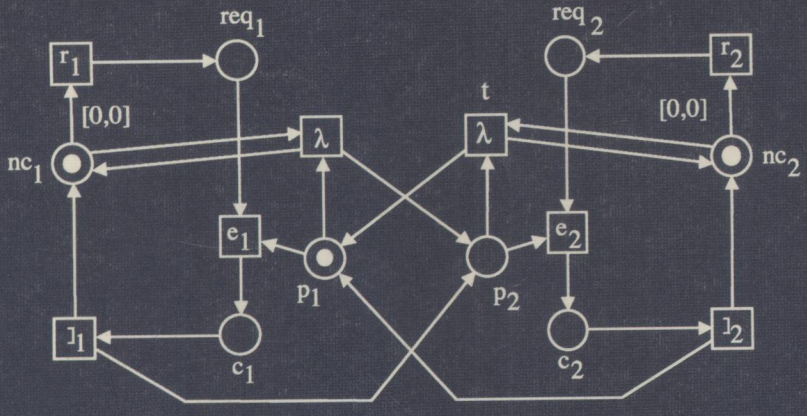
Marco Bernardo
Flavio Corradini (Eds.)

LNCS 3185 Tutorial

# Formal Methods for the Design of Real-Time Systems

**International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004 Bertinoro, Italy, September 2004, Revised Lectures**




Springer

Marco Bernardo Flavio Corradini (Eds.)

# Formal Methods for the Design of Real-Time Systems

International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004
Bertinoro, Italy, September 13-18, 2004
Revised Lectures



Springer

Volume Editors

Marco Bernardo
Università di Urbino "Carlo Bo", Istituto di Scienze e Tecnologie dell'Informazione
Piazza della Repubblica 13, 61029 Urbino, Italy
E-mail: bernardo@sti.uniurb.it

Flavio Corradini
Universitá di L'Aquila, Dipartimento di Informatica
E-mail: flavio@di.univaq.it

Library of Congress Control Number: 2004111362

CR Subject Classification (1998): D.2, D.3, F.3, C.3, C.2.4

ISSN 0302-9743
ISBN 3-540-23068-8 Springer Berlin Heidelberg New York



Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11315995 06/3142 5 4 3 2 1 0

# Lecture Notes in Computer Science 3185

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

## Editorial Board

# Preface

A large class of computing systems can be specified and verified by abstracting away from the temporal aspects of their behavior. In *real-time systems*, instead, time issues become essential. Their correctness depends not only on which actions they can perform, but also on the action execution time. Due to their importance and design challenges, real-time systems have attracted the attention of a considerable number of computer scientists and engineers from various research areas.

This volume collects a set of papers accompanying the lectures of the fourth edition of the *International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM)*. The school addressed the use of formal methods in computer science as a prominent approach to the rigorous design of computer, communication and software systems. The main aim of the SFM series is to offer a good spectrum of current research in foundations as well as applications of formal methods, which can be of help for graduate students and young researchers who intend to approach the field.

SFM-04:RT was devoted to real-time systems. It covered formal models and languages for the specification, modeling, analysis, and verification of these time-critical systems, the expressiveness of such models and languages, as well as supporting tools and related applications in different domains.

The opening paper by Rajeev Alur and Parthasarathy Madhusudan provides a survey of the theoretical results concerning decision problems of reachability, language inclusion, and language equivalence for timed automata. The survey is concluded with a discussion of some open problems. Elmar Bihler and Walter Vogler's paper presents timed extensions of Petri nets with continuous and discrete time and a natural testing-based faster-than relation for comparing asynchronous systems. Several applications of the theory are also presented. Jos C.M. Baeten and Michel A. Reniers present the theory and application of classical process algebras extended with different notions of time and time passing and compare their expressiveness via embeddings and conservative extensions. The PAR communication protocol is considered as a case study. The expressiveness of existing timed process algebras that deal with temporal aspects by following very different interpretations is also the main theme of Diletta R. Cacciagrano and Flavio Corradini's paper. In addition, they compare the expressiveness of urgent, lazy and maximal progress tests. Mario Bravetti presents a theory of probabilistic timed systems where durations are expressed by generally distributed random variables. The theory supports the specification of both real-time and stochastic time during the design and analysis of concurrent systems. Bran Selic, instead, provides an overview of the foundations of the run-time semantics underlying the Unified Modeling Language (UML) as defined in revision 2.0 of the official OMG standard.

After these contributions on formal timed models, timed languages and their expressiveness, the volume includes the description of three significant tools supporting the specification, modeling, analysis and verification of real-time systems. Gerd Behrmann, Alexandre David and Kim G. Larsen's tutorial paper on the tool Uppaal provides an introduction to the implementation of timed automata in the tool, the user interface, and the usage of the tool. Reference examples and modeling patterns are also presented. Marius Bozga, Susanne Graf, Ileana Ober, Iulian Ober, and Joseph Sifak present an overview on the IF toolset, which is an environment for the modeling and validation of heterogeneous real-time systems. The toolset is built upon a rich formalism, the IF notation, allowing structured automata-based system representations. A case study concerning the Ariane-5 Flight Program is presented. Finally, Joost-Pieter Katoen, Henrik Bohnenkamp, Ric Klaren, and Holger Hermanns survey the language Modest, a modeling and description language for stochastic and timed systems, and its accompanying tool environment MOTOR. The modeling and analysis with this tool of a device-absence-detecting protocol in plug-and-play networks is reported in the paper.

We believe that this book offers a quite comprehensive view of what has been done and what is going on worldwide at present in the field of real-time models and languages for the specification, analysis, and verification of time-critical systems. We wish to thank all the lecturers and all the participants for a lively and fruitful school. We also wish to thank the whole staff of the University Residential Center of Bertinoro (Italy) for the organizational and administrative support, as well as the sponsors of the school – AICA and ONRG – for making it possible through the provision of grants to some of the participants.

September 2004 Marco Bernardo and Flavio Corradini

# Lecture Notes in Computer Science

For information about Vols. 1–3103

please contact your bookseller or Springer

Vol. 3232: R. Heery, L. Lyon (Eds.), Research and Advanced Technology for Digital Libraries. XV, 528 pages. 2004.

Vol. 3223: K. Slind, A. Bunker, G. Gopalakrishnan (Eds.), Theorem Proving in Higher Order Logic. VIII, 337 pages. 2004.

Vol. 3221: S. Albers, T. Radzik (Eds.), Algorithms – ESA 2004. XVIII, 836 pages. 2004.

Vol. 3220: J.C. Lester, R.M. Vicari, F. Paraguaçu (Eds.), Intelligent Tutoring Systems. XXI, 920 pages. 2004.

Vol. 3208: H.J. Ohlbach, S. Schaffert (Eds.), Principles and Practice of Semantic Web Reasoning. VII, 165 pages. 2004.

Vol. 3207: L.T. Jang, M. Guo, G.R. Gao, N.K. Jha, Embedded and Ubiquitous Computing. XX, 1116 pages. 2004.

Vol. 3206: P. Sojka, I. Kopecek, K. Pala (Eds.), Text, Speech and Dialogue. XIII, 667 pages. 2004. (Subseries LNAI).

Vol. 3205: N. Davies, E. Mynatt, I. Siio (Eds.), UbiComp 2004: Ubiquitous Computing. XVI, 452 pages. 2004.

Vol. 3203: J. Becker, M. Platzner, S. Vernalde (Eds.), Field Programmable Logic and Application. XXX, 1198 pages. 2004.

Vol. 3199: H. Schepers (Ed.), Software and Compilers for Embedded Systems. X, 259 pages. 2004.

Vol. 3198: G.-J. de Vreede, L.A. Guerrero, G. Marín Raventós (Eds.), Groupware: Design, Implementation and Use. XI, 378 pages. 2004.

Vol. 3194: R. Camacho, R. King, A. Srinivasan (Eds.), Inductive Logic Programming. XI, 361 pages. 2004. (Subseries LNAI).

Vol. 3193: P. Samarati, P. Ryan, D. Gollmann, R. Molva (Eds.), Computer Security – ESORICS 2004. X, 457 pages. 2004.

Vol. 3192: C. Bussler, D. Fensel (Eds.), Artificial Intelligence: Methodology, Systems, and Applications. XIII, 522 pages. 2004. (Subseries LNAI).

Vol. 3189: P.-C. Yew, J. Xue (Eds.), Advances in Computer Systems Architecture. XVII, 598 pages. 2004.

Vol. 3186: Z. Bellahsène, T. Milo, M. Rys, D. Suciu, R. Unland (Eds.), Database and XML Technologies. X, 235 pages. 2004.

Vol. 3185: M. Bernardo, F. Corradini (Eds.), Formal Methods for the Design of Real-Time Systems. VII, 295 pages. 2004.

Vol. 3184: S. Katsikas, J. Lopez, G. Pernul (Eds.), Trust and Privacy in Digital Business. XI, 299 pages. 2004.

Vol. 3183: R. Traunmüller (Ed.), Electronic Government. XIX, 583 pages. 2004.

Vol. 3182: K. Bauknecht, M. Bichler, B. Pröll (Eds.), E-Commerce and Web Technologies. XI, 370 pages. 2004.

Vol. 3181: Y. Kambayashi, M. Mohania, W. Wöß (Eds.), Data Warehousing and Knowledge Discovery. XIV, 412 pages. 2004.

Vol. 3180: F. Galindo, M. Takizawa, R. Traunmüller (Eds.), Database and Expert Systems Applications. XXI, 972 pages. 2004.

Vol. 3179: F.J. Perales, B.A. Draper (Eds.), Articulated Motion and Deformable Objects. XI, 270 pages. 2004.

Vol. 3178: W. Jonker, M. Petkovic (Eds.), Secure Data Management. VIII, 219 pages. 2004.

Vol. 3177: Z.R. Yang, H. Yin, R. Everson (Eds.), Intelligent Data Engineering and Automated Learning – IDEAL 2004. XVIII, 852 pages. 2004.

Vol. 3176: O. Bousquet, U. von Luxburg, G. Rätsch (Eds.), Advanced Lectures on Machine Learning. VIII, 241 pages. 2004. (Subseries LNAI).

Vol. 3175: C.E. Rasmussen, H.H. Bülthoff, B. Schölkopf, M.A. Giese (Eds.), Pattern Recognition. XVIII, 581 pages. 2004.

Vol. 3174: F. Yin, J. Wang, C. Guo (Eds.), Advances in Neural Networks - ISNN 2004. XXXV, 1021 pages. 2004.

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. 3170: P. Gardner, N. Yoshida (Eds.), CONCUR 2004 - Concurrency Theory. XIII, 529 pages. 2004.

Vol. 3166: M. Rauterberg (Ed.), Entertainment Computing – ICEC 2004. XXIII, 617 pages. 2004.

Vol. 3163: S. Marinai, A. Dengel (Eds.), Document Analysis Systems VI. XII, 564 pages. 2004.

Vol. 3162: R. Downey, M. Fellows, F. Dehne (Eds.), Parameterized and Exact Computation. X, 293 pages. 2004.

Vol. 3160: S. Brewster, M. Dunlop (Eds.), Mobile Human-Computer Interaction – MobileHCI 2004. XVIII, 541 pages. 2004.

Vol. 3159: U. Visser, Intelligent Information Integration for the Semantic Web. XIV, 150 pages. 2004. (Subseries LNAI).

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. 3155: P. Funk, P.A. González Calero (Eds.), Advances in Case-Based Reasoning. XIII, 822 pages. 2004. (Subseries LNAI).

Vol. 3154: R.L. Nord (Ed.), Software Product Lines. XIV, 334 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 Augmented Reality. XII, 378 pages. 2004.

Vol. 3149: M. Danelutto, M. Vanneschi, D. Laforenza (Eds.), Euro-Par 2004 Parallel Processing. XXXIV, 1081 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. 3137: P. De Bra, W. Nejdl (Eds.), Adaptive Hypermedia and Adaptive Web-Based Systems. XIV, 442 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. 3132: B. Demoen, V. Lifschitz (Eds.), Logic Programming. XII, 480 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. 3122: K. Jansen, S. Khanna, J.D.P. Rolim, D. Ron (Eds.), Approximation, Randomization, and Combinatorial Optimization. IX, 428 pages. 2004.

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. 3115: P. Enser, Y. Kompatsiaris, N.E. O'Connor, A.F. Smeaton, A.W.M. Smeulders (Eds.), Image and Video Retrieval. XVII, 679 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.

# Preface

A large class of computing systems can be specified and verified by abstracting away from the temporal aspects of their behavior. In *real-time systems*, instead, time issues become essential. Their correctness depends not only on which actions they can perform, but also on the action execution time. Due to their importance and design challenges, real-time systems have attracted the attention of a considerable number of computer scientists and engineers from various research areas.

This volume collects a set of papers accompanying the lectures of the fourth edition of the *International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM)*. The school addressed the use of formal methods in computer science as a prominent approach to the rigorous design of computer, communication and software systems. The main aim of the SFM series is to offer a good spectrum of current research in foundations as well as applications of formal methods, which can be of help for graduate students and young researchers who intend to approach the field.

SFM-04:RT was devoted to real-time systems. It covered formal models and languages for the specification, modeling, analysis, and verification of these time-critical systems, the expressiveness of such models and languages, as well as supporting tools and related applications in different domains.

The opening paper by Rajeev Alur and Parthasarathy Madhusudan provides a survey of the theoretical results concerning decision problems of reachability, language inclusion, and language equivalence for timed automata. The survey is concluded with a discussion of some open problems. Elmar Bihler and Walter Vogler's paper presents timed extensions of Petri nets with continuous and discrete time and a natural testing-based faster-than relation for comparing asynchronous systems. Several applications of the theory are also presented. Jos C.M. Baeten and Michel A. Reniers present the theory and application of classical process algebras extended with different notions of time and time passing and compare their expressiveness via embeddings and conservative extensions. The PAR communication protocol is considered as a case study. The expressiveness of existing timed process algebras that deal with temporal aspects by following very different interpretations is also the main theme of Diletta R. Cacciagrano and Flavio Corradini's paper. In addition, they compare the expressiveness of urgent, lazy and maximal progress tests. Mario Bravetti presents a theory of probabilistic timed systems where durations are expressed by generally distributed random variables. The theory supports the specification of both real-time and stochastic time during the design and analysis of concurrent systems. Bran Selic, instead, provides an overview of the foundations of the run-time semantics underlying the Unified Modeling Language (UML) as defined in revision 2.0 of the official OMG standard.

# Table of Contents

## Part I: Models and Languages



## Part II: Tools and Applications

# Decision Problems for Timed Automata: A Survey*

Rajeev Alur and P. Madhusudan

University of Pennsylvania

**Abstract.** Finite automata and regular languages have been useful in a wide variety of problems in computing, communication and control, including formal modeling and verification. Traditional automata do not admit an explicit modeling of time, and consequently, *timed automata* [2] were introduced as a formal notation to model the behavior of real-time systems. Timed automata accept *timed languages* consisting of sequences of events tagged with their occurrence times. Over the years, the formalism has been extensively studied leading to many results establishing connections to circuits and logic, and much progress has been made in developing verification algorithms, heuristics, and tools. This paper provides a survey of the theoretical results concerning decision problems of reachability, language inclusion and language equivalence for timed automata and its variants, with some new proofs and comparisons. We conclude with a discussion of some open problems.

## 1 Timed Automata

A timed automaton is a finite automaton augmented with a finite set of (real-valued) *clocks*. The vertices of the automaton are called *locations*, and edges are called *switches*. While switches are instantaneous, time can elapse in a location. A clock can be reset to zero simultaneously with any switch. At any instant, the reading of a clock equals the time elapsed since the last time it was reset. With each switch we associate a clock constraint, and require that the switch may be taken only if the current values of the clocks satisfy this constraint. Timed automata accept (or, equivalently, generate) timed words, that is, strings of symbols tagged with occurrence times. Let $\mathbb{R}$ denote the set of nonnegative real numbers, and let $\mathbb{Q}$ denote the set of nonnegative rational numbers. A *timed word* over an alphabet $\Sigma$ is a sequence $(a_0, t_0), (a_1, t_1) \cdots (a_k, t_k)$, where each $a_i \in \Sigma$, each $t_i \in \mathbb{R}$, and the occurrence times increase monotonically: $t_0 \leq t_1 \leq \cdots \leq t_k$. The set of all timed words over $\Sigma$ is denoted $T\Sigma^*$. A *timed language* over $\Sigma$ is a subset of $T\Sigma^*$.

The *untimed* word corresponding to a timed word $(a_0, t_0), (a_1, t_1) \cdots (a_k, t_k)$ is the word $a_0 a_1 \ldots a_k$ obtained by deleting the occurrence times. The untimed language $untime(L)$ of a timed language $L$ consists of all the untimed words corresponding to the timed words in $L$. For an alphabet $\Sigma$, we use $\Sigma^\epsilon$ to denote

* This research was partially supported by NSF award ITR/SY 0121431.

M. Bernardo and F. Corradini (Eds.): SFM-RT 2004, LNCS 3185, pp. 1–24, 2004

$\Sigma \cup \{\epsilon\}$ (where $\epsilon$ is not in $\Sigma$), and for a subset $\Sigma' \subseteq \Sigma$, and a timed word $w = (a_0, t_0), (a_1, t_1) \cdots (a_k, t_k)$ over $\Sigma$, the projection of $w$ over $\Sigma'$ is obtained from $w$ by deleting all $(a_i, t_i)$ such that $a_i \notin \Sigma'$. The projection operation extends to timed languages as well.

To define timed automata formally, we need to say what type of clock constraints are allowed as guards. For a set $X$ of clocks, the set $\Phi(X)$ of *clock constraints* $g$ is defined by the grammar

$$g := x \leq c \mid c \leq x \mid x < c \mid c < x \mid g \wedge g$$

where $x \in X$ and $c \in \mathbb{Q}$. A *clock valuation* $\nu$ for a set $X$ of clocks assigns a real value to each clock; that is, it is a mapping from $X$ to $\mathbb{R}$. For $\delta \in \mathbb{R}$, $\nu + \delta$ denotes the clock valuation which maps every clock $x$ to the value $\nu(x) + \delta$. For $Y \subseteq X$, $\nu[Y := 0]$ denotes the clock valuation for $X$ which assigns 0 to each $x \in Y$, and agrees with $\nu$ over the rest of the clocks.

A *timed automaton* $A$ over an alphabet $\Sigma$ is a tuple $\langle V, V^0, V^F, X, E \rangle$, where

- $V$ is a finite set of locations,
- $V^0 \subseteq V$ is a set of initial locations,
- $V^F \subseteq V$ is a set of final locations,
- $X$ is a finite set of clocks,
- $E \subseteq V \times \Sigma^\epsilon \times \Phi(X) \times 2^X \times V$ is a set of switches. A switch $\langle s, a, g, \lambda, s' \rangle$ represents an edge from location $s$ to location $s'$ on symbol $a$. The *guard* $g$ is a clock constraint over $X$ that specifies when the switch is enabled, and the *update* $\lambda \subseteq X$ gives the clocks to be reset to 0 with this switch.

The semantics of a timed automaton $A$ is defined by associating an infinite-state automaton $S_A$ over the alphabet $\Sigma \cup \mathbb{R}$. A state of $S_A$ is a pair $(s, \nu)$ such that $s$ is a location of $A$ and $\nu$ is a clock valuation for $X$. A state $(s, \nu)$ is an initial state if $s$ is an initial location (i.e. $s \in V^0$) and $\nu(x) = 0$ for all clocks $x$. A state $(s, \nu)$ is a final state if $s$ is a final location (i.e. $s \in V^F$). There are two types of transitions in $S_A$:

**Elapse of time:** for a state $(s, \nu)$ and a time increment $\delta \in \mathbb{R}$, $(s, \nu) \xrightarrow{\delta} (s, \nu + \delta)$.

**Location switch:** for a state $(s, \nu)$ and a switch $\langle s, a, g, \lambda, s' \rangle$ such that $\nu$ satisfies the guard $g$, $(s, \nu) \xrightarrow{a} (s', \nu[\lambda := 0])$.

For a timed word $w = (a_0, t_0), (a_1, t_1) \cdots (a_k, t_k)$ over $\Sigma^\epsilon$, a *run* of $A$ over $w$ is a sequence

$$q_0 \xrightarrow{t_0} q_0' \xrightarrow{a_0} q_1 \xrightarrow{t_1 - t_0} q_1' \xrightarrow{a_1} q_2 \xrightarrow{t_2 - t_1} q_2' \xrightarrow{a_2} q_3 \rightarrow \cdots \xrightarrow{a_k} q_{k+1}$$

such that $q_0$ is an initial state of $S_A$. The run is *accepting* if $q_{k+1}$ is a final state of $S_A$. The timed automaton $A$ accepts a timed word $w$ over $\Sigma$ if there exists a timed word $w'$ over $\Sigma^\epsilon$ such that $A$ has an accepting run over $w'$ and the projection of $w'$ to $\Sigma$ is $w$. The set of timed words accepted by $A$ is denoted $L(A)$.

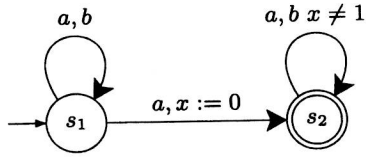



**Fig. 1.** A non-complementable timed automaton.

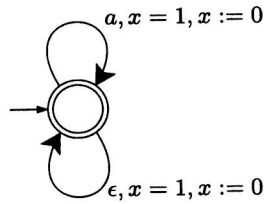



**Fig. 2.** $\epsilon$-transitions increase expressiveness.

A timed language $L \subseteq T\Sigma^*$ is said to be *timed regular* if there exists a timed automaton $A$ such that $L(A) = L$. The closure properties of timed regular languages are summarized below:

**Theorem 1.** *The set of timed regular languages is closed under union, intersection, and projection, but not under complementation [2].*

The closure under union and intersection is established by extending the classical product construction to timed automata. Closure under projection is immediate since switches can be labeled with $\epsilon$.

For the non-closure under complementation, we give a new proof here. Let $\Sigma = \{a, b\}$. Let $L$ be the timed language consisting of timed words $w$ containing an $a$ event at some time $t$ such that no event occurs at time $t + 1$. The (nondeterministic) timed automaton shown in Figure 1 (with initial location $s_1$ and final location $s_2$) accepts $L$.

We claim that $\overline{L}$, the complement of $L$, is not timed regular. Consider the timed language $L'$ consisting of timed words $w$ such that the untimed word of $w$ is in $a^*b^*$, all the $a$ events happen before time 1, and no two $a$ events happen at the same time. Verify that $L'$ is timed regular. Observe that a word of the form $a^n b^m$ belongs to $untime(\overline{L} \cap L')$ iff $m \geq n$. Since timed regular languages are closed under intersection, the untimed language of a timed regular language is regular (see Section 2), and the language $\{a^n b^m \mid m \geq n\}$ is not regular, it follows that $\overline{L}$ is not timed regular.

Unlike classical automata, $\epsilon$-labeled switches add to the expressive power of timed automata [10]. For example, the automaton of Figure 2 accepts timed words $w$ over $\{a\}$ such that every occurrence time is an integer and no two $a$-events occur at the same time. This language cannot be accepted by a timed automaton if $\epsilon$-labeled switches are disallowed: if the largest constant in a timed automaton $A$ is $c$ and $A$ does not have $\epsilon$-labeled switches, then $A$ cannot distinguish between the words $(a, c + 1)$ and$(a, c + 1.1)$.

The more recent definitions of timed automata also admit labeling of each location with a clock constraint called its *invariant*, and require that time can elapse in a location only as long as its invariant stays true [23]. While this is a useful modeling concept to enforce upper bounds (without introducing "error" locations), it does not add to the expressive power.

Timed languages can also be defined using *timed state sequences*: a timed state sequence is a mapping from a prefix of the reals to a finite alphabet that can be represented by a sequence $(a_o, I_0)(a_1, I_1)\ldots(a_k, I_k)$, where $I_0, I_1, \ldots I_k$ is a sequence of adjoining intervals (e.g. $[0, 1.1)[1.1, 1.2](1.2, 1.7)$). Timed state sequences can be generated by timed automata in which locations are labeled with observations [23, 3]. This dual view does not change the core results, but some expressiveness results do differ in the two views [34].

## 2 Reachability and Language Emptiness

### 2.1 Region Automata

Given a timed automaton $A$, to check whether the language $L(A)$ is empty, we must determine if some final state is reachable from an initial state in the infinite-state system $S_A$. The solution to this reachability problem involves construction of a finite quotient. The construction uses an equivalence relation on the state-space that equates two states with the same location if they agree on the integral parts of all clock values and on the ordering of the fractional parts of all clock values. The integral parts of the clock values are needed to determine whether or not a particular clock constraint is met, whereas the ordering of the fractional parts is needed to decide which clock will change its integral part first. This is formalized as follows. First, assume that all the constants in the given timed automaton $A$ are integers (if $A$ uses rational constants, we can simply multiply each constant with the least-common-multiple of all the denominators to get an automaton with the same timed language modulo scaling). For any $\delta \in \mathbb{R}$, $\langle\delta\rangle$ denotes the fractional part of $\delta$, and $\lfloor\delta\rfloor$ denotes the integral part of $\delta$; $\delta = \lfloor\delta\rfloor + \langle\delta\rangle$. For each clock $x \in X$, let $c_x$ be the largest integer $c$ such that $x$ is compared with $c$ in some clock constraint appearing in a guard. The equivalence relation $\cong$, called the *region equivalence*, is defined over the set of all clock valuations for $X$. For two clock valuations $\nu$ and $\mu$, $\nu \cong \mu$ iff all the following conditions hold:

1. For all clocks $x \in X$, either $\lfloor\nu(x)\rfloor$ and $\lfloor\mu(x)\rfloor$ are the same, or both $\nu(x)$ and $\mu(x)$ exceed $c_x$.
2. For all clocks $x, y$ with $\nu(x) \leq c_x$ and $\nu(y) \leq c_y$, $\langle\nu(x)\rangle \leq \langle\nu(y)\rangle$ iff $\langle\mu(x)\rangle \leq \langle\mu(y)\rangle$.
3. For all clocks $x \in X$ with $\nu(x) \leq c_x$, $\langle\nu(x)\rangle = 0$ iff $\langle\mu(x)\rangle = 0$.

A *clock region* for $A$ is an equivalence class of clock valuations induced by $\cong$. Note that there are only a finite number of regions, at most $k! \cdot 4^k \cdot \Pi_{x \in X}(c_x + 1)$, where $k$ is the number of clocks. Thus, the number of clock regions is exponential in the encoding of the clock constraints.

The key property of region equivalence is its stability: for any location $s$, and clock valuations $\nu$ and $\nu'$ such that $\nu \cong \nu'$, (a) for any $\delta \in \mathbb{R}$, if $(s,\nu) \xrightarrow{\delta} (s,\nu+\delta)$ then there exists $\delta' \in \mathbb{R}$ such that $(s,\nu') \xrightarrow{\delta'} (s,\nu'+\delta')$ and $(\nu+\delta) \cong (\nu'+\delta')$, and (b) for every label $a \in \Sigma^\epsilon$ and state $(t,\mu)$, if $(s,\nu) \xrightarrow{a} (t,\mu)$ then there exists $\mu'$ such that $(s,\nu') \xrightarrow{a} (t,\mu')$ and $\mu \cong \mu'$. Thus, if two states are equivalent, then an $a$-labeled discrete switch from one can be matched by a corresponding discrete switch from the other leading to an equivalent target state, and if the automaton can wait for $\delta$ units in one state, then it can wait for $\delta'$ units, possibly different from $\delta$, resulting in equivalent states. For this reason, the region equivalence is a *time-abstract* bisimulation.

For a timed automaton $A$, the quotient of $S_A$ with respect to the region equivalence is called the *region automaton* of $A$, and is denoted $R(A)$: vertices of $R(A)$ are of the form $(s,r)$, where $s$ is a location and $r$ is a clock region; there is an edge $(s,r) \xrightarrow{a} (s',r')$ in $R(A)$ for $a \in \Sigma^\epsilon$ iff for some clock valuations $\nu \in r$ and $\nu' \in r'$, $(s,\nu) \xrightarrow{a} (s',\nu')$ in $S_A$, or, $a = \epsilon$ and $(s,\nu) \xrightarrow{\delta} (s',\nu')$ for some $\delta \in \mathbb{R}$. The initial and final states of $S_A$ are used to define the initial and final vertices of $R(A)$. Now, the language of $R(A)$ is the untimed language of $L(A)$.

**Theorem 2.** *For a timed regular language $L$, untime($L$) is a regular language [2].*

Consequently, $R(A)$ can be used to solve language emptiness for $A$, and also to answer reachability queries for $A$. Thus, emptiness and reachability can be solved in time linear in the number of vertices and edges of the region automaton, which is linear in the number of locations and edges of $A$, exponential in the number of clocks, and exponential in the encoding of the constants. Technically, these problems are PSPACE-complete.

**Theorem 3.** *The language emptiness question for timed automata is* PSPACE*-complete, and can be solved in time $O(m \cdot k! \cdot 4^k \cdot (c \cdot c' + 1)^k)$, where $m$ is the number of switches in $A$, $k$ is the number of clocks in $A$, $c$ is largest numerator in the constants in the clock constraints in $A$, and $c'$ is the least-common-multiple of the denominators of all the constants in the clock constraints of $A$ [2].*

In [15] it was also shown that for timed automata with three clocks, reachability is already PSPACE-complete. A recent result [28] shows that for timed automata with one clock, reachability is NLOGSPACE-complete and for timed automata with two clocks, it is NP-hard. The reachability problem remains PSPACE-hard even if we bound the magnitudes of constants [15].

## 2.2 Cycle Detection

A timed $\omega$-word is an infinite sequence of the form $\alpha = (a_0,t_0)(a_1,t_1)\ldots (a_i,t_i),\ldots$, with $a_i \in \Sigma$, $t_i \in \mathbb{R}$, and $t_0 \leq t_1 \leq \cdots t_i \leq \cdots$, and timed $\omega$-language is a set of timed $\omega$-words. Reasoning in terms of infinite timed words, as in the untimed setting, is useful for checking liveness properties. The notion

of a run of a timed automaton $A$ naturally extends to timed $\omega$-words. A timed $\omega$-word $\alpha$ is accepted by $A$ using the Büchi condition, if there is a run of $A$ on $\alpha$ that repeatedly hits (infinitely often) some final location in $V^F$. The set of $\omega$-words accepted by $A$ is denoted by $L_\omega(A)$. Checking whether $L_\omega(A)$ is nonempty, for a given $A$, can be done by checking whether there is a cycle in the region graph of $A$ which is reachable from an initial state and contains some state in $V^F$.

For infinite words, it is natural to require that time diverges, that is, the sequence $t_0, t_1, \ldots t_i, \ldots$ grows without bound. Timed words that do not diverge depict an infinite number of events that occur in a finite amount of time. To restrict $L_\omega(A)$ only to divergent words, we can transform the timed automaton by adding a new clock $x$ which is reset to 0 whenever it becomes 1 (using an $\epsilon$-edge) and the timed automaton hits the new final set $V'_F$ only if the run had passed through $V_F$ in the last one unit of time.

**Theorem 4.** *Given a timed automaton $A$, the problem of checking emptiness of $L_\omega(A)$ is* PSPACE*-complete.*

Most of the results in this survey hold for timed $\omega$-languages also.

### 2.3 Sampled Semantics

In the discrete-time or sampled semantics for timed automata, the discrete switches, or the events, are required to occur only at integral multiples of a given sampling rate $f$. This can be formalized as follows. Given a timed automaton $A$ and a sampling rate $f \in \mathbb{Q}$, we define an automaton $S^f_A$: the states, initial states and final states of $S^f_A$ are the same as the states, initial states, and final states of $S_A$, and the transitions of $S^f_A$ are the transitions of $S_A$ that are labeled with either $a \in \Sigma^\epsilon$ or with $m.f$ (where $m \in \mathbb{N}$). The sampled timed language $L^f(A)$ is defined using the automaton $S^f_A$. Note that time of occurrence of any symbol in the timed words in $L^f(A)$ is an integral multiple of the sampling frequency $f$. To check emptiness of $L^f(A)$, observe that in any reachable state of $S^f_A$, the values of all clocks are integral multiples of $f$, and this can lead to a reduced search space compared to the region automata. However, the complexity class of the reachability and cycle-detection problems stays unchanged (here $L^f_\omega$ denotes the set of $\omega$-words where events occur at sampling rate $f$):

**Theorem 5.** *Given a timed automaton $A$ and a sampling rate $f \in \mathbb{Q}$, the problem of checking the emptiness of $L^f(A)$ (or $L^f_\omega(A)$) is* PSPACE*-complete.*

If the sampling rate $f$ is unknown, the resulting problems are the discrete-time reachability and discrete-time cycle-detection problems with unknown sampling rate: given a timed automaton $A$, does there exist a rational number $f \in \mathbb{Q}$ such that $L^f(A)$ (or $L^f_\omega(A)$) is nonempty. Discrete-time reachability for unknown sampling rate is decidable since it is equivalent to the question of whether $L(A)$ is empty: if $L(A)$ is nonempty, we can find a word in $L(A)$ where events occur at

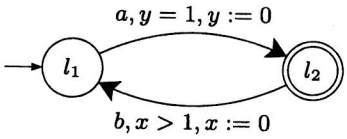



**Fig. 3.** Sampled semantics is different from the standard semantics.

rational times, and by choosing an appropriate $f$, show that it is an $f$-sampled word. However, the discrete-time cycle-detection problem with unknown sampling rate is undecidable:

**Theorem 6.** *Given $A$, the problem of checking whether $\bigcup_{f \in \mathbb{Q}} L^f_\omega(A)$ is nonempty, is undecidable [14].*

The undecidability proof is by reduction from the halting problem for two-counter machines. Given a two-counter machine $M$, one can construct a timed automaton $A_M$ and a location $s_F$ such that for any integer $n$, the location $s_F$ is reachable in the discrete-time semantics with the sampling rate $1/n$ iff the two-counter machine $M$ has a halting run in which both the counters do not exceed the value $n$.

To see that $L_\omega(A)$ can be nonempty while for each $f$, $L^f_\omega(A) = \emptyset$, consider the automaton in Figure 3. While the $a$-events occur at integer times, the $b$-events have to occur closer and closer to the $a$-events, and fixing any sampling rate $f$ makes the $\omega$-language empty.

## 2.4 Choice of Clock Constraints and Updates

The clock constraints in the guards of a timed automaton compare clocks with constants. Such constraints allow us to express (constant) lower and upper bounds on delays. Consider the following generalization of clock constraints: for a set $X$ of clocks, the set $\Phi^d(X)$ of *clock constraints* $g$ is defined by the grammar

$$g := x \leq c \mid c \leq x \mid x - y \leq c \mid x < c \mid c < x \mid x - y < c \mid g \wedge g$$

where $x, y$ are clocks in $X$ and $c \in \mathbb{Q}$. Including such "diagonal" clock constraints that compare clock differences with constants does not change the complexity of reachability. Similarly, we can relax the allowed updates on switches. In the original definition, each switch is tagged with a set $\lambda$ which specifies which clocks should be reset to zero. A more general *update map* $\lambda$ maps clocks in $X$ to $\mathbb{Q} \cup X$ specifying the assignments $x := \lambda(x)$. Thus, $x$ can be assigned to an arbitrary rational constant, or to the value of another clock. Both these modifications can be handled by modifying the region construction. In fact, both these extensions do not add to the expressive power.

**Theorem 7.** *If the clock constraints for guards are chosen from the set $\Phi^d(X)$, and the switches are annotated with the update maps, the expressive power of*