Control of Timed Systems

Franck Cassez

HABILITATION À DIRIGER LES RECHERCHES

English Version

Franck Cassez

CNRS/IRCCyN

Control of Timed Systems

École Doctorale : STIM

Specialisation : Computer Science

September 21st, 2007 at IRCCyN

Examining Committee

Reviewers:

Ahmed Bouajjani	Professor, University of Paris 7, France
Oded Maler	Research Director, CNRS, VERIMAG, Grenoble, France
Jean-François Raskin	Professor, Université Libre de Bruxelles, Belgium

Examiners:

Jean Bézivin	Professor, University of Nantes, France
Claude Jard	Professor, ENS Cachan, Antenne de Bretagne, Rennes, France
Kim G. Larsen	Professor, Aalborg University, Denmark
Jean-Jacques Loiseau	Research Director, CNRS, IRCCyN, Nantes, France
Olivier H. Roux	Assistant Professor (HDR), University of Nantes, France

Table of Contents

Foreword

To the reader who is not familiar with the French Higher Education system, the "Habilitation à Diriger les Recherches" (HDR) is a French degree which is supposed to acknowledge your ability to become a group leader, and more importantly which entitles you to officially supervise PhD students.

It is also mandatory to hold an HDR if you want to apply for a University Professor position or Research Director (INRIA, CNRS) position. Holding an HDR is a necessary but not sufficient condition.

Indeed, if you hold an HDR you can apply for a Research Director position, but there are not many positions per year and you might have to wait for a while.

An alternative is to apply for a University Professor position. Prior to this, you have to apply for a "Qualification": this "Qualification" is delivered (or not) by a National French Panel[1], and to obtain it you have to hold an HDR and also to write a careful application (of course the forms you have to fill in are different from the ones you had to fill in when you registered as an HDR student[2]). If you are "qualified" then you can apply for University Professor positions (and submit another application in yet another format).

The rules to obtain an HDR can sometimes appear to be inconsistent: you have to demonstrate you were able to write some research articles which is fine but also to show you could supervise PhD students. Here you can notice the French Cartesian tradition: without an HDR you cannot officially supervise PhD students, but to obtain it you must have supervised PhD students.

To sum up, obtaining an HDR requires you: 1) to write a report emphasising your research results and students' supervisions, and 2) to defend your report in front an Examining Committee. There are no rules for item 1), and you can write a hundred page book or just an extended abstract.

I chose to write an extended abstract and this report is organised as follows:

- Chapter 1 consists of a short Curriculum Vitæ;
- Chapter 2 gives an overview of my recent research contributions on the control of timed systems. This has been my main research interest in the last few years;
- Chapter 3 contains an overview of my other research activities and some hints for future work;
- finally Chapter 4 contains the references (sorted by categories) used in this report.

[1] the "Commission Nationale des Universités", CNU; there is one panel per domain *e.g.,* Computer Science.

[2] Indeed, when you want to defend an HDR, you register as a University student and you get a nice student's ID which entitles you the right you for many reduced fees.

Chapter 1

Curriculum Vitæ

Current Professional Address:

Institut de **R**echerche en **C**ommunication et **Cy**bernétique de **N**antes (IRCCyN)
UMR CNRS 6597
1 rue de la noë
BP 92101
44321 NANTES cedex 03
France

tel : (+33 | 0) 2 40 37 69 81
fax : (+33 | 0) 2 40 37 69 30
email: `franck.cassez@cnrs.irccyn.fr`
web: `http://www.irccyn.fr/franck`

Acronyms used in this Chapter like HDR[n] are listed page 15.

1.1 Positions Held

1993–1997 Associate Professor, Department of Computer Science, University of Western Brittany (Université de Bretagne Occidentale, UBO[1]), Brest, France.

1997–2007 Full-Time Researcher at the Centre National de la Recherche Scientifique (CNRS[2], French National Centre for Scientific Research), affiliated with the Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN[3], Research Institute in Communications and Cybernetics of Nantes), France

1.2 Professional Experience

1993–1997 At the University of Western Brittany: Lectures in Computer Science, "DEUG Sciences" (undergraduate), and Masters. *Algorithms and Data Structures, Operating Systems, Language Theory, Introduction to Complexity Theory.*

1999–2006 At the Ecole Nationale de la Statistique et de l'Analyse de l'Information (ENSAI[4]), Rennes, France: *Introduction to parsing* (17h). Graduate level, Engi-

neering Degree, 3rd year. Lab work with JFlex and JavaCup (Java versions of Lex & Yacc).

2001–2002 At Ecole Centrale de Nantes (ECN[5]), Nantes, France. *Modeling and Verification* (Lectures: 16h and Lab work: 10h). Specialisation in « Embedded Systems », Graduate level, Engineering Degree, 3rd year. Theory: transition systems, model-checking, temporal logic, timed automata. Lab work with SPIN[6] and UPPAAL[7].

2005 At Ecole Centrale de Nantes. *Operating Systems* (Lectures: 12h and Lab work: 6h). Graduate level, Engineering Degree, 3rd year, specialisation in Computer Science.

1.3 Research Activities

1990–1993 PhD in Computer Science, École Centrale de Nantes & University of Nantes, France.

1993–1997 Member of the research group *Langages et Interfaces pour Machines Intelligentes (LIMI)*, Department of Computer Science, University of Western Brittany, Brest, France.
I participated in the following projects:

- European project ESPRIT-WG 23531 Fireworks[8] from 1997 to 2000;
- Working group GRAFCET (AFCET);
- Working group GDR CNRS Protocoles-Réseaux-Systèmes (Protocol-Networks-Systems).

1997–2007 Member of the research group *Modélisation et Vérification des Systèmes Embarqués (MOVES, Modeling and Verification of Embedded Systems)* at IRCCyN, Nantes.
I am involve or have been involved in the following projects:

- *Distributed Open Timed Systems* (DOTS[9]) funded by ANR (French National Research Agency), started January 2007, duration: 4 years;
- *Control and Observation of Real-Time Open Systems* (CORTOS[10]) ACI program[11] from September 2003 to January 2007;
- *Tools and Algorithms for the Verification of Hybrid Systems* (CHRONO[12]) ACI program[11] from September 2001 to September 2004;
- Working group "GDR ARP" (Architecture, Networks, Parallelism).

1.4 Committees and Administrative Duties

2002– Member of the Steering Committee of the Summer School MOVEP[13].

2002–2006 Member of the Editorial Board of the French Journal *Technique et Science Informatiques* (TSI).

2002–2005 Member of the recruiting (or scientific) committee "commission de spécialistes" of the University of Evry, France.

1.5 Students Supervision and Collaborations

I am working regularly with my colleagues in Nantes on various research topics (timed automata, time Petri nets). The work described in this report was also carried out with PhD students I was co-supervising and with colleagues from France and abroad with whom I have ongoing collaborations.

1.5.1 Students Supervision

Post Doc Student:

- **Julien D'Orso**.
  Subject: Control of Timed Systems.
  Funding : ACI CORTOS, January to December 2004.

PhD Students:

- **Frédéric Herbreteau**, PhD defended in December 2002.
  Automates à file réactifs embarqués (Embedded Reactive FIFFO Automata).
  Supervision: Franck Cassez (50 %), Olivier Roux (50%).
  Funding: French Ministry of Research.
  PhD Committee: B. Boigelot, F. Cassez, A. Finkel, P. Gastin, O. Roux, F. Vernadat.
  Publications: one journal paper (*Journal of Real-Time Systems*, [J3]), and one international conference paper (LATIN'02, [C18]).
  Current activity: Associate professor at ENSEIRB, Bordeaux and researcher at LaBRI.

- **Claire Pagetti**, PhD defended in April 2004.
  Extension temps-réel d'AltaRica (Timed Extenstion of AltaRica).
  Supervision: Franck Cassez (50 %), Olivier Roux (50%).
  Funding: French Ministry of Research.
  PhD Committee: A. Arnold, F. Cassez, J.-P. Elloy, F. Laroussinie, A. Rauzy, O. Roux, R. Valette.
  Publications: one journal paper (*Fundamentæ Informatica*, [J2]) and one international workshop paper (*Formal Aspects of Component Software (FACS'03)*, [C16]).
  Current activity: Research Engineer at CERT/ONERA, Toulouse.

I have not co-supervised any other PhD thesis since 2004. In the meantime I have tried to strengthen and develop collaborations with my colleagues abroad (Aalborg, DK; Sydney, AU; Bruxelles, B) and in France (LSV, ENS-Cachan, VERIMAG, Grenoble) and also to expand my research activities. Since 2004, even if I was not supervising them, I have been working with PhD students in my lab: Didier Lime on the expressiveness of time Petri nets [C7, C8, C9] and Guillaume Gardey on the control of time Petri nets [I3, J7]. At the moment I am not considering any new supervision as I have applied for a Mobility Project (*Marie Curie* European Program) to spend two years abroad.

Master Students:

- Cédric Meuter, *Compilation et répartition de programmes ELECTRE sur LEGO Mindstorms* (Compiling and Distributing ELECTRE Programs for LEGO Mindstorms). Master Thesis in *Computer Science*, Université Libre de Bruxelles, Belgium, 2001/2002.

- Armelle Prigent, *Vérification des systèmes SDL par abstraction*, (Verification of SDL specifications), co-supervised with Philippe Dhaussy, ENSIETA, Brest and Olivier Roux, IRCCyN, Nantes). Master Thesis in *Automatique et Informatique Appliquée* (Automated Systems and Computer Science), Nantes, 1999/2000.
- Manuel David, *Test de Systèmes Temporisés*, Master Thesis in *Automatique et Informatique Appliquée*, Nantes, 1998/1999.
- François Lorillard, *Vérification de propriétés d'un logiciel de surveillance médicale avec SPIN*, (Verification of Medical Monitoring Software). Master Thesis in *Automatique et Informatique Appliquée*, Nantes,1997/1998. One publication in a national workshop [M1].

1.5.2 Collaborations.

National Collaborations. I work on a regular basis with my colleagues in the research group MOVES[14] (Modeling and Verification of Embedded Systems) and also with Olivier H. Roux from the Real-Time Systems group. Hereafter I just mention the collaborations I had or have with other colleagues in France:

- with Philippe Dhaussy (ENSIETA, Brest), during the Master Thesis of François Lorillard in 1998, we have designed a model of the software of a medical monitoring system (used in the main hospital of Brest) and formally verified it. The result of this work was published in a national workshop [M1]. Later on, during the Master Thesis of Armelle Prigent in 2000, we have proposed a translation of SDL in SPIN [C19].
- with Alain Finkel and Grégoire Sutre (LSV, ENS-Cachan) between 1998 and 2002, we have proposed a model for asynchronous infinite state systems, namely FIFFO automata (First In First Fireable Out) and studied the properties of this model (decidability of verification problems). The results we have obtained were published in [C22, C18].
- with François Laroussinie (LSV, ENS-Cachan), since 1999, we have been working on various problems. We were co-leaders of the project ACI CHRONO[12] (2001–2004) which focussed on tools and algorithms for hybrid systems and more particularly for *stopwatch automata*. *Stopwatch automata* are interesting because they can be used to model scheduling problems with preemption and complex synchronisation. This collaboration was pursued with Patricia Bouyer. We have worked on control problems for timed systems during the course of the project ACI-CORTOS[10] for which I was the site leader in Nantes. The results we have obtained were published in various conferences [I3, C10, C12, C14, C20, E1].
- with Gilles Bernot, Jean-Paul Comet and Franck Delaplace (LaMI, Evry), we have worked on modeling biological regulatory networks. This work was published in the BioCONCUR international workshop [C15].
- with Béatrice Berard and Serge Haddad (LAMSADE, Paris) we have been working since 2005 on the expressiveness of timed automata and time Petri nets. The results were published in [C7, C8, C9].
- with Claude Jard and Thomas Chatain (IRISA, Rennes) since 2005, we have been working on unfoldings of networks of timed automata. Our results were published in [C6]. This collaboration is now pursued in the French project DOTS[9] which started in January 2007.

- with Karine Altisen and Stavros Tripakis (VERIMAG, Grenoble) we have been working on diagnosis problems for discrete event systems and timed systems. Our results were published in [C4, C3, C5] and in the research report [RR1].
- with André Arnold, Alain Griffault (LaBRI, Bordeaux) and Antoine Rauzy (LMI, Marseille) during the PhD Thesis of Claire Pagetti (2004), we have proposed an extension of the language AltaRica for timed systems [J2, C16].

International Collaborations. The collaborations are listed in chronological order:

- with Mark Ryan (University of Birmingham, UK) and Pierre-Yves Schobbens (FUNDP, Namur, Belgium), during the course of the project FIREworks[8]. This project addressed the problem of *feature integration in requirements engineering* which is a real concern in the telecoms industry. The project started in May 1997 and ended in November 2000. Many European groups were involved in FIREworks, in particular the University of Birmingham, LaMI (Evry), FUNDP (Namur, Belgium), and Aalborg University, Denmark. I was the site leader in Nantes for this project.

  Together with Mark Ryan (University of Birmingham, UK) and Pierre-Yves Schobbens (FUNDP, Namur, Belgium) we have contributed some results on the detection of features interaction using the ATL logic. This work has been reported at the FIREworks annual meeting in May 2000 and published in [M2].

  Following this collaboration, Mark Ryan joined the Organising Committee of the summer school MOVEP[13] until 2004.
- with Kim G. Larsen, Aalborg University, Denmark, we started a collaboration in September 1999. Since then, I have visited Aalborg regularly (2 weeks a year). We have first contributed results on the expressiveness of *stopwatch automata* (published in CONCUR'2000, [C21]). More recently, we have been working on (Priced) Timed Games [C12, C14] and efficient algorithms for solving timed games [C1, C11].
- with Jean-François Raskin, Université Libre de Bruxelles (ULB) we started to collaborate in 2002. We contributed some results on the decidability of some sampling control problems for timed automata. This work was carried out together with Thomas Henzinger, UC Berkeley (USA) and published in HSCC'02 ([C17]). More recently we have worked (together with Kim G. Larsen) on the design of efficient algorithms for partially observable timed games [C1].

  In 2002, I supervised a Master's student, Cédric Meuter, from the Université Libre de Bruxelles.

  Lastly Jean-François Raskin has been in the organising committee of the school MOVEP since 2002 and hosted the school in Brussels in 2004.
- with Ralf Huuck and Ansgar Fehnker, from NICTA[15], Sydney, Australia, we have worked on modeling sensor networks with timed automata. This work was carried out during a 2-month visit I made at NICTA in 2005. I have applied for a *Marie Curie* International Outgoing Fellowship (OIF) of the 7^{th} Framework Programme to spend two years at NICTA from 2008.
- with John Mullins, CRAC[16], école Polytechnique de Montréal, Canada, we have worked on non-interference problems, more precisely on the synthesis of non-interferent systems [C2]. This work was done in collaboration with Olivier H. Roux.

1.5.3 Reviewing

I was a member of the Editorial Board of the French Journal *Technique et Science Informatiques* (TSI) from 2002 to 2006. As a member of the Editorial Board I helped in dispatching papers to reviewers and also reviewing papers for the journal TSI. In 2006, I was Associate Editor with François Laroussinie of a Special Issue of the journal about the *Control of Timed and Hybrid Systems* [E1].

I am regularly reviewing papers for international conferences: CAV[17], TACAS[18], FOSSACS[19], CONCUR[20], BioCONCUR[21], LICS[22], HSCC[23], FSTTCS[24], CSL[25] and international journals: *Journal of Real-Time Systems*, *Fundamenta Informaticæ*, *Discrete Mathematics and Theoretical Computer Science*, *Information & Computation*, *Theory and Practice of Logic Programming*, *Formal Methods in System Design* and *IEEE Transactions on Automatic Control* (since September 2006, I have reviewed one paper for each of the above mentioned journals.)

Since 2006, I am an expert evaluator for the French Research Agency (ANR) now in charge of dispatching the funding for public research in France, and I also served as an expert evaluator for the Canadian *Fonds de Recherche sur la nature et les technologies*.

1.5.4 Organisation of Summer Schools, Journal Special Issue

Since 1998, I have been in the organising committee of the summer school MOVEP[13]. The scope of the school is *MOdeling and VErification of parallel Processes* and the subjects covered are continuously updated following the new research advances. MOVEP used to be a French summer school (1994, 1996, 1998) and in 2000, I decided to give the school an international dimension. Renowned speakers from Europe and North America gave tutorials and invited talks. For this first international edition of MOVEP, the proceedings were published as a volume in the series *Lecture Notes in Computer Science Tutorials* [E5]. Ever since, around 120 participants from Europe, North America, India and North Africa have attended each edition of the school. MOVEP is now a well-established and internationally recognised event. It was organised in Brussels in 2004, Bordeaux in 2006 and will be held in Rennes in 2008. The school has been sponsored by national organisations (CNRS) and also European Network of Excellence (ARTIST2 NoE in 2006).

At the last international conference on *Concurrency Theory* (CONCUR'06, Bonn, Germany), I was in charge of organising the workshop CORTOS[26] on *Control and Observation of Real-Time Open Systems*. This workshop was sponsored by the French project CORTOS I was involved in, and gathered around 15 people working on the control of timed systems (6 invited talks were given at the workshop among them 2 were given by participants of the French project CORTOS).

In 2006, I was Associate Editor with François Laroussinie of a Special Issue of the journal *Technique et Science Informatiques (TSI)* about the *Control of Timed and Hybrid Systems* [E1].

Since 2004, I am also participating in the programme *rencontres élèves-chercheurs*, aimed at improving interaction between the research community and students (from Primary school to High school). A few times a year, I am giving short talks in classrooms to present researchers's activities. I am also participating in the *fête de la science*, held in October every year in France. For instance in 2007, I will give a talk in Nantes on *The limited power of computers*.

1.6 Funded Projects

Since I was hired as an Associate Professor in 1993, I have tried to participate in funded projects. Some of them are mentioned in this section.

1.6.1 Recent National Projects

- Since January 2007, I am involved in the project *Distributed Open Timed Systems* (DOTS[9]) funded by the ANR programme on *Sécurité Informatique* (Security). The project addresses verification and control problems for distributed and timed systems. The leader of the project is François laroussinie (LSV, ENS Cachan) and the other participants[1] are: IRISA Districom, Rennes (Claude Jard); IRCCyN, Nantes (Didier Lime); LaBRI, Bordeaux (Igor Waluckiewicz); LAMSADE, Paris Dauphine (Serge Haddad) and LSV, ENS Cachan (François Laroussinie). This project spans over 4 years and the total funding is 600 KEuros with 100 KEuros for our group at IRCCyN.
- from September 2003 to January 2007, I was the site leader of the project CORTOS[10] (Control and Observation of Real-Time Open Systems). CORTOS was also funded by the ANR programme on *Sécurité Informatique* (Security). This was a joint project with LSV (ENS Cachan) and VERIMAG (Grenoble) and the project leader was Patricia Bouyer (LSV, ENS Cachan). Total funding for this project was 300 KEuros with 100 KEuros for our group. The results of the project are available from the URL `http://www.lsv.ens-cachan.fr/aci-cortos/`.
- from September 2001 to September 2004, I was co-leader with François Laroussinie of the project CHRONO[12]. The theme of this project was using *stopwatch automata* to model scheduling problems. We were interested in studying decision problems for subclasses of stopwatch automata. The total funding for this project was 100 KEuros with 30 KEuros for our group.

1.6.2 European Project

From 1997 to 2000, I was involved in the Fireworks[8] project. This project addressed the problem of *feature integration in requirements engineering* which is a real concern in the telecoms industry. The project started in May 1997 and ended in November 2000. Many European groups were involved in FIREworks, in particular the University of Birmingham (UK, project leader), LaMI (Evry), FUNDP (Namur, Belgium), and Aalborg University, Denmark. I was the site leader in Nantes for this project.

Together with Mark Ryan (University of Birmingham, UK) and Pierre-Yves Schobbens (FUNDP, Namur, Belgium) we have contributed some results on the detection of features interactions using the ATL logic. This work has been reported at the FIREworks annual meeting in May 2000 and published in [M2].

The funding for our group in Nantes was around 20 KEuros.

1.6.3 Visits Abroad

I have visited a few research laboratories abroad. These visits were sponsored either by the host laboratory, or by the *Direction des Relations Internationales* (DRI) of CNRS, or by the

[1] Only site leaders are mentioned.

funds of the projects listed in Section 1.6.1. Here are some of the groups I visited recently:

- Professor Kim Guldstrand Larsen, Aalborg University, Denmark.
  - October–November 1999, 2 months, sponsored by BRICS[27], Denmark;
  - January 2001, 1 month, sponsored by BRICS, Denmark and DRI[28] CNRS;
  - August 2001, 1 month, sponsored by BRICS, Denmark;
  - December 2002, 3 weeks, sponsored by BRICS, Denmark and DRI CNRS;
  - December 2003, 2 weeks, sponsored by BRICS, Denmark;
  - February 2005, 2 weeks, sponsored by CISS[29] & ACI-CORTOS;
  - June 2006, 2 weeks, sponsored by CISS & ACI-CORTOS;
- Professor Jean-François Raskin, Université Libre de Bruxelles (ULB), Belgium.
  - May–June 2002, 2 months, sponsored by FNRS[30];
  - November–December 2007, 1 month, sponsored by FNRS;
- Professor John Mullins, Ecole Polytechnique de Montréal, Canada.
  - June 2005, 2 weeks, sponsored by ACI CORTOS;
  - April 2007, 2 weeks, sponsored by ACI CORTOS;
- Professors Ron Van der Meyden and Ralf Huuck, NICTA, Sydney, Australia.
  - November–December 2005, 2 months, sponsored by DREI[31] CNRS.

1.7 Seminars & Invited talks

Recently, I have presented the conference papers at ACSD'07, TASE'07, ACSD'06, ATVA'06, FORMATS'05, CONCUR'05, AVoCS'04, GDV'04.

In addition to this, I have given invited talks at projects meetings (FireWorks, ACI CHRONO, ACI CORTOS, ANR DOTS), or conferences or seminars:

1. *Efficient Algorithms for Timed Games* [I1], FORMATS'07, October 2007, Salzburg, Austria.
2. *Contrôle des systèmes temporisés* [I4], ETR'07, September 2007, Nantes, France.
3. *Sensor Minimization Problems for Finite Automata*, Seminar of the Centre Fédéré en vérification, November 2006, Brussels, Belgium.
4. *Fault Diagnosis with Digital Clocks*, CORTOS workshop, Bonn, August 2006.
5. *Sensor Minimization Problems for Finite Automata*, CISS Seminar, Aalborg, June 2006, Denmark.
6. *Sensor Minimization Problems for Finite Automata*, Joint Workshop of the French projects CORTOS[10], Versydis, Persee, Cachan, March 2006.

7. *Control of Timed Systems*, NICTA, Lectures on the theory of control of timed systems, implementation and optimal control. November–December 2005, Sydney, Australia.

8. *Introduction to Control of Timed Systems*, NICTA Workshop on Formal Methods, 7–9 November 2005, Sydney, Australia.

9. *Optimal Strategies for Priced Timed Games*, VERIMAG seminar, October 20th, 2005, Grenoble, France.

10. *Introduction to Control of Timed Systems*, Invited Talk, MSR'05, October 7th, 2005, Autrans, France.

11. *Introduction to Control of Timed Systems*, CRAC Seminar, June 2005, Ecole Polytechnique de Montréal, Canada

12. *Efficient Algorithms for reachability control of timed systems*, Seminar of Université Libre de Bruxelles, Brussels, May 28th, 2005.

13. *From Timed Petri Nets to Timed Automata*, CISS Seminar, Aalborg, March 2005, Denmark.

14. *Introduction to Control of Timed Systems*, ENS Bretagne, Rennes, January 2005.

15. *Optimal Strategies for Priced Timed Games*, Journées du Centre Fédéré en vérification, May 2004, Brussels, Belgium.

16. *Optimal Strategies for Priced Timed Games*, MVTSI Seminar, May 2004, LaBRI, Bordeaux, France.

17. *Vérification Qualitative – Model Checking et logiques temporelles* [I5], Ecole Jeunes Chercheurs *Ecole Temps Réel*, ETR'03, September 2003, Toulouse, France.

18. *Comparison of Control Problems for Timed and Hybrid Systems*, LIAFA seminar, Paris 7, June 2002, Paris, France.

19. *Comparison of Control Problems for Timed and Hybrid Systems*, 68NQRT Seminar, February 2003, IRISA, Rennes, France

20. *New results for Reactive FIFFO Automata*, Seminar of Université Libre de Bruxelles, Brussels, December 2002.

21. *Comparison of Control Problems for Timed and Hybrid Systems*, LaBRI seminar, January 2002, Bordeaux, France.

22. *Comparison of Control Problems for Timed and Hybrid Systems*, BRICS Seminar, November 2001, Aalborg, Denmark

23. *The Expressive Power of Stopwatches*, LSV Seminar, January 2000, Paris, France.

1.8 Publications

Most of the following papers are available from `http:www.irccyn.fr/franck`.

— Refereed Journal Papers —

International Journals

[J1] Franck Cassez and Olivier H. Roux. Structural translation from time petri nets to timed automata. *Journal of Software and Systems*, 29:1456–1468, 2006.

[J2] Franck Cassez, Claire Pagetti, and Olivier Roux. A timed extension for AltaRica. *Fundamenta Informatica*, 62(3–4):291–332, 2004.

[J3] Frédéric Herbreteau, Franck Cassez, and Olivier Roux. Application of Partial-Order Methods to Reactive Systems with Event Memorization. *Journal of Real-Time Systems*, 20(3):287–316, 2001.

[J4] Olivier Roux, Vlad Rusu, and Franck Cassez. Hybrid Verification of Reactive Systems. *Formal Aspects of Computing*, 11(4):448–471, 1999.

[J5] Franck Cassez. Formal Semantics for Reactive GRAFCET. *European Journal of Automation*, 31(3):581–603, 1997.

[J6] Franck Cassez and Olivier Roux. Compilation of the Electre Reactive Language into Finite Transition Systems. *Theoretical Computer Science*, 146(1–2):109–143, 1995.

National Journals

[J7] Karine Altisen, Patricia Bouyer, Thierry Cachat, Franck Cassez, and Guillaume Gardey. Introduction au contrôle des systèmes temps-réel. *Journal Européen des Systèmes Automatisés*, 39(1-2-3):367–380, 2005.

[J8] Olivier Roux, Denis Creusot, franck Cassez, and Jean-Pierre Elloy. Le langage réactif asynchrone ELECTRE. *Technique et Science Informatiques*, 11(5):35–66, 1992.

— Invited Contributions —

[I1] Franck Cassez. Efficient on-the-fly algorithms for partially observable timed games. In *Proc. of the 5th Int. Conf. on Formal Modeling and Analysis of Timed Systems (FORMATS'07)*, volume 4763 of *LNCS*, pages 5–24. Springer, 2007. Invited Paper.

[I2] Franck Cassez and Olivier H. Roux. *Petri Nets – Theory and Application*, chapter From Time Petri Nets to Timed Automata. Advanced Robotic Systems, Vienna, Austria, 2007. Forthcoming.

[I3] Karine Altisen, Patricia Bouyer, Thierry Cachat, Franck Cassez, and Guillaume Gardey. Introduction au contrôle des systèmes temps-réel. In *Actes du 5ème Colloque sur la Modélisation des Systèmes Réactifs (MSR'05)*, pages 367–380. Hermès Science, 2005. Invited Paper. Also appeared as [J7].

[I4] Franck Cassez and Nicolas Markey. Contrôle des systèmes temporisés. Actes de l'école d'été ETR'07, 2007. Nantes.

[I5] Franck Cassez. Vérification qualitative – Model-Checking et Logiques Temporelles. Actes de l'école d'été ETR'03, 2003. Toulouse.

— Refereed Conferences —

International Conferences

The acceptance ratio for the articles of this section are: [C2]: 32%; [C3]: 40%; [C4]: 22%; [C5]: 51%; [C6]: 25%; [C7]: 44%; [C8]: 23%; [C9]: 35%; [C10] et [C11]: 38%; [C13]: 67%; [C14]: 24% ; [C17]: 45%; [C18]: 42%; [C19]: 50%; [C20]: 38%; [C21]: 36%; [C22]: 36%.

[C1] Franck Cassez, Alexandre David, Kim G. Larsen, Didier Lime, and Jean-François Raskin. Timed control with observation based and stuttering invariant strategies. In *5th Int. Symp. on Automated Technology for Verification and Analysis (ATVA'07)*, LNCS, pages 307–321. Springer, 2007. Forthcoming.

[C2] Franck Cassez, John Mullins, and Olivier H. Roux. Synthesis of non-interferent systems. In *4th Int. Conf. on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS'07)*, volume 1 of *Communications in Computer and Inform. Science*, pages 307–321. Springer, 2007.

[C3] Franck Cassez, Stavros Tripakis, and Karine Altisen. Sensor minimization problems with static or dynamic observers for fault diagnosis. In *7th Int. Conf. on Application of Concurrency to System Design (ACSD'07)*, pages 90–99. IEEE Computer Society, 2007.

[C4] Franck Cassez, Stavros Tripakis, and Karine Altisen. Synthesis of optimal dynamic observers for fault diagnosis of discrete-event systems. In *1st IEEE & IFIP Int. Symp. on Theoretical Aspects of Soft. Engineering (TASE'07)*, pages 316–325. IEEE Computer Society, 2007.

[C5] Karine Altisen, Franck Cassez, and Stavros Tripakis. Monitoring and fault-diagnosis with digital clocks. In *6th Int. Conf. on Application of Concurrency to System Design (ACSD'06)*. IEEE Computer Society, 2006.

[C6] Franck Cassez, Thomas Chatain, and Claude Jard. Symbolic Unfoldings for Networks of Timed Automata. In *4th Int. Symp. on Automated Technology for Verification and Analysis (ATVA'06)*, volume 4218 of *LNCS*, pages 307–321. Springer, 2006.

[C7] Béatrice Bérard, Franck Cassez, Serge Haddad, Olivier H. Roux, and Didier Lime. Comparison of the Expressiveness of Timed Automata and Time Petri Nets. In *Proc. of the 3rd Int. Conf. on Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *LNCS*, pages 211–225. Springer, 2005.

[C8] Béatrice Bérard, Franck Cassez, Serge Haddad, Olivier H. Roux, and Didier Lime. When are Timed Automata weakly timed bisimilar to Time Petri Nets ? In *Proc. of the 25th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'05)*, volume 3821 of *LNCS*, pages 276–284. Springer, 2005.

[C9] Béatrice Bérard, Franck Cassez, Serge Haddad, Olivier H. Roux, and Didier Lime. Comparison of Different Semantics for Time Petri Nets. In *3rd Int. Symp. on Automated Technology for Verification and Analysis (ATVA'05)*, volume 3707 of *LNCS*, pages 293–307. Springer, 2005.

[C10] Patricia Bouyer, Franck Cassez, and François Laroussinie. Modal logics for timed control. In *Proc. of the 16th Int. Conf. on Concurrency Theory (CONCUR'05)*, volume 3653 of *LNCS*, pages 81–94. Springer, 2005.

[C11] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *Proc. of the 16th Int. Conf. on Concurrency Theory (CONCUR'05)*, volume 3653 of *LNCS*, pages 66–80. Springer, 2005.

[C12] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Synthesis of optimal strategies using HyTech. In *Proc. of the Workshop on Games in Design and Verification (GDV'04)*, volume 119 of *Elec. Notes in Theo. Comp. Science*, pages 11–31. Elsevier, 2005.

[C13] Franck Cassez and Olivier H. Roux. Structural Translation of Time Petri Nets into Timed Automata. In *Proceedings of the Workshop on Automated Verification of Critical Systems (AVoCS'04)*, volume 128 of *Elec. Notes in Theo. Comp. Science*, pages 145–160. Elsevier, 2005.

[C14] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Optimal strategies in priced timed game automata. In *Proc. of the 24th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *LNCS*, pages 148–160. Springer, 2004.

[C15] Gilles Bernot, Franck Cassez, Jean-Paul Comet, Franck Delaplace, Céline Müller, Olivier Roux, and Olivier Roux. Semantics of Biological Regulatory Networks. In *Proc. of the Workshop on Concurrent Models in Molecular Biology (BioConcur'03)*, Elec. Notes in Theo. Comp. Science. Elsevier, 2003.

[C16] Claire Pagetti, Franck Cassez, and Olivier Roux. Hierarchical Modeling and Verification of Timed Systems in Timed AltaRica. In *Workshop on Formal Aspects of Component Software (FACS'03)*, pages 63–80. UNU/IIST, Macau, 2003. Pisa, Italy, September 8–9, 2003.

[C17] Franck Cassez, Thomas A. Henzinger, and Jean-François Raskin. A Comparison of Control Problems for Timed and Hybrid Systems. In *Proc. of the Workshop on Hybrid Systems Computation and Control (HSCC'02)*, number 2289 in LNCS, pages 134–148. Springer, 2002.

[C18] Frédéric Herbreteau, Franck Cassez, Alain Finkel, Olivier Roux, and Grégoire Sutre. Verification of Embedded Reactive FIFFO Systems. In *Proc. of the Latin American Symp. on Theoretical Informatics (LATIN'02)*, volume 2286 of *LNCS*, pages 400–414. Springer, 2002.

[C19] Armelle Prigent, Franck Cassez, Philippe Dhaussy, and Olivier Roux. Extending the Translation from SDL to Promela. In *Proc. of the 9th International SPIN Workshop on Model Checking of Software (SPIN'02)*, volume 2318 of *LNCS*, pages 400–414. Springer, 2002.

[C20] Franck Cassez and François Laroussinie. Model-Checking of Hybrid Automata by Quotienting and Constraints Solving. In *Proc. of the Int. Conf. on Computer Aided Verification, (CAV'00)*, volume 1855 of *LNCS*, pages 373–388. Springer, 2000.

[C21] Franck Cassez and Kim Guldstrand Larsen. The Impressive Power of Stopwatches. In *Proc. of the 11th Int. Conf. on Concurrency Theory, (CONCUR'00)*, volume 1877 of *LNCS*, pages 138–152. Springer, 2000.

[C22] Grégoire Sutre, Alain Finkel, Olivier Roux, and Franck Cassez. Effective recognizability and model-checking of reactive FIFFO automata. In *Proc. of the 7th Int. Conf. on Algebraic Methodology and Software Technology (AMAST'98)*, volume 1548 of *LNCS*, pages 106–123. Springer, 1999.

National Conferences

[C23] Franck Cassez and Olivier Henri Roux. Traduction structurelle des Réseaux de Petri Temporels en Automates Temporisés. In *4ème Colloque Francophone sur la Modélisation des Systèmes Réactifs, (MSR'03)*, 2003.

[C24] Franck Cassez, Frédéric Herbreteau, and Olivier Roux. Reactive Systems with Unbounded Event Memorization. In *Conférence Africaine de Recherche en Informatique, CARI'2000*, pages 291–298. INRIA, 2000. Conférence francophone.

[C25] Franck Cassez. Sémantique pour le GRAFCET réactif. In *Actes du* 1^{er} *Colloque sur la Modélisation des Systèmes Réactifs (MSR'96)*, 1996.

— Proceedings, Journal Special Issue —

I was co-editor of the following books:

[E1] Franck Cassez and François Laroussinie, editors. *Contrôle des applications temps-réel: modèles temporisés et hybrides*, volume 25 of *Technique et Science Informatiques*. Hermès Science, 2006.

[E2] Franck Cassez, Therry Jéron, François Laroussinie, Jean-François Raskin, and Mark Ryan, editors. *Proceedings of the Summer School MOVEP'2006*. LaBRI, Bordeaux, France, Bordeaux, France, December 2006. 400 pages, in English.

[E3] Franck Cassez, Therry Jéron, François Laroussinie, Jean-François Raskin, and Mark Ryan, editors. *Proceedings of the Winter School MOVEP'2004*. Université Libre de Bruxelles, Belgique, Bruxelles, Belgium, December 2004. 400 pages, in English.

[E4] Franck Cassez, Claude Jard, François Laroussinie, and Mark Ryan, editors. *Proceedings of the Summer School MOVEP'2002*. Ecole Centrale de Nantes, 2002. 450 pages, in English.

[E5] Franck Cassez, Claude Jard, Brigitte Rozoy, and Mark D. Ryan, editors. *Proc. of the Int. Summer School on Modelling and Verification of Parallel Processes (MOVEP'2k)*, volume 2067 of *Lecture Notes in Computer Science Tutorials*. Springer, 2001.

[E6] Franck Cassez, Claude Jard, Brigitte Rozoy, and Mark Ryan, editors. *Proceedings of the Summer School MOVEP'2000*. Ecole Centrale de Nantes, 2000. 300 pages, in English.

[E7] Franck Cassez, Claude Jard, Olivier Roux, and Brigitte Rozoy, editors. *Actes de l'école d'été MOVEP'98*. Ecole Centrale de Nantes, 1998. 280 pages, in french.

— **Recent Research Reports** —

[RR1] Franck Cassez, Stavros Tripakis, and Karine Altisen. Sensor Minimization Problems with Static or Dynamic Observers for Fault Diagnosis. Technical Report RI-2007-1, IRCCyN/CNRS, Nantes, 2007.

[RR2] Franck Cassez, Thomas Chatain, and Claude Jard. Symbolic Unfoldings for Networks of Timed Automata. Technical Report RI-2006-4, IRCCyN/CNRS, Nantes, 2006.

[RR3] Patricia Bouyer, Franck Cassez, and François Laroussinie. Modal Logics for Timed Control. Technical Report RI-2005-2, IRCCyN/CNRS, Nantes, 2005.

[RR4] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Roux Olivier Henri. Comparison of the Expressiveness of Timed Automata and Time Petri Nets. Technical Report RI-2005-3, IRCCyN/CNRS, Nantes, 2005.

[RR5] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim Guldstrand Larsen. Optimal Strategies in Priced Timed Game Automata. BRICS Reports Series RS-04-0, BRICS, Denmark, 2004. ISSN 0909-0878.

— **Miscallenous** —

[M1] Ahmed Bouabdallah, Franck Cassez, Joël Champeau, Philippe Dhaussy, François Lorillard, and Olivier Roux. Vérification d'un logiciel temps-réel distribué: étude comparative des environnements B, SDT et SPIN. $7^{ième}$ journées thématiques *Informatique et électronique embarquées*, Brest, 1998.

[M2] Franck Cassez, Mark D. Ryan, and Pierre-Yves Schobbens. Proving Feature non-interaction with Alternating Time Temporal logic. In *3rd FIREworks Workshop – Language Constructs for Describing Features*, pages 85–104. Springer Verlag, London, 2000. Copyright Springer Verlag.

1.9 Acronyms

1 – **UBO**: Université de Bretagne Occidentale (University of Western Brittany), Brest, France.
2 – **CNRS**: Centre National de la Recherche Scientifique, `http://www.cnrs.fr`
3 – **IRCCyN**: Institut de Recherche en Communication et Cybernétique de Nantes (Research Institute in Communications and Cybernetics of Nantes), `http://www.irccyn.ec-nantes.fr`
4 – **ENSAI**: Ecole Nationale de la Statistique et de l'Analyse de l'Information (French National School in Statistics).
5 – **ECN**: Ecole Centrale de Nantes, France.
6 – **SPIN**: Model-checker for communicating automata, `http://spinroot.com/`
7 – **UPPAAL**: Model-checker for timed automata, `http://www.uppaal.com`
8 – **FIREworks**: Feature Integration in Requirements Engineering
`http://www.cs.bham.ac.uk/~mdr/fireworks/`
9 – **DOTS**: Distributed Open Timed Systems. Project funded by ANR (French National Research Agency) « Sécurité Informatique », started January 2007, duration: 4 years; `http://www.lsv.ens-cachan.fr/anr-dots/`
10 – **CORTOS**: ACI program « Sécurité Informatique », Control and Observation of Real-Time Open Systems `http://www.lsv.ens-cachan.fr/aci-cortos/`
11 – **ACI**: Action Concertée Incitative which is a French government program to sponsor research.
12 – **CHRONO**: ACI « Jeunes Chercheurs », Tools and Algorithms for the Verification of Hybrid Systems `http://www.irccyn.fr/franck/aci-chrono/`
13 – **MOVEP**: Modeling and Verification of Parallel Processes. Summer School. `http://movep.labri.fr/`
14 – **MOVES**: Modeling and Verification of Embedded Systems
15 – **NICTA**: National Information and Communications Technology Australia. `http://nicta.com.au`
16 – **CRAC**: Laboratoire de Conception et de Réalisation des Applications Complexes, Ecole Polytechnique de Montréal, Canada.
17 – **CAV**: Conference on Computer Aided verification.
18 – **TACAS**: Part of ETAPS, conference on Tools and Algorithms for the Construction and Analysis of Systems.
19 – **FOSSACS**: Foundations of Software Science and Computation Structures.
20 – **CONCUR**: Conference on Concurrency Theory.
21 – **BioCONCUR**: Workshop on Concurrent Models in Molecular Biology.
22 – **LICS**: Conference on Logics in Computer Science.
23 – **HSCC**: Hybrid Systems: Computation and Control.
24 – **FSTTCS**: Foundations of Software Technology and Theoretical Computer Science.
25 – **CSL**: Computer Science Logic
26 – **Workshop CORTOS:** `http://www.lsv.ens-cachan.fr/aci-cortos/workshop-concur06/`
27 – **BRICS**: Basic Research In Computer Science, Denmark.
28 – **DRI**: Direction Relations Internationales du CNRS.
29 – **CISS**: Center for Indlejrede Software Systemer (Center for Embedded Software Systems). `http://www.ciss.dk`
30 – **FNRS**: Fonds National de la Recherche Scientifique; Belgium.
31 – **DREI**: Direction Relations Européennes et Internationales du CNRS, formerly DRI.

Chapter 2

Overview of my Research Contributions

In this chapter, I give an overview of some results I have contributed to the domain of control of timed systems. In the last few years I have also been working on the expressive power of time Petri nets and timed automata. I have not developed this line of work as it was done with Olivier H. Roux, and a summary of the results we have obtained together is already available (in French) from Olivier's "Habilitation à Diriger les Recherches" [84].

Sections (2.1–2.3) give an introduction to the domain of control for timed systems. They were inspired from an earlier synthesis article [9] (in French). This paper was prepared for an invited session at the French conference *Modeling of Reactive Systems* (MSR in French) where we have presented the results obtained during the course of the project CORTOS (see section 1.3, page 2 and page 15 for the URL). I was in charge of this paper and I presented the invited talk at the conference. For further results, an excellent synthesis of the results on control for timed systems can be found in [8].

In the next section 2.4, I give an overview of some more advanced subjects in the domain of control for times systems, and emphasise some results I contributed to the domain.

2.1 Introduction

A classical approach to verification of critical systems, is *model-checking* [7]: this amounts to building a model (S) of the system we want to check (*e.g.,* the model can be a finite automaton), to give a formal statement of the correctness property (ψ) the system has to satisfy (*e.g.,* the property can be given in a temporal logic), and to use a *model-checking* algorithm to check whether S satisfies (or not) the property ψ ("S satisfies ψ" is often denoted $S \models \psi$). The model-checking technology is now mature enough to be used in an industrial context.

Nevertheless, models like finite automata cannot capture quantitative timing constraints easily nor efficiently. Consider a scheduling problem with complex synchronisation and non-deterministic tasks durations. It is possible to use finite automata and a discrete time domain to model the problem but there are some drawbacks: one has to encode manually all the behaviours (*e.g.,* if the duration of a task is within 1 and 3 time units, the model will contain explicitly the cases for 1, 2 and 3 time units), and moreover if the constants that appear in the specification have different order of magnitude, say 1 to 1000000, one has to consider

that the discrete time scale is 1 and the model will contain explicitly 1000000 different cases; this will result in a huge number of discrete states (state explosion problem) and thus makes verification a very hard task.

To handle features like non-deterministic tasks durations, and to have a compact and concise specification of a system, we can use an extension of the finite automaton model: *Timed Automata*, TA. Timed automata were introduced in [1], and are finite automata enriched with dense time *clocks*. Model-checking algorithms (as well as temporal logics) for discrete (finite) systems have been extended to handle the quantitative aspects of TA, and efficient tools were developed to model-check TA: UPPAAL [89], KRONOS [88], CMC [91, 86] or HyTech [87] for *Hybrid Automata* which are an extended version of TA that can handle more general types of clocks (like stopwatches that can be stopped and resumed).

The Controller Synthesis Problem. To verify the correctness of a system, for instance using a model-checking technique, it is necessary to have a complete model of the system: for a lift system, one has to build a model of the lift and a model of the control program in order to verify anything meaningful. Using an "automated systems" terminology, such systems are *closed loop* systems.

The environment S the control program has to supervise (*e.g.,* the lift) is called an *open* system. It is the purpose of *controller synthesis* to check that a control program, say C, exists such the closed system "S supervised by C" (denoted $(S \parallel C)$ in the sequel), is correct.

If the correctness property is ϕ, the verification problem is "Does $(S \parallel C)$ satisfy ϕ?". The model-checking problem can be formally defined by:

$$\text{Given } S, C \text{ and } \phi, \text{ does } (S \parallel C) \models \phi \ ? \tag{MC}$$

As said previously, this approach requires to build a model of the controller C, of the environment S and of the compound system $(S \parallel C)$. Guessing a controller might be a difficult task for complex systems, of for complex properties (*e.g.,* a property involving timing constraints). Moreover, if we can design a controller, it could happen that the property is not satisfied by the closed system. In this case, it is necessary to patch the controller. This process can be applied iteratively until a good controller is found. This guess/check method has some drawbacks: it could be that the handcrafted controller we designed is not very efficient (in terms of lines of code for instance) and a better one exists; even worse, it might be that no controller exists to enforce the property ϕ on the environment S and the iterative process of guess/check will end up in an infinite loop.

Ideally, we would like to start with a specification of the system to be controlled S, a property to be enforced, ϕ, and compute or *synthesise* a controller C s.t. $(S \parallel C) \models \phi$. Before computing a controller, we have to check that one exists. This is the *Control Problem* (CP) defined formally by:

$$\text{Given } S \text{ and } \phi, \text{ is there any } C \text{ s.t. } (S \parallel C) \models \phi \ ? \tag{CP}$$

The answer to the control problem (CP) depends on many parameters: for instance, which type of controllers we are looking for. We can decide to look for a finite state controller (with bounded memory) or a more powerful one which has its own clocks and can measure durations. If the answer to (CP) is "yes", a natural question is to compute a witness controller: this is the *Controller Synthesis Problem* (CSP):

$$\text{If the answer to the (CP) is "yes", can we compute a witness controller } C \ ? \tag{CSP}$$

The previous problems have been identified and studied for discrete event systems in the pioneering work of Ramadge and Wonham [12, 13]. Another way of addressing the control problem is to use the framework of *Game Theory*.

From Control to Games. The control problem can be seen as a *game problem* in which a controller has to win against an environment. As such, it can formalised using the *game theory* developed for discrete systems [14, 18, 15, 17] and later extended to timed systems [20, 25, 23, 29]. For instance, an open system can be specified using a *Timed Game Automaton* (TGA) like the one given in Figure 2.1. A TGA is a standard TA in which the actions are partitioned into two disjoints sets: *controllable* actions which can be ... controlled, and *uncontrollable* actions on which the controller has no control. In the TGA of Figure 2.1, controllable transitions

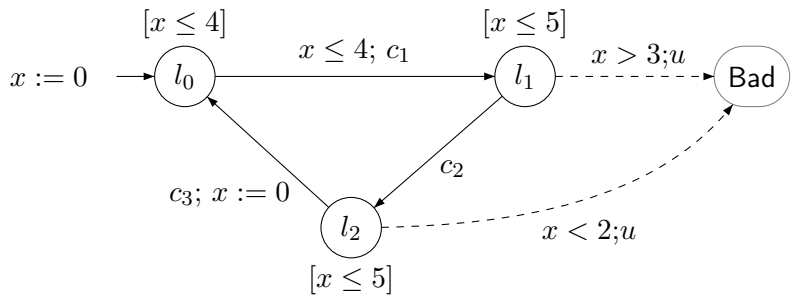

Figure 2.1: A Timed Game Automaton S.

are pictured using plain lines whereas uncontrollable transitions are pictured by dashed lines. The type of a transition is determined by the type of its action. Actions are partitioned into $\Sigma_c = \{c_1, c_2, c_3\}$ for controllable ones and and $\Sigma_u = \{u\}$ for uncontrollable ones.

The TGA of Figure 2.1 specifies a control problem where the controller is in charge of Σ_c *i.e.,* the controller can choose when to play a Σ_c-action, and Σ_u are uncontrollable actions played by the environment. In this example, the goal of controller is to avoid the state Bad, and this whatever the environment is doing. This type of game where one has to keep the system within a set of *good* states is called a *safety* game.

Timed Automata. A timed automaton (TA) is a finite automaton enriched with *clocks*, the values of which are in a dense time domain like $\mathbb{R}_{\geq 0}$. In the example of Figure 2.1, x is a clock of the TA S. The behaviours generated by a TA like S begin in the initial state $(\ell_0, x = 0)$. The semantics of TA is such that a state has two components: one discrete component which is the *location* in which the TA is, and the other component is an *assignment* of the clocks to real values in $\mathbb{R}_{\geq 0}$. This means that, when a dense time domain is considered, a TA has an infinite (sometimes uncountable) set of states.

In a TA, the values of the clocks increase synchronously as time elapses. Each location is tagged with an *invariant*[1] which is a constraint imposed on time elapsing. For instance, in location ℓ_0, the invariant is $[x \leq 4]$. This imposes that from state $(\ell_0, x = 0)$, the maximum amount of time that can elapse is 4 time units. For any $\delta \in \mathbb{R}_{\geq 0}$ with $\delta \leq 4$, time can elapse from $(\ell_0, x = 0)$ and the system S reaches state $(\ell_0, x = \delta)$: this is a *time* transition. A *discrete* transition can be fired if the constraints given by the *guard* of the transition are satisfied: for instance c_1 can be triggered when $x \leq 4$. The automaton S of Figure 2.1 imposes that transition c_1 occurs before 4 time units have elapsed from the initial state: the invariant

[1] Invariants are within square brackets in the figures.

$[x \leq 4]$ prevents time elapsing beyond $x = 4$.

The semantics of a TA is a *Timed Transition System* (TTS) which is a standard transition system with two types of transitions: *time* transition labelled by some $\delta \in \mathbb{R}_{\geq 0}$ and *discrete* transitions labelled by an action in $\Sigma_c \cup \Sigma_u$. A behaviour generated by a TA is a sequence of alternating time and discrete transitions. For example, the automaton S of Figure 2.1 can generate the following runs[2] ρ_1 and ρ_2:

$$\begin{aligned}
\rho_1 : \quad & (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\mathsf{Bad}, 3.22) \\
\rho_2 : \quad & (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2) \\
& \xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{c_3} (\ell_0, 0) \cdots
\end{aligned}$$

In a TTS there are infinitely many time transitions (with labels $\delta \in \mathbb{R}_{\geq 0}$) and infinitely many states. Nevertheless, for any TA A, the TTS that gives the semantics of A admits a finite abstraction (the *region automaton or graph* from Alur & Dill [1]) which contains enough information to prove most of the correctness properties of A. The nodes of this abstraction are *symbolic* states which are pairs of the form (ℓ, Z) where ℓ is a location of A and Z is a *zone i.e.,* a convex set of valuations of the clocks. In the example of Figure 2.1, we can have a symbolic node of the form $(\ell_0, 1 \leq x < 3)$ (actually this type of nodes with a zone defined by a non-elementary constraint $1 \leq x < 3$ cannot be in the region graph; it can be encountered in a coarser abstraction of this graph, the *simulation* graph). In case the set of clocks X has more than one element, zones are subsets of $\mathbb{R}^X_{\geq 0}$ and can be defined by diagonal constraints [1, 4].

To come back to the safety game defined by S in Figure 2.1, if the controller wants to avoid location Bad, it must restrict the set of behaviours: for instance it should not play c_1 in ℓ_0 if $x > 3$, as the environment could reply instantaneously by playing u from ℓ_1. When in location ℓ_1 with $x \leq 3$, it should not wait too long before playing c_2, because as soon as $x > 3$, the environment can again play u and S reaches location Bad.

2.2 Semantics of Timed Games

When playing a timed game, we must define the rules that the players must follow. In the case of a discrete (no time) game, the possible moves for each player are discrete actions. Each player has its own set of actions, Σ_c and Σ_u, and some of them are enabled in each state. The same semantics apply for discrete transitions in timed games: in the example of Figure 2.1, c_1 can be fired from $(\ell_0, x \leq 4)$, and u from $(\ell_1, x > 3)$ and $(\ell_2, x < 2)$. Still there is another available action in the timed setting: the "do nothing" or "wait" action. A player can choose to wait before playing a discrete action.

2.2.1 Rules for Playing Timed Games

In discrete games theory [18] (where games are finite automata, pushdown automata, etc), there are two types of games:

- *turn-based* games in which the moves of the players alternate. At each point in time, one player chooses the next state of the game;

[2] We use the notation (ℓ, v) for a state of S instead of $(\ell, x = v)$.

- *concurrent* games in which each player chooses a move and the next state is the result of the combination of the two choices.

The turn based semantics is not always adequate for timed games[3]: the moves of the players depend on the time elapsed since a particular instant and thus the semantics is similar to concurrent games. Two classical semantics have been proposed for timed games:

- timed games with *continuous observation* [20, 25]. With this semantics we assume that the players can observe each other continuously. They can choose an action based on the complete history of system's evolution. In the example of Figure 2.1, in location ℓ_1, the controller can choose to wait (until $x = 5$) or to fire c_2: this choice is based on the current state (full observation) of the system, for instance the controller can choose to wait for $x < 3$ and to fire c_2 when $x = 3$. Notice that there is no priority in this semantics: for instance, from the state $(\ell_1, x = 4)$, if the controller chooses to fire c_2, the environment can choose simultaneously to fire u. The next state of the system is chosen non-deterministically from the target states of these two actions *i.e.,* it is either ℓ_2 or Bad. If the controller has to avoid location Bad, it has to enforce the firing of c_2 before the "date" $x = 3$ (included). As the controller can observe the system continuously, it is never committed to wait for a particular amount of time. In this sense, the environment cannot play a move by surprise.

- timed games with an "element" of *surprise* [29]. In this semantics, each player must announce its choice for the next move. The choices are pairs $(\text{duration}, \text{action})$ where the action is either a discrete move or the special action "do nothing". A choice for a player is a pair (δ, a) and stands for: "I will do the a-action in δ time units". Once they have announced their choices, the players cannot change their mind. Assume the controller chooses (δ_1, a_1) and the environment chooses (δ_2, a_2). The next state of the system will be determined by the move that occurs first in the future. From state s, if $\delta_1 < \delta_2$, the next state of the system, s'', is obtained by computing the state of system after a continuous evolution of δ_1 time units, say s', and then the successor state of s' after the discrete move a_1. This semantics is symmetric and if $\delta_2 < \delta_1$ the next state is the one obtained after δ_2 and a_2. In case $\delta_1 = \delta_2$, the next state is non-deterministically chosen in next states obtained after (δ_1, a_1) and (δ_2, a_2). For instance, once the controller has announced a move (δ_1, a_1), it cannot change its mind, and in this sense there is an element of surprise: the environment can play a move before δ_1 time units have elapsed.

2.2.2 Strategies

In the framework of game theory, the control problem (CP) amounts to: "Is there a *strategy* for the controller to win the game?" (the winning condition is discussed in section 2.2.3) *i.e.,* a controller is a strategy and *vice versa*. The notion of strategy is the standard one which is commonly used in board games. Given a complete history of the game (from the initial state), a strategy indicates which action to play. In the case of *real-time* games with the continuous observation semantics, a strategy is a partial[4] function from the set of runs of the

[3]Such a semantics is meaningful when one considers a physical environment which evolves only by continuous moves, and a digital controller that can only react at predefined instants as in [26].

[4]It can happen that a run of the system ends in a state from which no Σ_c actions are enabled and time cannot elapse. In this case it is not possible to define a strategy for the controller. Either the system is in a deadlock state or the environment can play.

system (*i.e.,* sequences of alternating discrete and time steps) to the set $\{\lambda\} \cup \Sigma_c$ where λ is the special move "do nothing". For the example of Figure 2.1, we can define the strategy f by: $f(\rho.\ell_0, x < 2) = \lambda$, $f(\rho.\ell_0, x = 2) = c_1$, where $\rho.\ell_i$ stands for "any run ending in ℓ_i". This strategy means: wait when in ℓ_0 with $x < 2$, and do c_1 when $x = 2$. if we use this strategy to control the system S of Figure 2.1, it is impossible to generate a run ending in $(\ell_0, x > 2)$ from $(\ell_0, x = 0)$ as the strategy imposes that c_1 be taken when $x = 2$ is reached. A strategy restricts the set of behaviours the system can generate and consequently the set of reachable states. Notice also that a strategy must be consistent in the sense that the move it proposes is valid: a strategy f' with $f'(\rho.\ell_0, x > 4) = c_1$ is not consistent as c_1 is not enabled in $(\ell_0, x > 4)$.

A strategy like f above, which depends on the last state of a run and not on the complete history of the game, is a *positional* or *memoryless* strategy.

In the semantics with "surprise", a strategy is a function from the set of runs to the set of pairs of $\mathbb{R}_{\geq 0} \times (\{\lambda\} \cup \Sigma_c)$. If we impose that all the moves are of the form $(0, c)$ with $c \in \Sigma_c$, or (δ, λ) with $\delta \in \mathbb{R}_{\geq 0}$ (*i.e.,* both players always propose the same δ when they play the λ action), the element of surprise disappears.

2.2.3 Control Objectives

In section 2.1, the control problem (CP) is parametrised by the correctness property we want to enforce. The simplest properties are (state or location based) *safety* or *reachability* properties. A *safety* property ϕ consists of a set of safe (resp. unsafe) states and the objective of the controller (the control objective) is to keep the system within (resp. outside) the set of safe (resp. unsafe) states defined by ϕ. A control problem with a safety control objective is *safety control problem* (SCP). A reachability control objective ϕ is a set of states which must be forced: the control problem amounts to finding a strategy to force the system to reach a state in ϕ. This version of the control problem is the *reachability control problem* (RCP). Control objectives can be defined using temporal logic formulae like LTL [16] or TCTL [23]. It is then possible to write a control objective like "the system has to reach a ϕ_1-state infinitely often and avoid any ϕ_2-state". Finally, a more expressive class of control objectives is the class of ω-regular control objectives [18].

2.2.4 Winning Strategies

In this section we focus on safety control problems with the continuous observation semantics [20, 25]. Let G be a TGA (like the TGA S in Figure 2.1), and let *Reach*(G) be the set of states reachable in (the open system) G. Let ϕ be the safety property which consists of a set of safe states. We consider the SCP for G and ϕ. A strategy (equivalently a controller) is winning if it keeps the system G in ϕ states. Notice that even if a strategy f is deterministic, there can be more than one run that can be generated in G supervised by f. The game G supervised or controlled with the strategy f is denoted $f(G)$. A strategy only imposes restrictions on the date at which a controllable action is taken but uncontrollable actions cannot be affected by it. For example, using the strategy $f(\rho.\ell_2, x < 1) = \lambda$, and $f(\rho.\ell_2, x = 1) = c_3$ the closed system $f(G)$ can produce the runs $(\ell_2, x = \nu < 1) \xrightarrow{\delta} (\ell_2, x = \nu + \delta < 1)$ and $(\ell_2, x = \nu < 1) \xrightarrow{u} (\mathsf{Bad}, x = \nu)$. From the state $(\ell_2, x = 1)$, there is no priority to controllable actions (like c_2), and the two runs $(\ell_2, x = 1) \xrightarrow{c_3} (\ell_0, x = 1)$ and $(\ell_2, x = 1) \xrightarrow{u} (\mathsf{Bad}, x = 1)$

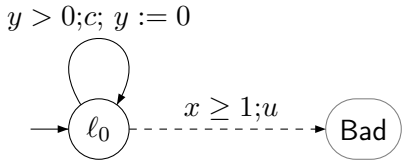

Figure 2.2: Zeno Controller

are possible outcomes of $f(G)$. As a result the set of runs of $f(G)$ is a subset of the set of runs of G and $\mathit{Reach}(f(G)) \subseteq \mathit{Reach}(G)$. The SCP reduces to: "Is there a strategy f s.t. $\mathit{Reach}(f(G)) \subseteq \phi$?". If a strategy f satisfies the previous inclusion, f is a *winning* strategy. It could be that the trivial strategy f_λ with $f_\lambda(\rho) = \lambda$ for any run ρ is winning: checking that f_λ is winning can be done using a model-checking algorithm to check that $\mathit{Reach}(f_\lambda(G)) \subseteq \phi$. Even if $\mathit{Reach}(f_\lambda(G)) \subseteq \phi$, this winning strategy is not always valid as it can create a deadlock in the supervised game. Of course, we want to avoid this and to look for non-blocking strategies [20, 22, 38]. Another trick a strategy can play in real-time games is to prevent time from diverging. It can do so by generating *zeno* behaviours *i.e.,* behaviours containing an infinite number of discrete actions in a finite amount of time. On the example of Figure 2.2, to avoid Bad and keep the system in the safe states $\phi = \{(\ell_0, x, y \geq 0)\}$, we can use the following strategy:

- let ρ_n be the run $(\ell_0, x = 0) \xrightarrow{(\delta_0, c)} \cdots \xrightarrow{(\delta_n, c)} (\ell_0, x = \sum_{i=0}^{n} \delta_i)$ where $\xrightarrow{(\delta_i, c)}$ means "wait δ_i time units and play c";
- define $f(\rho_n) = (\frac{1}{2} \cdot (1 - \sum_{i=0}^{n} \delta_i), c)$.

This strategy f is non-blocking if we take $0 < \delta_0 < 1$. The outcome of the strategy contains a single (infinite) run which avoids Bad. Nevertheless, it prevents time from diverging, as the total duration of this run is upper bounded by 1: f produces a zeno run. If we think of an implementation of this strategy, it has to react faster and faster as time increases. The amount of time between two c actions must become smaller each time c is played, and tends to zero. It is not a realistic or an implementable strategy and some methods have been proposed to build non-zeno strategies [28, 29].

2.2.5 Winning States

A *winning* state is a state from which there is a winning strategy. Let $\mathcal{W}$ be the set of winning states. The control problem can be solved easily if we can compute the set of winning states: if suffices to check that the initial state of the game belongs to $\mathcal{W}$. A good property of the continuous observation semantics is that timed games are *determined*: this means that each state s of the game is either winning for the controller or winning for the environment (in this case it is losing for the controller). Moreover, if the initial state is winning, there is a (global) strategy f, which is memoryless, and winning (*Cf.* Theorem 2 in section 2.3.3). The fact that timed games are determined under the continuous observation semantics enables one to reduce *reachability* timed games to safety games: the control objective "the controller must enforce ϕ" is transformed into "the environment must avoid ϕ".

Timed games under the "surprise" semantics are not determined. Moreover, winning a timed game under this semantics can be more complicated than for the continuous observation

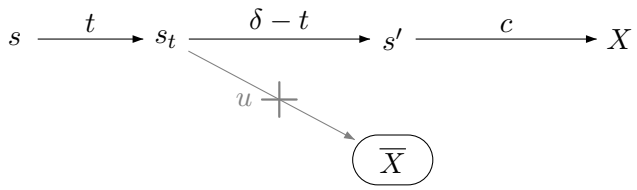

Figure 2.3: Controllable Predecessors

semantics and you may need unbounded memory [29] whereas memoryless strategies suffice under the continuous observation semantics (Theorem 2).

2.3 Algorithms for Controller Synthesis

Algorithms for the synthesis of timed controllers have been given in [20, 25] and later generalised in [27, 29]. In this section, we describe an algorithm to compute winning states of safety timed games taken from [20]. The result of the computation of this algorithm for the example of Figure 2.1 is given at the end of this section.

2.3.1 Controllable Predecessors

Let $CPre(X) = \{s \,|\, \exists c \in \Sigma_c \,|\, s \xrightarrow{c} s' \wedge s' \in X\}$ and $UPre(X) = \{s \,|\, \exists u \in \Sigma_u \,|\, s \xrightarrow{u} s' \wedge s' \in X\}$. These two sets give the states from which there is an outgoing controllable (resp. uncontrollable) transition leading to X.
Figure 2.3 pictures the conditions for a state s to be a *controllable predecessor* of a set of states X: the controller must be able to *force* the game into a state of X, first by time elapsing (δ) and then by firing a controllable transition c, and from each state s_t (with $t \leq \delta$) encountered during the time elapsing phase, the environment must not be able to fire an uncontrollable transition leading outside of X (denoted $\overline{X}$).
A state s is a controllable predecessor of X iff:

1. there is some δ, $\delta \geq 0, s \xrightarrow{\delta} s' \wedge s' \in CPre(X)$
2. for all $0 \leq t \leq \delta$ s.t. $s \xrightarrow{t} s_t$, we have $s_t \notin UPre(\overline{X})$

The set of controllable predecessors of X is denoted $\pi(X)$. In the example of Figure 2.1, the state $(\ell_1, x = 1)$ is a controllable predecessor of $(\ell_2, x = 2)$: a controller can let time elapse (1 time unit) until $(\ell_1, x = 2)$ and then fire c_2 from ℓ_1 to ℓ_2. During the time elapsing phase no uncontrollable transition can be fired.

2.3.2 Symbolic Controllable Predecessors

As said previously, the semantics of a TGA is a TTS and contains infinitely many (even uncountable) transitions and states. To analyse this type of models, states are represented symbolically: a *symbolic state* is a finite union of pairs of the form (ℓ, Z) where ℓ is a location and Z is zone *i.e.,* a convex set of clock valuations. The operators we have introduced in the previous section have two important properties. Let Z be a zone, then:

Computation of W^*

i	ℓ_0	ℓ_1	ℓ_2
0	$0 \leq x \leq 4$	$0 \leq x \leq 5$	$0 \leq x \leq 5$
1	$0 \leq x \leq 4$	$0 \leq x \leq 3$	$2 \leq x \leq 5$
2	$0 \leq x \leq 3$	–	–

Most Permissive Strategy f^*

	ℓ_0	ℓ_1	ℓ_2
λ	$x < 3$	$x < 3$	$x < 5$
c	$x \leq 3$	$2 \leq x$	$x \leq 5$

Table 2.1: Symbolic Computation for the example of Figure 2.1.

$\mathbf{P_1}$: $CPre(Z)$, $UPre(Z)$ and $\pi(Z)$ are unions of zones,

$\mathbf{P_2}$: $CPre(Z)$, $UPre(Z)$ et $\pi(Z)$ can be effectively computed[5].

For the automaton of Figure 2.1, we have $CPre(\ell_1, x \leq 3) = (\ell_0, x \leq 3)$ and if $Z = (\ell_0, x \leq 4) \cup (\ell_1, x \geq 0) \cup (\ell_2, x \geq 0)$ then $\pi(Z) = Z'$ avec $Z' = (\ell_0, x \leq 3) \cup (\ell_1, 0 \leq x \leq 3) \cup (\ell_2, x \geq 2)$.

2.3.3 Symbolic Computation of Winning States

For TGA and a safety control objective ϕ, the set of winning states $\mathcal{W}$ (defined in section 2.2.5) is the greatest fixpoint (see [25]) of the function $h(X) = \phi \cap \pi(X)$ (for reachability games this is the least fixpoint of $h(X) = \phi \cup \pi(X)$). Notice that $\mathcal{W}$ is the maximal set of winning states: thus $s \in \mathcal{W}$ iff s is winning. If ϕ is defined as union of zones, then the symbolic iterative computation $X_0 = \phi$, $X_{i+1} = h(X_i) = \phi \cap \pi(X_i)$ is such that each (i) X_i is a zone (the intersection of two zones is a zone) and (ii) there is some index k_0 s.t. $X_{k_0} = X_{k_0+1} = W^*$ and thus the computation of the fixpoint W^* terminates in a finite number of steps. Then we can effectively decide whether the initial state belongs to the symbolic representation W^* of $\mathcal{W}$ and thus we have the following result:

Theorem 1 ([25, 19]) *The control problems SCP and RCP are decidable for TGA and EXPTIME-complete.*

For the example of Figure 2.1, the iterative computation of W^* is given in Table 2.1: the current winning zone is given for each location in the corresponding column. For more complex properties, computing symbolically the winning states requires more involved fixpoints [20, 25, 27, 29].

2.3.4 Synthesis of Winning Strategies

To solve the Control Synthesis Problem for a safety game, we can define a *most liberal* or *most permissive* strategy. Actually, this strategy is not a strategy in the sense of section 2.2.2 as it maps states to sets of controllable actions and not to a single controllable action. The most permissive strategy f^* associates with each run ρ a set of actions from $\Sigma_c \cup \{\lambda\}$ which are safe: every non-blocking strategy f (in the sense of section 2.2.2) s.t. $f(\rho) \in f^*(\rho)$ is winning and conversely a strategy f is winning only if $f(\rho) \in f^*(\rho)$. This is why f^* is called the most permissive strategy. For the synthesis problem we have the following result:

[5] To compute $\pi(X)$, another operator is needed and we do not introduce it here. This operator satisfies properties $\mathbf{P_1}$ and $\mathbf{P_2}$.

Theorem 2 ([25]) *Let G be a TGA and ϕ a safety control objective. If the initial state of G is winning, there is a most permissive memoryless winning strategy.*

To compute this most permissive strategy, we begin by strengthening the guards of the controllable transitions such that, from a winning state s, firing the transition with the strengthened guard leads to a winning state. This amounts to computing a most liberal pre-condition. In our case, we strengthen (the guard of) transition c_1 by $x \leq 3$, c_2 by $x \geq 2$ (which ensures that c_2 is not fired too early). The most permissive strategy is then defined by: wait in each winning state s such that there is some $\delta > 0$ and $s \xrightarrow{\delta} s'$ with s' winning; fire a controllable transition c_i is the strenghtened guard is satisfied. For the example of Figure 2.1 the most permissive strategy is given in Table 2.1. Notice that to obtain a non-zeno strategy, one should not choose infinitely often λ in ℓ_0 (or ℓ_1, ℓ_2), but at some point fire c_1. Using the previous method we can compute a timed automaton C (given in Figure 2.4) which is a representation of the most permissive strategy f^*, and such that $\mathit{Reach}(G \parallel C) \subseteq \phi$ with $\phi = \mathit{Reach}(G) \setminus \{\mathsf{Bad}\}$. For reachability games, computing a winning strategy may be a bit more complicated [59].

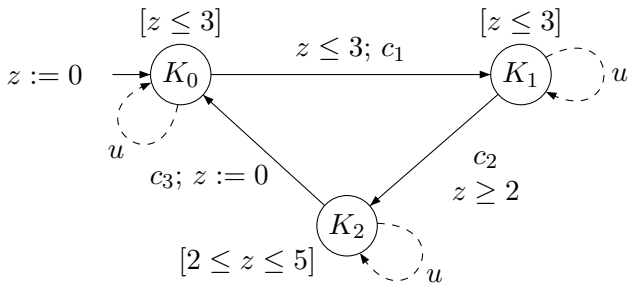

Figure 2.4: C: The Most Permissive Controller for the Example of Figure 2.1.

2.4 Contributed Results

In this section, I give a summary of some recent results obtained in the domain of controller synthesis for timed systems. The subsections address different advanced subjects about controller synthesis, and for each subsection, I describe my contribution to the domain.

All the results I will present were obtained in collaboration with colleagues in France and abroad. In particular, the co-authors of the papers presented hereafter are: Karine Altisen (VERIMAG, Grenoble, F.), Patricia Bouyer (LSV, ENS Cachan, F.), Alexandre David (CISS, Aalborg University, DK), Emmanuel Fleury (LaBRI, Bordeaux, F.), Thomas Henzinger (EPFL, Lausanne, CH), Didier Lime (IRCCyN, Nantes, F.), François Laroussinie (LSV, ENS Cachan, F.), Kim G. Larsen (CISS, Aalborg University, DK.), Jean-François Raskin (Université Libre de Brussels, B.) and Stavros Tripakis (VERIMAG, Grenoble, F. et Cadence, Berkeley, USA).

Some work presented in this section roughly consists in freezing a parameter (the model for timed games, the semantics or the type of control objective) of the control problem and studying the impact of the other ones:

- in the work presented in section 2.4.1, we focus on the SCP or the RCP, with a sampling

semantics, *i.e.,* (SCP,sampling), and study the class of timed automata for which (SCP, sampling) is decidable. This work was published in [28].

- in section 2.4.2, we study the decidability status of the control problem for control objectives expressed in a fixpoint modal logic. This work was published in [39].

In the following section 2.4.3, we study the optimal control problem for *Priced Timed Game Automata*. Timed automata were extended in [47, 46] with *cost* or *price*. The corresponding model is *Priced Timed Automata* (PTA) also called *Weighted Timed Automata* (WTA). Each run of a PTA has an associated cost. The cost of a run is computed by summing up the costs of the discrete transitions and time transitions. The *optimal path* problem for PTA, is to compute a path of minimal cost. The natural extension of this one player problem is to consider the optimal control problem for *Priced Timed Game Automata* (PTGA) in which some transitions are uncontrollable. This model and the control problem were defined in [51, 50].

In the next section 2.4.4, we focus on an important aspect of controller synthesis which is to design efficient algorithm for computing controllers. This section is based on the results of paper [68].

Finally, we conclude with some recent work on fault diagnosis for timed systems. Fault diagnosis is very similar to control under partial observation. This work is presented in section 2.4.5 and based on the paper [77].

2.4.1 Decidability Issues for Timed Control

In this section we focus on the safety control problem. As timed games with an element of surprise do not have good properties (they are non determined, may need unbounded memory controllers), we restrict our attention to continuous observation games. We can consider the following variants of this semantics:

- the controller can only react at predefined instants, and these instants are separated by a fixed amount of time. This is the *sampling* control problem [19]. Notice that the environment can only do continuous moves in this framework.
- the controller cannot control time elapsing: it can choose to do a discrete action or to wait, but in the latter case, the environment chooses the amount of time that will elapse. This variant is called *time-abstract control* [26].

Discrete and Continuous Control. Variants of the safety control problem have been proposed in many papers [26, 19, 25, 20]. In the paper

> **[28]** Franck Cassez, Thomas A. Henzinger, and Jean-François Raskin.
> **A comparison of control problems for timed and hybrid systems.**
> In *Proc. 5th International Workshop on Hybrid Systems: Computation and Control (HSCC'02)*, volume 2289 of *LNCS*, pages 134–148. Springer, 2002.

we have given a classification of these problems and we have compared them. First we have identified the following main classes:

- discrete time control: can we control the system with a controller that reacts every β time units? This is the *Known Sampling Rate* (KSR) control problem. A more general question is ask for the computation of a value β s.t. the system is controllable. This is the *Unknown Sampling Rate* (USR) control problem.

- dense time control: in this setting the controller can either (i) control time elapsing, which is the setting of the seminal papers [20, 25]; in this case this is the *Unknown Switch Condition* (USC) control problem; or (ii) cannot control time elapsing and in this case this is the *Known Switch Condition* (KSC) control problem.

The decidability status of the standard safety control problem also depends on the expressive power of the formalism used to specify timed games. Before giving the decidability status of the previous problems (KSR, USR, USC, KSC) we review some well-known classes of extensions of timed automata.

Hybrid Automata. The more general class that is usually considered for the control problem is the class of *Linear Hybrid Automata* (LHA) [5]. This class extends timed automata in the following manner: (i) the evolution rate of the variables[6] are given by intervals of the form $1 \leq \dot{x} \leq 3$; (ii) guards and resets are linear functions like $2x + 3y \leq 4$ et $x := 3y - 7z$.

An interesting subclass of LHA is the class of *Rectangular Automata* (RA) [19]. For RA, the guards and resets cannot involve two distinct variables: they are of the form $x \geq 1 \wedge y < 2$ and for resets $x :=]2, 3]$ to assign a value v with $2 < v \leq 3$ to x. This class has interesting and surprising decidability results: the reachability problem is decidable for RA [19]. A subclass of RA is the class of *Initialised Rectangular Automata* (IRA): if a derivative of a variable changes on a transition from ℓ to ℓ', then the variable must be reset on this transition.

The reachability problem is decidable for RA but for any subclass if becomes undecidable [3]. Another interesting class of hybrid systems with good decidability properties, is the class of *O-minimal Automata*, (OminA) [2]: in this model, the dynamics can be non-linear (exponential or any o-minimal theory) but all the variables must be reset when firing a discrete transition.

Decidability results for the safety control problem on LHA are summarised in Table 2.2:

Formalism	KSC	USC	KSR	USR
Timed Automata	$\surd$ [20]	$\surd$ [20]	$\surd$ [24]	$\times$ **[28]**
Initialised Rectangular Automata	$\surd$ [26]	$\times$ [3]	$\surd$ [19]	$\times$ **[28]**
Rectangular Automata	$\times$ [26]	$\times$ [26]	$\surd$ [19]	$\times$ **[28]**

Table 2.2: Safety Control Problem: Decidability Results. KSC = Known Switch Conditions controller, USC =Unknown Switch Conditions controller, KSR = Known Sampling Rate controller, and USR = Unknown Sampling Rate controller. $\surd$ = decidable, $\times$ = undecidable.

A recent result by Patricia Bouyer, Thomas Brihaye et Fabrice Chevalier [30] is the decidability of the Reachability control problem, USC variant, for o-minimal automata.

Contribution to the Domain. The main result of the paper [28] is that the USR control problem is undecidable, even for the class of timed automata. Another result of the paper is that, there are systems that can be controlled with a dense time controller (USC), but by no sampling controller (USR). We have given an example of such a system, given by the timed automaton of Figure 2.5. This timed automaton can be controlled with a non-zeno controller.

[6] We note $\dot{x}$ the derivative of x.

In this example, all the transitions are controllable. To avoid location Bad, it suffices to avoid firing a d-transition. This can be achieved by computing a most permissive controller: this controller C only requires to leave ℓ_2 when $x < 1$, and thus we add the invariant $[x < 1]$ to location ℓ_2

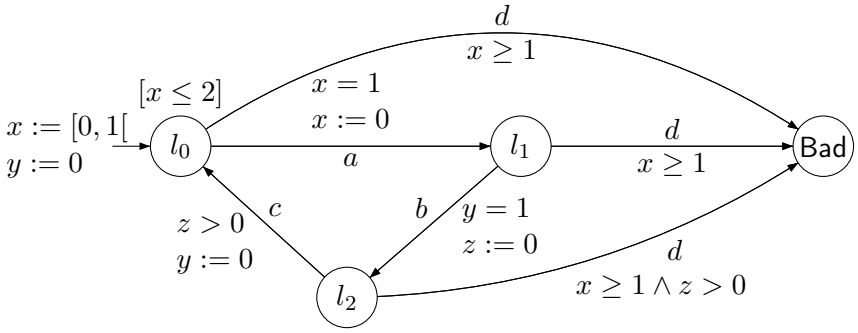

Figure 2.5: A System that Admits no Sampling Controller.

The controller C has to do the sequence of actions $(a.b.c)^\omega$, but the time elapsed between the $n+1$-th a and the $n+1$-th b must be smaller than for the previous round n. More precisely, if the time elapsed between the n-th a and the n-th b is δ_n, the controller has to ensure that $\sum_{i=1}^{+\infty} \delta_i \leq K$ where $K \leq 1$. Because of this, no sampling controller can control the system: given a sampling rate β, after a finite number of rounds m, the sum $\sum_{i=1}^{m} \delta_i$ is beyond 1. The controller C is non-zeno (and non-blocking) because the n-th a action occurs at time n. This shows dense time controllability does not imply sampling controllability. In other words, if the answer to the USC-CP is "yes", the most permissive controller is not always implementable on a digital controller.

The previous result show that the notion of *implementable controllers* is important and must be addressed in the control problem. This aspects of the control problem have been extensively studied by Jean-François Raskin and his group in [63, 66, 65, 64] and a synthesis of this work (in French) is given in [10].

2.4.2 Specification of Control Objectives

In the previous section, we have considered safety (or reachability) control objectives. In this section, we model the systems by timed automata and focus on more general types of control objectives.

Specification of Properties. In [27] the authors give an algorithm to deal with general ω-regular control objectives. It is to be noticed that those control objectives are often called *internal* in the sense that they refer to the state properties (and clocks) of the system to be controlled. The safety and reachability properties that we considered previously are internal property. In the case of timed systems they only refer to the untimed sequences of states of the system and thus have a restrictive expressiveness: it is possible to specify a property like "after p has been reached q will be reached" but nothing like "after p has been reached, q will be reached within less than d time units" (*bounded liveness*). In [37], the authors show that

the control problem is 2EXPTIME-complete for specifications given by LTL formulae[7].

A control objective is a property of a closed system, and it is natural to write properties using temporal logics or automata *i.e.,* to give the set of timed words the controlled system has to generate. The language[8] $\mathcal{L}(A)$ accepted by a timed automaton is a set of behaviours (timed words) and this can be considered to be a control objective: in this sense this is an *external* control objective. Notice that it can either specify good or bad behaviours. Assume we have two timed automata: S_d which defines the desired (permitted) behaviours of the system, and S_u which defines the undesired behaviours. Let E be the system we want to control. The *External Specification Control Problem*, (ES-CP) is the following:

$$\text{Is there any } C \text{ tel que (1) } \mathcal{L}(E \parallel C) \cap \mathcal{L}(S_u) = \emptyset \text{ and (2) } \mathcal{L}(E \parallel C) \subseteq \mathcal{L}(S_d) \text{ ?} \qquad \text{(ES-CP)}$$

In the paper [38], the authors prove that:

1. the sub-problems ES-CP.(1) et ES-CP.(2) are undecidable when S_d and S_u are non-deterministic timed automata;

2. these problems become decidable if S_d and S_u are deterministic; deterministic timed automata can be complemented and in this case the ES-CP can be reduced to a (internal) safety control problem;

3. the control problem ES-CP.(2) is decidable and 2EXPTIME-complete if the resources (number of clocks, granularity and constants used in the specification) of the controller are bounded, and this even if S_u is non-deterministic. ES-CP.(1) remains undecidable even for bounded resources.

Expressing control objectives using timed automata is a very powerful tool and there is a gap between the control objectives we considered previously (safety or reachability) and these very general class of ω-regular properties. Temporal logics are a less powerful yet often sufficient specification formalism for most properties needed in practise.

In [40], the authors consider a timed extension of LTL namely MTL [33]. The control problem is undecidable for MTL specifications but become decidable if the resources of the controller are bounded (it is still non primitive recursive). In [23], the authors consider TCTL specifications without equality and prove that the control problem (for timed automata) can be decided in exponential time for this type of specifications.

Recent work [15, 17] on control of discrete event systems has considered a more general framework in which properties of the controlled system are specified in various extensions of the μ-calculus: *loop* μ-calculus for [15] and *quantified* μ-calculus for [17]. In both cases the control problem is reduced to a model-checking problem. It is then natural to try and extend this work to timed systems.

Contribution to the Domain. In the paper

[7] This result does not contradict the complexity result of [29] because the specification is given as an LTL formula and not by the automaton that recognises the valid sequences.

[8] We consider languages over infinite words *e.g.,* ω-regular languages.

[39] Patricia Bouyer, Franck Cassez, and François Laroussinie.
Modal logics for timed control.
In Martín Abadi and Luca de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR'05)*, volume 3653 of *Lecture Notes in Computer Science*, pages 81–94, San Francisco, CA, USA, August 2005. Springer.

we have used the (timed) temporal logic L_ν which is a fragment of the timed μ-calculus that allows to specify *bounded safety* properties. More precisely, we consider a subset of L_ν, denoted L_ν^{det}, and study the control problem:

$$\text{Given } E \text{ and } \varphi \in L_\nu^{det}, \text{ is there any } C \text{ s.t. } E \parallel C \models \varphi \ ? \qquad (L_\nu\text{–CP})$$

The results we give in the paper are summarised by:

- as in the untimed case, the control problem L_ν–CP can be reduced to a model-checking problem. For this, we need to add a new operator to L_ν, which is a kind of *until* operator, and we define the extended logic L_ν^{cont}. We then show that the control problem L_ν–CP reduces to a model-checking problem of the type $E \models f(\varphi)$ where $f(\varphi)$ is a syntactical translation of φ and is an L_ν^{cont} formula.
- the extended logic L_ν^{cont} is strictly more expressive than L_ν.
- nevertheless, the model-checking problem for L_ν^{cont} against timed automata remains EXPTIME-complete (model-checking L_ν is already EXPTIME-complete [34]).

It is important to notice that the previous results concern the existence of a controller: in this work we have not yet proved that memoryless or bounded memory controllers exist for a control objective in L_ν^{det}. In other words, we have not solved the synthesis problem for L_ν^{det} specifications. Nevertheless, the result we have obtained is useful because it allows to decide at reasonable cost (model-checking EXPTIME) for the existence of a controller: if the answer is "no", we know for sure that there is no timed (nor hybrid) automaton that can control the system.

Open problems concerning the previous work are:

- we can decide the control problem for L_ν^{det} specifications but the controller synthesis is still open. This problem is closely related to the satisfiability problem for L_ν (Cf. [34]) which is also open.
- we have restricted the control objectives to L_ν^{det} but this may not be necessary. For instance, in the untimed case, the controller synthesis problem can be solved for general μ-calculus specifications.

In the previous work we have considered control objectives given in temporal logics used to specify properties of closed systems.

There is a nice logic which was designed to specify properties of open (discrete) systems: *Alternating-time Temporal Logic* (ATL) [35]. This logic generalises CTL in the sense that we can refer to the strategies of the players in the formula: it is possible to write formulae likes "The controller has a strategy to reach a state where it has a strategy to enforce a state q".

Recent work by François Laroussinie, Nicolas Markey and Ghassan Oreiby [41] proposed a timed extension of ATL for timed games with integer durations (time elapsing is a integer). They gave an algorithm to model-check timed ATL specifications against timed automata (using the discrete time step semantics).

2.4.3 Optimal Control

The control problem for timed systems (USC) was introduced and solved in 1995 in [20]. If we consider the reachability control problem, we want to synthesise a controller which enforces a particular state q. Once solved, we can ask for the controller to be *optimal e.g.,* the state q is reached as soon as possible with the controller. This version of optimal control called *time-optimal control* was solved in [57].

Timed Automata with Costs. A few years later in 2001, timed automata were extended with *costs* or *prices*: this extension was called *priced timed automata* (PTA) in [47, 48] and *weighted timed automata* (WTA) in [46] (both models are exactly the same thing but they were proposed independently at the same time by two different groups.)

The notion of cost or price generalises the notion of elapsed time: the cost is a linear function of the elapsed time in each location, and is an integer value on each discrete transition. In the example of Figure 2.6, the costs in the PTA $\mathcal{A}$ are given by the keyword cost. Assume δ time units elapse in location ℓ_0, and then c_1 is fired. The invariant on ℓ_1 prevents time from elapsing in ℓ_1. This location is a choice location in which we immediately go either to ℓ_2 or ℓ_3. Assume we choose ℓ_2 and let δ' time units elapse before firing c_2. In this case the total cost of this execution is $5 \cdot \delta + 10 \cdot \delta' + 1$. In PTA all the transitions are controllable.

The optimal-cost computation problem for PTA consists in computing the minimal cost to reach a particular location. The (smart) solutions to the optimal-cost computation problem for PTA were simultaneously proposed in [46, 47]. A symbolic algorithm [48] has been implemented in an extended version of UPPAAL called UPPAAL CORA [94].

If we consider PTA and a dual problem *i.e.,* keep the system in a set of states, we can define an optimal problem as well: here the model has *costs* and *rewards*, and the aim is to minimise the limit of the ration cost/reward on infinite runs. This latter problem was solved in [44, 49].

A step further consists in defining a logic which can refer to cost values: such a logic, WCTL, has been defined in [45], and extends TCTL with cost variables. Inside this logic, it is possible to write a formula which which states "it is possible to reach a state q and this at a cost which is less than K." The model-checking problem for WCTL against timed automata was proved undecidable (even if the time domain is discrete). Nevertheless, for a subset of WCTL, there are some classes of timed automata for which model-checking is decidable. Recently in [58], the authors have proved that WCTL was decidable for o-minimal PTA.

Optimal Cost in Timed Games. The previous optimal-cost problem can be defined for timed games. In this case we start with a model which is a *Priced Timed Game Automaton* (PTGA) *i.e.,* a TGA enriched with costs. We consider here reachability games: the aim is to reach a location at an optimal cost, and this optimal cost is the best the controller can ensure, whatever the adversary (environment) does. The automaton $\mathcal{A}$ of Figure. 2.6 is an example of a PTGA.

Uncontrollable transitions are labelled by u. Notice that in this game, there is a controller which can enforce location Goal: fire c_1 and then whatever the environment chooses in ℓ_1, fire c_2. In the sequel, we assume that the reachability control problem has been solved and the answer was "yes" *i.e.,* there is a controller which can enforce Goal. Our problem is now to compute the optimal cost.

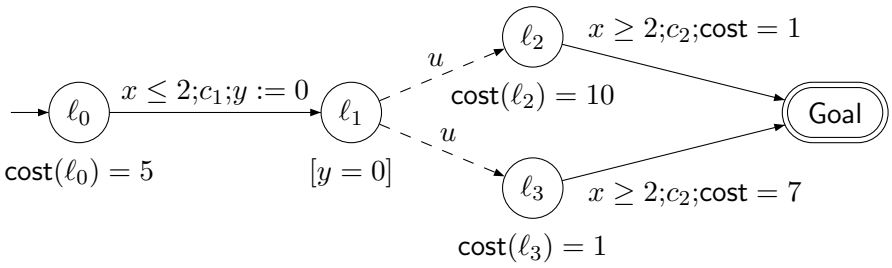

Figure 2.6: The Priced Timed Game Automaton $\mathcal{A}$

This optimal cost is defined as the value the controller can guarantee whatever the environment does. For example, in the PTGA $\mathcal{A}$ of Figure 2.6, there are two families of runs: the runs in which the environment chooses to go from ℓ_1 to ℓ_2 and the runs in which it chooses to go from ℓ_1 to ℓ_3. The only real decision the controller has to make is to choose how long to wait in ℓ_0 before firing c_1 (indeed when in ℓ_2 or ℓ_3 it chooses to fire c_2 as soon as $x \geq 2$.) Let δ be the amount of time the controller chooses to wait for in ℓ_0. The costs that can result from this choice depend on the choice of the environment: if it chooses $\ell_i, i = 2, 3$, the cost is respectively $\alpha_2(\delta) = 5 \cdot \delta + 10 \cdot (2 - \delta) + 1$ and $\alpha_3(\delta) = 5 \cdot \delta + 1 \cdot (2 - \delta) + 7$. Once the controller has fired c_1, the environment will try to maximise the cost of the current run: it will choose $\ell_i, i = 2, 3$ such that $\alpha_i(\delta)$ is maximal. The aim of the controller is to minimise this maximum.

Thus the optimal cost for the PTGA $\mathcal{A}$ is defined by:

$$\mathsf{OptCost} = \inf_{0 \leq \delta \leq 2} \max(5 \cdot \delta + 10 \cdot (2 - \delta) + 1, 5 \cdot \delta + 1 \cdot (2 - \delta) + 7)$$

For $\mathcal{A}$, this value is $\frac{43}{3}$ which means c_1 should be fired at the date $\frac{4}{3}$. The problem of computing automatically this optimal cost is the *Optimal Cost Reachability Control Problem* (OC-RCP).

Contribution to the Domain. Of course we can give a general definition of the optimal cost and generalise the above formula given for $\mathcal{A}$. With the model of PTGA we have introduced, the *time optimal control* problem is a particular version: in each location, the cost rate is 1 and each discrete transition has a null cost. A first step towards a solution of the optimal cost reachability control problem was given for *acyclic* timed games in [56]. As the PTGA does not have any cycle, the computation of the optimal cost can be reduced to a linear optimisation problem.

If we consider PTGA which can contain cycles, it is not obvious that the algorithm proposed in [56] can be tuned to accommodate cycles. Actually, a solution to this more general version of the computation of the optimal cost was simultaneously given by Alur, Bernadsky et Madhusudan [54] and in our papers

[50] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. **Optimal Strategies in Priced Timed Game Automata**. In Kamal Lodaya and Meena Mahajan, editors, *Proceedings of the 24th Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *Lecture Notes in Computer Science*, pages 148–160, Chennai, India, December 2004. Springer.

and

[51] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. **Synthesis of Optimal Strategies Using HyTech**. In Luca de Alfaro, editor, *Proceedings of the Workshop on Games in Design and Verification (GDV'04)*, volume 119 of *Electronic Notes in Theoretical Computer Science*, pages 11–31, Boston, Massachussetts, USA, February 2005. Elsevier Science Publishers.

The paper [54] was published at the ICALP'04 conference, and the results we have obtained use a different technique and were available at the same time in the BRICS research report [59] in February 2004. Compared to [54], we give decidability results for PTGA as well as structural properties of the class of strategies (or controllers) that can win this type of games. Moreover, we have implemented our result with HyTech [87] and this is reported in [51].

The idea we propose to solve the optimal cost problem[9] for PTGA is the following: the OC-RCP can be rephrased as: "What is the minimum amount of resources (time, petrol, etc) I should start with, to be able to enforce Goal, and when Goal is reached I have not run out of resources?" Thus to solve the OC-RCP we can proceed as follows:

1. we start with a PTGA $\mathcal{A}$ similar to the one given on Figure 2.6;
2. then we build a game $H_{\mathcal{A}}$, which is a timed game with a special variable *rsrc*: we end up with a timed game which is a special kind of linear hybrid automata (LHA). This variable stores the amount of resources which is left at each point in the game. Its value will then decrease in a location ℓ of $H_{\mathcal{A}}$ with a rate which is the opposite of the rate of the cost in the corresponding location in $\mathcal{A}$.

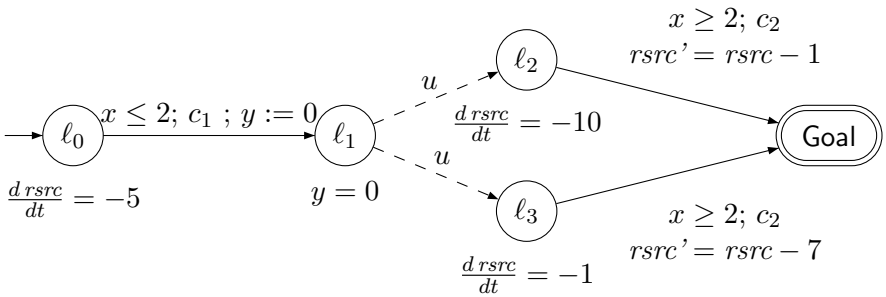

Figure 2.7: The LHA $H_{\mathcal{A}}$ built from $\mathcal{A}$.

Applied to $\mathcal{A}$ of Figure 2.6, this construction gives the automaton $H_{\mathcal{A}}$ on Figure 2.7. The OC-RCP for PTGA can then be reduced to a (standard) Reachability Control Problem for a linear hybrid game: the control objective is to enforce Goal and a value of *rsrc* which is larger or equal than zero *i.e.,* the control objective is $(\mathsf{Goal}, rsrc \geq 0)$. We know how to effectively compute the set of winning states for this type of (linear hybrid) games, and if the algorithm terminates we obtain a set of pairs of the form (ℓ, Z) where ℓ is a location and Z a polyhedron. In particular we may obtain a set of pairs (ℓ_0, Z) giving the set of winning states (values of the

[9] As emphasised earlier we assume there is a strategy to win the game *i.e.,* to force Goal. This ensures that the optimal cost exists and is an finite number.

variables) in the initial location. By definition of the algorithm to solve linear hybrid games, the set of such zones Z gives the maximal winning set of values for the variables in location ℓ_0.

The projection of the winning set in ℓ_0 on the variable *rsrc* gives the set of values of *rsrc* from which we can win the game *i.e.,* enforce $(\mathsf{Goal}, rsrc \geq 0)$. In our case of PTGA, we can prove that this set is of the form $rsrc \bowtie k$ with $\bowtie \{>, \geq\}$ and this means that the optimal cost is k. Decidability of the OC-RCP for PTGA is also reduced to the decidability of the RCP for the class of LHA that is obtained when we use our translation. We have obtained the following results:

- for bounded PTGA (all the clocks are bounded) such that on each cycle, the cost is at least $\beta > 0$ (we call this class of PTGA, *cost non-Zeno PTGA*), the classical backward algorithm that computes the winning states of the associated linear hybrid game terminates.

- the optimal cost can be obtained by computing the projection of the set of initial winning states on the variable *rsrc*. When the previous algorithm terminates, this set is always of the form $rsrc \bowtie k$ with $\bowtie \{>, \geq\}$ and k is the optimal cost. Compared to [54], we can decide whether there is a strategy that guarantees the optimal cost, *i.e.,* an optimal strategy: if the initial winning interval is of the form $rsrc \geq k$ the answer is "yes", otherwise it is of the form $rsrc > k$ and there is a family of strategies that can ensure a cost $k + \epsilon$ for any $\epsilon > 0$. Still there is no optimal strategy.

- optimal strategies when they exist, may need cost information *i.e.,* the accumulated amount of resources consumed since the game started. In this sense, the clocks of the PTGA are not always sufficient to decide what to do to ensure winning at optimal cost, and we may need the cost information.

- we have identified a syntactical class C of PTGA for which there is always an optimal strategy, and the strategy depends on the state of the PTGA (no cost information is needed).

- finally, we have implemented our algorithm [51] for computing the optimal cost and an optimal strategy for the subclass C.

Since these results were published, some new results about optimal cost for PTGA were obtained:

- in [53], the authors prove that the non-Zeno assumption we had made to prove termination of our algorithm is necessary. In other words, without this assumption, the optimal cost computation problem is undecidable. This proof was refined in [42] and the most recent result is that for PTGA with 3 clocks the optimal cost computation problem is undecidable.

- in [52], the authors prove that for PTGA with one clock, the optimal cost can be computed (3EXPTIME).

- finally, in [58], there is a proof that the optimal cost is computable for o-minimal automata.

2.4.4 Efficient Algorithms for Controller Synthesis

Tools for Analysing Timed Systems. In the last ten years, a lot of progress has been made in the design of efficient tools for the analysis (model-checking) of timed systems. Tools like Kronos [88] or Uppaal [93, 89] have become very efficient and widely used to check properties of timed automata but still no real efficient counterpart had been designed for timed games.

One of the reason may be that on-the-fly algorithms have been absolutely crucial to the success of these model-checking tools: on-the-fly algorithms were already crucial for model-checkers for discrete systems like SPIN [92]. Both reachability, safety as well as general liveness properties of such timed models may be decided using on-the-fly algorithms *i.e.,* by exploring the reachable state-space in a symbolic forward manner with the possibility of early termination. Timed automata technology has also been successfully applied to optimal scheduling problems with on-the-fly algorithms which quickly lead to near-optimal (time- or cost-wise) schedules [57, 46, 56, 55, 32].

The algorithms for computing controllers for timed games described in section 2.3.3 are based on *backwards* fix-point computations of the set of winning states [20, 25, 27]. For timed game automata, they were implemented in the tool Kronos [31] at the end of the 90's but lack efficiency because they require the computation of the complete set of winning states. Moreover the backward computation may sometimes not terminate or be very expensive for some extended classes of timed game automata.

Indeed, a backward algorithm does not take into account the fact that a state is reachable from the initial state or not. Backward computation may thus yield to an enormous set of winning states on which iterative computations are carried out. A solution is to first compute the set of reachable states and at each step of the iterative algorithm for computing winning states, take the intersection on the winning states with the reachable states. Even if these sets are represented with efficient data structures this can be very expensive and imply the generation of the whole state space of the system.

A more difficult problem is that for simple extensions of timed automata, it is very difficult and sometimes impossible to compute the controllable predecessor operator π. For instance, if transitions are of the form $i := 2j + k$ (which is allowed in UPPAAL), computing the controllable predecessors of the symbolic state $(\ell, j \geq 0 \wedge k \geq 0)$ means collecting all the values of i s.t. $i = 2j + k$. This computation can be very expensive. In practise, if we start from a given initial state and given j and k, there might be a few reachable (admissible) values for j and k and consequently a few for i. Or a least, for each new value $v(i)$ of i which is reached, we know the values of j and k that were used to obtain $v(i)$.

On-the-fly Algorithms . To obtain an *on-the-fly* algorithm for computing winning states, we should not have to compute the set of reachable states before we compute the winning set of states.

Regarding timed games, in [69, 70], Karine Altisen and Stavros Tripakis have proposed a partially on-the-fly method for solving timed games. However, this method involves an extremely expensive preprocessing step in which the quotient graph of the timed game w.r.t. time-abstracted bisimulation[10] needs to be built. Once obtained this quotient graph may be used with any on-the-fly game-solving algorithms for untimed systems.

[10] A time-abstracted bisimulation is a binary relation on states preserving discrete states and abstracted delay-transitions.

To design a truly on-the-fly algorithm for the computation of winning states for (reachability, timed) game automata we start from the on-the-fly algorithm suggested by Liu & Smolka in [36] for linear-time model-checking of finite-state systems.

Contribution to the Domain. In a recent paper

[68] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. **Efficient on-the-fly algorithms for the analysis of timed games.** In Martín Abadi and Luca de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR'05)*, volume 3653 of *Lecture Notes in Computer Science*, pages 66–80, San Francisco, CA, USA, August 2005. Springer. Copyright Springer-Verlag.

we have proposed an efficient, truly on-the-fly algorithm for the computation of winning states for (reachability) timed game automata.

For finite-state systems, on-the-fly model-checking algorithms has been an active and successful research area since the end of the 80's, with the algorithm proposed by Liu & Smolka [36] being particularly elegant (and optimal). In this paper, we have focused on reachability untimed an timed games.

The on-the-fly algorithm, OTFUR, we have proposed in [68] is given in Fig. 2.8. The idea for this algorithm is the following: we assume that some variables store two sets of transitions: *ToExplore* store the transitions that have explored and *ToBackPropagate* store the transitions the target states of which has been declared winning. Another variable, $Passed$, stores the set of states that have already been encountered. Each encountered state $q \in Passed$ has a status, $Win[q]$ which is either *winning* (1) or *unknown* (0). We also use a variable $Depend[q]$ that stores for each q, the set of explored transitions t s.t. q is a target of t. The initial values of the variables are set by lines 2 to 6.

To perform a step of the on-the-fly algorithm OTFUR, we pick transition (q, α, q') either in *ToExplore* or *ToBackPropagate* (line 10) and process it as follows:

- if the target state q' is encountered for the first time ($q' \notin Passed$), we update $Passed$, $Depend$ and $Win[q']$ (lines 14–16). We also initialise some counters (lines 12 and 13) $c(q')$ and $u(q')$ which have the following meaning: at each time, $c(q')$ represents the number of controllable transitions that can be taken to reach a winning state from q' and $u(s)$ represents the number of uncontrollable hazardous transitions from q' *i.e.,* those for which we do not know yet if they lead to a winning state. When q' is first encountered $u(q')$ is simply the number of outgoing uncontrollable transitions from q'. Finally (lines 17 to 20), depending on the status of q' we add the outgoing transitions to *ToExplore* or just schedule the current transition for back propagation if q' is winning.

- in case $q' \in Passed$, it means that either its status has been changed recently (and we just popped a transition from *ToBackPropagate*) or that a new transition leading to q' has been chosen (from *ToExplore*). We thus check whether the status of q' is winning and if yes, we update some information on q: lines 24 and 25 updates the counters $c(q)$ or $u(q)$ depending on the type of the transition being processed (controllable or not). The state q can be declared winning (line 27) if at least one controllable transition leads to a winning state ($c(q) \geq 1$) and all outgoing uncontrollable transitions lead to a winning state as well ($u(q) = 0$). In this case the transitions leading to q ($Depend[q]$) are

scheduled for back propagation (line 29). Otherwise we have just picked a new transition leading to q' and we only update $Depend[q']$ (line 31).

The formal proof of correctness of this algorithm is summarised by the following theorem:

Theorem 3 ([68]) *Upon termination of* OTFUR *on a given untimed game G the following holds:*

1. *If $q \in Passed$ and $Win[q] = 1$ then $q \in \mathcal{W}(G)$;*
2. *If $(ToExplore \cup ToBackPropagate) = \emptyset$ and $Win[q] = 0$ then $q \notin \mathcal{W}(G)$.*

In addition to being on-the-fly and correct, this algorithm terminates and is optimal in that it has linear time complexity in the size of the underlying untimed game: it is easy to see that each edge $e = (q, \alpha, q')$ will be added to *ToExplore* at most once and to *ToBackPropagate* at most once as well, the first time q is encountered (and added to $Passed$) and the second time when $Win[q']$ changes winning status from 0 to 1. Notice that to obtain an algorithm running in linear time in the size of G (*i.e.,* $|Q| + |E|$) it is important that the reevaluation of the winning status of a state q is performed using the two variables $c(q)$ and $u(q)$.

We can extend algorithm OTFUR to the timed case using a zone-based forward and on-the-fly algorithm for solving timed reachability games. The algorithm, SOTFTR, is given in Fig. 2.9 and may be viewed as an interleaved combination of *forward computation* of the *simulation graph* of the timed game automaton together with *back-propagation* of information *of winning states*. As in the untimed case the algorithm is based on two sets, *ToExplore* and *ToBackPropagate*, of symbolic edges in the simulation-graph, and a passed-list, $Passed$, containing all the symbolic states of the simulation-graph encountered so far by the algorithm.

The crucial point of our symbolic extension is that the winning status of an individual symbolic state is no more 0 or 1 but is now the *subset* $Win[S] \subseteq S$ (union of zones) of the symbolic state S which is currently known to be winning. The set $Depend[S]$ indicates the set of edges (or predecessors of S) which must be reevaluated (*i.e.,* added to *ToBackPropagate*) when new information about $Win[S]$ is obtained, *i.e.,* when $Win[S] \subsetneq Win^*$. Whenever a symbolic edge $e = (S, \alpha, S')$ is considered with $S' \in Passed$, the edge e is added to the dependency set of S' so that that possible future information about additional winning states within S' may also be back-propagated to S.

The correctness of the symbolic on-the-fly algorithm SOTFTR is given by the theorem:

Theorem 4 ([68]) *Upon termination of the algorithm* SOTFTR *on a given timed game automaton G the following holds:*

1. *If $(\ell, v) \in Win[S]$ for some $S \in Passed$ then $(\ell, v) \in \mathcal{W}(G)$;*
2. *If $ToExplore \cup ToBackPropagate = \emptyset$, $(\ell, v) \in S \setminus Win[S]$ for some $S \in Passed$ then $(\ell, v) \notin \mathcal{W}(G)$.*

Termination of the algorithm SOTFTR is guaranteed by the finiteness of the number of symbolic states of $SG(A)$. Moreover, each edge (S, α, T) will be present in the *ToExplore* and *ToBackPropagate* at most $1 + |T|$ times, where $|T|$ is the number of regions of T: (S, α, T) will be in *ToExplore* the first time that S is encountered and subsequently in *ToBackPropagate* each

```
1:  Initialisation
2:    Passed ← {q0};
3:    ToExplore ← {(q0, α, q') | α ∈ Σ, q --α--> q'};
4:    ToBackPropagate ← ∅;
5:    Win[q0] ← (q0 = Goal ? 1 : 0); // set status to 1 if q0 is Goal
6:    Depend[q0] ← ∅;
7:  Main
8:    while ((ToExplore ∪ ToBackPropagate ≠ ∅)) ∧ Win[q0] ≠ 1)) do
9:        // pick a transition from ToExplore or ToBackPropagate
10:       e = (q, α, q') ← pop(ToExplore) or pop(ToBackPropagate);
11:       if q' ∉ Passed then
12:           c(q') = 0;
13:           u(q') = |{(q' --a--> q'', a ∈ Σu}|;
14:           Passed ← Passed ∪ {q'};
15:           Depend[q'] ← {(q, α, q')};
16:           Win[q'] ← (q' = Goal ? 1 : 0);
17:           if Win[q'] = 0 then
18:               ToExplore ← ToExplore ∪ {(q', α, q'') | q' --α--> q''};
19:           else
20:               ToBackPropagate ← ToBackPropagate ∪ {e};
21:       else
22:           if Win[q'] = 1 then
23:               // update the counters of the state q
24:               if α ∈ Σc then c(q) ← c(q) + 1;
25:               else u(q) ← u(q) − 1;
26:               // re-evaluate the status of the state q
27:               Win[q] ← (c(q) ≥ 1) ∧ (u(q) = 0);
28:               if Win[q] then
29:                   ToBackPropagate ← ToBackPropagate ∪ Depend[q];
30:           else // Win[q'] = 0
31:               Depend[q'] ← Depend[q'] ∪ {e};
32:           endif
33:       endif
34:   endwhile
```

Figure 2.8: OTFUR: **O**n-**T**he-**F**ly Algorithm for **U**ntimed **R**eachability Games

1: Initialisation
2: $Passed \leftarrow \{S_0\}$ where $S_0 = \{(\ell_0, \overline{0})\}^{\nearrow}$;
3: $ToExplore \leftarrow \{(S_0, \alpha, S') \mid S' = \mathsf{Post}_\alpha(S_0)^{\nearrow}\}$;
4: $ToBackPropagate \leftarrow \emptyset$;
5: $Win[S_0] \leftarrow S_0 \cap (\{\mathsf{Goal}\} \times \mathbb{R}^X_{\geq 0})$;
6: $Depend[S_0] \leftarrow \emptyset$;
7: **Main**
8: **while** $((ToExplore \cup ToBackPropagate \neq \emptyset) \wedge (\ell_0, \overline{0}) \notin\neq Win[S_0]))$ **do**
9: *// pick a transition from ToExplore or ToBackPropagate*
10: $e = (S, \alpha, S') \leftarrow pop(ToExplore)$ or $pop(ToBackPropagate)$;
11: **if** $S' \notin Passed$ **then**
12: $Passed \leftarrow Passed \cup \{S'\}$;
13: $Depend[S'] \leftarrow \{(S, \alpha, S')\}$;
14: $Win[S'] \leftarrow S' \cap (\{\mathsf{Goal}\} \times \mathbb{R}^X_{\geq 0})$;
15: **if** $Win[S'] \subsetneq S'$ **then**
16: $ToExplore \leftarrow ToExplore \cup \{(S', \alpha, S'') \mid S'' = \mathsf{Post}_\alpha(S')^{\nearrow}\}$;
17: **if** $Win[S'] \neq \emptyset$ **then**
18: $ToBackPropagate \leftarrow ToBackPropagate \cup \{e\}$;
19: **else**
20: *// If $T \notin Passed$, we assume $Win[T] = \emptyset$*
21: $Good \leftarrow Win[S] \cup \bigcup_{S \xrightarrow{c} T} \mathsf{Pred}_c(Win[T])$;
22: $Bad \leftarrow \bigcup_{S \xrightarrow{u} T} \mathsf{Pred}_u(T \setminus Win[T])) \cap S$;
23: $Win^* \leftarrow \mathsf{Pred}_t(Good, Bad)$;
24: **if** $(Win[S] \subsetneq Win^*)$ **then**
25: $Waiting \leftarrow Waiting \cup Depend[S]$;
26: $Win[S] \leftarrow Win^*$;
27: $Depend[S'] \leftarrow Depend[S'] \cup \{e\}$;
28: **endif**
29: **endwhile**

Figure 2.9: SOTFTR: **S**ymbolic **O**n-**T**he-**F**ly Algo. for **T**imed **R**eachability Games

time the set $Win[T]$ increases. Now, any given region may be contained in several symbolic states of the simulation graph (due to overlap). Thus the SOTFTR algorithm is *not* linear in the region-graph and hence not theoretically optimal, as an algorithm with linear worst-case time-complexity could be obtained by applying the untimed algorithm directly to the region-graph. However, this is only a theoretical result and it turns out that the implementation of the algorithm in UPPAAL-TiGA is very efficient.

We can optimise (space-wise) the previous algorithm. When we explore the automaton forward, we check if any newly generated symbolic state S' belongs $Passed$. As an optimisation we may instead use the classical inclusion check: $\exists S'' \in Passed$ s.t. $S' \subseteq S''$, in which case, S' is discarded and we update $Depend[S'']$ instead. Indeed, new information learnt for S'' can be new information on S' but not necessarily. This introduces an overhead (time-wise) in the sense that we may back-propagate irrelevant information. On the other hand, back-propagating only the relevant information would be unnecessarily complex and would void most of the memory gain introduced by the use of inclusion. In practise, the reduction of the number of forward steps obtained by the inclusion check pays off for large systems and is a little overhead otherwise, as shown in our experiments. It is also possible to propagate information about losing states: in the case of reachability games, if a state is a deadlock state and is not winning, for sure it is losing. This can also speed-up the algorithm. For the example of Figure 2.1, we can propagate the information that $(\ell_5, x > 1)$ is losing which entails $(\ell_1, x > 1)$ is losing as well. Then it only remains to obtain the status of $(\ell_1, x \leq 1)$ to determine if the game is winning or not.

This algorithm has been implemented in the game version of UPPAAL which is called UPPAAL-Tiga [95].

2.4.5 Partial Observation: Control and Diagnosis

In the previous sections, we have assumed that the controller has full knowledge of what happens in the system: it can observe the global state of the system at any point in time and record the events that have occurred. In many practical applications, this is not realistic and the controller can only obtain partial information about the state of the system and some of the events that occurred might be *unobservable*. Nevertheless, we still want to control the system: this is the *Control Under Partial Observation Problem*, (PO-CP).

Partial observation usually means that the uncontrollable events, Σ_u, are partitioned into two classes: Σ_u^o which are observable and $\Sigma_u \setminus \Sigma_u^o$ which are unobservable. Thus the controller can only observe sequences $(a_0, t_0)(a_1, t_1) \cdots (a_n, t_n)$ of timed events in $((\Sigma_c \cup \Sigma_u^o) \times \mathbb{R}_{\geq 0})^*$ and must determine what to do according to this observation.

Thus we introduce the notion of *trace-based* strategies: a strategy is *trace-based* if, for any two runs ρ and ρ' with the same timed trace[11], we have $f(\rho) = f(\rho')$.

The PO-CP then consists in deciding whether there is a *trace-based* strategy f to win a game. As a controller is a strategy we can as well define *Trace-Based Controllers* (TBC) and formalise the control problem under partial observation as:

$$\text{Given } S \text{ and } \phi, \text{ is there a TBC } C \text{ s.t. } (S \parallel C) \models \phi \ ? \qquad \text{(PO-CP)}$$

Results about Control Under Partial Observation. The paper [11] (in French) gives an overview of the results about the control of partially observable (timed) systems. If the

[11] The timed trace of a run is a timed word in $((\Sigma_c \cup \Sigma_u^o) \times \mathbb{R}_{\geq 0})^*$.

specification is given by a deterministic TA, the PO-CP is undecidable [60]. If we consider the *safety* version (PO-SCP), and we want to synthesise a non-Zeno controller, the PO-SCP is still undecidable [8].

If the resources of controllers are bounded, we can extend the proof of section 2.4.2 for the synthesis of controllers for external specification to the case of partial observation. This yields the following result: if the resources of the controller are bounded, and the property ϕ is specified by a deterministic timed automaton, the PO-CP is EXPTIME-complete [60].

Recent results [62, 61] about partial observability for finite automata consider that the state of the system is partially observable. We have recently extended this work to timed systems [C1].

Algorithms for solving the control problem under partial observation are often used to solve the *diagnosis problem* which is a special type of control under partial observation.

Fault Diagnosis. We can define the *fault diagnosis* problem (FDP) as a special form of the PO-CP. In the FDP, we consider a system S, with observable events in Σ_o and the others in the complement set, $\overline{\Sigma_o}$, are unobservable. A special unobservable event f is the *fault* event. The goal in fault diagnosis, is to design an observer[12] that can detect that a fault has occurred *i.e.,* f has occurred. The diagnosis problem has been extensively studied for discrete event systems [72, 71, 73].

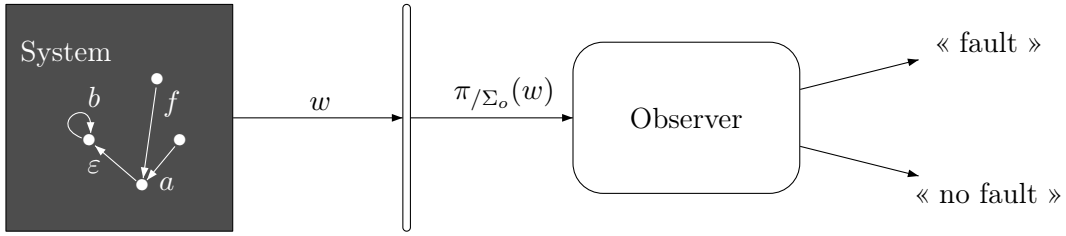

Figure 2.10: Fault-Diagnosis

The diagnosis problem framework is depicted on Figure 2.10. The system generates events, and only those that belong to Σ_o can be observed. If the system has generated a sequence of events w then the observer could only see the projection on Σ_o of this sequence *i.e.,* $\pi_{/\Sigma_o}(w)$. The goal of the observer is to watch observable events and to indicate as soon as possible if a fault occurred. Notice that the observer does not observe a complete input sequence w but it receives the actions one after the other.

The main parameter in the fault diagnosis problem is the amount of time it takes to detect a fault. In discrete event systems, this is the number of steps the system has made since the fault occurred. The runs of the system can be split into two categories:

- those that do not contain the fault event f; they are called *non-faulty* runs;
- and those that contain a fault event, followed by at least k steps: they are called *k-faulty* runs.

A *k-diagnoser* for a system S is an observer which behaves as follows:

- for each non-faulty run, it does not announce a fault;

[12] it is not a controller because it can only observe the system.

- for each k-faulty run it has to announce a fault.

Nothing is required for faulty runs with less than k steps after the fault. The k-fault diagnosis problem can be formally defined by:

$$\text{Given a system } S,\ k \in \mathbb{N}, \text{ is there a } k\text{-diagnoser for } S\ ? \qquad (k\text{-FDP})$$

S is *k-diagnosable* if there is a k-diagnoser for S and S is *diagnosable* if there is some k s.t. S is k-diagnosable. Deciding whether a system S is diagnosable is the *Fault-Diagnosis Problem* (FDP). Synthesising a diagnoser is the associated *Fault-Diagnosis Synthesis Problem* (FDSP). For discrete event systems, it was proved that the FDP is in PTIME and the FDSP is in EXPTIME [72, 71, 73].

These results have been extended to timed automata by Stavros Tripakis in [74]. For timed systems, the number of steps (k) following a fault is replaced by a duration $\Delta \in \mathbb{N}$, and the goal of a Δ-diagnoser is to announce a fault within Δ time units. The diagnoser can sense the (observable) events and the time at which they occurred. In Figure 2.10, it suffices to replace w by a pair (w, t) where w is a set of events generated by the system and t a sequence of dates. The diagnoser "sees" the observable events in w and the date of occurrence of each of them. In the paper mentioned above ([74]), it was proved that the FDP for timed automata is PSPACE-complete. In this paper, the FDSP is solved by building a Turing machine which is a Δ-diagnoser.

A natural follow-up question is then to decide whether there is a simpler diagnoser, which can be represented for instance by a timed automaton. This version of the FDP was stated and solved in [75] and the results are:

- the FDSP is 2EXPTIME-complete if the resources of the diagnoser are bounded, and if the diagnoser is a deterministic timed automaton;

- the FDSP becomes PSPACE-complete if the diagnoser is an *Event-Recording Automaton* (ERA) [6] (with bounded resources). ERA forms a subclass of TA where each clock is associated with an event and reset when this event occurs.

Contribution to the Domain. In the previous versions of the FDP, it is assumed that the diagnoser can rely on analog clocks which are infinitely precise. If we consider a real implementation, an observer will not be able to read clock values but rather it will sense periodically a tick event. Such a discrete clock that generates tick events is called a *digital* clock. To have a more fair or precise modeling of a system, it may be interesting to consider this notion of digital clocks. It has already been considered as the problem of implementing the ideal mathematical model of timed automata in [67] and in [63, 66, 65, 64].

This it makes sense to address the FDP using digital clocks: this version of the FDP is the *Digital-Clock Diagnosis Problem* (DC-DP). The architecture that we consider in this setting is given on Figure 2.11.

The digital clock issues tick events. The diagnoser can observe the observable events of the system plus the tick event. In contrast to the dense-time version, it cannot measure the time elapsed between two events (and thus it cannot obtain the date of an event). The diagnoser must announce of fault observing only the interleavings of events in $\Sigma_o \cup \{\mathsf{tick}\}$.

We have defined this problem in

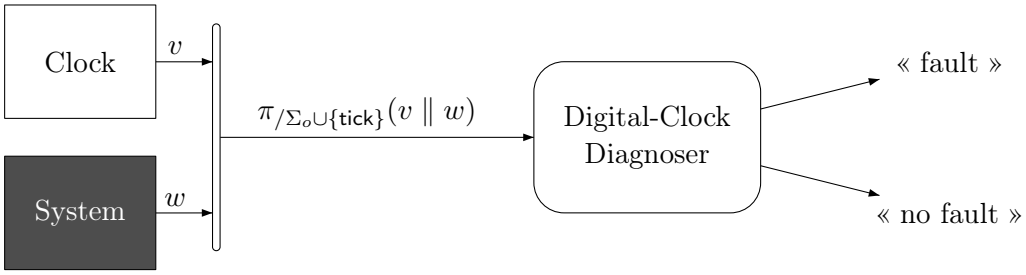

Figure 2.11: Fault Diagnosis with Digital Clocks

[77] Karine Altisen, Franck Cassez, and Stavros Tripakis.
Monitoring and fault-diagnosis with digital clocks.
In *Sixth International Conference on Application of Concurrency to System Design (ACSD'06)*, Turku, Finland, June 2006. IEEE Computer Society Press.

In this paper, we proved that, given a timed automaton S, a digital clock H which is also a timed automaton (and generates tick events), the digital fault diagnosis problem can be decided in EXPTIME. The size of the diagnoser is doubly exponential in the size of S and H (building a diagnoser usually requires a determinization step which adds an exponential to the diagnosis problem). One problem was left open in the previous paper: it is the digital clock *synthesis* problem: is there a digital clock H which is a TA, s.t. the system is diagnosable using this digital clock?

Notice that in [76], the authors also consider a diagnosis problem with digital clocks. Nevertheless, they do not mention the open problem we stated previously and the definition they give for the other problems are not totally formalised.

Chapter 3

Conclusion and Future Work

Conclusion

My main research contributions were obtained in the domain of Control of Timed Systems and developed in Chapter 2. The problems we have studied are of particular interest for the community working in this area. The results presented in Chapter 2 were obtained together with colleagues from French and European groups and were published in renowned conferences. The previous work has also been financially supported by the French Ministry of Research and by the French National Research Agency (ANR).

Another line of work I have contributed to recently is about the relations between Timed Automata (TA) and *Time Petri nets* (TPNs). First, together with colleagues in Nantes and Paris, we have first proved that TPNs could be translated into networks of TA [81]. Following this work, we have shown how to model-check the temporal real-time logic TCTL on TPNs [80]. This has been implemented in the tool Roméo [90, 85] developed by Olivier H. Roux in Nantes. Second, we have investigated how to translate TA into TPNs and proved that: (i) given a TA H, there is a TPN T_H which accepts[1] the same timed language [82] and (ii) there exists a TA H' such that no TPN (even unbounded) can be timed bisimilar to H' [82]; we have given a semantic condition for checking that a TA has an equivalent timed bisimilar TPN in [83].

Future Work

My future work will consist in applying the recent developments, theories and tools, in control theory for timed systems, to real commercialised systems. For instance the new tool UPPAAL-TiGA [95] enables one to solve timed games efficiently. This is a good starting point to address the problem of implementation of controllers for timed systems, and later the synthesis of optimal controllers. This domain of application is described more precisely in Section "Application of Control Theories and Tools to Real Systems" in the sequel. Of course I will also pursue the work I started on more theoretical aspects which is described hereafter.

[1] To do this, we extended the original TPN model with open intervals to specify the timing constraints of transitions.

Control of Distributed Open Timed Systems

The project DOTS (Distributed Open Timed Systems[2] began in January 2007, and during the course of this project we are going to work on many aspects related to the control problem. In this project I will be particularly interested in the following parts:

- Extending the results we have obtained for *optimal reachability control* of timed systems. This amounts to study the optimal control problem for *safety games* and hopefully general Büchi games;
- Extending the *on-the-fly* algorithm of section 2.4.4 to other type of games like *safety games* and Büchi games. Considering Büchi games is a rather important aspect when one wants to solve safety dense-time games: indeed, when one considers safety games, it could be that a controller exists, which prevents time from diverging, and this is the well-known Zenoness problem. Time divergence can be enforced by a Büchi objective and thus solving a safety+Büchi game will ensure the synthesis of non-Zeno controllers. This work is expected to be implemented in the next versions of the tool UPPAAL-TiGA [95].
- Designing logics for specifying timed control objectives. In a recent work [41], François Laroussinie *et al.* (LSV, ENS Cachan) showed how to solve the control problem in a discrete time setting, for a timed version of the logic ATL. We will study the extension of this work to dense time. This can also be viewed as a generalisation of some results we obtained earlier (section 2.4.2);
- Defining a concurrent semantics for timed systems. Recently with Thomas Chatain and Claude Jard (IRISA, Rennes, F.), we have proposed a method for computing unfoldings of networks of timed automata (see [C6] in section 1.8, page 10 of chapter 1). We will continue this work during the project DOTS and try to define a meaningful *concurrent* semantics for timed systems.
- Controller synthesis under partial observation. We have recently worked on: 1) controller synthesis under partial observation where the observations are given by some predicates on the timed systems we want to control as in [62, 61]; and 2) the extension of the *on-the-fly* algorithm of section 2.4.4 (which was designed for reachability games) to this setting. This work was carried out together with Didier Lime (IRCCyN, Nantes), Alexandre David and Kim G. Larsen (Aalborg, DK), and Jean-François Raskin (Bruxelles, B.) and will be presented at the next ATVA conference [C1]. There are many extensions of this work to be investigated and among them the data structures used to encode the winning set of states (Cf. [62, 61]).

Application of Control Theory to Other Domains

Fault diagnosis can be considered as a simplified version of the control problem and another line of work I am going to pursue concerns fault diagnosis of discrete and timed systems.

In dense time, the problems we have to solve are the ones which were left open in [77]. In this work we considered a plant and a digital clock (specified by a timed automaton). The digital clock emits "ticks" and the diagnoser can only sense the discrete events of the plant and the (discrete) "ticks". Given a plant and a clock, we have showed how to check that a discrete

[2]DOTS: `http://www.lsv.ens-cachan.fr/anr-dots/`.

(we say "digital") diagnoser exists. A more abstract version of this problem is to check whether there is a digital clock (given by a timed automaton) s.t. the plant is diagnosable. Actually we will consider two versions of the problem: one where the digital clock has a given (fixed) amount of resources (number of clocks, constants used in the specification of the digital clock) and the other one where we do not give a bound on the resources.

In discrete time, we have investigated some optimal fault diagnosis problems in [78, 79]. The purpose is to compute a least expensive diagnoser, given some measure of how much it costs to observe sequences of events. We will generalise the results of this work, obtained for fault diagnosis, to control under partial observation (for discrete event systems).

Finally, we have started to work on the synthesis of *non-interferent* systems. Non-interference is a security property of (discrete) systems that ensures no information can flow from a low level user to a high level user. We have already studied the problem of synthesis of non-interferent discrete systems in [C2] (section 1.8, page 10 of chapter 1). This work was carried out together with Olivier H. Roux (IRCCyN, Nantes, F.) and John Mullins (Ecole Polytechnique de Montréal, CA.). This subject is a work package of the project DOTS. Following [C2], we will study the decidability status of the various non-interferent problems for timed systems specified by timed automata.

Application of Control Theories and Tools to Real Systems

I have an ongoing collaboration with colleagues at NICTA, Sydney, Australia, and we try to model real systems (parts of embedded systems of sensor networks) and verify them or synthesise controllers for them. NICTA is a research laboratory working in close collaboration with industry. They have developed for instance a commercialised version of the micro-kernel L4, and have used formal methods to check correctness of the system. They have lots of specifications of real systems and need to use advanced theories and tools (like computing optimal controllers which can be used to design low power consumption devices).

One of the main issue in embedded systems at the moment is to come up with a methodology for designing such systems and to go from theory to practise, *i.e.,* confront the theory and tools recently developed for timed systems with real life systems.

I will try to contribute to this challenging issue in the next few years by modeling real systems, and try to obtain results that will help engineers in the design of advanced and complex embedded systems. In particular, I will address the problem of optimal scheduling both time-wise and energy-wise. This work will hopefully be carried out abroad during a long term visit at NICTA, Sydney.

Chapter 4

References

This chapter contains the references used in chapters 2 and 3. The refercences have been sorted using Nicolas Markey's `splitbib` package available from
`http://www.ctan.org/tex-archive/macros/latex/contrib/splitbib/`.

— Timed Automata and Their Extensions —

Journal Papers

[1] Rajeev Alur and David Dill. A theory of timed automata. *Theoretical Computer Science (TCS)*, 126(2):183–235, 1994.

[2] Gerardo Lafferriere, George Pappas, and Shankar Sastry. Ominimal hybrid systems. *Mathematics of Control Signals and Systems*, 13(1):1–21, 2000.

[3] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

[4] Patricia Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, May 2004.

Conference Papers

[5] Thomas A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292, 1996.

[6] Rajeev Alur, Limor Fix, and Thomas A. Henzinger. A determinizable class of timed automata. In *CAV: Computer-Aided Verification*, Lecture Notes in Computer Science 818, pages 1–13. Springer, 1994.

— Overview Articles and Books —

Book

[7] Philippe Schnoebelen, Béatrice Bérard, Michel Bidoit, François Laroussinie, and Antoine Petit. *Vérification de logiciels : Techniques et outils de model-checking*. Vuibert, 1999.

Journal Papers

[8] Patricia Bouyer and Fabrice Chevalier. On the control of timed and hybrid systems. *EATCS Bulletin*, 89:79–96, June 2006.

Conference Papers

[9] Karine Altisen, Patricia Bouyer, Thierry Cachat, Franck Cassez, and Guillaume Gardey. Introduction au contrôle des systèmes temps-réel. In *Actes du 5ème Colloque sur la Modélisation des Systèmes Réactifs (MSR'05)*, pages 367–380. Hermès Science, 2005. Invited Paper.

[10] Karine Altisen, Nicolas Markey, Pierre-Alain Reynier, and Stavros Tripakis. Implémentabilité des automates temporisés. In Hassane Alla and Éric Rutten, editors, *Actes du 5ème Colloque sur la Modélisation des Systèmes Réactifs (MSR'05)*, pages 395–406, Autrans, France, October 2005. Hermès. Invited paper.

[11] Patricia Bouyer, Fabrice Chevalier, Moez Krichen, and Stavros Tripakis. Observation partielle des systèmes temporisés. In Hassane Alla and Éric Rutten, editors, *Actes du 5ème Colloque sur la Modélisation des Systèmes Réactifs (MSR'05)*, pages 381–393, Autrans, France, October 2005. Hermès. Invited paper.

— Articles on Control of Discrete Event Systems —

Journal Papers

[12] P.J.G. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):1202–1218, 1987.

[13] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proc. of the IEEE*, 77(1):81–98, 1989.

[14] Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

[15] André Arnold, Aymeric Vincent, and Igor Walukiewicz. Games for synthesis of controllers with partial observation. *Theor. Comput. Sci.*, 1(303):7–34, 2003.

Conference Papers

[16] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symposium on Principles of Programming Languages (POPL'89)*, pages 179–190. ACM, 1989.

[17] Stéphane Riedweg and Sophie Pinchinat. Quantified mu-calculus for control synthesis. In *Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03)*, volume 2747 of *Lecture Notes in Computer Science*, pages 642–651. Springer, 2003.

[18] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900, pages 1–13. Springer, 1995. Invited talk.

— Articles on Control of Timed Systems —

Journal Papers

[19] Thomas A. Henzinger and Peter W. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221:369–392, 1999.

Conference Papers

[20] Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems. In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900 of *Lecture Notes in Computer Science*, pages 229–242. Springer, 1995.

[21] Karine Altisen and Stavros Tripakis. Tools for controller synthesis of timed systems. In *Proc. 2nd Workshop on Real-Time Tools (RT-TOOLS'02)*, 2002. Proc. published as Technical Report 2002-025, Uppsala University, Sweden.

[22] Luca de Alfaro, Thomas A. Henzinger, and M. Stoelinga. Timed interfaces. In *Proc. 2nd International Workshop on Embedded Software (EMSOFT'02)*, volume 2491 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2002.

[23] Marco Faëlla, Salvatore La Torre, and Aniello Murano. Dense real-time games. In *Proc. 17th IEEE Symposium on Logic in Computer Science (LICS'02)*, pages 167–176. IEEE Computer Society Press, 2002.

[24] G. Hoffmann and Howard Wong-Toi. The input-output control of real-time discrete-event systems. In *Proceedings of the 13th Annual Real-time Systems Symposium*, pages 256–265. IEEE Computer Society Press, 1992.

[25] Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469–474. Elsevier Science, 1998.

[26] Thomas A. Henzinger, Benjamin Horowitz, and Rupak Majumdar. Rectangular hybrid games. In *Proc. 10th International Conference on Concurrency Theory (CONCUR'99)*, volume 1664 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 1999.

[27] Luca de Alfaro, Thomas A. Henzinger, and Rupak Majumdar. Symbolic algorithms for infinite-state games. In *Proc. 12th International Conference on Concurrency Theory (CONCUR'01)*, volume 2154 of *Lecture Notes in Computer Science*, pages 536–550. Springer, 2001.

[28] Franck Cassez, Thomas A. Henzinger, and Jean-François Raskin. A comparison of control problems for timed and hybrid systems. In *Proc. 5th International Workshop on Hybrid Systems: Computation and Control (HSCC'02)*, volume 2289 of *LNCS*, pages 134–148. Springer, 2002.

[29] Luca de de Alfaro, Marco Faëlla, Thomas A. Henzinger, Rapuk Majumdar, and Mariëlla Stoelinga. The element of surprise in timed games. In *Proc. 14th International Conference on Concurrency Theory (CONCUR'2003)*, volume 2761 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2003.

[30] Patricia Bouyer, Thomas Brihaye, and Fabrice Chevalier. Control in o-minimal hybrid systems. In *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science (LICS'06)*, pages 367–378, Seattle, Washington, USA, August 2006. IEEE Computer Society Press.

[31] Karine Altisen, Gregor Goessler, Amir Pnueli, Joseph Sifakis, Stavros Tripakis, and Sergio Yovine. A framework for scheduler synthesis. In *Proc. International Real-Time Systems Symposium (RTSS'99)*, IEEE Computer Society, pages 154–163, 1999.

[32] Yasmina Abdeddaïm, Eugene Asarin, and Oded Maler. Scheduling with timed automata. *Theor. Comput. Sci.*, 354(2):272–300, 2006.

— Logics for Control and Games —

Journal Papers

[33] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[34] Luca Aceto and François Laroussinie. Is your model checker on time? On the complexity of model checking for timed modal logics. *Journal of Logic and Algebraic Programming*, 52-53:7–51, August 2002.

[35] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

Conference Papers

[36] X. Liu and S. A. Smolka. Simple Linear-Time Algorithm for Minimal Fixed Points. In *Proc. 26th Int. Conf. on Automata, Languages and Programming (ICALP'98)*, volume 1443 of *LNCS*, pages 53–66, Aalborg, Denmark, 1998. Springer.

[37] Marco Faella, Salvatore La Torre, and Aniello Murano. Automata-theoretic decision of timed games. In *3rd International Workshop on Verification, Model Checking, and Abstract Interpretation (VMCAI'02)*, volume 2294 of *Lecture Notes in Computer Science*, pages 240–254, Venezia, Italy, January 2002.

[38] Deepak D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proc. 19th International Symposium on Theoretical Aspects of Computer Science (STACS'02)*, volume 2285 of *Lecture Notes in Computer Science*, pages 571–582. Springer, 2002.

[39] Patricia Bouyer, Franck Cassez, and François Laroussinie. Modal logics for timed control. In *Proc. of the 16th Int. Conf. on Concurrency Theory (CONCUR'05)*, volume 3653 of *LNCS*, pages 81–94. Springer, 2005.

[40] Patricia Bouyer, Laura Bozzelli, and Fabrice Chevalier. Controller synthesis for MTL specifications. In Christel Baier and Holger Hermanns, editors, *Proceedings of the 17th International Conference on Concurrency Theory (CONCUR'06)*, volume 4137 of *Lecture Notes in Computer Science*, pages 450–464, Bonn, Germany, August 2006. Springer.

[41] François Laroussinie, Nicolas Markey, and Ghassan Oreiby. Model checking timed ATL for durational concurrent game structures. In Eugène Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *Lecture Notes in Computer Science*, pages 245–259, Paris, France, September 2006. Springer.

— Priced Timed Automata —

Journal Papers

[42] Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. Improved undecidability results on weighted timed automata. *Information Processing Letters*, 98(5):188–194, June 2006.

[43] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Updatable timed automata. *Theoretical Computer Science*, 321(2-3):291–345, August 2004.

[44] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 2007. To appear.

Conference Papers

[45] Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. Model-checking for weighted timed automata. In *FORMATS/FTRTFT*, pages 277–292, 2004.

[46] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In *Proc. 4th Int. Work. Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2001.

[47] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-cost reachability for priced timed automata. In *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

[48] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Efficient guiding towards cost-optimality in UPPAAL. In *Proc. 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, volume 2031 of *Lecture Notes in Computer Science*, pages 174–188. Springer, 2001.

[49] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Staying alive as cheaply as possible. In Rajeev Alur and George J. Pappas, editors, *Proceedings of the 7th International Conference on Hybrid Systems: Computation and Control (HSCC'04)*, volume 2993 of *Lecture Notes in Computer Science*, pages 203–218, Philadelphia, Pennsylvania, USA, March 2004. Springer.

[50] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Optimal strategies in priced timed game automata. In *Proc. of the 24th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *LNCS*, pages 148–160. Springer, 2004.

[51] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Synthesis of optimal strategies using HyTech. In *Proc. of the Workshop on Games in Design and Verification (GDV'04)*, volume 119 of *Elec. Notes in Theo. Comp. Science*, pages 11–31. Elsevier, 2005.

[52] Patricia Bouyer, Kim G. Larsen, Nicolas Markey, and Jacob Illum Rasmussen. Almost optimal strategies in one-clock priced timed automata. In Naveen Garg and S. Arun-Kumar, editors, *Proceedings of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *Lecture Notes in Computer Science*, pages 345–356, Kolkata, India, December 2006. Springer.

[53] Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On optimal timed strategies. In *FORMATS*, pages 49–64, 2005.

[54] Rajeev Alur, Mikhail Bernadsky, and P. Madhusudan. Optimal reachability in weighted timed games. In *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *Lecture Notes in Computer Science*, pages 122–133. Springer, 2004.

[55] Jacob Rasmussen, Kim G. Larsen, and K. Subramani. Resource-optimal scheduling using priced timed automata. In *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, volume 2988 of *Lecture Notes in Computer Science*, pages 220–235. Springer, 2004.

[56] Salvatore La Torre, Supratik Mukhopadhyay, and Aniello Murano. Optimal-reachability and control for acyclic weighted timed automata. In *Proc. 2nd IFIP International Conference on Theoretical Computer Science (TCS 2002)*, volume 223 of *IFIP Conference Proceedings*, pages 485–497. Kluwer, 2002.

[57] Eugene Asarin and Oded Maler. As soon as possible: Time optimal control for timed automata. In *Proc. 2nd International Workshop on Hybrid Systems: Computation and Control (HSCC'99)*, volume 1569 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 1999.

[58] Patricia Bouyer, Thomas Brihaye, and Fabrice Chevalier. Weighted o-minimal hybrid systems are more decidable than weighted timed automata! In Sergei N. Artemov, editor, *Proceedings of the Symposium on Logical Foundations of Computer Science (LFCS'07)*, Lecture Notes in Computer Science, New-York, NY, USA, June 2007. Springer. To appear.

Research Report

[59] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim Guldstrand Larsen. Optimal Strategies in Priced Timed Game Automata. BRICS Reports Series RS-04-0, BRICS, Denmark, 2004. ISSN 0909-0878.

— Control under Partial Observation —

[60] Patricia Bouyer, Deepak D'Souza, P. Madhusudan, and Antoine Petit. Timed control with partial observability. In Warren A. Hunt, Jr and Fabio Somenzi, editors, *Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03)*, volume 2725 of *Lecture Notes in Computer Science*, pages 180–192, Boulder, Colorado, USA, July 2003. Springer.

[61] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. A lattice theory for solving games of imperfect information. In *HSCC*, pages 153–168, 2006.

[62] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Algorithms for omega-regular games with imperfect information' . In *CSL*, pages 287–302, 2006.

— Implementation of Timed Automata —

Journal Papers

[63] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Almost ASAP semantics: From timed models to timed implementations. *Formal Aspects of Computing*, 17(3):319–341, 2005.

Conference Papers

[64] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Almost ASAP semantics: From timed models to timed implementations. In *Proceedings of HSCC 2004: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 2993, pages 296–310. Springer-Verlag, 2004.

[65] Martin De Wulf, Laurent Doyen, Nicoals Markey, and Jean-François Raskin. Robustness and implementability of timed automata. In *Proceedings of FORMATS-FTRTFT 2004: Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Lecture Notes in Computer Science 3253, pages 118–133. Springer-Verlag, 2004.

[66] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Systematic implementation of real-time models. In *Proceedings of FM 2005: Formal Methods*, Lecture Notes in Computer Science 3582, pages 139–156. Springer-Verlag, 2005.

[67] Karine Altisen and Stavros Tripakis. Implementation of timed automata: an issue of semantics or modeling? In *Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *Lecture Notes in Computer Science*. Springer, 2005.

— *On-the-fly* Algorithms —

[68] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *Proc. of the 16th Int. Conf. on Concurrency Theory (CONCUR'05)*, volume 3653 of *LNCS*, pages 66–80. Springer, 2005.

[69] Karine Altisen and Stavros Tripakis. On-the-fly controller synthesis for discrete and dense-time systems. In *World Congress on Formal Methods (FM'99)*, volume 1708 of *Lecture Notes in Computer Science*, pages 233–252. Springer, 1999.

[70] Karine Altisen and Stavros Tripakis. Tools for controller synthesis of timed systems. In *Proc. 2nd Workshop on Real-Time Tools (RT-TOOLS'02)*, 2002. Proc. published as Technical Report 2002-025, Uppsala University, Sweden.

— Fault Diagnosis —

Journal Papers

[71] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9), September 1995.

[72] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Transactions on Control Systems technology*, 4(2), March 1996.

[73] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial algorithm for testing diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46(8), August 2001.

Conference Papers

[74] Stavros Tripakis. Fault diagnosis for timed automata. In *7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'02)*, volume 2469 of *Lecture Notes in Computer Science*, pages 205–224, Oldenburg, Germany, September 2002.

[75] Patricia Bouyer, Fabrice Chevalier, and Deepak D'Souza. Fault diagnosis using timed automata. In Vladimiro Sassone, editor, *Proceedings of the 8th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'05)*, volume 3441 of *Lecture Notes in Computer Science*, pages 219–233, Edinburgh, U.K., April 2005. Springer.

[76] S. Jiang and Ratnesh Kumar. Diagnosis of dense-time systems using digital clocks. In *American Control Conferece (ACC'06)*, 2006. To appear.

[77] Karine Altisen, Franck Cassez, and Stavros Tripakis. Monitoring and fault-diagnosis with digital clocks. In *6th Int. Conf. on Application of Concurrency to System Design (ACSD'06)*. IEEE Computer Society, 2006.

[78] Franck Cassez, Stavros Tripakis, and Karine Altisen. Synthesis of optimal dynamic observers for fault diagnosis of discrete-event systems. In *1st IEEE & IFIP Int. Symp. on Theoretical Aspects of Soft. Engineering (TASE'07)*, pages 316–325. IEEE Computer Society, 2007.

[79] Franck Cassez, Stavros Tripakis, and Karine Altisen. Sensor minimization problems with static or dynamic observers for fault diagnosis. In *7th Int. Conf. on Application of Concurrency to System Design (ACSD'07)*, pages 90–99. IEEE Computer Society, 2007.

— Time Petri Nets —

Journal Papers

[80] Franck Cassez and Olivier H. Roux. Structural translation from time petri nets to timed automata. *Journal of Software and Systems*, 29:1456–1468, 2006.

Conference Papers

[81] Franck Cassez and Olivier H. Roux. Structural Translation of Time Petri Nets into Timed Automata. In *Proceedings of the Workshop on Automated Verification of Critical Systems (AVoCS'04)*, volume 128 of *Elec. Notes in Theo. Comp. Science*, pages 145–160. Elsevier, 2005.

[82] Béatrice Bérard, Franck Cassez, Serge Haddad, Olivier H. Roux, and Didier Lime. Comparison of the Expressiveness of Timed Automata and Time Petri Nets. In *Proc. of the 3rd Int. Conf. on Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *LNCS*, pages 211–225. Springer, 2005.

[83] Béatrice Bérard, Franck Cassez, Serge Haddad, Olivier H. Roux, and Didier Lime. When are Timed Automata weakly timed bisimilar to Time Petri Nets ? In *Proc. of the 25th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'05)*, volume 3821 of *LNCS*, pages 276–284. Springer, 2005.

Habililtation à diriger les recherches (in French)

[84] Olivier H. Roux. Vérification des réseaux de petri temporels et à chronomètres, December 2005. Habilitation à diriger les recherches, Ecole doctorale STIM, Nantes, France.

— Tools for Analysing Timed Systems —

Journal Papers or Conference Papers

[85] Guillaume Gardey, Didier Lime, Morgan Magnin, and Olivier H. Roux. Romeo: A tool for analyzing time petri nets. In *CAV*, pages 418–423, 2005.

[86] François Laroussinie and Kim G. Larsen. CMC: A tool for compositional model-checking of real-time systems. In *Proc. IFIP Joint International Conference on Formal Description Techniques & Protocol Specification, Testing, and Verification (FORTE-PSTV'98)*, pages 439–456. Kluwer Academic, 1998.

[87] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HyTech: A model-checker for hybrid systems. *Journal on Software Tools for Technology Transfer (STTT)*, 1(1–2):110–122, 1997.

[88] Sergio Yovine. KRONOS: A verification tool for real-time systems. *Journal of Software Tools for Technology Transfer (STTT)*, 1(1–2):123–133, 1997.

[89] Tobias Amnell, Gerd Behrmann, Johan Bengtsson, Pedro R. D'Argenio, Alexandre David, Angskar Fehnker, Thomas Hune, Bertrand Jeannet, Kim G. Larsen, Oliver Möller, Paul Pettersson, Carsten Weise, and Wang Yi. UPPAAL – now, next, and future. In *Proc. Modelling and Verification of Parallel Processes (*MOVEP*2k)*, volume 2067 of *Lecture Notes in Computer Science*, pages 99–124. Springer, 2001.

URLs

[90] Guillaume Gardey, Morgan Magnin, Didier Lime, and Olivier H. Roux. Roméo. `http://romeo.rts-software.org/`.

[91] François Laroussinie. CMC. `http://www.lsv.ens-cachan.fr/~fl/cmcweb.html`.

[92] Gerard Holtzmann. Spin. `http://spinroot.com`.

[93] Tobias Amnell, Gerd Behrmann, Johan Bengtsson, Pedro R. D'Argenio, Alexandre David, Angskar Fehnker, Thomas Hune, Bertrand Jeannet, Kim G. Larsen, Oliver Möller, Paul Pettersson, Carsten Weise, and Wang Yi. Uppaal. `http://www.uppaal.com`.

[94] Gerd Behrmann, Kim Guldstrand Larsen, and Jacob Illum Rasmussen. Uppaal-cora. `http://www.cs.aau.dk/~behrmann/cora/`.

[95] Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Uppaal-tiga. `http://www.cs.aau.dk/~adavid/tiga/`.