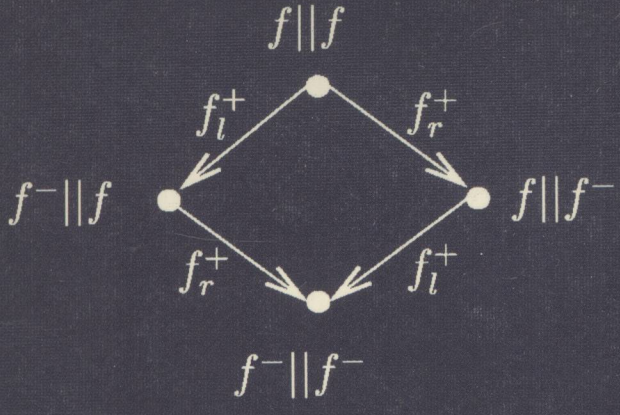
Christel Baier
Boudewijn R. Haverkort
Holger Hermanns
Joost-Pieter Katoen
Markus Siegle (Eds.)

Tutorial

LNCS 2925

Validation of Stochastic Systems

A Guide to Current Research



Springer

Christel Baier Boudewijn R. Haverkort
Holger Hermanns Joost-Pieter Katoen
Markus Siegle (Eds.)

Validation of Stochastic Systems

A Guide to Current Research

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Christel Baier
University of Bonn, Institute of Informatics I
Römerstr. 164, 53117 Bonn, Germany
E-mail: baier@cs.uni-bonn.de

Boudewijn R. Haverkort
University of Twente, Dept. of Computer Science and Electrical Engineering,
Design and Analysis of Communication Systems
P.O. Box 217, The Netherlands
E-mail: brh@cs.utwente.nl

Holger Hermanns
Saarland University, Dept. of Computer Science, Dependable Systems and Software
66123 Saarbrücken, Germany
E-mail: hermanns@cs.uni-sb.de

Joost-Pieter Katoen
University of Twente, Dept. of Computer Science, Formal Methods and Tools
P.O. Box 217, 7500 AE Enschede, The Netherlands
E-mail: katoen@cs.utwente.nl

Markus Siegle
University of Federal Armed Forces Munich, Dept. of Computer Science
85577 Neubiberg, Germany
E-mail: siegle@informatik.unibw-muenchen.de

Library of Congress Control Number: 2004110613

CR Subject Classification (1998): F.1, F.3, D.2, D.4, I.6, C.1

ISSN 0302-9743
ISBN 3-540-22265-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11015703 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 2925

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Preface

It is with great pleasure that we present to you this tutorial volume entitled *Validation of Stochastic Systems*. It is one of the results of the Dutch-German bilateral cooperation project "Validation of Stochastic Systems" (VOSS), financed by NWO and DFG (the Dutch and German science foundations, respectively).

In the early days of 2002, the idea emerged to organize a seminar at Schloss Dagstuhl, not the usual Dagstuhl seminar with primarily invited participants, but a seminar aimed at young(er) people, and for which the organizers assign themes to be worked upon and presented on. Following an open call announced via the Internet in the spring of 2002, we received many applications for participation. After a selection procedure, we decided to assign (mostly) teams of two researchers to work on specific topics, roughly divided into the following four theme areas: "Modelling of Stochastic Systems," "Model Checking of Stochastic Systems," "Representing Large State Spaces," and "Deductive Verification of Stochastic Systems." These are the titles of the four parts of this volume.

The seminar was held in Schloss Dagstuhl during December 8–11, 2002 as part of the so-called GI/Research Seminar series. This series of seminars is financially supported by the *Gesellschaft für Informatik*, the German Computer Society. At that point in time the papers had already undergone a first review round. Each of the tutorial papers was presented in a one-hour session, and on the basis of the presentations we decided to bring together a selection of them into a book. A second review round was performed throughout 2003; at the end of 2003 all contributions were finished. We are glad that Springer-Verlag was willing to publish it in their well-established *Lecture Notes in Computer Science* series, in particular in the "green cover" Tutorial subseries.

To conclude this preface, we would like to thank NWO and DFG for making the VOSS bilateral cooperation project possible in the first place. Secondly, we would like to thank the *Gesellschaft für Informatik* for supporting the participants of the seminar. We would like to thank the whole team at Schloss Dagstuhl for their willingness to host us and for their hospitality. We also thank the authors of the tutorial papers as well as the reviewers for their efforts; without you, there would not have been a workshop! Finally, we would like to thank José Martínez (of the University of Twente) for his work on the editing of this volume.

Christel Baier
Boudewijn Haverkort
Holger Hermanns
Joost-Pieter Katoen
Markus Siegle

Lecture Notes in Computer Science

For information about Vols. 1–3056

please contact your bookseller or Springer

Vol. 3172: M. Dorigo, M. Birattari, C. Blum, L. M.Gambardella, F. Mondada, T. Stützle (Eds.), Ant Colony, Optimization and Swarm Intelligence. XII, 434 pages. 2004.

Vol. 3158: I. Nikolaidis, M. Barbeau, E. Kranakis (Eds.), Ad-Hoc, Mobile, and Wireless Networks. IX, 344 pages. 2004.

Vol. 3157: C. Zhang, H. W. Guesgen, W.K. Yeap (Eds.), PRICAI 2004: Trends in Artificial Intelligence. XX, 1023 pages. 2004. (Subseries LNAI).

Vol. 3156: M. Joye, J.-J. Quisquater (Eds.), Cryptographic Hardware and Embedded Systems - CHES 2004. XIII, 455 pages. 2004.

Vol. 3153: J. Fiala, V. Koubek, J. Kratochvíl (Eds.), Mathematical Foundations of Computer Science 2004. XIV, 902 pages. 2004.

Vol. 3152: M. Franklin (Ed.), Advances in Cryptology – CRYPTO 2004. XI, 579 pages. 2004.

Vol. 3150: G.-Z. Yang, T. Jiang (Eds.), Medical Imaging and Virtual Reality. XII, 378 pages. 2004.

Vol. 3148: R. Giacobazzi (Ed.), Static Analysis. XI, 393 pages. 2004.

Vol. 3146: P. Érdi, A. Esposito, M. Marinaro, S. Scarpetta (Eds.), Computational Neuroscience: Cortical Dynamics. XI, 161 pages. 2004.

Vol. 3144: M. Papatriantafilou, P. Hunel (Eds.), Principles of Distributed Systems. XI, 246 pages. 2004.

Vol. 3143: W. Liu, Y. Shi, Q. Li (Eds.), Advances in Web-Based Learning – ICWL 2004. XIV, 459 pages. 2004.

Vol. 3142: J. Diaz, J. Karhumäki, A. Lepistö, D. Sannella (Eds.), Automata, Languages and Programming. XIX, 1253 pages. 2004.

Vol. 3140: N. Koch, P. Fraternali, M. Wirsing (Eds.), Web Engineering. XXI, 623 pages. 2004.

Vol. 3139: F. Iida, R. Pfeifer, L. Steels, Y. Kuniyoshi (Eds.), Embodied Artificial Intelligence. IX, 331 pages. 2004. (Subseries LNAI).

Vol. 3138: A. Fred, T. Caelli, R.P.W. Duin, A. Campilho, D.d. Ridder (Eds.), Structural, Syntactic, and Statistical Pattern Recognition. XXII, 1168 pages. 2004.

Vol. 3136: F. Meziane, E. Métais (Eds.), Natural Language Processing and Information Systems. XII, 436 pages. 2004.

Vol. 3134: C. Zannier, H. Erdogmus, L. Lindstrom (Eds.), Extreme Programming and Agile Methods - XP/Agile Universe 2004. XIV, 233 pages. 2004.

Vol. 3133: A.D. Pimentel, S. Vassiliadis (Eds.), Computer Systems: Architectures, Modeling, and Simulation. XIII, 562 pages. 2004.

Vol. 3131: V. Torra, Y. Narukawa (Eds.), Modeling Decisions for Artificial Intelligence. XI, 327 pages. 2004. (Subseries LNAI).

Vol. 3130: A. Syropoulos, K. Berry, Y. Haralambous, B. Hughes, S. Peter, J. Plaice (Eds.), TEX, XML, and Digital Typography. VIII, 265 pages. 2004.

Vol. 3129: Q. Li, G. Wang, L. Feng (Eds.), Advances in Web-Age Information Management. XVII, 753 pages. 2004.

Vol. 3128: D. Asonov (Ed.), Querying Databases Privately. IX, 115 pages. 2004.

Vol. 3127: K.E. Wolff, H.D. Pfeiffer, H.S. Delugach (Eds.), Conceptual Structures at Work. XI, 403 pages. 2004. (Subseries LNAI).

Vol. 3126: P. Dini, P. Lorenz, J.N.d. Souza (Eds.), Service Assurance with Partial and Intermittent Resources. XI, 312 pages. 2004.

Vol. 3125: D. Kozen (Ed.), Mathematics of Program Construction. X, 401 pages. 2004.

Vol. 3124: J.N. de Souza, P. Dini, P. Lorenz (Eds.), Telecommunications and Networking - ICT 2004. XXVI, 1390 pages. 2004.

Vol. 3123: A. Belz, R. Evans, P. Piwek (Eds.), Natural Language Generation. X, 219 pages. 2004. (Subseries LNAI).

Vol. 3121: S. Nikoletseas, J.D.P. Rolim (Eds.), Algorithmic Aspects of Wireless Sensor Networks. X, 201 pages. 2004.

Vol. 3120: J. Shawe-Taylor, Y. Singer (Eds.), Learning Theory. X, 648 pages. 2004. (Subseries LNAI).

Vol. 3118: K. Miesenberger, J. Klaus, W. Zagler, D. Burger (Eds.), Computer Helping People with Special Needs. XXIII, 1191 pages. 2004.

Vol. 3116: C. Rattray, S. Maharaj, C. Shankland (Eds.), Algebraic Methodology and Software Technology. XI, 569 pages. 2004.

Vol. 3114: R. Alur, D.A. Peled (Eds.), Computer Aided Verification. XII, 536 pages. 2004.

Vol. 3113: J. Karhumäki, H. Maurer, G. Paun, G. Rozenberg (Eds.), Theory Is Forever. X, 283 pages. 2004.

Vol. 3112: H. Williams, L. MacKinnon (Eds.), Key Technologies for Data Management. XII, 265 pages. 2004.

Vol. 3111: T. Hagerup, J. Katajainen (Eds.), Algorithm Theory - SWAT 2004. XI, 506 pages. 2004.

Vol. 3110: A. Juels (Ed.), Financial Cryptography. XI, 281 pages. 2004.

Vol. 3109: S.C. Sahinalp, S. Muthukrishnan, U. Dogrusoz (Eds.), Combinatorial Pattern Matching. XII, 486 pages. 2004.

Vol. 3108: H. Wang, J. Pieprzyk, V. Varadharajan (Eds.), Information Security and Privacy. XII, 494 pages. 2004.

Vol. 3107: J. Bosch, C. Krueger (Eds.), Software Reuse: Methods, Techniques and Tools. XI, 339 pages. 2004.

Vol. 3106: K.-Y. Chwa, J.I. Munro (Eds.), Computing and Combinatorics. XIII, 474 pages. 2004.

Vol. 3105: S. Göbel, U. Spierling, A. Hoffmann, I. Iurgel, O. Schneider, J. Dechau, A. Feix (Eds.), Technologies for Interactive Digital Storytelling and Entertainment. XVI, 304 pages. 2004.

Vol. 3104: R. Kralovic, O. Sykora (Eds.), Structural Information and Communication Complexity. X, 303 pages. 2004.

Vol. 3103: K. Deb, e. al. (Eds.), Genetic and Evolutionary Computation – GECCO 2004. XLIX, 1439 pages. 2004.

Vol. 3102: K. Deb, e. al. (Eds.), Genetic and Evolutionary Computation – GECCO 2004. L, 1445 pages. 2004.

Vol. 3101: M. Masoodian, S. Jones, B. Rogers (Eds.), Computer Human Interaction. XIV, 694 pages. 2004.

Vol. 3100: J.F. Peters, A. Skowron, J.W. Grzymała-Busse, B. Kostek, R.W. Świniarski, M.S. Szczuka (Eds.), Transactions on Rough Sets I. X, 405 pages. 2004.

Vol. 3099: J. Cortadella, W. Reisig (Eds.), Applications and Theory of Petri Nets 2004. XI, 505 pages. 2004.

Vol. 3098: J. Desel, W. Reisig, G. Rozenberg (Eds.), Lectures on Concurrency and Petri Nets. VIII, 849 pages. 2004.

Vol. 3097: D. Basin, M. Rusinowitch (Eds.), Automated Reasoning. XII, 493 pages. 2004. (Subseries LNAI).

Vol. 3096: G. Melnik, H. Holz (Eds.), Advances in Learning Software Organizations. X, 173 pages. 2004.

Vol. 3095: C. Bussler, D. Fensel, M.E. Orlowska, J. Yang (Eds.), Web Services, E-Business, and the Semantic Web. X, 147 pages. 2004.

Vol. 3094: A. Nürnberger, M. Detyniecki (Eds.), Adaptive Multimedia Retrieval. VIII, 229 pages. 2004.

Vol. 3093: S.K. Katsikas, S. Gritzalis, J. Lopez (Eds.), Public Key Infrastructure. XIII, 380 pages. 2004.

Vol. 3092: J. Eckstein, H. Baumeister (Eds.), Extreme Programming and Agile Processes in Software Engineering. XVI, 358 pages. 2004.

Vol. 3091: V. van Oostrom (Ed.), Rewriting Techniques and Applications. X, 313 pages. 2004.

Vol. 3089: M. Jakobsson, M. Yung, J. Zhou (Eds.), Applied Cryptography and Network Security. XIV, 510 pages. 2004.

Vol. 3087: D. Maltoni, A.K. Jain (Eds.), Biometric Authentication. XIII, 343 pages. 2004.

Vol. 3086: M. Odersky (Ed.), ECOOP 2004 – Object-Oriented Programming. XIII, 611 pages. 2004.

Vol. 3085: S. Berardi, M. Coppo, F. Damiani (Eds.), Types for Proofs and Programs. X, 409 pages. 2004.

Vol. 3084: A. Persson, J. Stirna (Eds.), Advanced Information Systems Engineering. XIV, 596 pages. 2004.

Vol. 3083: W. Emmerich, A.L. Wolf (Eds.), Component Deployment. X, 249 pages. 2004.

Vol. 3080: J. Desel, B. Pernici, M. Weske (Eds.), Business Process Management. X, 307 pages. 2004.

Vol. 3079: Z. Mammeri, P. Lorenz (Eds.), High Speed Networks and Multimedia Communications. XVIII, 1103 pages. 2004.

Vol. 3078: S. Cotin, D.N. Metaxas (Eds.), Medical Simulation. XVI, 296 pages. 2004.

Vol. 3077: F. Roli, J. Kittler, T. Windeatt (Eds.), Multiple Classifier Systems. XII, 386 pages. 2004.

Vol. 3076: D. Buell (Ed.), Algorithmic Number Theory. XI, 451 pages. 2004.

Vol. 3075: W. Lenski, Logic versus Approximation. VIII, 205 pages. 2004.

Vol. 3074: B. Kuijpers, P. Revesz (Eds.), Constraint Databases and Applications. XII, 181 pages. 2004.

Vol. 3073: H. Chen, R. Moore, D.D. Zeng, J. Leavitt (Eds.), Intelligence and Security Informatics. XV, 536 pages. 2004.

Vol. 3072: D. Zhang, A.K. Jain (Eds.), Biometric Authentication. XVII, 800 pages. 2004.

Vol. 3071: A. Omicini, P. Petta, J. Pitt (Eds.), Engineering Societies in the Agents World. XIII, 409 pages. 2004. (Subseries LNAI).

Vol. 3070: L. Rutkowski, J. Siekmann, R. Tadeusiewicz, L.A. Zadeh (Eds.), Artificial Intelligence and Soft Computing - ICAISC 2004. XXV, 1208 pages. 2004. (Subseries LNAI).

Vol. 3068: E. André, L. Dybkjær, W. Minker, P. Heisterkamp (Eds.), Affective Dialogue Systems. XII, 324 pages. 2004. (Subseries LNAI).

Vol. 3067: M. Dastani, J. Dix, A. El Fallah-Seghrouchni (Eds.), Programming Multi-Agent Systems. X, 221 pages. 2004. (Subseries LNAI).

Vol. 3066: S. Tsumoto, R. Słowiński, J. Komorowski, J.W. Grzymała-Busse (Eds.), Rough Sets and Current Trends in Computing. XX, 853 pages. 2004. (Subseries LNAI).

Vol. 3065: A. Lomuscio, D. Nute (Eds.), Deontic Logic in Computer Science. X, 275 pages. 2004. (Subseries LNAI).

Vol. 3064: D. Bienstock, G. Nemhauser (Eds.), Integer Programming and Combinatorial Optimization. XI, 445 pages. 2004.

Vol. 3063: A. Llamosí, A. Strohmeier (Eds.), Reliable Software Technologies - Ada-Europe 2004. XIII, 333 pages. 2004.

Vol. 3062: J.L. Pfaltz, M. Nagl, B. Böhlen (Eds.), Applications of Graph Transformations with Industrial Relevance. XV, 500 pages. 2004.

Vol. 3061: F.F. Ramos, H. Unger, V. Larios (Eds.), Advanced Distributed Systems. VIII, 285 pages. 2004.

Vol. 3060: A.Y. Tawfik, S.D. Goodwin (Eds.), Advances in Artificial Intelligence. XIII, 582 pages. 2004. (Subseries LNAI).

Vol. 3059: C.C. Ribeiro, S.L. Martins (Eds.), Experimental and Efficient Algorithms. X, 586 pages. 2004.

Vol. 3058: N. Sebe, M.S. Lew, T.S. Huang (Eds.), Computer Vision in Human-Computer Interaction. X, 233 pages. 2004.

Vol. 3057: B. Jayaraman (Ed.), Practical Aspects of Declarative Languages. VIII, 255 pages. 2004.

Table of Contents

Modelling Stochastic Systems

Model Checking of Stochastic Systems

Representing Large State Spaces

Deductive Verification of Stochastic Systems

Probabilistic Automata: System Types, Parallel Composition and Comparison

Ana Sokolova[1] and Erik P. de Vink[2]

[1] Department of Mathematics and Computer Science,
TU/e, Eindhoven
a.sokolova@tue.nl
[2] LIACS, Leiden University
evink@win.tue.nl

Abstract. We survey various notions of probabilistic automata and probabilistic bisimulation, accumulating in an expressiveness hierarchy of probabilistic system types. The aim of this paper is twofold: On the one hand it provides an overview of existing types of probabilistic systems and, on the other hand, it explains the relationship between these models. We overview probabilistic systems with discrete probabilities only. The expressiveness order used to built the hierarchy is defined via the existence of mappings between the corresponding system types that preserve and reflect bisimilarity. Additionally, we discuss parallel composition for the presented types of systems, augmenting the map of probabilistic automata with closedness under this compositional operator.

Keywords: probabilistic automata (transition systems), probabilistic bisimulation, preservation and reflection of bisimulation, non-determinism, parallel composition.

1 Introduction

The notion of a state machine has proved useful in many modelling situations, amongst others, the area of validation of stochastic systems. In the literature up to now, a great variety of types of probabilistic automata has been proposed and many of these have been actually used for verification purposes. In this paper we discuss a number of probabilistic automata with discrete probability distributions. For continuous-time probabilistic systems the interested reader is referred to [11, 33, 32, 17, 45, 4]. Models of stochastic systems that are not represented by transition systems can also be found in [22] and [70].

Due to the variety of proposed models it is often the case that results have to be interpreted from one type of systems to another. Therefore we compare the considered types of probabilistic automata in terms of their expressiveness. The comparison is achieved by placing a partial order on the classes of such automata, where one class is less then another if each automaton in the class can be translated to an automaton of the other class such that translations both reflect and preserve the respective notions of bisimilarity. Hence, bisimulation and

C. Baier et al. (Eds.): Validation of Stochastic Systems, LNCS 2925, pp. 1–43, 2004.

bisimilarity are central notions in this overview. Other comparison criteria are important as well, e.g. logical properties, logical characterization of bisimulation [61], complexity of algorithms for deciding bisimulation [9, 13, 31, 80] and so on. We choose the comparison criterion formulated in terms of strong bisimulation because of its simplicity and because we work with transition labelled systems, for which bisimulation semantics arises naturally from the step-by-step behavior.

A major distinction of probabilistic automata is that between fully probabilistic vs. non-deterministic ones. In a fully probabilistic automaton every choice is governed by a probability distribution (over set of states or states combined with actions). The probability distribution captures the uncertainty about the next state. If we abstract away from the actions in a fully probabilistic automaton, we are left with a discrete time Markov chain. Subsequently, standard techniques can be applied to analyze the resulting Markov chains. Sometimes, the incomplete knowledge about the system behavior can not be represented probabilistically. In these cases we should consider more than one transition possible. We speak in this case of a non-deterministic probabilistic automaton. Most of the models that we consider include some form of non-determinism and hence fall in the category of non-deterministic probabilistic automata. As pointed out by various authors, e.g. [47, 76, 3, 81] non-determinism is essential for modelling scheduling freedom, implementation freedom, the external environment and incomplete information. Furthermore, non-determinism is essential for the definition of an asynchronous parallel composition operator that allows interleaving. Often two kinds of non-deterministic choices are mentioned in the literature (see for e.g. [81]), *external* non-deterministic choices influenced by the environment, specified by having several transitions with different labels leaving from the same state, and *internal* non-determinism, exhibited by having several transitions with the same label leaving from a state. We use the term non-determinism for *full non-determinism* including both internal and external non-deterministic choices.

We introduce several classes of automata, ranging from the simplest models to more complex ones. The questions that we will address for each individual class are:

- the definition of the type of automaton and the respective notion of strong bisimulation;
- the relation of the model with other models;
- presence and form of non-determinism;
- the notion of a product or parallel composition in the model.

The set-up of the paper is as follows: Section 2 presents the necessary notions considering probability theory, automata (transition systems), and concurrency theory, in particular compositional operators. In section 3 we focus on the various definitions of probabilistic automata in isolation with their corresponding notions of bisimulation. In section 4 the operators of parallel composition are discussed. We address the interrelationship between the introduced types of automata in section 5. Section 6 wraps up with some conclusions.

Acknowledgements

We would like to thank Holger Hermanns for editorial support and for the plentitude of useful ideas and directions, as well as the other organizers of VOSS GI/Dagstuhl 2002 for initiating and organizing this nice event. We are in dept to the referees for various remarks. Special thanks go to Falk Bartels for his major contribution regarding the hierarchy of probabilistic systems, as well as for numerous comments, suggestions and his friendly cooperation.

2 Basic Ingredients

2.1 Probability Distributions

Let Ω be a set. A function $\mu\colon \Omega \to [0,1]$ is called a *discrete probability distribution*, or distribution for short, on Ω if $\{x \in \Omega |\ \mu(x) > 0\}$ is finite or countably infinite and $\sum_{x\in\Omega} \mu(x) = 1$. The set $\{x \in \Omega |\ \mu(x) > 0\}$ is called the *support* of μ and is denoted by $spt(\mu)$. If $x \in \Omega$, then μ_x^1 denotes the unique probability distribution with $\mu_x^1(x) = 1$, also known as the *Dirac distribution* for x. When μ is a distribution on Ω we use the notation $\mu[X]$ for $\sum_{x\in X} \mu(x)$ where $X \subseteq \Omega$. By $\mathcal{D}(\Omega)$ we denote the set of all discrete probability distributions on the set Ω. If μ is a distribution with finite support $\{s_1, \ldots, s_n\}$, we sometimes write $\{s_1 \mapsto \mu(s_1), \ldots, s_n \mapsto \mu(s_n)\}$. With this notation, $\mu_x^1 = \{x \mapsto 1\}$.

Let $\mu_1 \in \mathcal{D}(S)$ and $\mu_2 \in \mathcal{D}(T)$. The product $\mu_1 \times \mu_2$ of μ_1 and μ_2 is a distribution on $S \times T$ defined by $(\mu_1 \times \mu_2)(s,t) = \mu_1(s) \cdot \mu_2(t)$, for $\langle s,t\rangle \in S \times T$.

If $\mu \in \mathcal{D}(S \times T)$, we use the notation $\mu[s,T]$ for $\mu[\{s\} \times T]$ and $\mu[S,t]$ for $\mu[S \times \{t\}]$. We adopt from [51] the lifting of a relation between two sets to a relation between distributions on these sets.

Definition 1. *Let $R \subseteq S \times T$ be a relation between the sets S and T. Let $\mu \in \mathcal{D}(S)$ and $\mu' \in \mathcal{D}(T)$ be distributions. Define $\mu \equiv_R \mu'$ if and only if there exists a distribution $\nu \in \mathcal{D}(S \times T)$ such that*

1. *$\nu[s,T] = \mu(s)$ for any $s \in S$*
2. *$\nu[S,t] = \mu'(t)$ for any $t \in T$*
3. *$\nu(s,t) \neq 0$ if and only if $\langle s,t\rangle \in R$.*

The lifting of a relation R preserves the characteristic properties of preorders and equivalences (cf. [52]). For the special case of an equivalence relation there is a simpler way to define the lifting (cf. [52, 81, 9]).

Proposition 1. *Let R be an equivalence relation on the set S and let $\mu, \mu' \in \mathcal{D}(S)$. Then $\mu \equiv_R \mu'$ if and only if $\mu[C] = \mu'[C]$ for all equivalence classes $C \in S/R$.* □

Lifting of an equivalence relation on a set S to a relation $\equiv_{R,A}$ on the set $\mathcal{D}(A \times S)$, for a fixed set A, will also be needed.

Definition 2. *Let R be an equivalence relation on a set S, A a set, and let $\mu, \mu' \in \mathcal{D}(A \times S)$. Define*

$$\mu \equiv_{R,A} \mu' \iff \forall C \in S/R, \forall a \in A\colon \mu[a,C] = \mu'[a,C]$$

2.2 Non-probabilistic Automata, Markov Chains, Bisimilarity

Throughout the paper we will use the terms automaton, transition system or just system as synonyms.

Non-probabilistic Automata

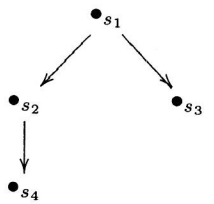
Definition 3. *A transition system, TS for short, is a pair $\langle S, \alpha \rangle$ where*

1. *S is a set of states*
2. *$\alpha : S \to \mathcal{P}(S)$ is a transition function, where $\mathcal{P}$ denotes the powerset of S.*

If $\langle S, \alpha \rangle$ is a transition system such that $s, s' \in S$ and $s' \in \alpha(s)$ we write $s \to s'$ and call it a transition.

Often in the literature a TS is given as a triple, including besides the set of states and the transition function also a subset of initial states, or a single initial state. In this paper we will consider **no initial states** and therefore they are not present in the definition. Instead of a transition function one could equivalently consider a transition relation as a subset of $S \times S$. Our choice here is to always present the transitions via a **transition function**.

A way of representing a TS is via its transition diagram. For example, the system $\langle S, \alpha \rangle$ where $S = \{s_1, s_2, s_3, s_4\}$ and $\alpha(s_1) = \{s_2, s_3\}$, $\alpha(s_2) = \{s_4\}$, $\alpha(s_3) = \alpha(s_4) = \emptyset$, is represented as follows:



The states s_3 and s_4 are *terminating* states, with no outgoing transitions.

It is often of use to model the phenomenon that a change of state in a system happens as a result of executing an *action*. Therefore, labelled transition systems evolve from transition systems. There are two ways to incorporate labels in a TS: by labelling the states (usually with some values of variables, or a set of propositions true in a state), or by explicitly labelling the transitions with actions or action names. In this paper we focus on **transition labelled systems**.

Definition 4. *A labelled transition system (LTS) (or a non-deterministic automaton) is a triple $\langle S, A, \alpha \rangle$ where*

1. *S is a set of states*
2. *A is a set of actions*
3. *$\alpha : S \to \mathcal{P}(A \times S)$ is a transition function.*

When $\langle S, A, \alpha \rangle$ is a LTS, then the transition function α can equivalently be considered as a function from S to $\mathcal{P}(S)^A$, the collection of functions from A

to $\mathcal{P}(S)$. As in the case of TSs, for any state $s \in S$ of a LTS, every element $\langle a, s' \rangle \in \alpha(s)$ determines a transition which is denoted by $s \xrightarrow{a} s'$.

The class of non-deterministic automata (LTSs) is denoted by **NA**. Deterministic automata, given by the next definition, form a subclass of **NA**.

Definition 5. *A deterministic automaton is a triple $\langle S, A, \alpha \rangle$ where*

1. *S is a set of states*
2. *A is a set of actions*
3. *$\alpha : S \to (S + 1)^A$ is a transition function.*

Notation 1 *We denote by $+$ the disjoint union of two sets. The set 1 is a singleton set containing the special element $*$, i.e. $1 = \{*\}$. We assume that $* \notin S$. The notation $(S + 1)^A$ stands for the collection of all functions from A to $S + 1$.*

The special set 1 and the disjoint union construction allow us to write partial functions as functions. Hence, in a deterministic automaton each state s is assigned a partial function $\alpha(s)\colon A \to S + 1$ from the set of actions to the set of states, meaning that whenever $\alpha(s)(a) = s'$ for some $s' \in S$, i.e. $\alpha(s) \neq *$, then there is a transition $s \xrightarrow{a} s'$ enabled in S. We denote the class of all deterministic automata by **DA**.

We note that the class of automata **DA** exhibits external non-determinism, while in **NA** there is full non-determinism.

external non-determinism full non-determinism

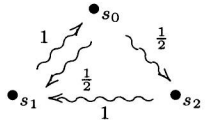
Markov Chains. The simplest class of fully probabilistic automata is the class of discrete time Markov chains. The theory of Markov chains is rich and huge (see, e.g., [57, 48, 16, 43]) and we only provide a simple definition of a discrete time Markov chain here.

Definition 6. *A Markov chain is a pair $\langle S, \alpha \rangle$ where*

1. *S is a set of states*
2. *$\alpha : S \to \mathcal{D}(S)$ is a transition function.*

Markov chains evolve from transition systems, when probability is added to each transition such that for any state the sum of the probabilities of all outgoing transitions equals 1. The class of all Markov chains is denoted by **MC**. If $s \in S$ and $\alpha(s) = \mu$ with $\mu(s') = p > 0$ then the Markov chain $\langle S, \alpha \rangle$ is said to go from a state s with probability p to a state s'. Notation: $s \rightsquigarrow \mu$ and $s \overset{p}{\rightsquigarrow} s'$.

Example 1.



$$S = \{s_0, s_1, s_2\}$$
$$\alpha(s_0) = \{s_0 \mapsto 0, s_1 \mapsto \tfrac{1}{2}, s_2 \mapsto \tfrac{1}{2}\}$$
$$\alpha(s_1) = \mu^1_{s_0}$$
$$\alpha(s_2) = \mu^1_{s_1}$$

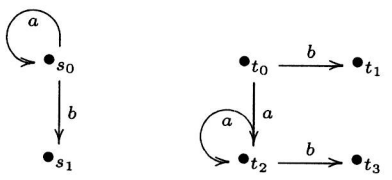
Bisimulation and Bisimilarity. Different semantics or notions of behavior can be given to labelled transition systems. We work with the bisimulation semantics (Milner [65, 66]) stating that two states in a system represented by LTSs are equivalent whenever there exists a bisimulation relation that relates them. A bisimulation relation compares the one-step behavior of two states and has a nice extension to the probabilistic case (as explored in [61]). In [54] probabilistic extensions of a number of other well known process equivalences have been studied like probability trace, completed trace, failure and ready equivalence. Other probabilistic process equivalences are probabilistic simulation and bisimulation by Segala and Lynch [78, 76], Yi and Larsen's testing equivalence [88], and CSP equivalences of Morgan et al. [67], Lowe [59] and Seidel [77]. An overview of several probabilistic process equivalences can be found in [58].

Definition 7. *Let $\langle S, A, \alpha\rangle$ and $\langle T, A, \alpha\rangle$ be two LTSs. A relation $R \subseteq S \times T$ is a* bisimulation *relation if for all $\langle s, t\rangle \in R$ and all $a \in A$ the following holds*

if $s \xrightarrow{a} s'$ then there exists $t' \in T$ such that $t \xrightarrow{a} t'$ and $\langle s', t'\rangle \in R$, and
if $t \xrightarrow{a} t'$ then there exists $s' \in S$ such that $s \xrightarrow{a} s'$ and $\langle s', t'\rangle \in R$.

Let $s \in S$ and $t \in T$. The states s and t are called bisimilar, *denoted by $s \approx t$ if there exists a bisimulation relation R with $\langle s, t\rangle \in R$.*

Example 2. For the following LTSs we have, for example, $s_0 \approx t_0$ since $R = \{\langle s_0, t_0\rangle, \langle s_0, t_2\rangle, \langle s_1, t_1\rangle, \langle s_1, t_3\rangle\}$ is a bisimulation.



Remark 1. Instead of comparing states in two systems $\langle S, A, \alpha\rangle$ and $\langle T, A, \beta\rangle$ we can always consider one joined system $\langle S + T, A, \gamma\rangle$ with $\gamma(s) = \alpha(s)$ for $s \in S$ and $\gamma(t) = \beta(t)$ for $t \in T$. Therefore bisimulation can be defined as a relation on the set of states of a system. Furthermore, if $R \subseteq S \times S$ is a bisimulation, then it is reflexive and symmetric, and the transitive closure of R is also a bisimulation. Hence bisimilarity $\approx$ is not affected by the choice of defining bisimulation as an equivalence.

Definition 8. *An equivalence relation R on a set of states S of a LTS is an equivalence bisimulation if for all $\langle s, t\rangle \in R$ and all $a \in A$*

if $s \xrightarrow{a} s'$ then $\exists t' \in S$: $t \xrightarrow{a} t'$, $\langle s', t'\rangle \in R$

The states s and t are called bisimilar, *denoted by $s \approx_e t$ if there exists an equivalence bisimulation R with $\langle s, t\rangle \in R$.*

By Remark 1, the following proposition holds.

Proposition 2. *Let $\langle S, A, \alpha \rangle$ and $\langle T, A, \beta \rangle$ be two LTSs, and let $s \in S$, $t \in T$. Then $s \approx t$ if and only if $s \approx_e t$.*

□

Bisimulation on **DA** is defined exactly the same as for **NA** i.e. with Definition 8.

The standard notion of probabilistic bisimulation is the one introduced by Larsen and Skou [61] originally formulated for reactive systems (see next subsection). An early reference to probabilistic bisimulation can be found in [23]. In the case of Markov chains, bisimulation corresponds to ordinary lumpability of Markov chains [57, 44, 27]. In [86, 85] it is shown that the concrete notion of bismulation for Markov-chains coincides with a general coalgebraic notion of bisimulation [68, 53, 74, 64].

The idea behind probabilistic bisimulation is as follows. Since bisimilar states are considered "the same", it does not matter which element within a bisimulation class is reached. Hence, a bisimulation relation should compare the probability to reach an equivalence class and not the probability to reach a single state. In order to define bisimulation for Markov chains the lifting of a relation on a state S to a relation on $\mathcal{D}(S)$, as defined in Definition 1 and explained with Proposition 1, is used. Note that the comments of Remark 1 are in place here as well.

Definition 9. *An equivalence relation R on a set of states S of a Markov chain $\langle S, \alpha \rangle$ is a bisimulation if and only if for all $\langle s, t \rangle \in R$*

$$\text{if } s \rightsquigarrow \mu \text{ then there is a transition } t \rightsquigarrow \mu' \text{ with } \mu \equiv_R \mu'.$$

The states s and t are called bisimilar, *denoted by $s \approx t$, if there exists a bisimulation R with $\langle s, t \rangle \in R$.*

Definition 9 will be used, with some variations, for defining bisimulation relations for all types of probabilistic automata that we consider in this overview. However, note that in the case of Markov chains any two states of any two Markov chains are bisimilar, according to the given definition, since $\nabla = S \times S$ is a bisimulation on the state set of any Markov chain $\langle S, \alpha \rangle$. Namely, let $\langle S, \alpha \rangle$ be a Markov chain and $s, t \in S$, such that $\alpha(s) = \mu, \alpha(t) = \mu'$, i.e., $s \rightsquigarrow \mu, t \rightsquigarrow \mu'$. Then for the only equivalence class of ∇, S, we have $\mu[S] = 1 = \mu'[S]$ i.e. $\mu \equiv_R \mu'$ which makes $s \approx t$. This phenomenon can be explained with the fact that bisimilarity compares the observable behavior of two states in a system and the Markov chains are very simple systems in which there is not much to observe. Therefore the need comes to enrich Markov chains with actions or at least termination.

Notation. In Section 3 we will introduce ten other types of probabilistic automata, with corresponding notions of bisimulation. In order to avoid repetition we collect the following.

- A type of automata will always be a triple $\langle S, A, \alpha \rangle$ where S is a set of states, A is a set of actions and α is a transition function. The difference between the system types is expressed with the difference in the codomains of the corresponding transition functions.
- A bisimulation relation will always be defined as an equivalence on the set of states of a system. Depending on the type of systems the "transfer conditions" in the definition of bisimulation vary.
- For a particular type of system, the bisimilarity relation, denoted by $\approx$ is defined by: $s \approx t$ if and only if there exists a bisimulation R that relates s and t, i.e. $\langle s, t \rangle \in R$. Although we use the same notation $\approx$ for bisimilarity in different types of systems, it should be clear that for each type of systems, $\approx$ is a different relation.

2.3 Parallel Composition of LTSs and MCs

Compositional operators serve the need of modular specification and verification of systems. They arise from process calculi, such as CCS ([66]), CSP ([47]) and ACP ([19]), where process terms (models of processes) are built from atomic process terms with the use of compositional operators. Usually a model of a process calculi is a suitable class of transition systems. Therefore it is often the case that process terms are identified with their corresponding transition systems, and the compositional operators of the process calculus can be considered as operators for combining transition systems. In this overview we focus on the parallel composition operator. The definition of parallel composition varies a lot throughout different process calculi. In this section we consider the non-probabilistic case (LTSs) in order to explain variants of different parallel compositions, and the parallel composition of Markov chains in order to present the basics of probabilistic parallel composition.

Labelled Transition Systems. A major distinction between different parallel composition operators is whether they are *synchronous*, where the components are forced to synchronize whenever they can, or *asynchronous* where the components can either synchronize or act independently. Furthermore, different approaches for synchronization exist. The result of the parallel composition of two automata $\mathcal{A}_1 = \langle S_1, A, \alpha_1 \rangle$ and $\mathcal{A}_2 = \langle S_2, A, \alpha_2 \rangle$ is an automaton $\mathcal{A}_1 \| \mathcal{A}_2 = \langle S_1 \times S_2, A, \alpha \rangle$ where the definition of α varies. Instead of a pair $\langle s, t \rangle \in S_1 \times S_2$ we will write $s \| t$ for a state in the composed automaton. Throughout this subsection we will use as running example, the parallel composition of the following two automata.