Paolo Ciancarini
Herbert Wiklicky (Eds.)

LNCS 4038

Coordination Models and Languages

8th International Conference, COORDINATION 2006
Bologna, Italy, June 2006
Proceedings

Springer

Paolo Ciancarini Herbert Wiklicky (Eds.)

Coordination Models and Languages

8th International Conference, COORDINATION 2006
Bologna, Italy, June 14-16, 2006
Proceedings

Springer

Volume Editors

Paolo Ciancarini
Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Zamboni 7, 40127 Bologna, Italy
E-mail: cianca@cs.unibo.it

Herbert Wiklicky
Imperial College, Department of Computing
Huxley Building, 180 Queen's Gate, London SW7 2AZ, UK
E-mail: herbert@doc.ic.ac.uk

Library of Congress Control Number: 2006926827

CR Subject Classification (1998): D.2.4, D.2, C.2.4, D.1.3, F.1.2, I.2.11

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-34694-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-34694-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11767954 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 4038

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

Editorial Board

Preface

This volume contains the proceedings of the 8th International Conference on Coordination Models and Languages, Coordination 2006. This year the conference was part of a set of federated conferences named DisCoTec 2006, held in Bologna in June 2006. It was held in the series of successful conferences whose proceedings were also published in LNCS, in volumes 1061, 1282, 1594, 1906, 2315, 2949, and 3454.

The conference was born in 1996, as a forum for researchers working on programming models, formalisms, and platforms for describing or supporting concurrent and distributed computations. Contemporary information systems increasingly rely on combining concurrent and distributed, and now also mobile, reconfigurable and heterogeneous components. New models, architectures, languages, and verification techniques are necessary to cope with the complexity induced by the demands of today's software industry. Coordination languages have emerged as a successful approach, in that they provide abstractions that cleanly separate behavior from communication, thereby supporting modular design, simplifying reasoning, and ultimately enhancing software development. Research on coordination models and languages is still playing a crucial role in addressing the technological concerns of widely distributed applications and services.

We received 50 submission. All papers were reviewed by four reviewers. The Program Committee used a tool for collaborative conference management to select 18 regular papers. The Program Committee invited Chris Hankin and Pierpaolo Degano to give talks.

The conference and this volume would not have been possible without the intellectual contributions of all the authors, the careful reviews and advice by members of the Program Committee, and the additional referees who helped us to evaluate the papers. We thank Gianluigi Zavattaro, the General Chair of DisCoTec 2006, for his helpful support with the conference management system.

April 2006

Paolo Ciancarini
Herbert Wiklicky

Organization

Coordination 2006, the 8th International Conference on Coordination Models and Languages, is part of the set of federated conferences DisCoTec 2006, also including DAIS 2006, the 6th IFIP International Conference on Distributed Applications and Interoperable Systems, and FMOODS 2006, the 8th IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems, as well as three workshops organized by the department of Computer Science,
University of Bologna.

Program Committee

Conference Chair:	Gianluigi Zavattaro, University of Bologna, Italy
Program Chairs:	Paolo Ciancarini, University of Bologna, Italy
	Herbert Wiklicky, Imperial College London, UK
Members:	Farhad Arbab, CWI Amsterdam, The Netherlands
	Luis Barbosa, Universidade do Minho, Portugal
	Antonio Brogi, University of Pisa, Italy
	Wolfgang Emmerich, University College London, UK
	Frank de Boer, CWI & Utrecht University, The Netherlands
	Jean-Marie Jacquet, University of Namur, Belgium
	Joost Kok, Leiden University, The Netherlands
	Toby Lehman, IBM Almaden, USA
	D.C. Marinescu, University of Central Florida, USA
	Ronaldo Menezes, Florida Institute of Technology, USA
	Andrea Omicini, University of Bologna, Italy
	Paolo Petta, OeFAI, Austria
	Gian Pietro Picco, Politecnico di Milano, Italy
	Ernesto Pimentel, University of Malaga, Spain
	Rosario Pugliese, University of Florence, Italy
	Gruia Catalin Roman, Washington University, USA
	Robert Tolksdorf, FU Berlin, Germany
	Emilio Tuosto, University of Leicester, UK
	Carlos Varela, Rensselaer Polytechnic Institute, USA
	Alan Wood, University of York, UK

Lecture Notes in Computer Science

For information about Vols. 1–3956

please contact your bookseller or Springer

Vol. 4060: K. Futatsugi, J.-P. Jouannaud, J. Meseguer (Eds.), Algebra, Meaning and Computation. XXXVIII, 643 pages. 2006.

Vol. 4058: L.M. Batten, R. Safavi-Naini (Eds.), Information Security and Privacy. XII, 446 pages. 2006.

Vol. 4055: J. Lee, J. Shim, S.-g. Lee, C. Bussler, S. Shim (Eds.), Data Engineering Issues in E-Commerce and Services. IX, 290 pages. 2006.

Vol. 4054: A. Horváth, M. Telek (Eds.), Formal Methods and Stochastic Models for Performance Evaluation. VIII, 239 pages. 2006.

Vol. 4053: M. Ikeda, K.D. Ashley, T.-W. Chan (Eds.), Intelligent Tutoring Systems. XXVI, 821 pages. 2006.

Vol. 4044: P. Abrahamsson, M. Marchesi, G. Succi (Eds.), Extreme Programming and Agile Processes in Software Engineering. XII, 230 pages. 2006.

Vol. 4043: A.S. Atzeni, A. Lioy (Eds.), Public Key Infrastructure. XI, 261 pages. 2006.

Vol. 4041: S.-W. Cheng, C.K. Poon (Eds.), Algorithmic Aspects in Information and Management. XI, 395 pages. 2006.

Vol. 4040: R. Reulke, U. Eckardt, B. Flach, U. Knauer, K. Polthier (Eds.), Combinatorial Image Analysis. XII, 482 pages. 2006.

Vol. 4039: M. Morisio (Ed.), Reuse of Off-the-Shelf Components. XIII, 444 pages. 2006.

Vol. 4038: P. Ciancarini, H. Wiklicky (Eds.), Coordination Models and Languages. VIII, 299 pages. 2006.

Vol. 4037: R. Gorrieri, H. Wehrheim (Eds.), Formal Methods for Open Object-Based Distributed Systems. XVII, 474 pages. 2006.

Vol. 4036: O. H. Ibarra, Z. Dang (Eds.), Developments in Language Theory. XII, 456 pages. 2006.

Vol. 4034: J. Münch, M. Vierimaa (Eds.), Product-Focused Software Process Improvement. XVII, 474 pages. 2006.

Vol. 4033: B. Stiller, P. Reichl, B. Tuffin (Eds.), Performability Has its Price. X, 103 pages. 2006.

Vol. 4031: M. Ali, R. Dapoigny (Eds.), Innovations in Applied Artificial Intelligence. XXIII, 1353 pages. 2006. (Sublibrary LNAI).

Vol. 4027: H.L. Larsen, G. Pasi, D. Ortiz-Arroyo, T. Andreasen, H. Christiansen (Eds.), Flexible Query Answering Systems. XVIII, 714 pages. 2006. (Sublibrary LNAI).

Vol. 4026: P. Gibbons, T. Abdelzaher, J. Aspnes, R. Rao (Eds.), Distributed Computing in Sensor Systems. XIV, 566 pages. 2006.

Vol. 4025: F. Eliassen, A. Montresor (Eds.), Distributed Applications and Interoperable Systems. XI, 355 pages. 2006.

Vol. 4024: S. Donatelli, P. S. Thiagarajan (Eds.), Petri Nets and Other Models of Concurrency - ICATPN 2006. XI, 441 pages. 2006.

Vol. 4021: E. André, L. Dybkjær, W. Minker, H. Neumann, M. Weber (Eds.), Perception and Interactive Technologies. XI, 217 pages. 2006. (Sublibrary LNAI).

Vol. 4020: A. Bredenfeld, A. Jacoff, I. Noda, Y. Takahashi, RoboCup 2005: Robot Soccer World Cup IX. XVII, 727 pages. 2006. (Sublibrary LNAI).

Vol. 4018: V. Wade, H. Ashman, B. Smyth (Eds.), Adaptive Hypermedia and Adaptive Web-Based Systems. XVI, 474 pages. 2006.

Vol. 4016: J.X. Yu, M. Kitsuregawa, H.V. Leong (Eds.), Advances in Web-Age Information Management. XVII, 606 pages. 2006.

Vol. 4013: L. Lamontagne, M. Marchand (Eds.), Advances in Artificial Intelligence. XIII, 564 pages. 2006. (Sublibrary LNAI).

Vol. 4012: T. Washio, A. Sakurai, K. Nakashima, H. Takeda, S. Tojo, M. Yokoo (Eds.), New Frontiers in Artificial Intelligence. XIII, 484 pages. 2006. (Sublibrary LNAI).

Vol. 4011: Y. Sure, J. Domingue (Eds.), The Semantic Web: Research and Applications. XIX, 726 pages. 2006.

Vol. 4010: S. Dunne, B. Stoddart (Eds.), Unifying Theories of Programming. VIII, 257 pages. 2006.

Vol. 4009: M. Lewenstein, G. Valiente (Eds.), Combinatorial Pattern Matching. XII, 414 pages. 2006.

Vol. 4007: C. Àlvarez, M. Serna (Eds.), Experimental Algorithms. XI, 329 pages. 2006.

Vol. 4006: L.M. Pinho, M. González Harbour (Eds.), Reliable Software Technologies – Ada-Europe 2006. XII, 241 pages. 2006.

Vol. 4005: G. Lugosi, H.U. Simon (Eds.), Learning Theory. XI, 656 pages. 2006. (Sublibrary LNAI).

Vol. 4004: S. Vaudenay (Ed.), Advances in Cryptology - EUROCRYPT 2006. XIV, 613 pages. 2006.

Vol. 4003: Y. Koucheryavy, J. Harju, V.B. Iversen (Eds.), Next Generation Teletraffic and Wired/Wireless Advanced Networking. XVI, 582 pages. 2006.

Vol. 4001: E. Dubois, K. Pohl (Eds.), Advanced Information Systems Engineering. XVI, 560 pages. 2006.

Vol. 3999: C. Kop, G. Fliedl, H.C. Mayr, E. Métais (Eds.), Natural Language Processing and Information Systems. XIII, 227 pages. 2006.

Vol. 3998: T. Calamoneri, I. Finocchi, G.F. Italiano (Eds.), Algorithms and Complexity. XII, 394 pages. 2006.

Vol. 3997: W. Grieskamp, C. Weise (Eds.), Formal Approaches to Software Testing. XII, 219 pages. 2006.

Vol. 3996: A. Keller, J.-P. Martin-Flatin (Eds.), Self-Managed Networks, Systems, and Services. X, 185 pages. 2006.

Vol. 3995: G. Müller (Ed.), Emerging Trends in Information and Communication Security. XX, 524 pages. 2006.

Vol. 3994: V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, J. Dongarra (Eds.), Computational Science – ICCS 2006, Part IV. XXXV, 1096 pages. 2006.

Vol. 3993: V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, J. Dongarra (Eds.), Computational Science – ICCS 2006, Part III. XXXVI, 1136 pages. 2006.

Vol. 3992: V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, J. Dongarra (Eds.), Computational Science – ICCS 2006, Part II. XXXV, 1122 pages. 2006.

Vol. 3991: V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, J. Dongarra (Eds.), Computational Science – ICCS 2006, Part I. LXXXI, 1096 pages. 2006.

Vol. 3990: J. C. Beck, B.M. Smith (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. X, 301 pages. 2006.

Vol. 3989: J. Zhou, M. Yung, F. Bao, Applied Cryptography and Network Security. XIV, 488 pages. 2006.

Vol. 3987: M. Hazas, J. Krumm, T. Strang (Eds.), Location- and Context-Awareness. X, 289 pages. 2006.

Vol. 3986: K. Stølen, W.H. Winsborough, F. Martinelli, F. Massacci (Eds.), Trust Management. XIV, 474 pages. 2006.

Vol. 3984: M. Gavrilova, O. Gervasi, V. Kumar, C.J. K. Tan, D. Taniar, A. Laganà, Y. Mun, H. Choo (Eds.), Computational Science and Its Applications - ICCSA 2006, Part V. XXV, 1045 pages. 2006.

Vol. 3983: M. Gavrilova, O. Gervasi, V. Kumar, C.J. K. Tan, D. Taniar, A. Laganà, Y. Mun, H. Choo (Eds.), Computational Science and Its Applications - ICCSA 2006, Part IV. XXVI, 1191 pages. 2006.

Vol. 3982: M. Gavrilova, O. Gervasi, V. Kumar, C.J. K. Tan, D. Taniar, A. Laganà, Y. Mun, H. Choo (Eds.), Computational Science and Its Applications - ICCSA 2006, Part III. XXV, 1243 pages. 2006.

Vol. 3981: M. Gavrilova, O. Gervasi, V. Kumar, C.J. K. Tan, D. Taniar, A. Laganà, Y. Mun, H. Choo (Eds.), Computational Science and Its Applications - ICCSA 2006, Part II. XXVI, 1255 pages. 2006.

Vol. 3980: M. Gavrilova, O. Gervasi, V. Kumar, C.J. K. Tan, D. Taniar, A. Laganà, Y. Mun, H. Choo (Eds.), Computational Science and Its Applications - ICCSA 2006, Part I. LXXV, 1199 pages. 2006.

Vol. 3979: T.S. Huang, N. Sebe, M.S. Lew, V. Pavlović, M. Kölsch, A. Galata, B. Kisačanin (Eds.), Computer Vision in Human-Computer Interaction. XII, 121 pages. 2006.

Vol. 3978: B. Hnich, M. Carlsson, F. Fages, F. Rossi (Eds.), Recent Advances in Constraints. VIII, 179 pages. 2006. (Sublibrary LNAI).

Vol. 3977: N. Fuhr, M. Lalmas, S. Malik, G. Kazai (Eds.), Advances in XML Information Retrieval and Evaluation. XII, 556 pages. 2006.

Vol. 3976: F. Boavida, T. Plagemann, B. Stiller, C. Westphal, E. Monteiro (Eds.), Networking 2006. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems. XXVI, 1276 pages. 2006.

Vol. 3975: S. Mehrotra, D.D. Zeng, H. Chen, B.M. Thuraisingham, F.-Y. Wang (Eds.), Intelligence and Security Informatics. XXII, 772 pages. 2006.

Vol. 3973: J. Wang, Z. Yi, J.M. Zurada, B.-L. Lu, H. Yin (Eds.), Advances in Neural Networks - ISNN 2006, Part III. XXIX, 1402 pages. 2006.

Vol. 3972: J. Wang, Z. Yi, J.M. Zurada, B.-L. Lu, H. Yin (Eds.), Advances in Neural Networks - ISNN 2006, Part II. XXVII, 1444 pages. 2006.

Vol. 3971: J. Wang, Z. Yi, J.M. Zurada, B.-L. Lu, H. Yin (Eds.), Advances in Neural Networks - ISNN 2006, Part I. LXVII, 1442 pages. 2006.

Vol. 3970: T. Braun, G. Carle, S. Fahmy, Y. Koucheryavy (Eds.), Wired/Wireless Internet Communications. XIV, 350 pages. 2006.

Vol. 3969: Ø. Ytrehus (Ed.), Coding and Cryptography. XI, 443 pages. 2006.

Vol. 3968: K.P. Fishkin, B. Schiele, P. Nixon, A. Quigley (Eds.), Pervasive Computing. XV, 402 pages. 2006.

Vol. 3967: D. Grigoriev, J. Harrison, E.A. Hirsch (Eds.), Computer Science – Theory and Applications. XVI, 684 pages. 2006.

Vol. 3966: Q. Wang, D. Pfahl, D.M. Raffo, P. Wernick (Eds.), Software Process Change. XIV, 356 pages. 2006.

Vol. 3965: M. Bernardo, A. Cimatti (Eds.), Formal Methods for Hardware Verification. VII, 243 pages. 2006.

Vol. 3964: M. Ü. Uyar, A.Y. Duale, M.A. Fecko (Eds.), Testing of Communicating Systems. XI, 373 pages. 2006.

Vol. 3963: O. Dikenelli, M.-P. Gleizes, A. Ricci (Eds.), Engineering Societies in the Agents World VI. XII, 303 pages. 2006. (Sublibrary LNAI).

Vol. 3962: W. IJsselsteijn, Y. de Kort, C. Midden, B. Eggen, E. van den Hoven (Eds.), Persuasive Technology. XII, 216 pages. 2006.

Vol. 3960: R. Vieira, P. Quaresma, M.d.G.V. Nunes, N.J. Mamede, C. Oliveira, M.C. Dias (Eds.), Computational Processing of the Portuguese Language. XII, 274 pages. 2006. (Sublibrary LNAI).

Vol. 3959: J.-Y. Cai, S. B. Cooper, A. Li (Eds.), Theory and Applications of Models of Computation. XV, 794 pages. 2006.

Vol. 3958: M. Yung, Y. Dodis, A. Kiayias, T. Malkin (Eds.), Public Key Cryptography - PKC 2006. XIV, 543 pages. 2006.

Table of Contents

Stochastic Reasoning About Channel-Based Component Connectors

Christel Baier[1] and Verena Wolf[2]

[1] Universität Bonn, Germany
baier@cs.uni-bonn.de
[2] Universität Mannheim, Germany
wolf@informatik.uni-mannheim.de

Abstract. Constraint automata have been used as an operational model for component connectors that coordinate the cooperation and communication of the components by means of a network of channels. In this paper, we introduce a variant of constraint automata (called continuous-time constraint automata) that allows us to specify time-dependent stochastic assumptions about the channel connections or the component interfaces, such as the arrival rates of communication requests, the average delay of enabled I/O-operations at the channel ends or the stochastic duration of internal computations. This yields the basis for a performance analysis of channel-based coordination mechanisms. We focus on compositional reasoning and discuss several bisimulation relations on continuous-time constraint automata. For this, we adapt notions of strong and weak bisimulation that have been introduced for similar stochastic models and introduce a new notion of weak bisimulation which abstracts away from invisible non-stochastic computations as well as the internal stochastic evolution.

1 Introduction

Coordination models and languages provide a formalization of the *glue-code* that binds individual components and organizes the communication and cooperation between them. In the past 15 years, various types of coordination models have been proposed that they yield a clear separation between the internal structure of the components and their interactions. See e.g. [19, 24, 13, 25, 15].

The purpose of this paper is to introduce an operational model for reasoning about *stochastic properties* of coordination languages similar to the approaches of Priami [27] and Di Pierro et al. [16]. In contrast to these approaches our focus is on exogenous channel-based coordination languages, such as Reo [2] (see also [5, 29, 1, 17, 14]) and stochastic models with nondeterminism. The rough idea of Reo is that complex component connectors are synthesized from channels via certain composition operators, thus yielding a *network of channels* (called Reo connector circuit) that coordinates the interactions between the components. An operational semantics of Reo connector circuits has been provided by means of *constraint automata* [4]. These are variants of labelled transition systems and encode the configurations of the network by their states and the possible data flow at the ports of the components and the nodes "inside" the network by their transitions. Extensions of constraint automata have been presented in [3] to study real-time constraints of component connectors and in [6] to reason about

P. Ciancarini and H. Wiklicky (Eds.): COORDINATION 2006, LNCS 4038, pp. 1–15, 2006.

channels with a probabilistic effect. The latter approach is time-abstract and deals with discrete probabilities, e.g., to model the faulty behaviours of buffered channels that might loose or corrupt stored messages, while the former approach focusses on a purely timed setting, e.g., to reason about hard deadlines, but does not deal with probabilities. The contribution of this paper is orthogonal to the extensions proposed in [3, 6] as it introduces a stochastic variant of constraint automata where transitions might have a certain delay according to some probability distribution over a continuous time domain. This model, called continuous-time constraint automata (CCA for short), combines the features of ordinary constraint automata with the race conditions in continuous-time Markov chains. CCA are close to *interactive Markov chains* (IMCs), which have been introduced by Hermanns [20] for specifying reactive systems with internal stochastic behaviours. CCA can be used – as ordinary constraint automata – to formalize the stepwise behaviour of the interfaces of the components and the channels connecting them, as well as an operational model for the composite system. In addition, CCA provide the possibility to specify, e.g., the average rate of communication requests of a certain component or the mean time that have to be passed between two consecutive I/O operations at a certain channel. Thus, CCA yield a simple and intuitive model that allow for a performance analysis of channel-based coordination mechanisms.

In this paper, we concentrate on compositional reasoning by means of *bisimulation* relations on CCA. We first consider strong and weak bisimulation, that have been introduced for interactive Markov chains [20]. These notions adapted to CCA yield equivalences that are congruences for the two basic operators (product and hiding) for generating the CCA for a complex component connector out of the CCA for its channels and the component interfaces. Furthermore, we introduce a new notion of weak bisimulation, called *very weak bisimulation* which abstracts away from the stochastic branching behaviour and cumulates the effect of sequences of stochastic transitions. Very weak bisimulation equivalence is coarser than weak bisimulation equivalence, but preserves the probabilities for trace-based linear time properties and can be checked in polynomial-time.

Organization. Section 2 recalls the basic concepts of ordinary constraint automata. CCA and a product and hiding operator on CCA are introduced in Section 3. Section 4 deals with bisimulation relations on CCA. Section 5 concludes the paper. Due to length restrictions, proofs for the theorems are omitted. They can be found in the full version (see `http://pi2.informatik.uni-mannheim.de/HomePages/vwolf/cca.ps`).

2 Constraint Automata

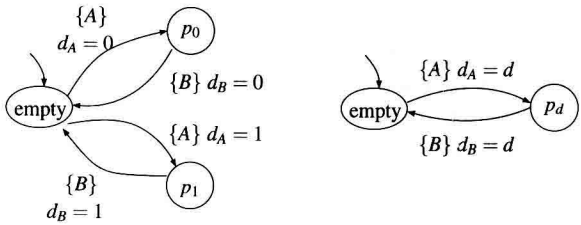
This section summarizes the basis concepts of constraint automata [4] and their use as operational model for channel-based component connectors. Constraint automata, CA for short, are variants of labelled transition systems where transitions are augmented with pairs $\langle N, g \rangle$ rather than action labels. The states of a constraint automata stand for the network configurations, e.g., the contents of the buffers for FIFO channels. The transition labels $\langle N, g \rangle$ can be viewed as *sets of I/O-operations* that will be performed *in parallel*. More precisely, N is a set of nodes in the network where data-flow is observed simultaneously, and g is a boolean condition on the observed data items. Thus,

transitions going out of a state q represent the possible data-flow in the corresponding configuration and its effect on the configuration.

Data assignments and data constraints. CA use a finite set $\mathcal{N}$ of nodes. The nodes can play the role of input and output ports of components, but they can appear outside the interfaces of components as "intermediate" stations of the network where several channels are glued together and the transmission of data items can be observed. For the purposes of this paper, there is no need to distinguish between write and read operations at the nodes. Instead CA only refer to the data items that can be "observed" at a node. Throughout the paper, we assume a fixed, non-empty and finite data domain *Data* consisting of the data items that can be transmitted through the channels. A data assignment for $N \subseteq \mathcal{N}$ means a function $\delta : N \to Data$. We write $\delta.A$ for the data item assigned to node $A \in N$ under δ and $DA(N)$ for the set of all data assignments for node-set N. CA use a symbolic representation of data assignments by data constraints which mean propositional formulae built from the atoms "$d_A = d_B$", "$d_A \in P$" or "$d_A = d$" where A, B are nodes, d_A is a symbol for the observed data item at node A and $d \in Data$, $P \subseteq Data$. The symbol $\models$ stands for the obvious satisfaction relation which results from interpreting data constraints over data assignments. Satisfiability and logical equivalence $\equiv$ of data constraints are defined as usual. We write $DC(N)$ to denote the set of satisfiable data constraints using only the symbols d_A for $A \in N$, but not d_B for $B \in \mathcal{N} \setminus N$.

Constraint automata (CA) [4]. A CA is a tuple $\mathcal{A} = (Q, \mathcal{N}, \longrightarrow, Q_0)$ where Q is a set of states, also called configurations, $\mathcal{N}$ a finite set of nodes, $Q_0 \subseteq Q$ the set of initial states and $\longrightarrow$ a subset of $\bigcup_{N \subseteq \mathcal{N}} Q \times \{N\} \times DC(N) \times Q$, called the transition relation.

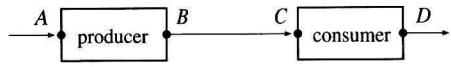
We write $q \xrightarrow{N,g} p$ instead of $(q,N,g,p) \in \longrightarrow$ and refer to N as the node-set and g the guard. Transitions where the node-set N is non-empty are called *visible*, while transitions with the empty node-set are called *hidden*. Each transition represents a set of possible interactions given by the *transition instances* that result by replacing the guard g with a data assignment δ where g holds. The intuitive behaviour of a CA is as follows. The automaton starts in an initial state. If the current state is q then an instance $q \xrightarrow{N,\delta} p$ of the outgoing transitions from q is chosen, the corresponding I/O-operations are performed and the next state is p. If there are several outgoing transitions from state q the next transition is chosen nondeterministically. A formalization of the possible (finite or infinite) observable data flow of a constraint automaton is obtained by the notion of a run. A run in $\mathcal{A}$ denotes a (finite or infinite) sequence of consecutive transition instances $q_0 \xrightarrow{N_0,\delta_0} q_1 \xrightarrow{N_1,\delta_1} q_2 \xrightarrow{N_2,\delta_2} \ldots$ where $q_0 \in Q_0$. For finite runs we require that the last state q does not have an outgoing hidden transition. This can be understood as a *maximal progress assumption* for hidden transitions, i.e., steps that do not require any interaction with the environment.



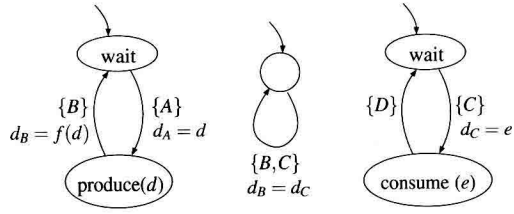
The picture above shows a CA for a FIFO1 channel with the node-set $\mathcal{N} = \{A, B\}$ and $Data = \{0, 1\}$. Node A serves as input port where data items can be written into the channel, while node B can be regarded as an output port where the stored data element is taken from the buffer and delivered to the environment.

State "empty" stands for the configuration where the buffer is empty, while state p_d encodes the configuration where d is stored in the buffer. Often, we use simplified parametric pictures for CA with meta symbols for data items as in the right of the picture (and formally explained in [4]).

For another example, we regard a simple system consisting of a producer and a consumer which are linked via a synchronous channel for transmitting the generated products from the producer's output port B to the consumer's input port C.



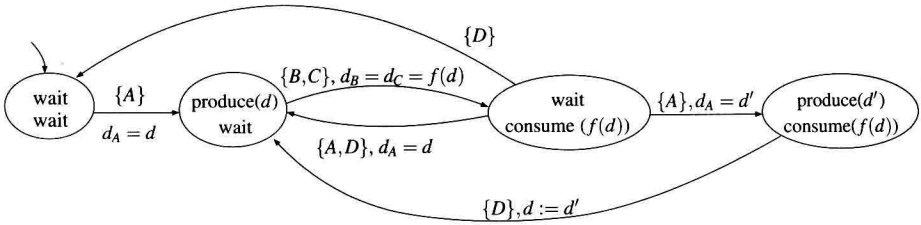
We model both components (producer and consumer) and the synchronous channel BC by CA. Parametric pictures are shown below. We assume here that the producer is activated by obtaining an input value d from the environment at its input port A. It then generates a certain product $f(d)$ which is synchronously delivered to the consumer. After having received $e = f(d)$, the consumer starts the consume-phase and finally sends a signal via output port D to the environment. We assume here that the value send off at D is arbitrary, that is, we deal with the valid guard true (which is omitted in the picture).



Product. To obtain a constraint automata for the composite producer-consumer-system, we apply a product construction to the three CA. The product of two CA $\mathcal{A}_1 = (Q_1, \mathcal{N}_1, \longrightarrow_1, Q_{0,1})$ and $\mathcal{A}_2 = (Q_2, \mathcal{N}_2, \longrightarrow_2, Q_{0,2})$ is defined as follows. $\mathcal{A}_1 \bowtie \mathcal{A}_2$ is a CA with the components $(Q_1 \times Q_2, \mathcal{N}_1 \cup \mathcal{N}_2, \longrightarrow, Q_{0,1} \times Q_{0,2})$ where $\longrightarrow$ is given by the following rules:

- If $q_1 \xrightarrow{N_1, g_1}_1 p_1$, $q_2 \xrightarrow{N_2, g_2}_2 p_2$, $N_1 \cap \mathcal{N}_2 = N_2 \cap \mathcal{N}_1 \neq \emptyset$ then $\langle q_1, q_2 \rangle \xrightarrow{N_1 \cup N_2, g_1 \wedge g_2} \langle p_1, p_2 \rangle$, provided that $g_1 \wedge g_2$ is satisfiable.
- If $q_1 \xrightarrow{N,g}_1 p_1$ where $N \cap \mathcal{N}_2 = \emptyset$ then $\langle q_1, q_2 \rangle \xrightarrow{N,g} \langle p_1, q_2 \rangle$.
- If $q_2 \xrightarrow{N,g}_2 p_2$ where $N \cap \mathcal{N}_1 = \emptyset$ then $\langle q_1, q_2 \rangle \xrightarrow{N,g} \langle q_1, p_2 \rangle$.

The former rule expresses the synchronization case which means that both automata have to "agree" on the I/O-operations at their common nodes, while the I/O-operations at their individual nodes is arbitrary. The latter two rules are in the style of classical interleaving rules for labelled transition systems. They formalize the case where no synchronization is required since no common nodes are involved. A parametric picture for the product of the CA of the producer, the consumer and the synchronous channel BC has the following form:



Hiding. Another operator that is helpful for abstraction purposes and can be used in Reo to built components from networks by declaring the internal structure of the network as hidden (i.e., invisible for the environment) is the hiding operator. It takes as input a CA $\mathcal{A} = (Q, \mathcal{N}, \longrightarrow, Q_0)$ and a non-empty node-set $M \subseteq \mathcal{N}$. The result is a CA $\mathsf{hide}(\mathcal{A}, M)$ that behaves as $\mathcal{A}$, except that data flow at the nodes $A \in M$ is made invisible. Formally, $\mathsf{hide}(\mathcal{A}, M) = \big(Q, \mathcal{N} \setminus M, \longrightarrow_M, Q_{0,M}\big)$ where

$$q \xrightarrow{\bar{N},\bar{g}}_M p \text{ iff there exists a transition } q \xrightarrow{N,g} p \text{ such that } \bar{N} = N \setminus M \text{ and } \bar{g} = \exists M[g].$$

$\exists M[g]$ stands short for $\bigvee_{\delta \in DA(M)} g[d_A/\delta.A \mid A \in M]$, where $g[d_A/\delta.A \mid A \in M]$ denotes the syntactic replacement of all occurrences of d_A in g for $A \in M$ with $\delta.A$. Thus, $\exists M[g]$ formalizes the set of data assignments for $\bar{N} = N \setminus M$ that are obtained from a data assignment δ for N where g holds by dropping the assignments for the nodes $A \in N \cap M$. For example, hiding nodes B and C in the CA $\mathcal{A}$ for the producer-consumer system yields a CA $\mathcal{A}' = \mathsf{hide}(\mathcal{A}, \{B,C\})$ with node-set $\{A,D\}$. $\mathcal{A}'$ has the same structure as $\mathcal{A}$, the only difference being that the $\{B,C\}$-transition in $\mathcal{A}$ becomes a hidden transition in $\mathcal{A}'$.

3 Continuous-Time Constraint Automata

We now present a stochastic extension of constraint automata that yields the basis for a performance analysis of channel-based component connectors, e.g. to reason about expected response times, the average number of messages that are stored in a buffer of a FIFO channel, the stochastic long-run behaviour or verifying soft deadlines such as "there is a 95% chance to obtain a message at input port B within 10 time units after having sent a request from output port A". Continuous-time constraint automata (CCA for short) rely on the assumption that hidden transitions are performed as soon as possible, while enabled I/O-operations at (non-hidden) nodes can occur at any moment or even can be refused. The idea is that the environment might connect to the non-hidden nodes and might either agree to perform a communication immediately, might cause a delay of a certain communication or might even be not willing to cooperate. CCA are most in the spirit of interactive Markov chains (IMC) that have been introduced by Hermanns [20] and that are closely related to continuous-time Markov decision processes [28]. As in IMCs we have two types of transitions:

- *interactive* transitions $q \xrightarrow{N,g} p$ as in ordinary constraint automata, and
- *Markovian transitions* $q \xrightarrow{\lambda} p$ where λ is a positive real number, called rate.

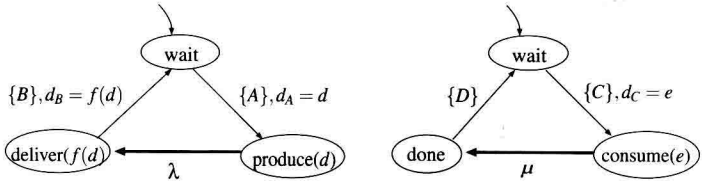
The interpretation of the rates is as in continuous-time Markov chains, see e.g. [22], i.e., with probability $1 - e^{-\lambda t}$ the delay of a Markovian transition with rate λ is less

or equal t. If there are two or more outgoing Markovian transitions from q and no interactive transition is taken from q then the transition with the least delay (i.e., the transition that is enabled first) will fire. Note that rates and average delays are dual in the sense that average delay Λ stands for the rate $\lambda = 1/\Lambda$.

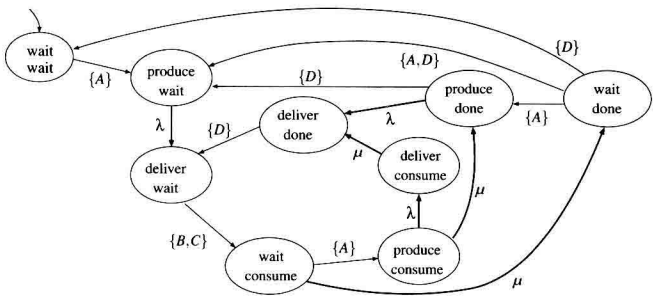
Definition 1 (Continuous-time constraint automata (CCA)). A CCA is a tuple $C = (Q, \mathcal{N}, \longrightarrow, Q_0)$ where Q is a countable set of states, $Q_0 \subseteq Q$ the set of initial states, $\mathcal{N}$ is a finite set of nodes and $\longrightarrow \subseteq (Q \times \mathbb{R}_{>0} \times Q) \cup \left(\bigcup_{N \subseteq \mathcal{N}} Q \times \{N\} \times DC(N) \times Q\right)$ such that for all states q and p there is at most one Markovian transition from q to p. □

We write $\mathbf{R}(q,p) = \lambda$ if $q \xrightarrow{\lambda} p$ and $\mathbf{R}(q,p) = 0$ if there is no Markovian transition from q to p. For mathematical reasons, we require that for each state the exit rate $E(q)$ defined by $\sum_{p \in Q} \mathbf{R}(q,p)$ is finite and that there does not exist an infinite path consisting of consecutive interactive transitions. (The latter assumptions are irrelevant for the purposes of this paper, but they are necessary to ensure non-zenoness.) When state q is entered then either *immediately* a hidden transition instance is taken or the system stays in state q until one of the Markovian transitions becomes enabled and fires or a visible interactive transition is taken. A visible transition instance $q \xrightarrow{N,\delta} p$ can only be taken if all involved nodes $A \in N$ agree to perform the I/O-operations specified by (N,δ). If N is non-empty then this agreement depends on the (unknown) environment which might refuse to provide the required I/O-operations at the nodes $A \in N$. Thus, none of the visible transitions might be taken. If, however, the current state q has one or more outgoing hidden transitions there is a *nondeterministic choice* which selects one of the interactive (visible or hidden) transitions. Thus, Markovian transitions can only be taken from state q if there is no hidden transition that starts in q, in which case q is called a *Markovian state*.

The possible stepwise behaviours of a CCA can be made precise by means of the runs and the induced stochastic processes. A run in a CCA C is a sequence of consecutive transition instances $q_0 \xrightarrow{\alpha_0} q_1 \xrightarrow{\alpha_1} q_2 \xrightarrow{\alpha_2} \ldots$ where the α_i's are either triples (t,N,δ) such that $q_i \xrightarrow{N,\delta} q_{i+1}$ is an instance of an interactive transition and $t \geq 0$ (the time passage between entering state q_i and performing the I/O-operations specified by (N,δ)) or $\alpha_i \in \mathbb{R}_{>0}$ and there is a Markovian transition from q_i to q_{i+1}. In the latter case, α_i stands for the amount of time the system spends in state q_i until the first Markovian transition fires. According to the maximal progress assumption we require that Markovian transitions and that $\alpha_i = (t,N,\delta)$ for some $t > 0$ can only occur if no hidden transition can be taken in q_i and that finite runs end in a state where all outgoing transitions are visible. To reason about the probabilities of runs, the concept of schedulers, also often called policy, strategy or adversary, is needed. The details, which can be found e.g. in [28], are not of importance here. We just mention that a scheduler takes as input the history of the system, formalized by a finite prefix of a run, and either selects one of the enabled interactive transition instances or, if no hidden transition can be taken, decides to take a visible transition instance with some delay t unless a Markovian transition fires first or decides to take no interactive transition and to wait for the first enabled Markovian transition. For any given scheduler a probability measure on the induced runs can be defined which, for instance, allows to speak about the probability to reach a certain configuration within t time units or the expected time until a certain communication takes place.



Example 1. The picture above shows a stochastic variant for the CA of the producer and the consumer. Here, we assume that the production time takes in average $1/\lambda$ time units, while the mean time of the consume phase is $1/\mu$. The data-abstract behavior of the composite system can then be specified by the CCA shown below. This CCA can now be subject of a stochastic analysis. For example, it can be verified that the average time of one production-consume cycle is $1/\lambda + 1/\mu$, or that the probability for the event "after being activated through an input at A, the time for delivering the product via channel BC is less or equal t" is given by $1 - e^{-\lambda t}$. □



Beside specifying stochastic phenomena that are internal to certain components, also channels might have stochastic behaviours and can be modelled by CCA. E.g., if a component Comp, that is linked to the sink end of a FIFO1 channel c, is waiting for a message along c then Comp cannot immediately read when a message is written at the source end. Instead it has to wait for a certain (possibly very small) amount of time until the read operation can be performed. As long as we consider any channel in isolation these delays might be very small or even negligible. However, for complex networks where several channels are composed, the effect of delays becomes less clear and can play a crucial role for performability issues. Assume a FIFO channel c with 1.000 buffer cells is composed from 1.000 copies of FIFO1 channel with average delay Λ then the mean time passage between writing a data item d into c's source end and the instant where d can be taken at the sink end is $1.000 \cdot \Lambda$.

Example 2. A FIFO1 channel with average delay $1/\lambda$ between the read and write operations can be modelled by one of the CCA shown below. In both CCA, after the write operation at the source A the state wait(d) is reached where a Markovian transition with rate λ is emanating, leading to state take(d) where the sink B can take the element. In the CCA on the left, no proper delay between the read operation at sink B and the next write operation at A is specified, while the automaton on the right relies on the assumption that the physical properties of the buffer yield an average delay $1/\mu$ for enabling a write operation after a read operation. □