Farhad Arbab
Marjan Sirjani (Eds.)

LNCS 4767

International Symposium on Fundamentals of Software Engineering

International Symposium, FSEN 2007
Tehran, Iran, April 2007
Proceedings

Springer

Farhad Arbab Marjan Sirjani (Eds.)

International Symposium on Fundamentals of Software Engineering

International Symposium, FSEN 2007
Tehran, Iran, April 17-19, 2007
Proceedings

Springer

Volume Editors

Farhad Arbab
CWI, Leiden University
Kruislaan 413, Amsterdam, The Netherlands
E-mail: Farhad.Arbab@cwi.nl

Marjan Sirjani
University of Tehran, IPM
North Karegar Ave., Tehran, Iran
E-mail: msirjani@ut.ac.ir

Library of Congress Control Number: 2007936606

CR Subject Classification (1998): D.2, D.2.4, F.4.1, D.2.2

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-75697-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-75697-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12175260 06/3180 5 4 3 2 1 0

Preface

The present volume contains the post-proceedings of the second IPM International Symposium on Fundamentals of Software Engineering (FSEN), Tehran, Iran, April 17–19, 2007. This event, FSEN 2007, was organized by the School of Computer Science at the Institute for Studies in Fundamental Sciences (IPM) in Iran, in cooperation with the ACM SIGSOFT and IFIP WG 2.2, and was additionally supported by the University of Tehran, Sharif University of Technology, and the International Scientific Meetings Office (ISMO). This symposium brought together researchers and practitioners working on different aspects of formal methods in software engineering. FSEN 2007 covered many aspects of formal methods, especially those related to advancing the application of formal methods in the software industry and promoting their integration with practical engineering techniques.

A truly international program committee of top researchers from 23 different academic institutes in 9 countries selected the technical content of this symposium. We received a total of 73 submissions, out of which the PC selected 22 as regular papers and 8 as short papers to be published in the post-proceedings, and 6 papers accepted for poster presentations at the symposium. Each submission was reviewed by at least 3 independent referees, for its quality, originality, contribution, clarity of presentation, and its relevance to the symposium topics. We had 93 registered participants at the symposium from 12 countries.

We had 4 distinguished keynote speakers at FSEN 2007: James C. Browne, University of Texas at Austin, Texas, USA, on *Unification of Verification and Validation Methods for Software*; Masahiro Fujita, University of Tokyo, Japan, on *Hardware-Software Co-design for SoC with Separated Verification Between Computation and Communication;* Davide Sangiorgi, University of Bologna, Italy, on *Bisimulation in Higher-Order Languages*; and Peter D. Mosses, Swansea University, Wales, UK, on *Fundamentals of Semantics Engineering.*

In conjunction with FSEN 2007, the Working Group 2.2 of IFIP organized 2 full-day tutorials by internationally recognized researchers on the semantics of programming languages (Peter D. Mosses), and the semantics of concurrency (Davide Sangiorgi). These well-attended and well-received tutorials strengthened the impact of FSEN 2007, and we would like to take this opportunity to express our appreciation for the contribution of IFIP WG 2.2 and the tutorial speakers.

We are grateful for the support and the assistance of the IPM staff in the organization of this symposium, especially Dr. Larijani, Dr. Sarbazi-Azad, Ms. Arfai and Mr. Shahrabi. We thank the members of the program committee for their time, effort, and contributions to making FSEN 2007 a quality symposium. Last but not least, our thanks go to our authors and workshop partici-

pants, without whose submissions and participation FSEN 2007 would not have materialized.

April 2007

Farhad Arbab
Marjan Sirjani

Organization

Program Chairs

Farhad Arbab, CWI, Netherlands; Leiden University, Netherlands; University of Waterloo, Canada
Marjan Sirjani, University of Tehran, Iran; IPM, Iran

Program Committee

Gul Agha, University of Illinois at Urbana - Champaign, USA
Farhad Arbab, CWI, Netherlands; Leiden University, Netherlands; University of Waterloo, Canada
Mohammad Ardeshir, Sharif University of Technology, Iran
Christel Baier, University of Bonn, Germany
Frank S. de Boer, CWI, Netherlands; Leiden University, Netherlands
Marcello Bonsangue, Leiden University, Netherlands
Mario Bravetti, University of Bologna
James C. Browne, University of Texas at Austin, USA
Michael Butler, University of Southampton, UK
Nancy Day, University of Waterloo, Canada
Masahiro Fujita, University of Tokyo, Japan
Maurizio Gabbrielli, University of Bologna, Italy
Jan Friso Groote, Technical University of Eindhoven, Netherlands
Radu Grosu, State University of New York at Stony Brook, USA
Michael Huth, Imperial College of London, UK
Joost Kok, Leiden University, Netherlands
Marta Kwiatkowska, University of Birmingham, UK
Mohammad Reza Meybodi, AmirKabir University of Technology, Iran
Seyed-Hassan Mirian-Hosseinabadi, Sharif University of Technology, Iran
Ugo Montanari, University of Pisa, Italy
Mohammad Reza Mousavi, Technical University of Eindhoven, Netherlands
Ali Movaghar, IPM, Iran; Sharif University of Technology, Iran
Andrea Omicini, University of Bologna, Italy
George Papadopoulos, University of Cyprus, Cyprus
Jan Rutten, CWI, Netherlands; Vrije University Amsterdam, Netherlands
Sandeep Shukla, Virginia Tech, USA
Marjan Sirjani, IPM, Iran; University of Tehran, Iran
Carolyn Talcott, SRI International, USA

Local Organization

Hamidreza Shahrabi, IPM, Iran (Chair)
Samira Tasharofi, IPM, Iran; University of Tehran, Iran
Hossein Hojjat, IPM, Iran; University of Tehran, Iran

Referees

Sumit Ahuja
Elisabeth Ball
Simonetta Balsamo
Massimo Bartoletti
Debayan Bhaduri
Armin Biere
Stefano Bistarelli
Frank S. de Boer
Benedikt Bollig
Maria Paola Bonacina
Pascal Bouvry
Pyrros Bratskas
Mario Bravetti
Adam Brown
James Browne
Michael Butler
Marco Carbone
Liping Chen
Pericles Leng Cheng
Tom Chothia
Dave Clarke
Claudio Sacerdoti Cohen
John Colley
David Costa
Pieter Cuijpers
Marco Danelutto
Nancy Day
Wan Fokkink
Matthias Fruth
Fatemeh Ghassemi
Vittorio Ghini
Cinzia di Giusto
Jan Friso Groote
Hossein Hojjat
Hamed Iravanchi
Mohammad Izadi
Mohammad-Mahdi Jaghouri
Kevin Kane
Stephanie Kemper
Ramtin Khosravi
Minyoung Kim
Alexander Knapp
Christian Kohler
Adam Koprowski
Marcel Kyas
Timo Latvala
Moreno Marzolla
Deepak Abraham Mathaikutty
Kirill Mechitov
Seyyed Hassan Mirian
Mohammad-Reza Mousavi
Ali Movaghar
Gethin Norman
Farhad Oroumchian
Karel Van Oudheusden
David Parker
Nearchos Paspallis
Hiren Patel
Bas Ploeger
Jaco van de Pol
Jose Proenca
Niloofar Razavi
Michel Reniers
Abdolbaghi Rezazadeh
Shamim Ripon
Jan Rutten
Werner Sandmann
Laura Semini
Gaurav Singh
Marjan Sirjani
Colin Snook
Jeremy Sproston
Andres Stam
Martin Steffen
Mark-Oliver Stehr
Syed Suhaib
Meng Sun
Sameer Sundresh
Paolo Tacchella
Edward Turner
Aimilia Tzanavari
Yaroslav Usenko
Daniele Veracca
Fons Verbeek
Eric Verbeek
Erik de Vink
Dimitrios Vogiatzis
Marc Voorhoeve
Michael Weber
Muck van Weerdenburg
Marco Wiering
Hans Zantema

Sponsoring Institutions

ACM Special Interest Group on Software Engineering (SIGSOFT)
International Federation for Information Processing (IFIP WG 2.2)

International Scientific Meetings Office (ISMO)
Iran Telecommunications Research Center (ITRC)
Hi-Tech Industries Center of Iran
Electronic Computing Machine Service Company

University of Tehran
Sharif University of Technology
Centrum voor Wiskunde en Informatica (Center for Mathematics and Computer Science - CWI)

Lecture Notes in Computer Science 4767

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

Editorial Board

Lecture Notes in Computer Science

Sublibrary 2: Programming and Software Engineering

For information about Vols. 1– 4143
please contact your bookseller or Springer

Vol. 4767: F. Arbab, M. Sirjani (Eds.), International Symposium on Fundamentals of Software Engineering. XIII, 450 pages. 2007.

Vol. 4764: P. Abrahamsson, N. Baddoo, T. Margaria, R. Messnarz (Eds.), Software Process Improvement. XI, 225 pages. 2007.

Vol. 4758: F. Oquendo (Ed.), Software Architecture. XVI, 340 pages. 2007.

Vol. 4757: F. Cappello, T. Herault, J.J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XVI, 396 pages. 2007.

Vol. 4753: E. Duval, R. Klamma, M. Wolpers (Eds.), Creating New Learning Experiences on a Global Scale. XII, 518 pages. 2007.

Vol. 4749: B.J. Krämer, K.-J. Lin, P. Narasimhan (Eds.), Service-Oriented Computing – ICSOC 2007. XIX, 629 pages. 2007.

Vol. 4748: K. Wolter (Ed.), Formal Methods and Stochastic Models for Performance Evaluation. X, 301 pages. 2007.

Vol. 4741: C. Bessière (Ed.), Principles and Practice of Constraint Programming – CP 2007. XV, 890 pages. 2007.

Vol. 4735: G. Engels, B. Opdyke, D.C. Schmidt, F. Weil (Eds.), Model Driven Engineering Languages and Systems. XV, 698 pages. 2007.

Vol. 4716: B. Meyer, M. Joseph (Eds.), Software Engineering Approaches for Offshore and Outsourced Development. X, 201 pages. 2007.

Vol. 4680: F. Saglietti, N. Oster (Eds.), Computer Safety, Reliability, and Security. XV, 548 pages. 2007.

Vol. 4670: V. Dahl, I. Niemelä (Eds.), Logic Programming. XII, 470 pages. 2007.

Vol. 4652: D. Georgakopoulos, N. Ritter, B. Benatallah, C. Zirpins, G. Feuerlicht, M. Schoenherr, H.R. Motahari-Nezhad (Eds.), Service-Oriented Computing ICSOC 2006. XVI, 201 pages. 2007.

Vol. 4634: H. Riis Nielson, G. Filé (Eds.), Static Analysis. XI, 469 pages. 2007.

Vol. 4615: R. de Lemos, C. Gacek, A. Romanovsky (Eds.), Architecting Dependable Systems IV. XIV, 435 pages. 2007.

Vol. 4610: B. Xiao, L.T. Yang, J. Ma, C. Muller-Schloer, Y. Hua (Eds.), Autonomic and Trusted Computing. XVIII, 571 pages. 2007.

Vol. 4609: E. Ernst (Ed.), ECOOP 2007 – Object-Oriented Programming. XIII, 625 pages. 2007.

Vol. 4608: H.W. Schmidt, I. Crnković, G.T. Heineman, J.A. Stafford (Eds.), Component-Based Software Engineering. XII, 283 pages. 2007.

Vol. 4591: J. Davies, J. Gibbons (Eds.), Integrated Formal Methods. IX, 660 pages. 2007.

Vol. 4589: J. Münch, P. Abrahamsson (Eds.), Product-Focused Software Process Improvement. XII, 414 pages. 2007.

Vol. 4574: J. Derrick, J. Vain (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2007. XI, 375 pages. 2007.

Vol. 4556: C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction, Part III. XXII, 1020 pages. 2007.

Vol. 4555: C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction, Part II. XXII, 1066 pages. 2007.

Vol. 4554: C. Stephanidis (Ed.), Universal Acess in Human Computer Interaction, Part I. XXII, 1054 pages. 2007.

Vol. 4553: J.A. Jacko (Ed.), Human-Computer Interaction, Part IV. XXIV, 1225 pages. 2007.

Vol. 4552: J.A. Jacko (Ed.), Human-Computer Interaction, Part III. XXI, 1038 pages. 2007.

Vol. 4551: J.A. Jacko (Ed.), Human-Computer Interaction, Part II. XXIII, 1253 pages. 2007.

Vol. 4550: J.A. Jacko (Ed.), Human-Computer Interaction, Part I. XXIII, 1240 pages. 2007.

Vol. 4542: P. Sawyer, B. Paech, P. Heymans (Eds.), Requirements Engineering: Foundation for Software Quality. IX, 384 pages. 2007.

Vol. 4536: G. Concas, E. Damiani, M. Scotto, G. Succi (Eds.), Agile Processes in Software Engineering and Extreme Programming. XV, 276 pages. 2007.

Vol. 4530: D.H. Akehurst, R. Vogel, R.F. Paige (Eds.), Model Driven Architecture - Foundations and Applications. X, 219 pages. 2007.

Vol. 4523: Y.-H. Lee, H.-N. Kim, J. Kim, Y.W. Park, L.T. Yang, S.W. Kim (Eds.), Embedded Software and Systems. XIX, 829 pages. 2007.

Vol. 4498: N. Abdennahder, F. Kordon (Eds.), Reliable Software Technologies - Ada-Europe 2007. XII, 247 pages. 2007.

Vol. 4486: M. Bernardo, J. Hillston (Eds.), Formal Methods for Performance Evaluation. VII, 469 pages. 2007.

Vol. 4470: Q. Wang, D. Pfahl, D.M. Raffo (Eds.), Software Process Dynamics and Agility. XI, 346 pages. 2007.

Vol. 4468: M.M. Bonsangue, E.B. Johnsen (Eds.), Formal Methods for Open Object-Based Distributed Systems. X, 317 pages. 2007.

Vol. 4467: A.L. Murphy, J. Vitek (Eds.), Coordination Models and Languages. X, 325 pages. 2007.

Vol. 4454: Y. Gurevich, B. Meyer (Eds.), Tests and Proofs. IX, 217 pages. 2007.

Vol. 4444: T. Reps, M. Sagiv, J. Bauer (Eds.), Program Analysis and Compilation, Theory and Practice. X, 361 pages. 2007.

Vol. 4440: B. Liblit, Cooperative Bug Isolation. XV, 101 pages. 2007.

Vol. 4408: R. Choren, A. Garcia, H. Giese, H.-f. Leung, C. Lucena, A. Romanovsky (Eds.), Software Engineering for Multi-Agent Systems V. XII, 233 pages. 2007.

Vol. 4406: W. De Meuter (Ed.), Advances in Smalltalk. VII, 157 pages. 2007.

Vol. 4405: L. Padgham, F. Zambonelli (Eds.), Agent-Oriented Software Engineering VII. XII, 225 pages. 2007.

Vol. 4401: N. Guelfi, D. Buchs (Eds.), Rapid Integration of Software Engineering Techniques. IX, 177 pages. 2007.

Vol. 4385: K. Coninx, K. Luyten, K.A. Schneider (Eds.), Task Models and Diagrams for Users Interface Design. XI, 355 pages. 2007.

Vol. 4383: E. Bin, A. Ziv, S. Ur (Eds.), Hardware and Software, Verification and Testing. XII, 235 pages. 2007.

Vol. 4379: M. Südholt, C. Consel (Eds.), Object-Oriented Technology. VIII, 157 pages. 2007.

Vol. 4364: T. Kühne (Ed.), Models in Software Engineering. XI, 332 pages. 2007.

Vol. 4355: J. Julliand, O. Kouchnarenko (Eds.), B 2007: Formal Specification and Development in B. XIII, 293 pages. 2006.

Vol. 4354: M. Hanus (Ed.), Practical Aspects of Declarative Languages. X, 335 pages. 2006.

Vol. 4350: M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C. Talcott, All About Maude - A High-Performance Logical Framework. XXII, 797 pages. 2007.

Vol. 4348: S. Tucker Taft, R.A. Duff, R.L. Brukardt, E. Plödereder, P. Leroy, Ada 2005 Reference Manual. XXII, 765 pages. 2006.

Vol. 4346: L. Brim, B. Haverkort, M. Leucker, J. van de Pol (Eds.), Formal Methods: Applications and Technology. X, 363 pages. 2007.

Vol. 4344: V. Gruhn, F. Oquendo (Eds.), Software Architecture. X, 245 pages. 2006.

Vol. 4340: R. Prodan, T. Fahringer, Grid Computing. XXIII, 317 pages. 2007.

Vol. 4336: V.R. Basili, D. Rombach, K. Schneider, B. Kitchenham, D. Pfahl, R.W. Selby (Eds.), Empirical Software Engineering Issues. XVII, 193 pages. 2007.

Vol. 4326: S. Göbel, R. Malkewitz, I. Iurgel (Eds.), Technologies for Interactive Digital Storytelling and Entertainment. X, 384 pages. 2006.

Vol. 4323: G. Doherty, A. Blandford (Eds.), Interactive Systems. XI, 269 pages. 2007.

Vol. 4322: F. Kordon, J. Sztipanovits (Eds.), Reliable Systems on Unreliable Networked Platforms. XIV, 317 pages. 2007.

Vol. 4309: P. Inverardi, M. Jazayeri (Eds.), Software Engineering Education in the Modern Age. VIII, 207 pages. 2006.

Vol. 4294: A. Dan, W. Lamersdorf (Eds.), Service-Oriented Computing – ICSOC 2006. XIX, 653 pages. 2006.

Vol. 4290: M. van Steen, M. Henning (Eds.), Middleware 2006. XIII, 425 pages. 2006.

Vol. 4279: N. Kobayashi (Ed.), Programming Languages and Systems. XI, 423 pages. 2006.

Vol. 4262: K. Havelund, M. Núñez, G. Roşu, B. Wolff (Eds.), Formal Approaches to Software Testing and Runtime Verification. VIII, 255 pages. 2006.

Vol. 4260: Z. Liu, J. He (Eds.), Formal Methods and Software Engineering. XII, 778 pages. 2006.

Vol. 4257: I. Richardson, P. Runeson, R. Messnarz (Eds.), Software Process Improvement. XI, 219 pages. 2006.

Vol. 4242: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development II. IX, 289 pages. 2006.

Vol. 4229: E. Najm, J.-F. Pradat-Peyre, V.V. Donzeau-Gouge (Eds.), Formal Techniques for Networked and Distributed Systems - FORTE 2006. X, 486 pages. 2006.

Vol. 4227: W. Nejdl, K. Tochtermann (Eds.), Innovative Approaches for Learning and Knowledge Sharing. XVII, 721 pages. 2006.

Vol. 4218: S. Graf, W. Zhang (Eds.), Automated Technology for Verification and Analysis. XIV, 540 pages. 2006.

Vol. 4214: C. Hofmeister, I. Crnković, R. Reussner (Eds.), Quality of Software Architectures. X, 215 pages. 2006.

Vol. 4204: F. Benhamou (Ed.), Principles and Practice of Constraint Programming - CP 2006. XVIII, 774 pages. 2006.

Vol. 4199: O. Nierstrasz, J. Whittle, D. Harel, G. Reggio (Eds.), Model Driven Engineering Languages and Systems. XVI, 798 pages. 2006.

Vol. 4192: B. Mohr, J.L. Träff, J. Worringen, J.J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XVI, 414 pages. 2006.

Vol. 4184: M. Bravetti, M. Núñez, G. Zavattaro (Eds.), Web Services and Formal Methods. X, 289 pages. 2006.

Vol. 4166: J. Górski (Ed.), Computer Safety, Reliability, and Security. XIV, 440 pages. 2006.

Vol. 4158: L.T. Yang, H. Jin, J. Ma, T. Ungerer (Eds.), Autonomic and Trusted Computing. XIV, 613 pages. 2006.

Vol. 4157: M. Butler, C.B. Jones, A. Romanovsky, E. Troubitsyna (Eds.), Rigorous Development of Complex Fault-Tolerant Systems. X, 403 pages. 2006.

Table of Contents

Finite Abstract Models for Deterministic Transition Systems: Fair Parallel Composition and Refinement-Preserving Logic

Harald Fecher and Immo Grabe

Christian-Albrechts-University at Kiel, Germany
{hf,igb}@informatik.uni-kiel.de

Abstract. Since usually no scheduler is given at the programming or modeling language level, abstract models together with a refinement notion are necessary to model concurrent systems adequately. Deterministic transition systems are an appropriate model for implementations of (concurrent) reactive programs based on synchronous communication. In this paper, we develop a suitable setting for modeling and reasoning about deterministic transition systems. In particular, we (i) develop a class of abstract models together with a refinement notion; (ii) define parallel composition guaranteeing fairness; and (iii) develop a 3-valued logic with a satisfaction relation that is preserved under refinement.

1 Introduction

The execution of concurrent reactive programs, where the scheduler is given, e.g., by the operating system, behaves (if no real random generator exists) deterministically up to the environment, i.e., the system behaves in the same way whenever the environment behaves in the same way (including points in time). Deterministic transition systems, where no two transitions leaving the same state have the same label, are an appropriate model for reactive systems based on synchronous communication, whenever the environment will provide at most one action (resp. will request at most one of the actions provided by the system) at once. For example, they are in particular an appropriate model for implementations of a UML state machine [1], where only synchronous communication between the state machine and its event pool, which can provide at most one 'event' at the same time, occurs.

Deterministic transition systems are also appropriate as model for components of closed concurrent systems, whenever every component has its own scheduler, i.e., determines which process(es) of the component performs the next action. Here, a global scheduler decides if a (and which) communication between the component and its environment takes place or if an internal computation takes place.

Models for programming languages that contain concurrency are usually nondeterministic, since the scheduler is not known at that level (i.e., will be provided by the operating system). Therefore, those models as well as models for

F. Arbab and M. Sirjani (Eds.): FSEN 2007, LNCS 4767, pp. 1–16, 2007.

modeling languages should contain nondeterminism, which will be resolved (via refinements) in later design phases and/or by the operating system until deterministic computations are reached. Properties valid on the abstract level, i.e., on the model containing nondeterminism, should be preserved under refinement to maintain the relation between the model and the system. Furthermore, a model for the abstract level should provide a compact and finite description of sets of implementations, especially to improve verification. Moreover, it should be closed under standard operators to be suitable for defining semantics of programming languages and for compositional reasoning. Note that often programmers, software engineers, and computer scientists stay on the abstract level and never reach the concrete level in their contribution to the software development process. Nevertheless, it is important to know what exactly the systems are, since the definition of, e.g., sound satisfaction at the abstract level heavily depends on this information.

Contribution. We develop a setting for modeling and reasoning about deterministic transition systems.

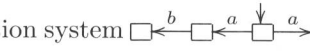
In particular:

- We develop a class of abstract models together with a refinement notion, where exactly the deterministic transition systems are the concrete ones. Our model allows finite/compact modeling by (i) abstracting labels, (ii) having a predicate over labels indicating whether the removal of all transitions having a label is allowed as a refinement step or not, and (iii) having Streett acceptance conditions for restricting infinite computations.
- We define parallel composition for our model that (i) preserves refinement, (ii) preserves satisfiability (i.e., the existence of a refining implementation), and (iii) guarantees fairness, in that, roughly speaking, every component as well as internal synchronization gets an infinite number of opportunities to execute. Here, Streett acceptance conditions are naturally generated by parallel composition between deterministic transition systems.
- We develop a logic together with its satisfaction relation. The logic has as its basic operator $\langle\!|\, \alpha \,|\!\rangle q$ indicating that α can be executed and after executing α property q is guaranteed to hold. This logic yields a 3-valued satisfaction relation on our model, but is 2-valued on concrete abstractions (implementations). We show soundness, i.e., that satisfaction is preserved under refinement. Furthermore, deciding our satisfaction relation is in NP and approximates the EXPTIME-hard language inclusion problem which asks whether all implementations that refine abstraction $\mathcal{M}$ satisfy property ϕ. The PSPACE-complete LTL model checking problem is also approximated.

Related work. Kripke structures (with Streett fairness constraints) together with trace inclusion as refinement notion are used as abstract settings for linear time, where implementations are traces. In this context, LTL [2] is an appropriate logic. Abstract models used for abstraction of linear time settings are not appropriate for our purpose, since they do not model the branching time sensitivity obtained by communications on different actions.

Transition systems with (forward or backward) simulation [3, 4] are not an appropriate setting for abstraction of deterministic transition systems, since deterministic transition systems can be refined further and, therefore, refinement preserving satisfaction relations are in general not 2-valued on them. Therefore, alternating refinement [5] also yields no appropriate setting for our purpose, since it coincides with simulation on labeled transition systems.

On the other hand, transition systems with ready simulation [6] yield an appropriate setting if deterministic transition systems are the implementations. The predicate over labels and the fairness constraint in our setting allow a more compact representation than ready simulation, which will be illustrated later. Note that ready simulation coincides with our refinement notion for the canonical embedding of transition systems into our setting. Transition systems are already extended in [7, 8] by a predicate over labels indicating divergence (infinitely many internal computations are possible). Therefore, the relation introduced there, called *prebisimulation*, does not yield a comparable refinement notion. The refinement notions of failure, failure trace, ready, and ready trace inclusion [9] are also appropriate settings if deterministic transition systems are the implementations. Their trace based approach makes it hard to define an approximated, compositional satisfaction relation that is preserved under refinement.

Standard branching time logics, which are interpreted on transition systems, are, e.g., CTL [10] and the μ-calculus [11]. But these logics are not appropriate for our setting, since these logics are not preserved under ready simulation: the property that "there is a transition labeled a such that b is possible afterwards" holds in the labeled transition system 

but not in its refinement 

. μ-automata [12], (disjunctive) modal transition systems [13, 14] and their variants [15, 16, 17, 18] are used as abstraction model for transition systems in order to improve verification of full branching time properties, as, e.g., in [19, 20, 21]. These models are not appropriate for our purpose, since they consider transition systems rather than deterministic transition systems as implementations. Consequently, these models contain additional complex structures that are unnecessary if the implementations are guaranteed to be deterministic. For example, a state in a modal transition system can have more than one outgoing must-transitions, which makes it, e.g., hard to determine satisfiability w.r.t. deterministic transition systems.

To the best of our knowledge there is no abstract model (beside the model developed here) that can create finite abstraction of labeled (deterministic) transition systems in case infinitely many different transition labels are used.

Outline. Our model together with its refinement notion is formally introduced in Section 2, whereas in Section 3 the parallel composition is presented. Section 4 introduces the logic together with the satisfaction relation and Section 5 presents illustration how the setting can be used for modeling and for verification. Section 6 concludes the paper and discusses future work.