901: P.M. Hill (Ed.), Logic Based Program Synthesis and Transformation. X, 179 pages. 2006.

Vol. 3899: S. Frintrop, VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search. XIV, 216 pages. 2006. (Sublibrary LNAI).

Vol. 3895: O. Goldreich, A.L. Rosenberg, A.L. Selman (Eds.), Essays in Theoretical Computer Science. XII, 399 pages. 2006.

Vol. 3894: W. Grass, B. Sick, K. Waldschmidt (Eds.), Architecture of Computing Systems - ARCS 2006. XII, 496 pages. 2006.

Vol. 3890: S.G. Thompson, R. Ghanea-Hercock (Eds.), Defence Applications of Multi-Agent Systems. XII, 141 pages. 2006. (Sublibrary LNAI).

Vol. 3889: J. Rosca, D. Erdogmus, J.C. Príncipe, S. Haykin (Eds.), Independent Component Analysis and Blind Signal Separation. XXI, 980 pages. 2006.

Vol. 3888: D. Draheim, G. Weber (Eds.), Trends in Enterprise Application Architecture. IX, 145 pages. 2006.

Vol. 3887: J. Correa, A. Hevia, M. Kiwi (Eds.), LATIN 2006: Theoretical Informatics. XVI, 814 pages. 2006.

Vol. 3886: E.G. Bremer, J. Hakenberg, E.-H.(S.) Han, D. Berrar, W. Dubitzky (Eds.), Knowledge Discovery in Life Science Literature. XIV, 147 pages. 2006. (Sublibrary LNBI).

Vol. 3885: V. Torra, Y. Narukawa, A. Valls, J. Domingo-Ferrer (Eds.), Modeling Decisions for Artificial Intelligence. XII, 374 pages. 2006. (Sublibrary LNAI).

Vol. 3884: B. Durand, W. Thomas (Eds.), STACS 2006. XIV, 714 pages. 2006.

Vol. 3881: S. Gibet, N. Courty, J.-F. Kamp (Eds.), Gesture in Human-Computer Interaction and Simulation. XIII, 344 pages. 2006. (Sublibrary LNAI).

Vol. 3880: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development I. IX, 335 pages. 2006.

Vol. 3879: T. Erlebach, G. Persinao (Eds.), Approximation and Online Algorithms. X, 349 pages. 2006.

Vol. 3878: A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing. XVII, 589 pages. 2006.

Vol. 3877: M. Detyniecki, J.M. Jose, A. Nürnberger, C. J. '. van Rijsbergen (Eds.), Adaptive Multimedia Retrieval: User, Context, and Feedback. XI, 279 pages. 2006.

Vol. 3876: S. Halevi, T. Rabin (Eds.), Theory of Cryptography. XI, 617 pages. 2006.

Vol. 3875: S. Ur, E. Bin, Y. Wolfsthal (Eds.), Haifa Verification Conference. X, 265 pages. 2006.

Vol. 3874: R. Missaoui, J. Schmidt (Eds.), Formal Concept Analysis. X, 309 pages. 2006. (Sublibrary LNAI).

Vol. 3873: L. Maicher, J. Park (Eds.), Charting the Topic Maps Research and Applications Landscape. VIII, 281 pages. 2006. (Sublibrary LNAI).

Vol. 3872: H. Bunke, A. L. Spitz (Eds.), Document Analysis Systems VII. XIII, 630 pages. 2006.

Vol. 3870: S. Spaccapietra, P. Atzeni, W.W. Chu, T. Catarci, K.P. Sycara (Eds.), Journal on Data Semantics V. XIII, 237 pages. 2006.

Vol. 3869: S. Renals, S. Bengio (Eds.), Machine Learning for Multimodal Interaction. XIII, 490 pages. 2006.

Vol. 3868: K. Römer, H. Karl, F. Mattern (Eds.), Wireless Sensor Networks. XI, 342 pages. 2006.

Vol. 3866: T. Dimitrakos, F. Martinelli, P.Y.A. Ryan, S. Schneider (Eds.), Formal Aspects in Security and Trust. X, 259 pages. 2006.

Vol. 3865: W. Shen, K.-M. Chao, Z. Lin, J.-P.A. Barthès (Eds.), Computer Supported Cooperative Work in Design II. XII, 359 pages. 2006.

Vol. 3863: M. Kohlhase (Ed.), Mathematical Knowledge Management. XI, 405 pages. 2006. (Sublibrary LNAI).

Vol. 3862: R.H. Bordini, M. Dastani, J. Dix, A.E.F. Seghrouchni (Eds.), Programming Multi-Agent Systems. XIV, 267 pages. 2006. (Sublibrary LNAI).

Vol. 3861: J. Dix, S.J. Hegner (Eds.), Foundations of Information and Knowledge Systems. X, 331 pages. 2006.

Vol. 3860: D. Pointcheval (Ed.), Topics in Cryptology – CT-RSA 2006. XI, 365 pages. 2006.

Vol. 3858: A. Valdes, D. Zamboni (Eds.), Recent Advances in Intrusion Detection. X, 351 pages. 2006.

Vol. 3857: M.P.C. Fossorier, H. Imai, S. Lin, A. Poli (Eds.), Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. XI, 350 pages. 2006.

Vol. 3855: E. A. Emerson, K.S. Namjoshi (Eds.), Verification, Model Checking, and Abstract Interpretation. XI, 443 pages. 2005.

Vol. 3854: I. Stavrakakis, M. Smirnov (Eds.), Autonomic Communication. XIII, 303 pages. 2006.

Vol. 3853: A.J. Ijspeert, T. Masuzawa, S. Kusumoto (Eds.), Biologically Inspired Approaches to Advanced Information Technology. XIV, 388 pages. 2006.

Vol. 3852: P.J. Narayanan, S.K. Nayar, H.-Y. Shum (Eds.), Computer Vision – ACCV 2006, Part II. XXXI, 977 pages. 2006.

Vol. 3851: P.J. Narayanan, S.K. Nayar, H.-Y. Shum (Eds.), Computer Vision – ACCV 2006, Part I. XXXI, 973 pages. 2006.

Vol. 3850: R. Freund, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing. IX, 371 pages. 2006.

Vol. 3849: I. Bloch, A. Petrosino, A.G.B. Tettamanzi (Eds.), Fuzzy Logic and Applications. XIV, 438 pages. 2006. (Sublibrary LNAI).

Vol. 3848: J.-F. Boulicaut, L. De Raedt, H. Mannila (Eds.), Constraint-Based Mining and Inductive Databases. X, 401 pages. 2006. (Sublibrary LNAI).

Vol. 3847: K.P. Jantke, A. Lunzer, N. Spyratos, Y. Tanaka (Eds.), Federation over the Web. X, 215 pages. 2006. (Sublibrary LNAI).

Vol. 3846: H. J. van den Herik, Y. Björnsson, N.S. Netanyahu (Eds.), Computers and Games. XIV, 333 pages. 2006.

Vol. 3845: J. Farré, I. Litovsky, S. Schmitz (Eds.), Implementation and Application of Automata. XIII, 360 pages. 2006.

Vol. 3844: J.-M. Bruel (Ed.), Satellite Events at the MoDELS 2005 Conference. XIII, 360 pages. 2006.

Vol. 3843: P. Healy, N.S. Nikolov (Eds.), Graph Drawing. XVII, 536 pages. 2006.

Vol. 3842: H.T. Shen, J. Li, M. Li, J. Ni, W. Wang (Eds.), Advanced Web and Network Technologies, and Applications. XXVII, 1057 pages. 2006.

Vol. 3841: X. Zhou, J. Li, H.T. Shen, M. Kitsuregawa, Y. Zhang (Eds.), Frontiers of WWW Research and Development - APWeb 2006. XXIV, 1223 pages. 2006.

Vol. 3840: M. Li, B. Boehm, L.J. Osterweil (Eds.), Unifying the Software Process Spectrum. XVI, 522 pages. 2006.

Vol. 3839: J.-C. Filliâtre, C. Paulin-Mohring, B. Werner (Eds.), Types for Proofs and Programs. VIII, 275 pages. 2006.

Vol. 3838: A. Middeldorp, V. van Oostrom, F. van Raamsdonk, R. de Vrijer (Eds.), Processes, Terms and Cycles: Steps on the Road to Infinity. XVIII, 639 pages. 2005.

Vol. 3837: K. Cho, P. Jacquet (Eds.), Technologies for Advanced Heterogeneous Networks. IX, 307 pages. 2005.

Vol. 3836: J.-M. Pierson (Ed.), Data Management in Grids. X, 143 pages. 2006.

Vol. 3835: G. Sutcliffe, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. XIV, 744 pages. 2005. (Sublibrary LNAI).

Vol. 3834: D.G. Feitelson, E. Frachtenberg, L. Rudolph, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing. VIII, 283 pages. 2005.

Vol. 3833: K.-J. Li, C. Vangenot (Eds.), Web and Wireless Geographical Information Systems. XI, 309 pages. 2005.

Vol. 3832: D. Zhang, A.K. Jain (Eds.), Advances in Biometrics. XX, 796 pages. 2005.

Vol. 3831: J. Wiedermann, G. Tel, J. Pokorný, M. Bieliková, J. Štuller (Eds.), SOFSEM 2006: Theory and Practice of Computer Science. XV, 576 pages. 2006.

Vol. 3830: D. Weyns, H. V.D. Parunak, F. Michel (Eds.), Environments for Multi-Agent Systems II. VIII, 291 pages. 2006. (Sublibrary LNAI).

Vol. 3829: P. Pettersson, W. Yi (Eds.), Formal Modeling and Analysis of Timed Systems. IX, 305 pages. 2005.

Vol. 3828: X. Deng, Y. Ye (Eds.), Internet and Network Economics. XVII, 1106 pages. 2005.

Vol. 3827: X. Deng, D.-Z. Du (Eds.), Algorithms and Computation. XX, 1190 pages. 2005.

Vol. 3826: B. Benatallah, F. Casati, P. Traverso (Eds.), Service-Oriented Computing - ICSOC 2005. XVIII, 597 pages. 2005.

Vol. 3824: L.T. Yang, M. Amamiya, Z. Liu, M. Guo, F.J. Rammig (Eds.), Embedded and Ubiquitous Computing – EUC 2005. XXIII, 1204 pages. 2005.

Vol. 3823: T. Enokido, L. Yan, B. Xiao, D. Kim, Y. Dai, L.T. Yang (Eds.), Embedded and Ubiquitous Computing – EUC 2005 Workshops. XXXII, 1317 pages. 2005.

Vol. 3822: D. Feng, D. Lin, M. Yung (Eds.), Information Security and Cryptology. XII, 420 pages. 2005.

Vol. 3821: R. Ramanujam, S. Sen (Eds.), FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science. XIV, 566 pages. 2005.

Vol. 3820: L.T. Yang, X.-s. Zhou, W. Zhao, Z. Wu, Y. Zhu, M. Lin (Eds.), Embedded Software and Systems. XXVIII, 779 pages. 2005.

Vol. 3819: P. Van Hentenryck (Ed.), Practical Aspects of Declarative Languages. X, 231 pages. 2005.

Vol. 3818: S. Grumbach, L. Sui, V. Vianu (Eds.), Advances in Computer Science – ASIAN 2005. XIII, 294 pages. 2005.

Vol. 3817: M. Faundez-Zanuy, L. Janer, A. Esposito, A. Satue-Villar, J. Roure, V. Espinosa-Duro (Eds.), Nonlinear Analyses and Algorithms for Speech Processing. XII, 380 pages. 2006. (Sublibrary LNAI).

Vol. 3816: G. Chakraborty (Ed.), Distributed Computing and Internet Technology. XXI, 606 pages. 2005.

Vol. 3815: E.A. Fox, E.J. Neuhold, P. Premsmit, V. Wuwongse (Eds.), Digital Libraries: Implementing Strategies and Sharing Experiences. XVII, 529 pages. 2005.

Vol. 3814: M. Maybury, O. Stock, W. Wahlster (Eds.), Intelligent Technologies for Interactive Entertainment. XV, 342 pages. 2005. (Sublibrary LNAI).

Vol. 3813: R. Molva, G. Tsudik, D. Westhoff (Eds.), Security and Privacy in Ad-hoc and Sensor Networks. VIII, 219 pages. 2005.

Vol. 3812: C. Bussler, A. Haller (Eds.), Business Process Management Workshops. XIII, 520 pages. 2006.

Vol. 3811: C. Bussler, M.-C. Shan (Eds.), Technologies for E-Services. VIII, 127 pages. 2006.

Vol. 3810: Y.G. Desmedt, H. Wang, Y. Mu, Y. Li (Eds.), Cryptology and Network Security. XI, 349 pages. 2005.

Vol. 3809: S. Zhang, R. Jarvis (Eds.), AI 2005: Advances in Artificial Intelligence. XXVII, 1344 pages. 2005. (Sublibrary LNAI).

Vol. 3808: C. Bento, A. Cardoso, G. Dias (Eds.), Progress in Artificial Intelligence. XVIII, 704 pages. 2005. (Sublibrary LNAI).

Vol. 3807: M. Dean, Y. Guo, W. Jun, R. Kaschek, S. Krishnaswamy, Z. Pan, Q.Z. Sheng (Eds.), Web Information Systems Engineering – WISE 2005 Workshops. XV, 275 pages. 2005.

# Table of Contents

## Optimization



## Code Generation



## Register Allocation

# Using Dependent Types to Port Type Systems to Low-Level Languages

George Necula

University of California, Berkeley, USA

A major difficulty when trying to apply high-level type systems to low-level languages is that we must reason about relationships between values. For example, in a low-level implementation of object-oriented dynamic dispatch we must ensure that the "self" argument passed to the method is the same object from whose virtual table we fetched the pointer to the method. Similarly, in low-level code using arrays we must relate the array address with the variables that store the bounds. We show for several examples that the high-level type system must be extended with dependent types in order to reason about low-level code. The novel feature in this use of dependent types is that they can be used in presence of pointers and mutation.

We discuss three case studies. First, we show a variant of bytecode verification that operates on the assembly language output of a native code compiler. Second, we show how to express and check at the assembly level the invariants enforced by CCured, a source-level instrumentation tool that guarantees type safety in legacy C programs. Finally, we show that dependent types are a natural specification mechanism for enforcing common safe programming practices in C programs. We have used this mechanism to efficiently enforce memory safety for several Linux device drivers.

A. Mycroft and A. Zeller (Eds.): CC 2006, LNCS 3923, p. 1, 2006.

# Interprocedural Dataflow Analysis in the Presence of Large Libraries

Atanas Rountev[1], Scott Kagan[1], and Thomas Marlowe[2]

[1] Ohio State University, Columbus, OH, USA
[2] Seton Hall University, South Orange, NJ, USA

**Abstract.** Interprocedural dataflow analysis has a large number of uses for software optimization, maintenance, testing, and verification. For software built with reusable components, the traditional approaches for *whole-program analysis* cannot be used directly. This paper considers *component-level analysis* of a main component which is built on top of a pre-existing library component. We propose an approach for computing summary information for the library and for using it to analyze the main component. The approach defines a general theoretical framework for dataflow analysis of programs built with large extensible library components, using pre-computed summary functions for library-local execution paths. Our experimental results indicate that the cost of component-level analysis could be substantially lower than the cost of the corresponding whole-program analysis, without any loss of precision. These results present a promising step towards practical analysis techniques for large-scale software systems built with reusable components.

## 1 Introduction

*Interprocedural dataflow analysis* is a widely-used form of static program analysis. Dataflow analysis techniques play an important role in tools for performance optimization, program understanding and maintenance, software testing, and verification of program properties. Unfortunately, the use of interprocedural dataflow analysis in real-world software tools is hindered by several serious challenges. One of the central problems is the underlying analysis model implicit in most of the work in this area. The key feature of this model is the assumption of a *whole-program analysis for a homogeneous program.* Interprocedural whole-program analysis takes as input an entire program and produces information about the behavior of that program. This classical dataflow analysis model [28] assumes that the source code for the whole program is available for analysis.

Modern software presents serious challenges for this traditional model. For example, systems often contain reusable components. Whole-program analysis assumes that it is appropriate to analyze the source code of the entire program as a single unit. However, for software built with reusable components,

- Some program components may be available only in binary form, without source code, which makes whole-program analysis impossible.

A. Mycroft and A. Zeller (Eds.): CC 2006, LNCS 3923, pp. 2–16, 2006.

- It is necessary to re-analyze a component every time this component is used as part of a new system. For example, a library may be used in many applications, and whole-program analysis requires re-analysis of this library from scratch in the context of each such application.
- Code changes in one component typically require complete re-analysis of the entire application.
- The cost of whole-program analysis is often dominated by the analysis of the underlying large library components (e.g., standard libraries, middleware, frameworks, etc.). To achieve practical cost, analysis designers are often forced to use semantic approximations that reduce the precision and usefulness of the analysis solution.

These issues limit the usefulness of many existing analyses. In some cases the analyses cannot be used at all. Even if they are possible, the analyses have to be relatively approximate in order to scale for large-scale software with hundreds of thousands (or even millions) lines of code. Such approximations lead to under-optimized code in optimizing compilers, spurious dependencies in program understanding tools, false warnings in verification tools, and infeasible coverage requirements in testing tools.

*Component-Level Dataflow Analysis.* In this paper we consider a model of interprocedural dataflow analysis which we refer to as *component-level analysis* (CLA). A component-level analysis processes the source code of a single program component, given some information about the environment of this component. The general CLA model is discussed in [20] (without any formalisms, proofs, or experiments.) Here, we focus on one particular scenario for CLA: analysis of a main component *Main* which is built on top of a library component *Lib*. In this scenario, the source code of *Lib* is pre-analyzed independently of any library clients. This pre-analysis produces *summary information* for *Lib*. This information is used subsequently for component-level analysis of the source code of any main component built on top of *Lib*.

This form of CLA has significant real-world relevance. In particular, there are large standard libraries that are associated with languages such as C++, Java, and C#. A library could be considered as component *Lib*, while a program written on top of it is component *Main*. CLA allows (1) analysis of *Main* without the source code of *Lib*, by using the summary information, (2) reduction in the cost of analyzing *Main*, because the source code of *Lib* has already been analyzed, (3) reuse of the summary information across multiple main components, in order to avoid repeated re-analysis of *Lib*, and (4) reduced work to handle code changes, since changes in *Main* do not require re-analysis of *Lib*.

*Contributions.* The main goal of our work is to define general theoretical machinery for designing component-level analyses of *Main*. We achieve this goal by generalizing the "functional approach" to whole-program analysis due to Sharir and Pnueli [28]. The key technical issue that this generalization needs to address is *the lack of complete call graph information when performing pre-analysis of a library.* An example of this problem is the presence of callbacks from the library to the main component. The contributions of our work are: