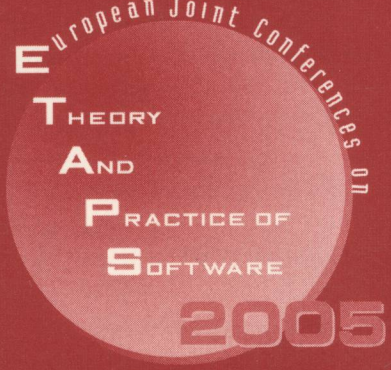
Maura Cerioli (Ed.)

LNCS 3442

Fundamental Approaches to Software Engineering

8th International Conference, FASE 2005
Held as Part of the Joint European Conferences
on Theory and Practice of Software, ETAPS 2005
Edinburgh, UK, April 2005, Proceedings



Springer

Maura Cerioli (Ed.)

Fundamental Approaches to Software Engineering

8th International Conference, FASE 2005
Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005
Edinburgh, UK, April 4-8, 2005
Proceedings

Springer

Volume Editor

Maura Cerioli
Università di Genova, DISI
Via Dodecaneso 35, 16146 Genova, Italy
E-mail: cerioli@disi.unige.it

Library of Congress Control Number: 2005922879

CR Subject Classification (1998): D.2, F.3, D.3

ISSN 0302-9743
ISBN-10 3-540-25420-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-25420-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11405955 06/3142 5 4 3 2 1 0

Foreword

ETAPS 2005 was the eighth 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), 17 satellite workshops (AVIS, BYTECODE, CEES, CLASE, CMSB, COCV, FAC, FESCA, FINCO, GCW-DSE, GLPL, LDTA, QAPL, SC, SLAP, TGC, UITP), seven invited lectures (not including those that were specific to the satellite events), and several tutorials. We received over 550 submissions to the five conferences this year, giving acceptance rates below 30% for each one. Congratulations to all the authors who made it to the final program! 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 program 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 synchronized 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 2005 was organized by the School of Informatics of the University of Edinburgh, 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).

The organizing team comprised:

- Chair: Don Sannella
- Publicity: David Aspinall
- Satellite Events: Massimo Felici

- Secretariat: Dyane Goodchild
- Local Arrangements: Monika-Jeannette Lekuse
- Tutorials: Alberto Momigliano
- Finances: Ian Stark
- Website: Jennifer Tenzer, Daniel Winterstein
- Fundraising: Phil Wadler

ETAPS 2005 received support from the University of Edinburgh.

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 Bodik (Berkeley), Maura Cerioli (Genoa), Evelyn Duesterwald (IBM, USA), Hartmut Ehrig (Berlin), José Fiadeiro (Leicester), Marie-Claude Gaudel (Paris), Roberto Gorrieri (Bologna), Reiko Heckel (Paderborn), Holger Hermanns (Saarbrücken), Joost-Pieter Katoen (Aachen), Paul Klint (Amsterdam), Jens Knoop (Vienna), Kim Larsen (Aalborg), Tiziana Margaria (Dortmund), Ugo Montanari (Pisa), Hanne Riis Nielson (Copenhagen), Fernando Orejas (Barcelona), Mooly Sagiv (Tel Aviv), Don Sannella (Edinburgh), Vladimiro Sassone (Sussex), Peter Sestoft (Copenhagen), Michel Wermelinger (Lisbon), Igor Walukiewicz (Bordeaux), Andreas Zeller (Saarbrücken), Lenore Zuck (Chicago).

I would like to express my sincere gratitude to all of these people and organizations, the program committee chairs and PC members of the ETAPS conferences, the organizers of the satellite events, the speakers themselves, the many reviewers, and Springer for agreeing to publish the ETAPS proceedings. Finally, I would like to thank the organizer of ETAPS 2005, Don Sannella. He has been instrumental in the development of ETAPS since its beginning; it is quite beyond the limits of what might be expected that, in addition to all the work he has done as the original ETAPS Steering Committee Chairman and current ETAPS Treasurer, he has been prepared to take on the task of organizing this instance of ETAPS. It gives me particular pleasure to thank him for organizing ETAPS in this wonderful city of Edinburgh in this my first year as ETAPS Steering Committee Chair.

Edinburgh, January 2005

Perdita Stevens
ETAPS Steering Committee Chair

Preface

The conference on Fundamental Approaches to Software Engineering (FASE) is one of the European Joint Conferences on Theory and Practice of Software (ETAPS). As such, it provides a common forum for practitioners and researchers to discuss theories for supporting and improving software engineering practices and their practical application in real contexts.

Contributions were sought targeting both pragmatic concepts and their formal foundations which could lead to new engineering practices and a higher level of reliability, robustness, and evolvability of heterogeneous software federations.

The record submission of 99 research papers and 6 tool demos was the response of the scientific community, with contributions ranging from theoretical aspects, such as graph grammars, graph transformation, agent theory and algebraic specification languages, to applications to industrially used languages, methods, technologies, and tools, including UML, Web services, product lines, component-based development, Java, and Java cards.

The scientific program was complemented by the invited lectures of Gérard Berry on *Esterel v7: from Verified Formal Specification to Efficient Industrial Designs* and of Thomas A. Henzinger on *Checking Memory Safety with Blast.*

The authors of the submissions were from 29 countries, both within Europe (Belgium, Denmark, Finland, France, Germany, Hungary, Ireland, Italy, Luxembourg, Macedonia, Portugal, Spain, Sweden, Switzerland, The Netherlands, United Kingdom) and outside (Australia, Brazil, Canada, China, India, Japan, Korea, Pakistan, Russia, Thailand, Tunisia, Turkey, USA). It is a pleasure to note the increasing number of submissions from eastern Europe and from outside Europe altogether, showing that FASE is gaining importance as a world-wide conference.

The help of the Program Committee was invaluable in selecting just 25 papers (3 of them tool demos) from the large number of high-quality submissions, and I take the opportunity to thank warmly all its members and the other referees for supporting the selection process with their precious time.

FASE 2005 was held in Edinburgh, hosted and organized by the School of Informatics of the University of Edinburgh. Next year FASE will take place in Vienna (Austria).

Being part of ETAPS, FASE shares the sponsoring and support described by the ETAPS Chair in the Foreword. Heartfelt thanks are also due to José Fiadeiro and Perdita Stevens for their great efforts in the global ETAPS organization and to Don Sannella and his staff for the wonderful job as local organizers.

Finally, a special thanks to the contributors to and participants of FASE, who in the end are the people making the conference worthwhile.

Genoa, January 2005 Maura Cerioli

Organization

Program Committee

Silvia Teresita Acuña (Universidad Autónoma de Madrid, Spain)
Leonor Barroca (Open University, UK)
Yolande Berbers (Katholieke Universiteit Leuven, Belgium)
Jean Bézivin (University of Nantes, France)
Jean-Michel Bruel (University of Pau, France)
Maura Cerioli (Università di Genova, Italy)
Marsha Chechik (University of Toronto, Canada)
Gianpaolo Cugola (Politecnico di Milano, Italy)
Colin Fidge (University of Queensland, Australia)
Anthony Finkelstein (University College London, UK)
Chris George (UNU/IIST, Macao 1, China)
Martin Große-Rhode (Fraunhofer-Institut für Software und Systemtechnik, Germany)
Tomasz Janowski (University of Gdańsk, Poland)
Mehdi Jazayeri (Technical University of Vienna, Austria)
Cliff Jones (University of Newcastle upon Tyne, UK)
Antónia Lopes (University of Lisbon, Portugal)
Tiziana Margaria (University of Dortmund, Germany)
Stephan Merz (INRIA Lorraine, LORIA, France)
Carlo Montangero (Università di Pisa, Italy)
Doron Peled (University of Warwick, UK)
Ernesto Pimentel (Universidad de Málaga, Spain)
Michel Wermelinger (Universidade Nova de Lisboa, Portugal)
Roel Wieringa (University of Twente, The Netherlands)
Alexander Wolf (University of Colorado at Boulder, USA)

Reviewers

Vincenzo Ambriola
Jonathan Amir
Giovanni Cignoni
Joey Coleman
Maya Daneva
Valeria de Castro
Juan de Lara
Benet Devereux
Manuel Díaz
Oscar Dieste
Francisco Durán
Rik Eshuis
Pascal Fenkam
Fabio Gadducci
Vincenzo Gervasi
Mihaela Gheorghiu
Vittoria Gianuzzi
Arie Gurfinkel
Markus Hardt
David N. Jansen
Ioannis Kassios
Engin Kirda
Giovanni Lagorio
Albert Lai
José A. Macías Iglesias
Paola Magillo
Antonio Maña

Stefan Mann
Pedro Merino
Henry Muccini
Shiva Nejati
Johann Oberleitner
José Luis Pastrana
Mónica Pinto
Stanislav Pokraev
Andrea Polini
Claudia Pons
Lucia Rapanotti
Gianna Reggio
Arend Rensink
Matthew Rutherford
Laura Semini
Almudena Sierra
Klaas Sikkel
Judith Stafford
Bedir Tekinerdogan
Klaas Van den Berg
Pascal van Eck
Ou Wei
Elena Zucca

Lecture Notes in Computer Science 3442

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

Editorial Board

Lecture Notes in Computer Science

For information about Vols. 1–3339

please contact your bookseller or Springer

Vol. 3452: F. Baader, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. XI, 562 pages. 2005. (Subseries LNAI).

Vol. 3448: G.R. Raidl, J. Gottlieb (Eds.), Evolutionary Computation in Combinatorial Optimization. XI, 271 pages. 2005.

Vol. 3442: M. Cerioli (Ed.), Fundamental Approaches to Software Engineering. XIII, 373 pages. 2005.

Vol. 3441: V. Sassone (Ed.), Foundations of Software Science and Computational Structures. XVIII, 521 pages. 2005.

Vol. 3440: N. Halbwachs, L.D. Zuck (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. XVII, 588 pages. 2005.

Vol. 3436: B. Bouyssounouse, J. Sifakis (Eds.), Embedded Systems Design. XV, 492 pages. 2005.

Vol. 3433: S. Bhalla (Ed.), Databases in Networked Information Systems. VII, 319 pages. 2005.

Vol. 3432: M. Beigl, P. Lukowicz (Eds.), Systems Aspects in Organic and Pervasive Computing - ARCS 2005. X, 265 pages. 2005.

Vol. 3427: G. Kotsis, O. Spaniol, Wireless Systems and Mobility in Next Generation Internet. VIII, 249 pages. 2005.

Vol. 3423: J.L. Fiadeiro, P.D. Mosses, F. Orejas (Eds.), Recent Trends in Algebraic Development Techniques. VIII, 271 pages. 2005.

Vol. 3422: R.T. Mittermeir (Ed.), From Computer Literacy to Informatics Fundamentals. X, 203 pages. 2005.

Vol. 3421: P. Lorenz, P. Dini (Eds.), Networking - ICN 2005, Part II. XXXV, 1153 pages. 2005.

Vol. 3420: P. Lorenz, P. Dini (Eds.), Networking - ICN 2005, Part I. XXXV, 933 pages. 2005.

Vol. 3419: B. Faltings, A. Petcu, F. Fages, F. Rossi (Eds.), Constraint Satisfaction and Constraint Logic Programming. X, 217 pages. 2005. (Subseries LNAI).

Vol. 3418: U. Brandes, T. Erlebach (Eds.), Network Analysis. XII, 471 pages. 2005.

Vol. 3416: M. Böhlen, J. Gamper, W. Polasek, M.A. Wimmer (Eds.), E-Government: Towards Electronic Democracy. XIII, 311 pages. 2005. (Subseries LNAI).

Vol. 3415: P. Davidsson, B. Logan, K. Takadama (Eds.), Multi-Agent and Multi-Agent-Based Simulation. X, 265 pages. 2005. (Subseries LNAI).

Vol. 3414: M. Morari, L. Thiele (Eds.), Hybrid Systems: Computation and Control. XII, 684 pages. 2005.

Vol. 3412: X. Franch, D. Port (Eds.), COTS-Based Software Systems. XVI, 312 pages. 2005.

Vol. 3411: S.H. Myaeng, M. Zhou, K.-F. Wong, H.-J. Zhang (Eds.), Information Retrieval Technology. XIII, 337 pages. 2005.

Vol. 3410: C.A. Coello Coello, A. Hernández Aguirre, E. Zitzler (Eds.), Evolutionary Multi-Criterion Optimization. XVI, 912 pages. 2005.

Vol. 3409: N. Guelfi, G. Reggio, A. Romanovsky (Eds.), Scientific Engineering of Distributed Java Applications. X, 127 pages. 2005.

Vol. 3408: D.E. Losada, J.M. Fernández-Luna (Eds.), Advances in Information Retrieval. XVII, 572 pages. 2005.

Vol. 3407: Z. Liu, K. Araki (Eds.), Theoretical Aspects of Computing - ICTAC 2004. XIV, 562 pages. 2005.

Vol. 3406: A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing. XVII, 829 pages. 2005.

Vol. 3404: V. Diekert, B. Durand (Eds.), STACS 2005. XVI, 706 pages. 2005.

Vol. 3403: B. Ganter, R. Godin (Eds.), Formal Concept Analysis. XI, 419 pages. 2005. (Subseries LNAI).

Vol. 3401: Z. Li, L.G. Vulkov, J. Waśniewski (Eds.), Numerical Analysis and Its Applications. XIII, 630 pages. 2005.

Vol. 3399: Y. Zhang, K. Tanaka, J.X. Yu, S. Wang, M. Li (Eds.), Web Technologies Research and Development - APWeb 2005. XXII, 1082 pages. 2005.

Vol. 3398: D.-K. Baik (Ed.), Systems Modeling and Simulation: Theory and Applications. XIV, 733 pages. 2005. (Subseries LNAI).

Vol. 3397: T.G. Kim (Ed.), Artificial Intelligence and Simulation. XV, 711 pages. 2005. (Subseries LNAI).

Vol. 3396: R.M. van Eijk, M.-P. Huget, F. Dignum (Eds.), Agent Communication. X, 261 pages. 2005. (Subseries LNAI).

Vol. 3395: J. Grabowski, B. Nielsen (Eds.), Formal Approaches to Software Testing. X, 225 pages. 2005.

Vol. 3394: D. Kudenko, D. Kazakov, E. Alonso (Eds.), Adaptive Agents and Multi-Agent Systems III. VIII, 313 pages. 2005. (Subseries LNAI).

Vol. 3393: H.-J. Kreowski, U. Montanari, F. Orejas, G. Rozenberg, G. Taentzer (Eds.), Formal Methods in Software and Systems Modeling. XXVII, 413 pages. 2005.

Vol. 3391: C. Kim (Ed.), Information Networking. XVII, 936 pages. 2005.

Vol. 3390: R. Choren, A. Garcia, C. Lucena, A. Romanovsky (Eds.), Software Engineering for Multi-Agent Systems III. XII, 291 pages. 2005.

Vol. 3389: P. Van Roy (Ed.), Multiparadigm Programming in Mozart/OZ. XV, 329 pages. 2005.

Vol. 3388: J. Lagergren (Ed.), Comparative Genomics. VII, 133 pages. 2005. (Subseries LNBI).

Vol. 3387: J. Cardoso, A. Sheth (Eds.), Semantic Web Services and Web Process Composition. VIII, 147 pages. 2005.

Vol. 3386: S. Vaudenay (Ed.), Public Key Cryptography - PKC 2005. IX, 436 pages. 2005.

Vol. 3385: R. Cousot (Ed.), Verification, Model Checking, and Abstract Interpretation. XII, 483 pages. 2005.

Vol. 3383: J. Pach (Ed.), Graph Drawing. XII, 536 pages. 2005.

Vol. 3382: J. Odell, P. Giorgini, J.P. Müller (Eds.), Agent-Oriented Software Engineering V. X, 239 pages. 2005.

Vol. 3381: P. Vojtáš, M. Bieliková, B. Charron-Bost, O. Sýkora (Eds.), SOFSEM 2005: Theory and Practice of Computer Science. XV, 448 pages. 2005.

Vol. 3380: C. Priami, Transactions on Computational Systems Biology I. IX, 111 pages. 2005. (Subseries LNBI).

Vol. 3379: M. Hemmje, C. Niederee, T. Risse (Eds.), From Integrated Publication and Information Systems to Information and Knowledge Environments. XXIV, 321 pages. 2005.

Vol. 3378: J. Kilian (Ed.), Theory of Cryptography. XII, 621 pages. 2005.

Vol. 3377: B. Goethals, A. Siebes (Eds.), Knowledge Discovery in Inductive Databases. VII, 190 pages. 2005.

Vol. 3376: A. Menezes (Ed.), Topics in Cryptology – CT-RSA 2005. X, 385 pages. 2005.

Vol. 3375: M.A. Marsan, G. Bianchi, M. Listanti, M. Meo (Eds.), Quality of Service in Multiservice IP Networks. XIII, 656 pages. 2005.

Vol. 3374: D. Weyns, H.V.D. Parunak, F. Michel (Eds.), Environments for Multi-Agent Systems. X, 279 pages. 2005. (Subseries LNAI).

Vol. 3372: C. Bussler, V. Tannen, I. Fundulaki (Eds.), Semantic Web and Databases. X, 227 pages. 2005.

Vol. 3371: M.W. Barley, N. Kasabov (Eds.), Intelligent Agents and Multi-Agent Systems. X, 329 pages. 2005. (Subseries LNAI).

Vol. 3370: A. Konagaya, K. Satou (Eds.), Grid Computing in Life Science. X, 188 pages. 2005. (Subseries LNBI).

Vol. 3369: V.R. Benjamins, P. Casanovas, J. Breuker, A. Gangemi (Eds.), Law and the Semantic Web. XII, 249 pages. 2005. (Subseries LNAI).

Vol. 3368: L. Paletta, J.K. Tsotsos, E. Rome, G.W. Humphreys (Eds.), Attention and Performance in Computational Vision. VIII, 231 pages. 2005.

Vol. 3367: W.S. Ng, B.C. Ooi, A. Ouksel, C. Sartori (Eds.), Databases, Information Systems, and Peer-to-Peer Computing. X, 231 pages. 2005.

Vol. 3366: I. Rahwan, P. Moraitis, C. Reed (Eds.), Argumentation in Multi-Agent Systems. XII, 263 pages. 2005. (Subseries LNAI).

Vol. 3365: G. Mauri, G. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing. IX, 415 pages. 2005.

Vol. 3363: T. Eiter, L. Libkin (Eds.), Database Theory - ICDT 2005. XI, 413 pages. 2004.

Vol. 3362: G. Barthe, L. Burdy, M. Huisman, J.-L. Lanet, T. Muntean (Eds.), Construction and Analysis of Safe, Secure, and Interoperable Smart Devices. IX, 257 pages. 2005.

Vol. 3361: S. Bengio, H. Bourlard (Eds.), Machine Learning for Multimodal Interaction. XII, 362 pages. 2005.

Vol. 3360: S. Spaccapietra, E. Bertino, S. Jajodia, R. King, D. McLeod, M.E. Orlowska, L. Strous (Eds.), Journal on Data Semantics II. XI, 223 pages. 2005.

Vol. 3359: G. Grieser, Y. Tanaka (Eds.), Intuitive Human Interfaces for Organizing and Accessing Intellectual Assets. XIV, 257 pages. 2005. (Subseries LNAI).

Vol. 3358: J. Cao, L.T. Yang, M. Guo, F. Lau (Eds.), Parallel and Distributed Processing and Applications. XXIV, 1058 pages. 2004.

Vol. 3357: H. Handschuh, M.A. Hasan (Eds.), Selected Areas in Cryptography. XI, 354 pages. 2004.

Vol. 3356: G. Das, V.P. Gulati (Eds.), Intelligent Information Technology. XII, 428 pages. 2004.

Vol. 3355: R. Murray-Smith, R. Shorten (Eds.), Switching and Learning in Feedback Systems. X, 343 pages. 2005.

Vol. 3354: M. Margenstern (Ed.), Machines, Computations, and Universality. VIII, 329 pages. 2005.

Vol. 3353: J. Hromkovič, M. Nagl, B. Westfechtel (Eds.), Graph-Theoretic Concepts in Computer Science. XI, 404 pages. 2004.

Vol. 3352: C. Blundo, S. Cimato (Eds.), Security in Communication Networks. XI, 381 pages. 2005.

Vol. 3351: G. Persiano, R. Solis-Oba (Eds.), Approximation and Online Algorithms. VIII, 295 pages. 2005.

Vol. 3350: M. Hermenegildo, D. Cabeza (Eds.), Practical Aspects of Declarative Languages. VIII, 269 pages. 2005.

Vol. 3349: B.M. Chapman (Ed.), Shared Memory Parallel Programming with Open MP. X, 149 pages. 2005.

Vol. 3348: A. Canteaut, K. Viswanathan (Eds.), Progress in Cryptology - INDOCRYPT 2004. XIV, 431 pages. 2004.

Vol. 3347: R.K. Ghosh, H. Mohanty (Eds.), Distributed Computing and Internet Technology. XX, 472 pages. 2004.

Vol. 3346: R.H. Bordini, M. Dastani, J. Dix, A.E.F. Seghrouchni (Eds.), Programming Multi-Agent Systems. XIV, 249 pages. 2005. (Subseries LNAI).

Vol. 3345: Y. Cai (Ed.), Ambient Intelligence for Scientific Discovery. XII, 311 pages. 2005. (Subseries LNAI).

Vol. 3344: J. Malenfant, B.M. Østvold (Eds.), Object-Oriented Technology. ECOOP 2004 Workshop Reader. VIII, 215 pages. 2005.

Vol. 3343: C. Freksa, M. Knauff, B. Krieg-Brückner, B. Nebel, T. Barkowsky (Eds.), Spatial Cognition IV. Reasoning, Action, and Interaction. XIII, 519 pages. 2005. (Subseries LNAI).

Vol. 3342: E. Şahin, W.M. Spears (Eds.), Swarm Robotics. IX, 175 pages. 2005.

Vol. 3341: R. Fleischer, G. Trippen (Eds.), Algorithms and Computation. XVII, 935 pages. 2004.

Vol. 3340: C.S. Calude, E. Calude, M.J. Dinneen (Eds.), Developments in Language Theory. XI, 431 pages. 2004.

Table of Contents

Invited Contributions

Web Services

Graph Grammars and Graph Transformations

Components

Product Lines

Theory

Code Understanding and Validation

The UML

Automatic Proofs and Provers

Esterel v7: From Verified Formal Specification to Efficient Industrial Designs

Gérard Berry

Chief Scientist, Esterel Technologies Member, Academie des Sciences

Synchronous languages were developed in the mid-80's specifically to deal with embedded systems. They are based on mathematical semantics and support formal compilation to classical software or hardware languages as well as formal verification. Esterel v7 is a major industrial evolution of the original Esterel synchronous language, mostly directed to complex hardware applications. The language is supported by the Esterel Studio integrated development environment, which provides a smooth path from verifiable executable specification to efficient circuit synthesis. The graphical Safe States Machines derived from Esterel are also used in the SCADE tool which is widely used for safety-critical software applications in avionics.

Through the examples of Esterel v7 and SCADE, we discuss the impact and evolution of formal methods for actual industrial design. In particular, we discuss some issues that are central for actual applications but are usually either not considered as such or viewed as too difficult to handle in research or R&D projects. We demonstrate that the difference between industrial success and failure often lies in precisely these aspects.

M. Cerioli (Ed.): FASE 2005, LNCS 3442, p. 1, 2005.

Checking Memory Safety with Blast*

Dirk Beyer[1], Thomas A. Henzinger[1,2], Ranjit Jhala[3], and Rupak Majumdar[4]

[1] EPFL, Switzerland
[2] University of California, Berkeley
[3] University of California, San Diego
[4] University of California, Los Angeles

Abstract. BLAST is an automatic verification tool for checking temporal safety properties of C programs. Given a C program and a temporal safety property, BLAST statically proves that either the program satisfies the safety property or the program has an execution trace that exhibits a violation of the property. BLAST constructs, explores, and refines abstractions of the program state space based on lazy predicate abstraction and interpolation-based predicate discovery. We show how BLAST can be used to statically prove memory safety for C programs. We take a two-step approach. First, we use CCURED, a type-based memory safety analyzer, to annotate with run-time checks all program points that cannot be proved memory safe by the type system. Second, we use BLAST to remove as many of the run-time checks as possible (by proving that these checks never fail), and to generate for the remaining run-time checks execution traces that witness them fail. Our experience shows that BLAST can remove many of the run-time checks added by CCURED and provide useful information to the programmer about many of the remaining checks.

1 Introduction

Invalid memory access is a major source of program failures. If a program statement dereferences a pointer that points to an invalid memory cell, the program is either aborted by the operating system or, often worse, the program continues to run with an undefined behavior. To avoid the latter, one can perform checks before every memory access at run time. For some programming languages (e.g., Java) this is done automatically by the compiler/run-time environment. For the language C, neither the compiler nor the run-time environment enforces memory-safety policies. CCURED [7, 24] is a program-transformation tool for C which transforms any given C program to a memory-safe version. CCURED uses a type-based program analysis to prove as many memory accesses as possible

* This research was supported in part by the NSF grants CCR-0234690, CCR-0225610, and ITR-0326577.

M. Cerioli (Ed.): FASE 2005, LNCS 3442, pp. 2–18, 2005.

memory safe, and it inserts run-time checks before the remaining memory accesses. The resulting, "cured" C program is memory safe in the sense that it alarms the user if the program was about to execute an unsafe operation. Despite the manyfold advantages of this approach, it has two drawbacks: first, the run-time checks consume additional processor time, and second, the checks give late feedback, just before the program aborts.

We address these two points by combining CCured with a more powerful, path-sensitive program analysis. The additional analysis is performed by the model checker Blast [19]. For each memory access that the type-based analysis of CCured fails to prove safe, we invoke the more precise, more expensive analysis of Blast. There are three possible outcomes. First, Blast may be able to prove that the memory access is safe (even though CCured was not able to prove this). In this case, no run-time check needs to be inserted, thus reducing the overhead in the cured program. Second, Blast may be able to generate an execution trace to an invalid pointer dereference at the considered control location, i.e., an execution trace along which the run-time check inserted by CCured would fail. This may expose a program bug, which can, based on the error trace provided by Blast, then be fixed by the programmer. Third, Blast may time-out attempting to check whether or not a given memory access is always safe. In this case, the run-time check inserted by CCured remains in the cured program. It is important to note that Blast, even though often more powerful than CCured, is not invoked by itself, but only after a type-based pointer analysis fails. This is because where successful, the CCured analysis is more efficient, and it may also succeed in cases that overwhelm the model checker. However, the combination of CCured and Blast guarantees memory-safe programs with less run-time overhead than the use of CCured alone, and it provides useful compile-time feedback about memory-safety violations to the programmer.

Blast performs an abstract reachability analysis to check if a given error location of a C program can be visited during program execution. All paths of the program are checked symbolically and abstractly, by tracking only some relevant facts (called *predicates*) about program variables, instead of the full program state. If a path to the error location is found, the path may be due to the imprecision in the abstraction (a so-called *spurious* counterexample) or it may correspond to a feasible program path (a *genuine* counterexample). In the former case, additional relevant predicates are discovered automatically to remove the spurious error trace. The process is repeated, by tracking an increasing number of predicates, until either a genuine error trace (program bug) is found, or the abstraction is precise enough to prove the absence of error traces. This scheme of counterexample-guided predicate abstraction refinement was first implemented for verifying software by the Slam project [3]. Blast improves on the general scheme in two main ways. First, relevant predicates are discovered locally and independently at each program location as interpolants between the past and the future fragments of a spurious error trace [15]. Second, the discovered new predicates are added and tracked locally only in those parts of an abstract reachability tree where the spurious error trace occurred (*lazy abstraction*) [18]. This