

Jens Grabowski  
Brian Nielsen (Eds.)

LNCS 3395

# Formal Approaches to Software Testing

4th International Workshop, FATES 2004  
Linz, Austria, September 2004  
Revised Selected Papers

TP311.5-53  
F723  
2004

Jens Grabowski Brian Nielsen (Eds.)

# Formal Approaches to Software Testing

4th International Workshop, FATES 2004  
Linz, Austria, September 21, 2004  
Revised Selected Papers



E200500904



Springer

## Volume Editors

Jens Grabowski  
University of Göttingen  
Institute for Informatics  
Lotzestr. 16-18, 37083 Göttingen, Germany  
E-mail: grabowski@informatik.uni-goettingen.de

Brian Nielsen  
Aalborg University  
Department of Computer Science  
Fredrik Bajersvej 7B, 9220 Aalborg, Denmark  
E-mail: bnielsen@cs.auc.dk

Library of Congress Control Number: 2005921470

CR Subject Classification (1998): D.2, D.3, F.3, K.6

ISSN 0302-9743

ISBN 3-540-25109-X Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 11400752 06/3142 5 4 3 2 1 0

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*University of Dortmund, Germany*

Madhu Sudan

*Massachusetts Institute of Technology, MA, USA*

Demetri Terzopoulos

*New York University, NY, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Moshe Y. Vardi

*Rice University, Houston, TX, USA*

Gerhard Weikum

*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

# Lecture Notes in Computer Science

For information about Vols. 1–3315

please contact your bookseller or Springer

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. 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. 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. Vulkov, J. Waśniewski (Eds.), *Numerical Analysis and Its Applications*. XIII, 630 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.), *Advances in 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. 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. 3388: J. Lagergren (Ed.), *Comparative Genomics*. VIII, 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. 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. 2004.

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

Vol. 3365: G. Mauri, G. Päun, 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. Orłowska, 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. 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.
- Vol. 3339: G.I. Webb, X. Yu (Eds.), *AI 2004: Advances in Artificial Intelligence*. XXII, 1272 pages. 2004. (Subseries LNAI).
- Vol. 3338: S.Z. Li, J. Lai, T. Tan, G. Feng, Y. Wang (Eds.), *Advances in Biometric Person Authentication*. XVIII, 699 pages. 2004.
- Vol. 3337: J.M. Barreiro, F. Martin-Sanchez, V. Maojo, F. Sanz (Eds.), *Biological and Medical Data Analysis*. XI, 508 pages. 2004.
- Vol. 3336: D. Karagiannis, U. Reimer (Eds.), *Practical Aspects of Knowledge Management*. X, 523 pages. 2004. (Subseries LNAI).
- Vol. 3335: M. Malek, M. Reitenspieß, J. Kaiser (Eds.), *Service Availability*. X, 213 pages. 2005.
- Vol. 3334: Z. Chen, H. Chen, Q. Miao, Y. Fu, E. Fox, E.-p. Lim (Eds.), *Digital Libraries: International Collaboration and Cross-Fertilization*. XX, 690 pages. 2004.
- Vol. 3333: K. Aizawa, Y. Nakamura, S. Satoh (Eds.), *Advances in Multimedia Information Processing - PCM 2004, Part III*. XXXV, 785 pages. 2004.
- Vol. 3332: K. Aizawa, Y. Nakamura, S. Satoh (Eds.), *Advances in Multimedia Information Processing - PCM 2004, Part II*. XXXVI, 1051 pages. 2004.
- Vol. 3331: K. Aizawa, Y. Nakamura, S. Satoh (Eds.), *Advances in Multimedia Information Processing - PCM 2004, Part I*. XXXVI, 667 pages. 2004.
- Vol. 3330: J. Akiyama, E.T. Baskoro, M. Kano (Eds.), *Combinatorial Geometry and Graph Theory*. VIII, 227 pages. 2005.
- Vol. 3329: P.J. Lee (Ed.), *Advances in Cryptology - ASIACRYPT 2004*. XVI, 546 pages. 2004.
- Vol. 3328: K. Lodaya, M. Mahajan (Eds.), *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science*. XVI, 532 pages. 2004.
- Vol. 3327: Y. Shi, W. Xu, Z. Chen (Eds.), *Data Mining and Knowledge Management*. XIII, 263 pages. 2005. (Subseries LNAI).
- Vol. 3326: A. Sen, N. Das, S.K. Das, B.P. Sinha (Eds.), *Distributed Computing - IWDC 2004*. XIX, 546 pages. 2004.
- Vol. 3325: C.H. Lim, M. Yung (Eds.), *Information Security Applications*. XI, 472 pages. 2005.
- Vol. 3323: G. Antoniou, H. Boley (Eds.), *Rules and Rule Markup Languages for the Semantic Web*. X, 215 pages. 2004.
- Vol. 3322: R. Klette, J. Žunić (Eds.), *Combinatorial Image Analysis*. XII, 760 pages. 2004.
- Vol. 3321: M.J. Maher (Ed.), *Advances in Computer Science - ASIAN 2004. Higher-Level Decision Making*. XII, 510 pages. 2004.
- Vol. 3320: K.-M. Liew, H. Shen, S. See, W. Cai (Eds.), *Parallel and Distributed Computing: Applications and Technologies*. XXIV, 891 pages. 2004.
- Vol. 3319: D. Amyot, A.W. Williams (Eds.), *System Analysis and Modeling*. XII, 301 pages. 2005.
- Vol. 3318: E. Eskin, C. Workman (Eds.), *Regulatory Genomics*. VII, 115 pages. 2005. (Subseries LNBI).
- Vol. 3317: M. Domaratzki, A. Okhotin, K. Salomaa, S. Yu (Eds.), *Implementation and Application of Automata*. XII, 336 pages. 2005.
- Vol. 3316: N.R. Pal, N.K. Kasabov, R.K. Mudi, S. Pal, S.K. Parui (Eds.), *Neural Information Processing*. XXX, 1368 pages. 2004.

# Preface

Testing often accounts for more than 50% of the required effort during system development. The challenge for research is to reduce these costs by providing new methods for the specification and generation of high-quality tests. Experience has shown that the use of formal methods in testing represents a very important means for improving the testing process. Formal methods allow for the analysis and interpretation of models in a rigorous and precise mathematical manner. The use of formal methods is not restricted to system models only. Test models may also be examined. Analyzing system models provides the possibility of generating complete test suites in a systematic and possibly automated manner whereas examining test models allows for the detection of design errors in test suites and their optimization with respect to readability or compilation and execution time. Due to the numerous possibilities for their application, formal methods have become more and more popular in recent years.

The Formal Approaches in Software Testing (FATES) workshop series also benefits from the growing popularity of formal methods. After the workshops in Aalborg (Denmark, 2001), Brno (Czech Republic, 2002) and Montréal (Canada, 2003), FATES 2004 in Linz (Austria) was the fourth workshop of this series. Similar to the workshop in 2003, FATES 2004 was organized in affiliation with the IEEE/ACM Conference on Automated Software Engineering (ASE 2004). FATES 2004 received 41 submissions. Each submission was reviewed by at least three independent reviewers from the Program Committee with the help of some additional reviewers. Based on their evaluations, 14 full papers and one work-in-progress paper from 11 different countries were selected for presentation.

This volume contains revised versions of the presented papers. The revisions reflect the lively discussions among the presenters and participants during the FATES workshop. The papers use different formal methods and languages, e.g., automata, labelled transition systems, TTCN-3 or UPPAAL, and apply them to symbolic test generation, the use of model-checking techniques in testing, the test of nonfunctional properties, and test optimization. This diversity of formal methods and application domains in conjunction with the high number of submissions to and participants of the FATES 2004 workshop emphasize the increased importance attributed to the research on formal approaches in software testing.

We would like to express our gratitude to all authors for their valuable contributions and to the Workshop Organizing Committee of the ASE 2004 conference. In addition, we would like to thank all members of the FATES Program Committee and the additional reviewers, who were given the essential task of reviewing many papers in a short period of time. The individuals who contributed to this effort are listed on the following pages.

December 2004  
Goettingen and Aalborg

Jens Grabowski and Brian Nielsen  
Program Chairs  
FATES 2004

# Organization

## Program Chairs

**Jens Grabowski**  
**Brian Nielsen**

University of Goettingen, Germany  
Aalborg University, Denmark

## Program Committee

<b>Rachel Cardell-Oliver</b>	University of Western Australia, Crawley, Australia
<b>Shing-Chi Cheung</b>	Hong Kong University of Science and Technology, Hong Kong, China
<b>Marie-Claude Gaudel</b>	Université de Paris-Sud, France
<b>Wolfgang Grieskamp</b>	Microsoft Research, USA
<b>Robert M. Hierons</b>	Brunel University, UK
<b>Thierry Jron</b>	IRISA/INRIA, France
<b>David Lee</b>	Bell Labs, Beijing, China
<b>Jose Carlos Maldonado</b>	University of Sao Paulo, Brazil
<b>Manuel Nunez</b>	Universidad Complutense de Madrid, Spain
<b>Jeff Offutt</b>	George Mason University, USA
<b>Alexandre Petrenko</b>	Computer Research Institute of Montréal, Canada
<b>Ina Schieferdecker</b>	Fraunhofer FOKUS, Berlin, Germany
<b>Jan Tretmans</b>	Radboud University, Nijmegen, The Netherlands
<b>Andreas Ulrich</b>	Siemens AG, Corporate Technology, Munich, Germany
<b>Carsten Weise</b>	Ericsson Eurolab Deutschland GmbH, Germany
<b>Clay Williams</b>	IBM Research, Thomas J. Watson Research Center, New York, USA

## Additional Reviewers

<b>Aynur Abdurazik</b>	George Mason University, USA
<b>Roger Alexander</b>	Colorado State University, USA
<b>Ellen Francine Barbosa</b>	University of Sao Paulo, Brazil
<b>Machiel van der Bijl</b>	University of Twente, The Netherlands
<b>Henrik Bohnenkamp</b>	University of Twente, The Netherlands
<b>Sergiy Boroday</b>	CRIM, Canada
<b>Ricky W.K. Chan</b>	University of Hong Kong, Hong Kong, China
<b>Caixia Chi</b>	Bell Labs, China
<b>Lars Frantzen</b>	Radboud University, Nijmegen, The Netherlands
<b>Tim French</b>	University of Western Australia, Australia
<b>David de Frutos-Escrig</b>	Universidad Complutense de Madrid, Spain
<b>Roland Groz</b>	INPG-ENSIMAG, Canada
<b>Yuri Gurevich</b>	Microsoft Research, Redmond, USA
<b>Cedric S.C. Ho</b>	Hong Kong University of Science and Technology, Hong Kong, China
<b>Jin Bei-Hong</b>	Chinese Academy of Sciences, China
<b>Jia Le Huo</b>	CRIM, Canada
<b>Matthew Kaplan</b>	IBM Research, Thomas J. Watson Research Center, New York, USA
<b>Tim Klinge</b>	IBM Research, Thomas J. Watson Research Center, New York, USA
<b>Pieter Koopman</b>	Radboud University, Nijmegen, The Netherlands
<b>Keqin Li</b>	Bell Labs, China
<b>Zhijun Liu</b>	Ohio State University, USA
<b>Marius Mikucionis</b>	Aalborg University, Denmark
<b>Helmut Neukirchen</b>	University of Goettingen, Germany
<b>Vikram Reddy</b>	Ohio State University, USA
<b>Ismael Rodriguez</b>	Universidad Complutense de Madrid, Spain
<b>Fernando Rubio</b>	Universidad Complutense de Madrid, Spain
<b>Guoqiang Shu</b>	Ohio State University, USA
<b>Arne Skou</b>	Aalborg University, Denmark
<b>Tatiana Sugeta</b>	University of Sao Paulo, Brazil
<b>Nikolai Tillmann</b>	Microsoft Research, Redmond, USA
<b>Margus Veanes</b>	Microsoft Research, Redmond, USA
<b>Auri M.R. Vincenzi</b>	University of Sao Paulo, Brazil
<b>Bijendra Vishal</b>	Ohio State University, USA
<b>Frdric Voisin</b>	LRI, Université de Paris-Sud and CNRS, France
<b>Dong Wang</b>	Bell Labs, China
<b>Edith Werner</b>	University of Goettingen, Germany
<b>Tim Willemse</b>	Radboud University Nijmegen, The Netherlands

# Table of Contents

## Symbolic Test Generation

Test Generation Based on Symbolic Specifications <i>Lars Frantzen, Jan Tretmans, Tim A.C. Willemse</i> .....	1
Symbolic Test Case Generation for Primitive Recursive Functions <i>Achim D. Brucker, Burkhart Wolff</i> .....	16
Preserving Contexts for Soft Conformance Relation <i>David de Frutos Escrig, Carlos Gregorio Rodríguez</i> .....	33

## Testing Non-functional Properties

Testing of Symbolic-Probabilistic Systems <i>Natalia López, Manuel Núñez, Ismael Rodríguez</i> .....	49
A Test Generation Framework for <i>quiescent</i> Real-Time Systems <i>Laura Brandán Briones, Ed Brinksma</i> .....	64
Online Testing of Real-Time Systems Using UPPAAL <i>Kim G. Larsen, Marius Mikucionis, Brian Nielsen</i> .....	79
Testing Deadlock-Freeness in Real-Time Systems; A Formal Approach <i>Behzad Bordbar, Kozo Okano</i> .....	95

## Test Development with Model Checking Techniques

Using Model Checking for Reducing the Cost of Test Generation <i>Hyoung Seok Hong, Hasan Ural</i> .....	110
Specifying and Generating Test Cases Using Observer Automata <i>Johan Blom, Anders Hessel, Bengt Jonsson, Paul Pettersson</i> .....	125
Semi-formal Development of a Fault-Tolerant Leader Election Protocol in Erlang <i>Thomas Arts, Koen Claessen, Hans Svensson</i> .....	140
An Automata-Theoretic Approach for Model-Checking Systems with Unspecified Components <i>Gaoyan Xie, Zhe Dang</i> .....	155

Test Optimization

Test Patterns with TTCN-3  
    *Alain Vouffo-Feudjio, Ina Schieferdecker* ..... 170

High-Level Restructuring of TTCN-3 Test Data  
    *Antal Wu-Hen-Chang, Dung Le Viet, Gabor Batori, Roland Gecse, Gyula Csopaki* ..... 180

Ordering Mutants to Minimise Test Effort in Mutation Testing  
    *Kalpesh Kapoor, Jonathan P. Bowen* ..... 195

Testing COM Components Using Software Fault Injection and Mutation Analysis, and Its Empirical Study  
    *Hoijin Yoon, Eunhee Kim, Joo Young Seo, Byoungju Choi* ..... 210

Author Index ..... 225

# Test Generation Based on Symbolic Specifications

Lars Frantzen\*, Jan Tretmans, and Tim A.C. Willemse\*\*

Nijmegen Institute for Computing and Information Sciences (NIII),  
Radboud University Nijmegen – The Netherlands  
{lf, tretmans, timw}@cs.ru.nl

**Abstract.** Classical state-oriented testing approaches are based on simple machine models such as Labelled Transition Systems (LTSs), in which data is represented by concrete values. To implement these theories, data types which have infinite universes have to be cut down to finite variants, which are subsequently enumerated to fit in the model. This leads to an explosion of the state space. Moreover, exploiting the syntactical and/or semantical information of the involved data types is non-trivial after enumeration. To overcome these problems, we lift the family of testing relations  $\text{ioco}_{\mathcal{F}}$  to the level of Symbolic Transition Systems (STSs). We present an algorithm based on STSs, which generates and executes tests on-the-fly on a given system. It is sound and complete for the  $\text{ioco}_{\mathcal{F}}$  testing relations.

## 1 Introduction

Testing is an important technique to assess the quality of systems. In testing, experiments are conducted with a System Under Test (SUT) to determine whether it behaves as expected. There are many different kinds of testing. We focus on formal, specification based, black box, functionality testing. This basically means that the SUT can only be observed (and controlled) via its external interfaces. Moreover, a mathematical, unambiguous specification of the causal order between (appropriate) inputs and expected outputs of the SUT is the starting point for the generation and the analysis of the test results.

Several (formal) test generation tools have been developed for specification based, black box testing. Most of these tools use (variations of) state machines or transition systems as the underlying model for test generation. We refer to these types of tools as *state oriented* tools. For an overview of such tools see [2]. A problem, often encountered in such tools is the *state space explosion*, which is

---

\* Lars Frantzen is supported by the Netherlands Organisation for Scientific Research (NWO) under project: STRESS – Systematic Testing of Realtime Embedded Software Systems.

\*\* Tim Willemse carried out this work as part of the TANGRAM project under the responsibility of the Embedded Systems Institute. Tangram is partially supported by the Netherlands Ministry of Economic Affairs under grant TSIT2026.

due to the fact that they use an explicit internal representation for the states of the specification. This is particularly true when the specification uses complex data structures with large or infinite data domains, because each value in the data domain potentially leads to another state. Consequently, many tools can only cope with very restricted data structures with finite domains.

Opposed to state oriented tools are *data type oriented* tools, which are tools tailored to deal with test generation for complicated data structures, such as QUICKCHECK [3] and GAST [5]. These tools employ the structure of data types to generate test data. However, they lack a built-in concept of state, which makes them less suited to test, e.g., concurrent systems. The way to handle state in such tools is to explicitly define a data structure that represents a state space, but this is not always satisfactory.

The combination of the state oriented and the data type oriented approaches looks promising, and it is exactly this what we investigate in this paper. As our basis we take a state oriented approach to testing, viz. the **ioco** test theory [8]. To the underlying model of Labelled Transition Systems, we add the concept of location variables, and the concept of data, which can be communicated over gates. Both influence the flow of control, thereby allowing us to specify data-dependent behaviour. We refer to these augmented Labelled Transition Systems as *Symbolic Transition Systems* (STSs). We subsequently lift the **ioco** test theory to STSs. As a result, we obtain a sound and complete test derivation algorithm from specifications expressed as STSs.

The test derivation algorithm for STSs allows to treat data symbolically. Rather than elaborating our approach for a specific data formalism, data types are treated as sets of values (algebras) and first order formulas are used to specify values or predicates. This allows to combine STSs with any formalism of choice (with corresponding test tools) for the specification and manipulation of data. This is further elaborated into a tractable algorithm.

From a theoretical point of view, it is also interesting to give an algorithm which generates *symbolic test cases* (STCs). This requires a purely symbolic version of the **ioco** <sub>$\mathcal{F}$</sub>  relations. This is depicted in Fig. 1. The front triangle

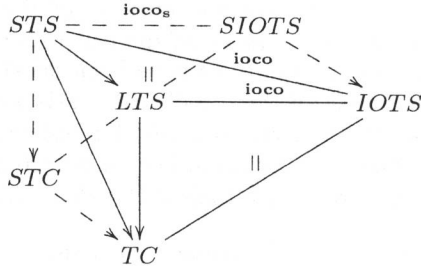


Fig. 1. Classical **ioco** test theory and symbolic **ioco** test theory

represents the classical **ioco** test theory, as presented in [8]. Test cases ( $TC$ ) are generated out of a specification LTS, and subsequently executed ( $||$ ) on an SUT, assumed to be modelled by an IOTS. The rear triangle consists of a purely symbolic test theory. In this paper, we concentrate on the relation between STSs, LTSs and IOTSs, and on the generation and execution of test cases, i.e. the relation between STSs and TCs. Elaborating on the dashed lines and the corresponding models is another line of research we are pursuing.

*Related Work.* The idea of combining data type oriented and state oriented approaches is not entirely new in testing. We mention a few noteworthy approaches.

The approach which comes closest to ours is the one described in [7]. There, Input-Output Symbolic Transition Systems (IOSTSs) are used, which are very similar to our STSs. The conformance relation they use corresponds to  $\mathbf{iocnf} = \mathbf{ioco}_{traces(\mathcal{L})}$ , but they do not deal with quiescence. In [7] test purposes are chosen as a way to tackle the state space explosion problem. These are used to compute a subgraph of the IOSTS representing a specific issue of interest. Such test purposes are again (special) IOSTSs. The result is a test case which is still symbolic in the sense that it is a deterministic IOSTS with special states *Pass*, *Fail* and *Inconclusive*. The verdict *Inconclusive* is necessary to judge a behaviour which conforms to a given specification, but does not satisfy the given test purpose. Our approach does not rely on test purposes, even though the set  $\mathcal{F}$  which identifies the relation  $\mathbf{ioco}_{\mathcal{F}}$  can be seen as some form of test purpose.

The data-type oriented GAST tool [5] was recently extended in [6] to deal with specifications given as (possibly nondeterministic) Extended Finite State Machines (EFSMs). Such EFSMs are also symbolic specifications, but in some senses more restrictive than STSs or IOSTSs. GAST basically implements a generic algorithm to enumerate the elements of an arbitrary algebraic data type. Such a type can be an input value, but also a whole path through the EFSM. Since the list of all elements of a recursive type is infinitely long, lazy evaluation is employed to generate only the fraction of this list that is actually needed. The elements are generated in increasing size, both the executed paths and the input values. GAST can be used to execute the generated tests on an SUT in an on-the-fly manner.

*Overview.* This paper is structured as follows. In Sect. 2 we briefly repeat notions from first order logic. The **ioco** test theory is summarised in Sect. 3. The framework of Symbolic Transition Systems is introduced in Sect. 4. We present an on-the-fly implementation for generating and executing test cases for Symbolic Transition Systems in Sect. 5. We finish with conclusions and future extensions in Sect. 6.

## 2 First Order Logic

We use basic concepts from first order logic as our framework for dealing with data. For a general introduction into logic we refer to [4]. From hereon we assume a first order structure as given, i.e.:

- A logical signature  $\mathfrak{S} = (F, P)$  with
  - $F$  is a set of *function symbols*. Each  $f \in F$  has a corresponding arity  $n \in \mathbb{N}$ . If  $n = 0$  we call  $f$  a *constant*.
  - $P$  is a set of *predicate symbols*. Each  $p \in P$  has a corresponding arity  $n > 0$ .
- A model  $\mathfrak{M} = (\mathcal{U}, (f_{\mathfrak{M}})_{f \in F}, (p_{\mathfrak{M}})_{p \in P})$  with
  - $\mathcal{U}$  being a nonempty set called *universe*.
  - For all  $f \in F$  with arity  $n$ ,  $f_{\mathfrak{M}}$  is a function of type  $\mathcal{U}^n \rightarrow \mathcal{U}$ .
  - For every  $p \in P$  with arity  $n$  we have  $p_{\mathfrak{M}} \subseteq \mathcal{U}^n$ .

For simplicity, and without loss of generality we restrict to one-sorted signatures. Let  $\mathfrak{X}$  be a set of *variables*. *Terms* over  $X$ , denoted  $\mathfrak{T}(X)$ , are built from function symbols  $F$  and variables  $X \subseteq \mathfrak{X}$ . We write  $\text{var}(t)$  to denote the set of variables appearing in a term  $t$ . Terms  $t \in \mathfrak{T}(\emptyset)$  are called *ground terms*.

*Example 1.* Assume we have  $X = \{x, y\}$ . Let  $\mathfrak{S} = (F, P)$  be given by  $F = \{\text{zero}, \text{succ}, \text{add}\}$  (with arities 0, 1 and 2, resp.), and  $P = \{\text{leq}\}$  (with arity 2). An obvious model for this signature is the natural numbers with 0, *successor*, *addition* and the less-or-equal predicate; any other model that sticks to the given arities is fine too. Terms are, e.g.  $x$ ,  $\text{succ}(x)$  and  $\text{add}(\text{succ}(x), y)$ . Ground terms are, e.g.  $\text{zero}$  and  $\text{add}(\text{zero}, \text{succ}(\text{zero}))$ .  $\square$

A *term-mapping* is a function  $\sigma: \mathfrak{X} \rightarrow \mathfrak{T}(\mathfrak{X})$ . The term-mapping id, referred to as the *identity mapping*, is defined as  $\text{id}(x) = x$  for all  $x \in \mathfrak{X}$ . We use the following notation. For sets  $X, Y$  with  $X \cup Y \subseteq \mathfrak{X}$ , we write  $\mathfrak{T}(Y)^X$  for the set of term-mappings that assign to each variable  $x \in X$  a term  $t \in \mathfrak{T}(Y)$ , and to each variable  $x \notin X$  the term  $x$ . Given a term-mapping  $\sigma \in \mathfrak{T}(Y)^X$  we overload the  $\text{var}$ -notation as follows:  $\text{var}(\sigma) =_{\text{def}} \bigcup_{x \in X} \text{var}(\sigma(x))$ .

The set of free variables of a first order formula  $\varphi$  is denoted  $\text{free}(\varphi)$ ; the set of bound variables is denoted  $\text{bound}(\varphi)$ . The set of first order formulas  $\varphi$  over  $X \subseteq \mathfrak{X}$  is denoted  $\mathfrak{F}(X)$ ; we have  $\text{free}(\varphi) \cup \text{bound}(\varphi) \subseteq X$ . A tautology is represented by  $\top$ . The *existential closure* of a formula  $\varphi$ , denoted  $\exists\varphi$ , is defined as  $\exists\varphi =_{\text{def}} \exists x_1 \exists x_2 \dots \exists x_n : \varphi$  with  $\{x_1, \dots, x_n\} = \text{free}(\varphi)$ .

Given a term-mapping  $\sigma$  and a formula  $\varphi$ , the *substitution* of  $\sigma(x)$  for  $x \in \text{free}(\varphi)$  in  $\varphi$  is denoted  $\varphi[\sigma]$ . Substitutions are side-effect free, i.e. they do not add bound variables. This is achieved using  $\alpha$ -renaming. The substitution of terms  $\sigma(x)$  for variables  $x \in \text{var}(t)$ , in a term  $t$  using a term-mapping  $\sigma$ , is denoted  $t[\sigma]$ .

*Example 2.* An example of a term mapping for  $X = \{x, y\}$  is  $\sigma = \{x \mapsto \text{succ}(y), y \mapsto \text{zero}\} \in \mathfrak{T}(X)^X$ , with  $\text{var}(\sigma) = \{y\}$ . The existential closure of the formula  $\varphi = \forall y : \text{leq}(x, y)$  with  $\text{bound}(\varphi) = \{y\}$  and  $\text{free}(\varphi) = \{x\}$  is  $\exists\varphi = \exists x \forall y : \text{leq}(x, y)$ . The substitution of  $\sigma$  in  $\varphi$  is not side-effect free, but can be achieved by renaming variable  $y$  to  $z$ , i.e.  $\varphi[\sigma] = \forall z : \text{leq}(\text{succ}(y), z)$ .  $\square$

A *valuation*  $\vartheta$  is a function  $\vartheta: \mathfrak{X} \rightarrow \mathcal{U}$ . We denote the set of all valuations as  $\mathcal{U}^{\mathfrak{X}} =_{\text{def}} \{\vartheta: \mathfrak{X} \rightarrow \mathcal{U} \mid \vartheta \text{ is a valuation of } \mathfrak{X}\}$ . For a given  $X \subseteq \mathfrak{X}$  we write  $\vartheta \in \mathcal{U}^X$  when only the values of the variables in  $X$  are of interest. For all the other variables  $y \in \mathfrak{X} \setminus X$  we set  $\vartheta(y) = *$ , where  $*$  is an arbitrary element of set  $\mathcal{U}$ .

Having two valuations  $\vartheta \in \mathcal{U}^X$  and  $\varsigma \in \mathcal{U}^Y$  with  $X \cap Y = \emptyset$ , their union is defined as:

$$(\vartheta \cup \varsigma)(x) =_{\text{def}} \begin{cases} \vartheta(x) & \text{if } x \in X \\ \varsigma(x) & \text{if } x \in Y \\ * & \text{otherwise} \end{cases}$$

The *satisfaction* of a formula  $\varphi$  w.r.t. a given valuation  $\vartheta$  is denoted  $\vartheta \models \varphi$ . When  $\text{free}(\varphi) = \emptyset$  we write  $\mathfrak{M} \models \psi$  because the satisfaction is independent of a concrete valuation.

The extension to evaluate whole terms based on a valuation  $\vartheta$  is called a *term-evaluation* and denoted  $\vartheta_{\text{eval}}: \mathfrak{T}(\mathfrak{X}) \rightarrow \mathcal{U}$ . The evaluation of ground terms is denoted  $\text{eval}: \mathfrak{T}(\emptyset) \rightarrow \mathcal{U}$ .

To ease notation, we often treat a tuple  $\langle x_1, \dots, x_n \rangle \in A_1 \times \dots \times A_n$  as the set  $\{x_1, \dots, x_n\}$ . We denote the composition of functions  $f: B \rightarrow C$  and  $g: A \rightarrow B$  as  $f \circ g$ .

*Example 3.* Assuming the standard model for natural numbers as given in example 1, an example valuation is  $\vartheta = \{x \mapsto 24, y \mapsto 7\} \in \mathcal{U}^{\{x, y\}}$ . For the formula  $\varphi$  of example 2, the valuation  $\vartheta$  and the standard model for natural numbers we find  $\vartheta \not\models \varphi$  and  $\mathfrak{M} \models \exists \varphi$  and we get  $\vartheta_{\text{eval}}(\text{add}(x, \text{succ}(y))) = 32$ .  $\square$

Our example of a logical structure for natural numbers shows that many, even infinite ground terms may evaluate to the same value, e.g. the ground terms **zero** and  $\text{add}(\text{zero}, \text{zero})$  both evaluate to 0. We assume we have a unique ground term representative for every value to facilitate the bidirectional translation.

### 3 Testing Labelled Transition Systems

We briefly review the **ioco<sub>F</sub>** test theory on which this paper is based. For a more detailed overview, we refer to [8]. The semantical model we use to model reactive systems is based on *Labelled Transition Systems* (LTSs).

**Definition 1.** A Labelled Transition System is a tuple  $\mathcal{L} = \langle S, s_0, \Sigma, \rightarrow \rangle$ , where

- $S$  is a (possibly infinite) set of states.
- $s_0 \in S$  is the initial state.
- $\Sigma$  is a (possibly infinite) set of action labels. The special action label  $\tau \notin \Sigma$  denotes an unobservable action. In contrast, all other actions are observable. We write  $\Sigma_\tau$  to denote the set  $\Sigma \cup \{\tau\}$ .
- $\rightarrow \subseteq S \times \Sigma_\tau \times S$  is the transition relation. When  $(s, \mu, s') \in \rightarrow$  we write  $s \xrightarrow{\mu} s'$ .

We often identify an LTS  $\mathcal{L}$  with its initial state  $s_0$ .

Unobservable actions can be used to model events that cannot be seen by an observer of a system. The generalised transition relation  $\Longrightarrow \subseteq S \times \Sigma^* \times S$  captures this phenomenon: it abstracts from  $\tau$  actions preceding, in-between and following a (possibly empty) sequence of observable actions. Given an LTS

**Table 1.** Deduction rules for generalised transitions

$s \xRightarrow{\epsilon} s$	$\frac{s \xRightarrow{\sigma} s'' \quad s'' \xrightarrow{\tau} s'}{s \xRightarrow{\sigma} s'}$	$\frac{s \xRightarrow{\sigma} s'' \quad s'' \xrightarrow{\mu} s' \quad \mu \neq \tau}{s \xRightarrow{\sigma\mu} s'}$
------------------------------	--	--

$\mathcal{L} = \langle S, s_0, \Sigma, \rightarrow \rangle$ , this relation is defined by the deduction rules of Table 1. We define two operations on LTSs. Given an LTS  $\mathcal{L} = \langle S, s_0, \Sigma, \rightarrow \rangle$  and a (possibly new) action  $\mu$ . The *action prefix*  $\mu; \mathcal{L}$  is defined as

$$\mu; \mathcal{L} =_{\text{def}} \langle S \cup \{s\}, s, \Sigma \cup \{\mu\}, \rightarrow \cup \{s \xrightarrow{\mu} s_0\} \rangle \quad (1)$$

with  $s \notin S$  being a fresh state. For a set of LTSs  $\bar{\mathcal{L}} = \{\mathcal{L}_1, \dots, \mathcal{L}_n\}$  with  $n \geq 0$  of the form  $\mathcal{L}_i = \langle S_i, s_{0i}, \Sigma_i, \rightarrow_i \rangle$ , we define the *alternative composition* of all LTSs  $\mathcal{L}_i$ , denoted  $\sum(\bar{\mathcal{L}})$ , as follows:

$$\sum(\bar{\mathcal{L}}) =_{\text{def}} \langle \bigcup_{i \leq n} S_i \cup \{s\}, s, \bigcup_{i \leq n} \Sigma_i, \bigcup_{i \leq n} (\rightarrow_i \cup \{s \xrightarrow{\mu} s' \mid s_{0i} \xrightarrow{\mu} s'\}) \rangle \quad (2)$$

with  $s \notin \bigcup_{i \leq n} S_i$  being a fresh state. The operator  $\sum$  is associative and commutative. We sometimes write  $\mathcal{L}_1 + \mathcal{L}_2$  instead of  $\sum\{\mathcal{L}_1, \mathcal{L}_2\}$ .

### 3.1 The Test Relation $\text{ioco}_{\mathcal{T}}$

We introduce the following shorthand notation. For a  $\mu \in \Sigma_{\tau}$  we write  $s \xrightarrow{\mu}$  when there is a state  $s'$  such that  $s \xrightarrow{\mu} s'$ , and, likewise, given a  $\sigma \in \Sigma^*$  we write  $s \xRightarrow{\sigma}$  when there is a state  $s'$  such that  $s \xRightarrow{\sigma} s'$ .

**Definition 2.** Let  $\mathcal{L} = \langle S, s_0, \Sigma, \rightarrow \rangle$  be an LTS and let  $s \in S$ .

1.  $\text{init}(s) =_{\text{def}} \{ \mu \in \Sigma_{\tau} \mid s \xrightarrow{\mu} \}$ .
2.  $\text{traces}(s) =_{\text{def}} \{ \sigma \in \Sigma^* \mid s \xRightarrow{\sigma} \}$ .
3.  $\mathcal{L}$  has finite behaviour if all  $\sigma \in \text{traces}(s_0)$  satisfy  $|\sigma| < n$  for some  $n \in \mathbb{N}$ .
4.  $\mathcal{L}$  is deterministic if for all  $\sigma \in \Sigma^*$ ,  $|\{s' \mid s_0 \xRightarrow{\sigma} s'\}| \leq 1$ .

We assume that implementations of a reactive system can be given as an *input-output transition system* (IOTSs). An IOTS is an LTS in which the set of action labels  $\Sigma$  is partitioned in a set of *input actions*  $\Sigma_I$  and a set of *output actions*  $\Sigma_U$ , and for which it is assumed that all input actions are enabled in all states.

**Definition 3.** Let  $\mathcal{L} = \langle S, s_0, \Sigma_I \cup \Sigma_U, \rightarrow \rangle$  be an LTS. A state  $s \in S$  is *quiescent*, denoted by  $\delta(s)$ , if  $\forall \mu \in \Sigma_U \cup \{\tau\} : s \not\xrightarrow{\mu}$ .

Let  $\delta$  be a special action label, not part of any action label set. For a given set of action labels  $\Sigma$ , we abbreviate  $\Sigma \cup \{\delta\}$  with  $\Sigma_{\delta}$ . The suspension transitions  $\Rightarrow_{\delta} \subseteq S \times \Sigma_{\delta}^* \times S$  are given by the deduction rules of Table 2. The set of all *suspension traces* of  $\mathcal{L}$  is denoted  $\text{Straces}(\mathcal{L}) = \{\sigma \in \Sigma_{\delta}^* \mid \mathcal{L} \xRightarrow{\sigma} \delta\}$ .