Games for the Verification of Timed Systems

by

Vinayak Prabhu

B. Tech. (Indian Institute of Technology, Kanpur)

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering — Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Thomas A. Henzinger, Chair
Professor John Steel
Professor Pravin Varaiya

Fall, 2008

The dissertation of Vinayak Prabhu is approved:

Chair Date

Date

Date

University of California at Berkeley

Fall, 2008

Games for the Verification of Timed Systems

Abstract

Games for the Verification of Timed Systems

by

Vinayak Prabhu
Doctor of Philosophy in Engineering — Electrical Engineering and Computer Sciences

University of California at Berkeley

Professor Thomas A. Henzinger, Chair

Models of timed systems must incorporate not only the sequence of system events, but the timings of these events as well to capture the real-time aspects of physical systems. Timed automata are models of real-time systems in which states consist of discrete locations and values for real-time clocks. The presence of real-time clocks leads to an uncountable state space. This thesis studies verification problems on timed automata in a game theoretic framework.

For untimed systems, two systems are close if every sequence of events of one system is also observable in the second system. For timed systems, the difference in timings of the two corresponding sequences is also of importance. We propose the notion of bisimulation distance which quantifies timing differences; if the bisimulation distance between two systems is ε, then (a) every sequence of events of one system has a corresponding matching sequence in the other, and (b) the timings of matching events in between the two corresponding traces do not differ by more than ε. We show that we can compute the bisimulation distance between two timed automata to within any desired degree of accuracy. We also show that the timed verification logic TCTL is robust with respect to our notion of quantitative bisimilarity, in particular, if a system satisfies a formula, then every close system satisfies a close formula.

Timed games are used for distinguishing between the actions of several agents, typically a controller and an environment. The controller must achieve its objective against all possible choices of the environment. The modeling of the passage of time leads to the presence of zeno executions, and corresponding unrealizable strategies of the controller which may achieve objectives by blocking time. We disallow such unreasonable strategies by restricting all agents to use only receptive strategies — strategies which while not being required to ensure time divergence by any agent, are such that no agent is responsible for blocking time. Time divergence is guaranteed when all players use receptive strategies. We show that timed automaton games with receptive strategies can be solved by a reduction to finite state turn based game graphs. We define the logic timed alternating-time temporal logic for verification of timed automaton games and show that the logic can be model checked in EXPTIME. We also show that the minimum time required by an agent to reach a desired location, and the maximum time an agent can stay safe within a set of locations, against all possible actions of its adversaries are both computable.

We next study the memory requirements of winning strategies for timed automaton games. We prove that finite memory strategies suffice for safety objectives, and that winning strategies for reachability objectives may require infinite memory in general. We introduce randomized strategies in which an agent can propose a probabilistic distribution of moves and show that finite memory randomized strategies suffice for all ω-regular objectives. We also show that while randomization helps in simplifying winning strategies, and thus allows the construction of simpler controllers, it does not help a player in winning at more states, and thus does not allow the construction of more powerful controllers.

Finally we study robust winning strategies in timed games. In a physical system, a controller may propose an action together with a time delay, but the action cannot be assumed to be executed at the exact proposed time delay. We present robust strategies which incorporate such jitters and show that the set of states from which an agent can win robustly is computable.

Professor Thomas A. Henzinger
Dissertation Committee Chair

Contents

List of Figures **v**

1 Introduction **1**

2 Quantifying Similarities between Timed Systems **11**

2.1 Introduction 11

2.2 Quantitative Timed Simulation Functions 14

2.2.1 Simulation Relations and Quantitative Extensions 14

2.2.2 Algorithms for Simulation Functions 17

2.3 Robustness of Timed Computation Tree Logic 26

2.4 Discounted CTL for Timed Systems 31

3 Timed Automaton Games **36**

3.1 Introduction 36

3.2 Timed Games 38

3.2.1 Timed Game Structures 38

3.2.2 Timed Winning Conditions 40

3.2.3 Timed Automaton Games 43

3.3 Solving Timed Automaton Games 46

3.4 Efficient Solution of Timed Automaton Games 49

4 Timed-Alternating Time Logic **59**

4.1 Introduction 59

4.2 TATL Syntax and Semantics 60

4.3 TATL* 63

4.4 Model Checking TATL 65

5 Minimum-Time Reachability in Timed Games **68**

5.1 Introduction 68

5.2 The Minimum-Time Reachability Problem 69

5.3 Reduction to Reachability with Büchi and co-Büchi Constraints 72

5.4 Termination of the Fixpoint Iteration 76

6 Trading Memory for Randomness in Timed Games 83

6.1 Introduction 83

6.2 Randomized Strategies in Timed Games 87

6.3 Safety Objectives: Pure Finite-memory Receptive Strategies Suffice 93

6.4 Reachability Objectives: Randomized Finite-memory Receptive Strategies Suffice 96

6.5 Parity Objectives: Randomized Finite-memory Receptive Strategies Suffice 108

7 Robust Winning of Timed Games 110

7.1 Introduction 110

7.2 Robust Winning of Timed Parity Games 113

7.3 Winning with Bounded Jitter and Response Time 118

8 Conclusions 124

Bibliography 129

List of Figures

2.1 Two similar timed automata 13
2.2 A_r is 2-similar to A_s 17
2.3 First automata for game with an ε of $\Theta(2 \cdot n_1 \cdot n_2 \cdot W)$ 19
2.4 Second automata for game with an ε of $\Theta(2 \cdot n_1 \cdot n_2 \cdot W)$ 20

3.1 A timed automaton game. 37

5.1 A timed automaton game. 69
5.2 An extended region with $C_< = C \cup \{z\}, C_= = \emptyset, C_> = \emptyset$ 78
5.3 An extended region with $C_< = C \cup \{z\}, C_= = \emptyset$ and its time successor. . . . 78
5.4 An extended region with $C_< \neq \emptyset, C_= \neq \emptyset, C_> \neq \emptyset$ and its time successor. . . 79

6.1 A timed automaton game. 84

7.1 A timed automaton game $\mathcal{T}$. 111
7.2 The timed automaton game $\mathcal{T}^{\varepsilon_j, \varepsilon_r}$ obtained from $\mathcal{T}$. 121

Acknowledgements

I am indebted to my advisor Prof. Thomas A. Henzinger for his support, guidance and the generous funding for the (many) years of this endeavor. I came to Berkeley not even knowing the term “formal methods”, and fell in love with the field after taking his course on verification. I have constantly referred to his excellent CAV book (co-authored with Prof. Rajeev Alur). He has taught me about a myriad of issues, from how to think about and do research, and be precise in formulating problems, to how to write theorems in papers, how to correctly use LaTeX and how to punctuate properly when writing. He allowed me to be the teaching assistant in his CS172 course which was a wonderful learning experience. Even after his move to Switzerland, he has managed to devote more time to his students at Berkeley than most of the students get with their resident advisors. He has been ready to call late at night his time, so that I could talk in the afternoon (because of the time difference) and to work through a paper word by word. I also thank him for the many productive visits to EPFL, and for bringing me in contact with the excellent research groups he assembled at Berkeley and at EPFL. His influence will be present in all my future work, in addition to this thesis.

I am also grateful to Prof. Pravin Varaiya for serving as my co-advisor after Prof. Henzinger moved to EPFL, and for being the chair of my qualifying exam committee. He has always been ready to listen to my research (and other) problems and has provided valuable feedback and guidance. I have been incredibly lucky to have had not one but *two* such excellent advisors.

I am thankful to Prof. John Steel for being on my dissertation committee, and for teaching three beautiful courses on mathematical logic and recursion theory; to Prof. Thomas Scanlon for being on my qualifying exam committee (and for teaching two course on set theory and logic), and to Prof. David Aldous for agreeing to serve on the qualifying exam committee 18 hours before the exam when Ruth, our graduate assistant, told me she had not received confirmation from Prof Scanlon and that the exam could not take place without an outside member making me run over to Evans Hall in panic mode to Prof. Aldous at 4PM, though eventually we did receive the confirmation.

In my penultimate year, I wandered over to Prof. Kurt Keutzer’s MOT class at the recommendation of Arkadeb, and it was a revelation, a different world. I thank Prof. Keutzer for that excellent class (which must have required an enormous amount of effort and time on

his part to manage the many visitors and firms) and his efforts to cultivate entrepreneurship amongst EECS students.

I have also had the good fortune of being in the company of Prof. Rupak Majumdar who served as an oracle for various academic and non-academic issues during my stay at Berkeley. He introduced me to verification and logic and has always been ready with helpful advice and pointers, and has been an inspiration.

In the past two years I have had the pleasure and good luck of collaborating with the ultra-prolific Krishnendu Chatterjee with his infinite patience on my neverending questions on μ-calculus and parity automata (and my flawed proofs). He always made time for our papers, even when he had to write an entire (seperate) paper in two days; and lately he has spent hours traveling from Mountain View to Berkeley (and back) to discuss our work. I hope we will have many future joint projects. I have been fortunate to have collaborated with Prof. Jean-François Raskin and Thomas Brihaye on the 2007 ICALP paper. I also wish to thank Prof. Antar Bandyopadhyay for his patient help on Probability Theory when I was struggling with the STAT205 course.

The administrative staff at Berkeley has been exemplary. Ruth Gjerde has always been on top of things and always ready with solutions to the various problems of graduate students. I have not encountered a better assistant (or a nicer person) anywhere. I am also grateful to the Sylvie Vaucher in EPFL for taking care of my many visits to EPFL, and to Fabien Salvi for providing computer support, and for giving me the script for expanding ps files for printing which I've been using constantly.

I have had many good friends at Berkeley, Animesh Kumar and Biswo Poudel have helped me enormously with my move when I literally had to leave stuff at Berkeley in their capable hands and fly away, and have provided many hours of stimulating conversations; Arindam Chakrabarti has taught me to precisely analyze arguments and to question the most basic assumptions we have (and with who I hope to collaborate in the future); Arkadeb Ghosal has been an excellent officemate and collaborator; Prof. Marcin Jurdziński has shared his viewpoints on various issues and provided many hours of discussion on research. I have also enjoyed the company of Divesh Bhatt and Ushnish Basu who together with Prof. Majumdar have hosted me on several occasions when I was apartment hunting; Prof. Rahul Jain, Kaushik Ravindran, Mohan Vamsi Dunga, Satrajit Chatterjee, Karl Chen, James Wu, Minxi Gao, Arnab Nilim, Adam Cataldo, Digvijay Raorane, and Satish Kumar amongst many others. The people in Prof. Henzinger's group, past and present, have provided a stimulating work environment. I have been lucky to have been a part of UCMAP, the experience will stay with me for life. I am grateful to the many instructors at UCMAP who have provided countless hours of excellent instruction to other students, and to Shri Balram Yadav at IIT Kanpur for introducing me to this field.

I would not have been able to come to Berkeley were it not for the support from my teachers at Kanpur. Prof. Prabha Sharma's course on Linear Algebra taught me the beauty of mathematics, Prof. Katyal taught me that JEE physics could be tackled systematically and Prof. Mitra cultivated a love of physical chemistry in us in high school. I also took many other excellent courses during my BTech program, and am grateful to all my Professors. The 4-top gang at IIT Kanpur provided an incredible support group during my BTech. I am also grateful to Vivek Tandon for his extensive help during my first year at Kanpur, and to Sumedh Wale for introducing me to Linux and who was always ready to troubleshoot systems (and my assignments) and who worked straight through many nights until the problems were solved or when he was able to show that the problems were not really solvable. I enjoyed working with Tushar Kumar on countless projects throughout my last two years at Kanpur. I could not have asked for a better project partner.

I am thankful to my brother for having been there whenever I needed help, and for dragging me into outdoor adventures. Finally, I am most indebted to my parents for providing unwavering support throughout my life, and for having endured my many quirks. My BTech at IIT Kanpur (and hence this thesis) would not have been possible without their understanding, the wonderful campus environment made possible by Father, and the delicious food by Mother. I am also grateful to them for providing patient backing and for bolstering my spirits whenever I needed it during my time at Berkeley.

Chapter 1

Introduction

Timed systems. The finite state model checking approach abstracts away from time, retaining only the sequence of events of a reactive system for *qualitative* reasoning about temporal properties (see [CGP00] and [Sch04] for an introduction to model checking and verification of reactive systems). In this thesis we focus on properties of systems for which time *cannot* be abstracted away, for example, an airplane controller must not only provide inputs to the airplane, it must also do so in a timely fashion. Such systems are modeled as *timed systems* in which the passage of time is made explicit. The *discrete-time* approach models the time sequence as a monotonically increasing sequence of integers. This approach is appropriate for synchronous digital systems where states are assumed to change at only the times that are integer multiples of a known clock time period. Since physical systems may not obey this restriction, the discrete-time model is only an approximation to real-time systems (see [HMP92] for scenarios where a discrete-time approach suffices). The *dense-time* approach models time as a dense set where the event timings are monotonically increasing real (or rational) numbers.

Timed automata. Timed automata [AD94] are a well established dense-time formalism for modeling and analysis of timed systems. A timed automaton is a finite state automaton augmented with real-time clocks and clock constraints. The automaton has a finite set of locations and a finite set of clocks. All clocks increase at unit rate, and transitions in between locations are governed by clock constraints in which clocks are compared to rational constants. A transition might also reset some clocks to 0. A state in such a system consists of a location together with the values of the individual clocks. The presence of dense real-time imposes challenges for verification of properties, for example, the universality and subset inclusion problems are undecidable for timed automata (see [AM04] for a survey on decision problems). However, a wide class of verification problems on timed automata have been shown to be decidable [CY92, ACD93, CJ99, WDMR04]. In parallel with these theoretical results, efficient verification tools for real-time and hybrid systems have been implemented and successfully applied to industrial relevant case studies [HHWT95, LPY97, Fre05].

Robust models of timed systems. Timed automata and related models can distinguish between actions that are arbitrarily close in time. A state may satisfy a property, with an arbitrary small deviation in the clock values of the state leading to a violation of initial property. Since formal models for timed systems are only approximations of the real world, and are subject to estimation errors, this presents a serious shortcoming in the theory. Several attempts have been made to obtain a more robust theory for timed systems. The robust timed automata of [GHJ97] are such that if an automaton accepts a trajectory, then it must accept neighboring trajectories also; and if a robust timed automaton rejects a trajectory, then it must reject neighboring trajectories also. Another model of robustness is to introduce arbitrarily small but non-zero drifts in the rates of clocks as in [Pur98, WDMR04]. We may also explicitly model a bound on the clock drifts and delays, as in [AT04, AT05].

Games. Often we want to distinguish between the actions of different agents in a system, for example between a controller and an environment, where the controller must achieve its objective irrespective of the behavior of the environment. The actions can be differentiated by considering *games* played by interacting agents. We shall consider two player games. A game proceeds in an infinite sequence of rounds where players propose moves. Each round results in a new state, and the outcome of the game is a set of *runs*, a run being an infinite sequence of states of the system. The game may be *turn based* or *concurrent*. In turn based games, the states are partitioned into player-1 states and player-2 states: in player-1 states, player-1 chooses the successor state; and in player-2 states, player-2 chooses the successor state. In concurrent games, in every round both players simultaneously and independently choose from a set of available moves, and the combination of both choices determines the successor state. We may further categorize games as being *pure*, in which moves consist of a unique desired successor state; and *stochastic* games where moves determine a probability distribution on the possible successor states. For the most part, we shall focus on pure games. An objective Φ for a player consists of a set of desired runs, and the player wins from a given initial state if she has a strategy to ensure that no matter what the opponent does, the set of resulting runs is a subset of Φ (in stochastic games, the player maximizes her probability of winning). We mention two classes of objectives here: *safety* objectives require that the game never gets outside a designated set of states; *reachability* objectives require that player 1 ensure that the game gets to designated set of states eventually. The *synthesis* problem (or *control* problem) for reactive systems asks for the construction of a winning strategy in a game [Chu62, Büc62]. Game-theoretic formulations have proved useful not only for synthesis, but also for the modeling [Dil89, ALW89], refinement [HKR02], verification [dAHM00, AHK02], testing [BGNV05], and compatibility checking [dAH01] of reactive systems. See [Cha07] for a survey of the results of game theory relevant to reactive systems.

Timed automaton games. In timed automaton games, a player must not only indicate what transition she wants to take, but also *when* she wants to take it. Since a state consists of a location together with values of the clocks, even the set of successors from a state is uncountably infinite. Construction of algorithms for timed games hence also needs extra work to ensure termination. Termination is typically ensured by demonstrating that one can work on a finite bisimulation quotient of the timed automaton, the *region graph* of the system. Timed automaton games usually have a turn based flavor, there are controllable transitions controlled by player 1, and uncontrollable transitions controlled by player 2. A successor state in a round is determined by an action of either player 1 or player 2. Typically, player 2 transitions can occur any time; player 1 only has control over her own actions.

Since players have a choice of when they take transitions, some strategies result in runs where time does not diverge, so called *zeno* runs. Zeno runs are not physically meaningful, and hence various approaches are taken to ensure players do not win by blocking time. The simplest approach is to discretize time so that players can only take transitions at integer multiples of some fixed time period [HK99]. The second approach is to syntactically ensure that players cannot block time. The syntactic restriction is usually presented as the *strong non-zenoness* assumption where the attention is restricted to timed automata where every cycle is such that in it some clock is reset to 0 and is also greater than an integer value at some point [AM99, BBL04, PAMS98]. The third approach is to work in continuous time, and put restrictions on strategies so that a player can win only if time diverges as in [DM02, BDMP03]. This approach works for safety objectives but is unfair when player 1 wants to win a reachability objective, in this case player 2 may prevent player 1 from reaching the desired state simply by blocking time. We follow the fourth approach, first presented in [dAFH$^+$03] which treats both reachability and safety objectives (and other ω-regular objectives [Tho97]) in an equitable manner. To win a safety objective, player 1 is required to not block time; and for reachability objectives she is guaranteed that player 2 will not block time. We show in the thesis that this approach is equivalent to requiring that both players use only *receptive* strategies [SGSAL98, AH97]. To define receptive strategies, we first need to assign blame to a player in case time converges. Given a run of a game, we say player i is responsible for the run if her moves determine successor states infinitely often in the run. We blame player i for blocking time in a zeno run if she was responsible for the run. Note that we may blame both players in case of a time convergent run. A receptive strategy for player i is then such that no matter what the opponent does, player i is *never responsible for blocking time.* Restriction to receptive strategies is also fair as a player is not required to *guarantee* time divergence. We also have that if both players use receptive strategies, then time must diverge in the resulting runs.

Simulation and bisimulation relations. A *trace* of a system is a sequence of observable state predicates for a given system execution, it could, for example, be the sequence of states for a particular execution. Given an abstract system specification model S and a more detailed model I for implementation, we want to know whether I is a faithful implementation of A. This is the *trace inclusion* problem — we want to know whether every trace of I is also a trace of S to ensure that no undesirable behaviors are present in I. Unfortunately, as shown in [AD94], trace inclusion is undecidable for timed automata. The existence of a *simulation relation* is a sufficient (but not necessary) condition for trace inclusion. Let Q_s and Q_i denote the state spaces of two systems S and I respectively, and let $\mu(q)$ denote the observation on the state q. We denote $q \overset{m}{\to} q'$ to denote that that the system moves from the state q to q' for the move m. For timed automata the move m can be a simple timed move (denoting time passage), or it can be a discrete transition where the location changes. A binary relation $\preceq \subseteq Q \times Q$ is a *simulation* if $q_i \preceq q_s$ implies the following conditions:

1. $\mu(q_i) = \mu(q_s)$.

2. If $q_i \xrightarrow{m} q_i'$, then there exists q_s' such that $q_s \xrightarrow{m} q_s'$, and $q_i' \preceq q_s'$.

The state q is simulated by the state q' if there exists a simulation $\preceq$ such that $q \preceq q'$. A *bisimulation* is a symmetric simulation relation. It can be seen that if $\langle q_i, q_s \rangle \in \preceq$, then every trace from q_i is also a trace of q_s (the other direction also holds in case of a bisimulation). To see whether q_i simulates q_s, we can consider the following game: player 2 plays from I states and player 1 plays from S states. The goal of player 2 is to show that q_i is not similar to q_s, and player 1 is trying to prove otherwise by matching every move of player 2. In each round first player 2 proposes a move, which player 1 trys to match. The state q_i simulates q_s iff player 1 has a strategy for matching every move of player 2 in every round.

Simulation relations between a design and an implementation often exist in practice if the implementation is closely coupled to the specification. This happens in case the implementation follows the specification at the transition level. For timed systems, we can have *time-abstract* simulation relations where the durations of matching simple timed moves need not be the same. A time-abstract simulation relation ensures that the sequence of observations from the first state can be observed from the second state, with their timings being possibly unrelated. A *timed* simulation requires that the durations of the simple timed moves must be the same, this ensures that the timed trace from the first state is observable from the second, with timings matching exactly. Computation of the maximal time-abstract and timed simulation relations is decidable for timed automata [HHK95, Cer92, Tas98].

Quantitative extensions of simulation relations. Quantitative extensions of simulation relations define pseudo-metrics on states. For example, for discrete systems, the distance between two states may be based on *how long* player 1 can match player-2's moves in the simulation game. Such extensions are useful as a system that follows the specification for 10^{10} steps, and then diverges is clearly better than a system that diverges from the specification after 2 steps.

Temporal logics and quantitative interpretations. Temporal logics are a system for qualitatively describing and reasoning about how the truth values of assertions change over time (see [Eme90] for a survey). These logics can reason about properties like "eventually the specified assertion becomes true", or "the specified assertion is true infinitely often". Typical temporal operators include $\Diamond P$ which is true if the assertion P is true eventually, and $\Box Q$ which is true at a state if the assertion Q holds at the state and all states which can follow in a system execution. A state satisfies a temporal formula φ if the executions from the state satisfy the specification of φ. Quantitative interpretations of temporal logics reason about *how well* a state satisfies the specification. The value of a logic formula at a state is a real number rather than just a boolean value. For example, the reachability formula $\Diamond P$ may have a higher value the sooner the assertion P holds, and the safety property $\Box Q$ may have a higher value the longer Q holds.

Organization and results. We now present the organization of the thesis and the main results of each chapter.

1. (**Chapter 2**). We first present the definitions for timed transition systems and timed automata. The main results of the chapter are as follows:

   - We define quantitative timed simulation functions and show that the value of these functions can be computed to within any desired degree of accuracy for timed automata.
   - We show that the logic of timed CTL is robust with respect to our quantitative version of bisimilarity.
   - We define a quantitative temporal logic DCTL over timed systems which assigns to every CTL formula a real value that is obtained by discounting real time. We show that DCTL is robust with respect to the bisimilarity metic. We also present a model checking algorithm for a subset of the logic over timed automata.

2. (**Chapter 3**). We first present the definitions for timed games, objectives, strategies and receptiveness. Then, we demonstrate that the condition of receptiveness can be pushed into objectives; that is the winning set for a timed game with objective Φ in which only receptive strategies are allowed is the same as the winning set for the game with an objective $\mathsf{WC}(\Phi)$ in which *all* strategies are allowed. We then present a reduction of our (semi)concurrent timed automaton games to classical turn based games on finite state graphs. This reduction allows us to use the rich literature of algorithms for finite game graphs for solving timed automaton games. It also leads to algorithms with better complexity than known before.

3. (**Chapter 4**). We define the logics TATL and TATL* to specify timed properties of timed game structures and show that while model checking for TATL* is not decidable for timed automaton games, model checking for TATL is complete for EXPTIME.

4. (**Chapter 5**). We consider the optimal reachability problem for timed automata which asks for the minimal time required by a player to satisfy a proposition irrespective of what the other player does. We present an EXPTIME algorithm for computing this minimal time from all states of a timed automaton.

5. (**Chapter 6**). Chapter 6 contains the following results:

   - We show that player 1 provably needs infinite memory to win reachability objectives in certain timed automaton games.
   - We show that finite memory strategies suffice for winning safety objectives.
   - We extend earlier deterministic strategies to strategies that can use randomization.
   - We show that randomization does not help player 1 in winning at more states, and also that player 2 cannot spoil player 1 from winning from additional states with the help of randomization. Thus, the winning sets remain the same in the presence of randomized strategies.
   - We show that with the use of randomization, finite-memory strategies suffice to win all ω-regular objectives.

6. (**Chapter 7**). We define robust models of timed automaton games where player 1 must accommodate "jitter" and finite response times in her actions. We propose two jitter models.

   - In the first robust model, each move of player 1 must allow some jitter in when the action of the move is taken. The jitter may be arbitrarily small, but it must be greater than 0.
   - In the second robust model, we give a lower bound on the jitter, i.e., every move of player 1 must allow for a fixed jitter, which is specified as a parameter for the game.

   We show that the states from which player 1 can win with robust strategies under both models is computable for all ω-regular objectives.

7. (**Chapter 8**). We conclude by reviewing some of the results in the thesis and presenting directions for future work.

Related work.

Metrics and quantitative logics. Most of the work in this area has been done in the untimed setting. The work in [dAFS04] studies metrics on finite state (untimed) quantitative transition systems where propositions may have values in the set $[0, 1]$. The distance function is computed by looking at the distance between the observations at corresponding steps in the executions. Quantitative versions of the (untimed) logics LTL and μ-calculus are presented together with robustness theorems. A discounted theory for (untimed) probabilistic systems is presented in [dAHM03] where matching observations is given higher value in the present than in the future. Robustness of the discounted μ-calculus is shown with respect to discounted bisimilarity. Model checking algorithms for discounted logics on (untimed) finite state stochastic systems are presented in [dAFH$^+$05]. A measure theoretic treatment of probabilistic bisimulation distances and quantitative probabilistic logics over labeled Markov processes is presented in [DGJP04]. Bisimulation distances for generalized semi-Markov processes are studied in [GJP06]. A robustness theorem for MITL is presented for timed systems in [HVG04], where distances between states are approximated as those obtained from testing. Properties relating to skorokhod metric for trace distances of hybrid systems are studied in [CB02]. The metric combines timing mismatches with output mismatches for continuous state systems. The present thesis provides, to our knowledge, the first algorithms for computing refinement metrics on timed systems. Recently [GJP08] has explored approximate simulation relations for hybrid systems. That work however assumes that the discrete dynamics of the two hybrid systems are the same, and also requires that the time durations of the corresponding steps match exactly, with the value of the continuous variables possibly being different. Algorithms for computing approximate simulation relations for certain classes of hybrid systems are given. These metrics and related properties are also explored in [GP07b, GP07a] and it is shown that they are computable for certain classes of linear systems by solving Lyapunov-like differential equations.

Timed games. Timed automaton games were first presented in [MPS95] with the implicit assumption that it is not possible for the controller to block time. Often work has been done under the explicit assumption of strong-nonzenoness for ensuring time progress, see e.g., [PAMS98, FTM02]. The work in [AH97] looks at safety objectives, and correctly requires that a player might not stay safe simply by blocking time. It however also requires that the player achieve her objective even if the opponent blocks time, and hence is deficient for reachability objectives. In [DM02] the authors require player 1 to always allow player 2 moves, thus, in particular, player 2 can foil a reachability objective of player 1 by blocking time. The strategies of player 1 are also assumed to be region strategies by *definition*, automatically giving a finite abstraction of the game. However, they also explore the resources required by player 1 to win, in terms of the number of clocks, and the granularity of the constants that the clocks are compared to. The work is extended in [BDMP03] where player 1 cannot fully observe the states of the system. The notion of receptive strategies for timed systems is presented in [SGSAL98], however, no algorithm under the receptiveness restriction is given. The work of [dAFH$^+$03] introduced the framework where player 1 can win either by achieving the objective irrespective of what player 2 does, or she wins if her moves are allowed only finitely often by player 2. The connection to requiring that players use receptive strategies was not explored. Decidability of an extension to the logic TATL to include ATL* was presented in [BLMO07], after TATL was introduced in our paper [HP06]. There has also been work done on *weighted* timed games, where each location is given a cost rate together with a discrete cost on transitions, and the objective of player 1 is to minimize this cost for reachability or limit-average objectives. This problem is decidable under the strong non-zenoness assumption [BCFL04, BBL04], but undecidable in the general case [BBBR07]. Limit-average discrete-time games are presented in [AdAF05] where timed moves are restricted to be of durations either 0 or 1 time unit.

Robustness for timed systems. Much work has been done on obtaining robust semantics of timed automata (in the case of a single player). The robust timed automata of [GHJ97] introduce fuzziness in accepting trajectories, the automata must not just accept single trajectories, they must accept tubes of trajectories. In [HR00] it is shown that the universality problem remains undecidable for robust timed automata (and hence that they are not complementable). Another model is to introduce drifts in the rates of clocks, as explored in [Pur98, ATM05, WDMR04, WLR05]. As shown in [ATM05], timed automata with just one drifting clock are determinizable from which decidability of subset inclusion follows. The work also shows the undecidability of language problems in the multi-clock case. The work of [Pur98, Dim07] computes reachable sets in the presence of *some* clock drift. The framework of a given controller executing in parallel with the system, and where the controller has both an observation delay, and also an action delay when its actions take effect is explored in [WDMR04, WLR05]. The first paper explores whether there exists *some* delay for which a destination is reachable, the second explores the problem for a known delay parameter. Robust model checking of LTL is shown to be decidable in [BMR06] and that of coFlat-MTL in [BMR08]. The work in [AT04, AT05] explores hybrid automata in the presence of known observation and action delays.

Bibliography.

Chapter 2 is based on the paper [HMP05] co-authored with Prof. Thomas A. Henzinger, and Prof. Rupak Majumdar. Chapter 3 is based on the papers [HP06, CHP08c, CHP08a] and the technical reports [CHP08d, CHP08b] co-authored with Prof. Thomas A. Henzinger and Krishnendu Chatterjee. Chapter 4 is based on the paper [HP06] co-authored with Prof. Thomas A. Henzinger. Chapter 5 is based on the paper [BHPR07a] and the technical report [BHPR07b] co-authored with Prof. Thomas A. Henzinger, Prof. Jean-François Raskin and Thomas Brihaye. Chapter 6 is based on the paper [CHP08c] and the technical report [CHP08d] co-authored with Prof. Thomas A. Henzinger and Krishnendu Chatterjee. Chapter 7 is based on the paper [CHP08a] and the technical report [CHP08b] co-authored with Prof. Thomas A. Henzinger and Krishnendu Chatterjee.

Chapter 2

Quantifying Similarities between Timed Systems

2.1 Introduction

Most formal models for timed systems are too precise: two states can be distinguished even if there is an arbitrarily small mismatch between the timings of an event. For example, traditional timed language inclusion requires that each trace in one system be matched *exactly* by a trace in the other system. Since formal models for timed systems are only approximations of the real world, and subject to estimation errors, this presents a serious shortcoming in the theory, and has been well noted in the literature [Frä99, Pur98, WLR05, WDMR04, HR00, GHJ97, ATM05, HVG03, GJP06]. On the other hand, *untimed* notions of refinement, where each trace in one system must match only the event sequence, throws out timing altogether, an important aspect of the models.

We develop a theory of refinement for timed systems that is *robust* with respect to small timing mismatches. The robustness is achieved by generalizing timed refinement relations to metrics on timed systems that quantitatively estimate the closeness of two systems. That is, instead of looking at refinement between systems as a boolean true/false relation, we assign a positive real number between zero and infinity to a pair of timed systems (T_r, T_s) which indicates how well T_r refines T_s. In the linear setting, we define the distance between two traces as ∞ if the untimed sequences differ, and as the supremum of the difference of corresponding time points otherwise. The distance between two systems is then taken to be the supremum of closest matching trace differences from the initial states. For example, the distance between the traces $a \xrightarrow{1} b$ and $a \xrightarrow{2} b$ is 1 unit, and occurs due to the second trace lagging the first by 1 unit at b. Similarly, the distance between the first trace and the trace $a \xrightarrow{100} b$ is 99. Intuitively, the first trace is "closer" to the second than the third; our metric makes this intuition precise.

Timed trace inclusion is undecidable on timed automata [AD94]. To compute a refinement distance between timed automata, we therefore take a branching view. We define quantitative notions of timed similarity and bisimilarity which generalize timed similarity and bisimilarity relations [Cer92, Tas98] to metrics over timed systems. Given a positive real number ε, we define a state r to be ε-similar to another state s, if (1) the observations at the states match, and (2) if for every timed step from r there is a timed step from s such that the timing of events on the traces from r and s remain within ε. We provide algorithms to compute the similarity distance between two timed systems modeled as timed automata to within any given precision.

We show that bisimilarity metrics provide a robust refinement theory for timed systems by relating the metrics to timed computation tree logic (TCTL) specifications. We prove a *robustness theorem* that states close states in the metric satisfy TCTL specifications that have "close" timing requirements. For example, if the bisimilarity distance between states r and s is ε, and r satisfies the TCTL formula $\exists\Diamond_{\leq 5}a$ (i.e., r can get to a state where a holds within 5 time units), then s satisfies $\exists\Diamond_{\leq 5+2\varepsilon}a$. A similar robustness theorem for MITL was studied in [HVG04]. However, they do not provide algorithms to compute distances between systems, relying on system execution to estimate the bound.

As an illustration, consider the two timed automata in Figure 2.1. Each automaton has four locations and two clocks x, y. Observations are the same as the locations. Let the initial states be $\langle a, x = 0, y = 0\rangle$ in both automata. The two automata seem close on inspection, but traditional language refinement of T_s by T_r does not hold. The trace $\langle a, x = 0, y = 0\rangle \xrightarrow{0} \langle b, 0, 0\rangle \xrightarrow{4} \langle c, 4, 4\rangle \ldots$ in T_r cannot be matched by a trace in T_s. The automaton T_s however, does have a similar trace, $\langle a, x = 0, y = 0\rangle \xrightarrow{0} \langle b, 0, 0\rangle \xrightarrow{3} \langle c, 3, 3\rangle \ldots$ (the trace difference is 1 time unit). We want to be able to quantify this notion of similar traces. Our metric gives a directed distance of 1 between T_r and T_s: for every (timed) move of T_r from the starting state, there is a move for T_s such that the trace difference is never more than 1 unit. The two automata do have the same untimed languages, but are not timed similar. Thus, the traditional theory does not tell us if the timed languages are close,

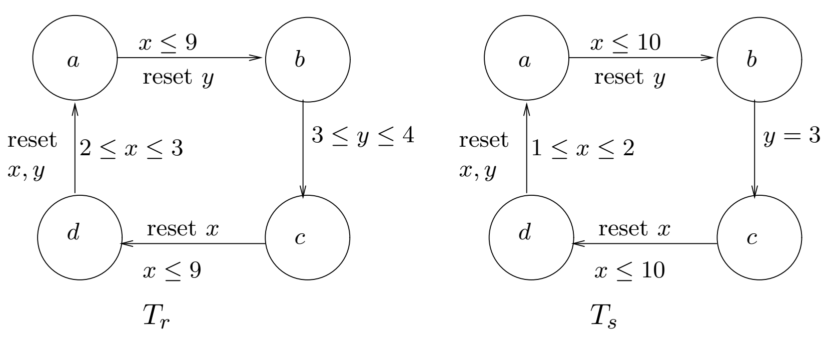


Figure 2.1: Two similar timed automata

or widely different. Looking at TCTL specifications, we note T_s satisfies $\exists\Diamond(c \wedge \exists\Diamond_{\geq 7} d)$, while T_r only satisfies the more relaxed specification $\exists\Diamond(c \wedge \exists\Diamond_{\geq 5} d)$. Robustness guarantees a bound on the relaxation of timing requirements.

Once we generalize refinement to quantitative metrics, a natural progression is to look at logical formulae as functions on states, having real values in the interval $[0, 1]$. We use *discounting* [dAFH+05, dAHM03] for this quantification and define DCTL, a quantitative version of CTL for timed systems. Discounting gives more importance to near events than to those in the far future. For example, for the reachability query $\exists\Diamond a$, we would like to see a as soon as possible. If the shortest time to reach a from the state s is t_a, then we assign β^{t_a} to the value of $\exists\Diamond a$ at s, where β is a positive discount factor less than 1 in our multiplicative discounting. The subscript constraints in TCTL (e.g., ≤ 5 in $\exists\Diamond_{\leq 5} a$) may be viewed as another form of discounting, focusing only on events before 5 time units. Our discounting in DCTL takes a more uniform view; the discounting for a time interval depends only on the duration of the interval. We also show that the DCTL values are well behaved in the sense that close bisimilar states have close values for all DCTL specifications. For the discounted CTL formula $\exists\Diamond c$, the value in T_r is β^9 and β^{10} in T_s (shortest time to reach c on time diverging paths is 9 in T_r and 10 in T_s). They are again close (on the β scale).

Outline. The rest of the chapter is organized as follows. In Section 2.2 we define the standard notions of refinement, similarity relations, trace metrics, and quantitative notions of simulation and bisimilarity, and exhibit an algorithm to compute these functions to within any desired degree of accuracy for timed automata. In Section 2.3 we prove the robustness theorem for quantitative bisimilarity with respect to timed computation tree logic. In Section 2.4, we define dCTL, show its robustness, and give a model checking algorithm for a subset of dCTL over timed automata.

2.2 Quantitative Timed Simulation Functions

We define *quantitative* refinement functions on timed systems. These functions allow approximate matching of timed traces and generalize timed and untimed simulation relations.

2.2.1 Simulation Relations and Quantitative Extensions

A *timed transition system* (TTS) is a tuple $A = \langle Q, \Sigma, \rightarrow, \mu, Q_0 \rangle$ where

- Q is the set of states.
- Σ is a set of atomic propositions (the observations).
- $\rightarrow \subseteq Q \times \mathbb{R}^+ \times Q$ is the transition relation.
- $\mu : Q \mapsto 2^\Sigma$ is the observation map which assigns a truth value to atomic propositions true in a state.
- $Q_0 \subseteq Q$ is the set of initial states.

We write $q \xrightarrow{t} q'$ if $(q, t, q') \in \rightarrow$. A *state trajectory* is an infinite sequence $q_0 \xrightarrow{t_0} q_1 \xrightarrow{t_1} \dots$, where for each $j \geq 0$, we have $q_j \xrightarrow{t_j} q_{j+1}$. The state trajectory is *initialized* if $q_0 \in Q_0$ is an initial state. A state trajectory $q_0 \xrightarrow{t_0} q_1 \dots$ induces a *trace* given by the observation sequence $\mu(q_0) \xrightarrow{t_0} \mu(q_1) \xrightarrow{t_1} \dots$. To emphasize the initial state, we say q_0-trace for a trace induced by a state trajectory starting from q_0. A trace is initialized if it is induced by an initialized state trajectory. A TTS A_i *refines* or *implements* a TTS A_s if every initialized trace of A_i is also an initialized trace of A_s. The general trace inclusion problem for timed systems is undecidable [AD94], simulation relations allow us to restrict our attention to a computable relation.

Let A be a TTS. A binary relation $\preceq \subseteq Q \times Q$ is a *timed simulation* if $q_1 \preceq q_2$ implies the following conditions:

1. $\mu(q_1) = \mu(q_2)$.

2. If $q_1 \xrightarrow{t} q_1'$, then there exists q_2' such that $q_2 \xrightarrow{t} q_2'$, and $q_1' \preceq q_2'$.

The state q is timed simulated by the state q' if there exists a timed simulation $\preceq$ such that $q \preceq q'$. A binary relation $\equiv$ is a *timed bisimulation* if it is a symmetric timed simulation. Two states q and q' are timed bisimilar if there exists a timed bisimulation $\equiv$ with $q \equiv q'$. Timed bisimulation is stronger than timed simulation which in turn is stronger than trace inclusion. If state q is timed simulated by state q', then every q-trace is also a q'-trace.

Untimed simulation and bisimulation relations are defined analogously by ignoring the duration of time steps. Formally, a binary relation $\preceq \subseteq Q \times Q$ is an (untimed) simulation if condition (2) above is replaced by
$(2)'$ If $q_1 \xrightarrow{t} q_1'$, then there exists q_2' and $t' \in \mathbb{R}^+$ such that $q_2 \xrightarrow{t'} q_2'$, and $q_1' \preceq q_2'$.
A symmetric untimed simulation relation is called an untimed bisimulation.

Timed simulation and bisimulation require that times be matched exactly. This is often too strict a requirement, especially since timed models are approximations of the real world. On the other hand, untimed simulation and bisimulation relations ignore the times on moves altogether. We now define *approximate* notions of refinement, simulation, and bisimulation that quantify if the behavior of an implementation TTS is "close enough" to a specification TTS. We begin by defining a metric on traces. Given two traces $\pi = r_0 \xrightarrow{t_0} r_1 \xrightarrow{t_1} r_2 \dots$ and $\pi' = s_0 \xrightarrow{t_0'} s_1 \xrightarrow{t_1'} s_2 \dots$, the distance $\mathcal{D}(\pi, \pi')$ is defined by

$$\mathcal{D}(\pi, \pi') = \begin{cases} \infty & : \text{ if } r_j \neq s_j \text{ for some } j \\ \sup_j \{ \lvert \sum_{n=0}^{j} t_n - \sum_{n=0}^{j} t_n' \rvert \} & : \text{ otherwise} \end{cases}$$

The trace metric $\mathcal{D}$ induces a *refinement distance* between two TTS. Given two timed transition systems A_r, A_s, with initial states Q_r, Q_s respectively, the *refinement distance* of A_r with respect to A_s is given by $\sup_{\pi_q} \inf_{\pi_{q'}'} \{\mathcal{D}(\pi_q, \pi_{q'}')\}$ where π_q (respectively, $\pi_{q'}'$) is a q-trace (respectively, q'-trace) for some $q \in Q_r$ (respectively, $q' \in Q_s$). Notice that the refinement distance is asymmetric: it is a *directed distance* [dAFS04].

We also generalize the simulation relation to a directed distance in the following way. For states r, s and $\delta \in \mathbb{R}$, the *simulation function* $\mathcal{S} : Q \times Q \times \mathbb{R} \to \mathbb{R}$ is the least fixpoint (in the absolute value sense) of the following equation:

$$\mathcal{S}(r, s, \delta) = \begin{cases} \infty & \text{if } \mu(r) \neq \mu(s) \\ \sup_{t_r}' \inf_{t_s}' \{ \max{}' (\delta,\ \mathcal{S}(r', s', \delta + t_r - t_s)) \mid r \xrightarrow{t_r} r', s \xrightarrow{t_s} s' \} & \text{otherwise} \end{cases}$$

where $\sup', \inf', \max'$ consider only the modulus in the ordering, i.e., $x <' y$ iff $|x| < |y|$ in the standard real number ordering. We say r is *ε-simulated* by s if $|\mathcal{S}(r, s, 0)| \leq \varepsilon$. Note the ε-simulation is *not* transitive in the traditional sense. If r is ε-simulated by s, and s is ε-simulated by w, then r is (2ε)-simulated by w.

Given two states r, s, it is useful to think of the value of $\mathcal{S}(r, s, \delta)$ as being the outcome of a game. Environment plays on r (and its successors), and chooses a move at each round. We play on s and choose moves on its successors. Each round adds another step to both traces (from r and s). The goal of the environment is to maximize the trace difference, our goal is to minimize. The value of $\mathcal{S}(r, s, \delta)$ is the maximum lead of the r trace with respect to the s trace when the simulation game starts with the r trace starting with a lead of δ. If from r, s the environment can force the game into a configuration in which we cannot match its observation, we assign a value of ∞ to $\mathcal{S}(r, s, \cdot)$. Otherwise, we recursively compute the maximum trace difference for each step from the successor states r', s'. For the successors r', s', the lead at the first step is $(\delta + t_r - t_s)$. The lead from the first step onwards is then $\mathcal{S}(r', s', \delta + t_r - t_s)$. The maximum trace difference is either the starting trace difference (δ), or some difference after the first step $(\mathcal{S}(r', s', \delta + t_r - t_s))$.

Note that different accumulated differences in the times in the two traces may lead to different strategies, we need to keep track of the accumulated delay or lead. For example, suppose the environment is generating a trace and is currently at state r, and our matching trace has ended up at state s. Suppose r can only take a step of length 1, and s can take two steps of lengths 0 and 100. If the two traces ending at r and s have an accumulated difference of 0 (the times at which r and s occur are exactly the same), then s should take the step of length 0. But if the r trace leads the s trace by say 70 time units, then s should take the step of length 100, the trace difference after the step will then be $|70 + 1 - 100| = 29$, if s took the 0 step, the trace difference would be $70 + 1 - 0 = 71$.

We also define the corresponding bisimulation function. For states $r, s \in Q$ and a real number δ, the *bisimulation function* $\mathcal{B} : Q \times Q \times \mathbb{R} \to \mathbb{R}$ is the least fixpoint (in the absolute value sense) of the equations $\mathcal{B}(r, s, \delta) = \infty$ if $\mu(r) \neq \mu(s)$, and

$$\mathcal{B}(r, s, \delta) = \max \left\{ \begin{array}{l} \sup'_{t_r} \inf'_{t_s} \{\max'(\delta,\ \mathcal{B}(r', s', \delta + t_r - t_s)) \mid r \xrightarrow{t_r} r', s \xrightarrow{t_s} s'\}, \\ \sup'_{t_s} \inf'_{t_r} \{\max'(\delta,\ \mathcal{B}(r', s', \delta + t_r - t_s)) \mid r \xrightarrow{t_r} r', s \xrightarrow{t_s} s'\} \end{array} \right\}$$

otherwise, where $\sup', \inf', \max'$ consider only the modulus in the ordering. The *bisimilarity distance* between two states r, s of a TTS is defined to be $\mathcal{B}(r, s, 0)$. States r, s are ε-bisimilar

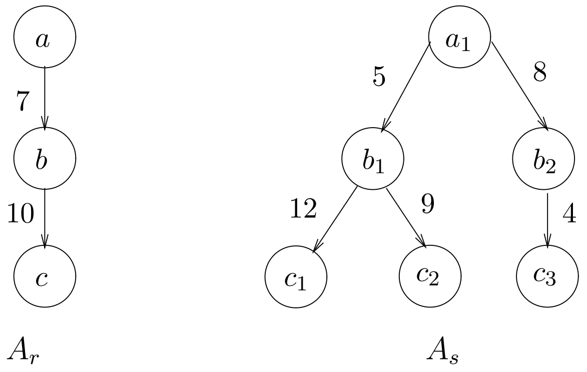


Figure 2.2: A_r is 2-similar to A_s

if $\mathcal{B}(r, s, 0) \leq \varepsilon$. Notice that $\mathcal{B}(r, s, 0) = 0$ iff r, s are timed bisimilar.

Proposition 1. *Let r and s be two states of a* TTS*. For every trace π_r from r, there is a trace π_s from s such that $\mathcal{D}(\pi_r, \pi_s) \leq |\mathcal{S}(r, s, 0)|$. The bisimilarity distance $\mathcal{B}(r, s, 0)$ is a pseudo-metric on the states of TTSs.*

Example 1. *Consider the example in Fig. 2.2. The observations have been numbered for simplicity: $\mu(a_1) = a, \mu(b_i) = b, \mu(c_i) = c$. We want to compute $\mathcal{S}(a, a_1, 0)$. It can be checked that a is untimed similar to a_1 All paths have finite weights, so $\mathcal{S}(a, a_1, 0) < \infty$. Consider the first step, a takes a step of length 7 in A_r. A_s has two options, it can take a step to b_1 of length 5 or a step to b_2 of length 8, and to decide which one to take, it needs $\mathcal{S}(b, b_1, 2)$ and $\mathcal{S}(b, b_2, -1)$. $\mathcal{S}(b, b_2, -1)$ is $-1 + 10 - 4 = 5$. To compute $\mathcal{S}(b, b_1, 2)$, we look at b_1's options. In the next step, if we move to c_2, then the trace at the (c, c_2) configuration will be $2 + 10 - 9 = 3$. If we move to c_1, the trace difference will be $2 + 10 - 12 = 0$ (this is the better option). Thus $\mathcal{S}(b, b_1, 2) = 2$ (the 2 is due to the initial lead). Thus $\mathcal{S}(a, a_1, 0) = 2$.* □

2.2.2 Algorithms for Simulation Functions

Finite Weighted Graphs.

We first look at computing ε-simulation on a special case of timed transition systems. A *finite timed graph* $T = (Q, \Sigma, E, \mu, W)$ consists of a finite set of locations Q, a set Σ of atomic propositions, an edge relation $E \subseteq Q \times Q$, an observation function $\mu : V \rightarrow 2^{\Sigma}$ on the locations, and an integer weight function $W : E \rightarrow \mathbb{N}^{+}$ on the edges. For vertices

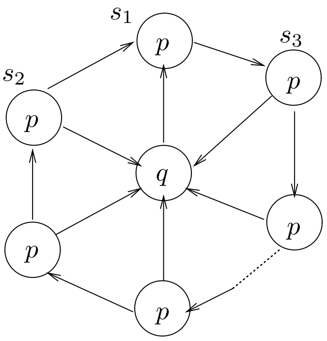
$s, s' \in Q$, we write $s \xrightarrow{t} s'$ iff there is an edge $(s, s') \in E$ with $W(s, s') = t$. The following theorem provides a bound on simulation functions on a finite timed graph.

Theorem 1. *Let A be a finite timed graph and let $n = |Q|$ be the number of nodes and $W_{\max} = \max_{e \in E}\{W(e)\}$ the maximum weight of any edge. Let $f \in \{\mathcal{S}, \mathcal{B}\}$. (1) For every pair of vertices $r, s \in Q$, if $|f(r, s, 0)| < \infty$, then $|f(r, s, 0)| \leq 2n^2 \cdot W_{\max}$. (2) The values $\mathcal{S}(r, s, 0)$ and $\mathcal{B}(r, s, 0)$ are computable over finite timed graphs in time polynomial in n and $W_{\max}$.*

Proof. The proof is by contradiction, we give the argument for $\mathcal{S}(r, s, 0)$. Since we are working on a finite graph, the sup-inf in the definition of $\mathcal{S}$ can be replaced by a max-min. Consider the product graph $A \times A$ where if $r \xrightarrow{t_r} r'$ and $s \xrightarrow{t_s} s'$ in A, then $\langle r, s\rangle \xrightarrow{t_r - t_s} \langle r', s'\rangle$ in $A \times A$. The value of the max-min can be viewed as the outcome of a game, where the environment chooses a (maximizing) move for the first vertex in the product graph, we choose a (minimizing) move for the second vertex, and the game moves to the resulting vertex pair.

Suppose $n^2 W_{\max} < |\mathcal{S}(r, s, 0)| < \infty$. Since there are only n^2 locations in the pair graph, and since each composite move can cause at most $W_{\max}$ lead or lag, there must be a cycle of composite locations in the game, with non-zero accumulative weight. When the game starts, we would do our best to not to get into such a cycle. If we cannot *avoid* getting into such a cycle because of observation matching of the environment moves, $|\mathcal{S}(v, s, 0)|$ will be ∞, because the environment will force us to loop around that cycle forever. If $|\mathcal{S}(r, s, 0)| < \infty$, and we *choose* to go into such a cycle, it must be the case that there is an alternative path/cycle that we can take which has accumulated delay of the opposite sign. For example, it may happen that at some point in the game we have an option of going into two loops, loop 1 has total gain 10, loop 2 has total gain -1000. We will take loop 1 the first 500 times, then loop 2 once, then repeat with loop 1. The leads and lags cancel out in part keeping $\mathcal{S}(r, s, 0)$ bounded. A finite value value of $\mathcal{S}(r, s, 0)$ is then due to 1) some initial hard observation matching constraint steps, with the number of steps being less than n^2 (no cycles), and 2) presence of different weight cycles (note we never need to go around the maximum weight cycle more than once). A cycle in the pair graph can have weight at most $n^2 W_{\max}$. Hence the value of $|\mathcal{S}(r, s, 0)|$ is bounded by $2n^2 W_{\max}$. □

Given the upper bound, the value of $\mathcal{S}()$ can then be computed using dynamic programming (since all edges are integer valued, it suffices to restrict our attention to



T1: All edges of weight 2

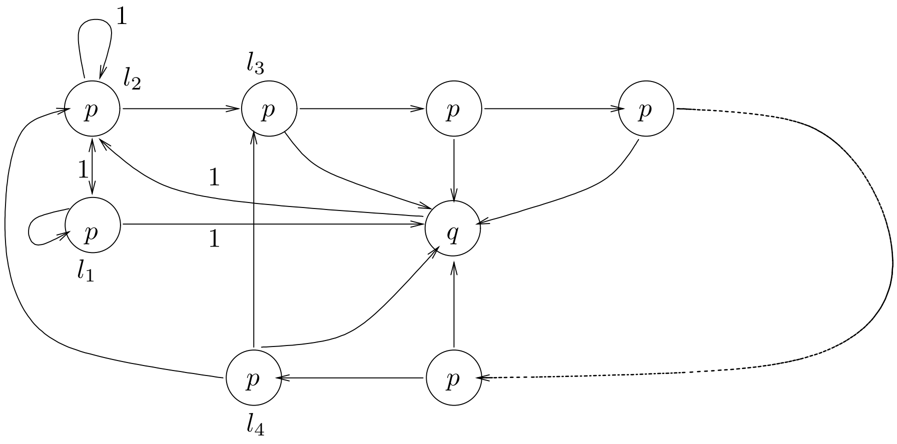
Figure 2.3: First automata for game with an ε of $\Theta(2 \cdot n_1 \cdot n_2 \cdot W)$

$\mathcal{S}(\cdot,\cdot,\delta)$ for integer valued δ). Further, this bound is tight: there is a finite timed graph A and two states r, s of A with $\mathcal{S}(r,s,0)$ in $\Theta(2 \cdot n^2 \cdot W_{\max})$.

Example 2 (Game with a simulation distance of $\Theta(2 \cdot n_1 \cdot n_2 \cdot W)$)**.** *Consider the game in Figures 2.3 and 2.4. Suppose we start in locations $\langle s_1, l_3\rangle$. Player 2 reaches l_4 in $n_2 - 4$ moves, player 1 reaches s_2 in $n_1 - 2$ steps. If $n_2 - 4 \neq n_1 - 2$, then player 2 will have to move into the long p cycle again, for it can only take the $l_4 \rightarrow l_2$ transition when player 1 is at s_2, and his next state does not have a transition to q. This process continues, and player 2 is forced to loop around the long cycle k times, where is the least number such that $k(n_2-3)-1 = m(n_1-1)-1$. At the end of the k loopings, player 1 is at s_2, and player 2 at l_4. The worst case occurs when $n_1 - 1$ and $n_2 - 3$ are relatively prime, and $k = n_1 - 1$. Accumulated weight of player 2 $= ((n_1-1)(n_3-3)-1)W$. Accumulated weight of player 1 $= ((n_1-1)(n_3-3)-1)2$, so $\varepsilon = ((n_1-1)(n_3-3)-1)(W-2)$. At the end of the loopings, player 2 can move to l_2 and then to l_1, and allow player 1 to "catch up".*

Proposition 2. *The following Algorithm computes ε-similarity for simple timed graphs.*

```
$r := 0$;
MAX_EPS:= $2 \cdot n_1 \cdot n_2 \cdot W$
visited$_r[1 \ldots n_1][1 \ldots n_2][-$MAX_EPS$\ldots$MAX_EPS$][-$MAX_EPS$\ldots$MAX_EPS$+1] :=$ FALSE
for $1 \le s_a \le n_1, 1 \le s_b \le n_2, |i| \le$ MAX_EPS do
  if $\text{obs}(s_a) = \text{obs}(s_b)$ then
    $f_0(\langle s_a, s_b, i\rangle) = g_0(\langle s_a, s_b, i\rangle) = i$
```



T2 (Default edge weight is $W >> 2$)

Figure 2.4: Second automata for game with an ε of $\Theta(2 \cdot n_1 \cdot n_2 \cdot W)$

```
  else
    $f_0(\langle s_a, s_b, i\rangle) = g_0(\langle s_a, s_b, i\rangle) = \infty$
  end if
end for
for $1 \leq s_a \leq n_1, 1 \leq s_b \leq n_2, \texttt{MAX\_EPS} + 1 \leq |i| \leq \texttt{MAX\_EPS} + W$ do
  $f_0(\langle s_a, s_b, i\rangle) = g_0(\langle s_a, s_b, i\rangle) = \infty$
end for
repeat
  $r := r + 1$
  $f_r := \Pi(f_{r-1})$
  if $|f_r(\langle s_a, s_b, i\rangle)| > 2 \cdot n_1 \cdot n_2 \cdot W$ then
    $f_r(\langle s_a, s_b\rangle) := \infty$
  end if
  $g_r := \max\{g_{r-1}, |f_r|\}$
  $\texttt{visited}_r := \texttt{visited}_{r-1}$
  for $1 \leq s_a \leq n_1, 1 \leq s_b \leq n_2, |i| \leq \texttt{MAX\_EPS}$ do
    if $f_r(\langle s_a, s_b, i\rangle) \neq \infty$ then
      $\texttt{visited}_r(\langle s_a, s_b, i, f_r(\langle s_a, s_b, i\rangle)\rangle) := \text{TRUE}$
    else
      $\texttt{visited}_r(\langle s_a, s_b, i, \texttt{MAX\_EPS} + 1\rangle) := \text{TRUE}$
    end if
  end for
until visited$_r$ = visited$_{r-1}$
$\varepsilon(\langle s_a, s_b\rangle) := g_r(\langle s_a, s_b, 0\rangle)$
```

Where for $\mathcal{H}_r(s_a, s'_a, s_b, s'_b, i) = w(s_a, s'_a) - w(s_b, s'_b) + f_{r-1}(\langle s'_a, s'_b, i + w(s_a, s'_a) - w(s_b, s'_b)\rangle$ we have

$$\Pi(f_{r-1})(\langle s_a, s_b, i\rangle) = \begin{cases} \mathcal{H}_r(s_a, s'_a, s_b, s'_b, i) \text{ such that } \operatorname{obs}(s''_a) = \operatorname{obs}(s''_b) \text{ and} \\ \quad |\mathcal{H}_r(s_a, s'_a, s_b, s'_b, i)| = \\ \quad\quad \max\limits_{s''_a : s_a \to s''_a} \left(\min\limits_{s''_b : s_b \to s''_b, \operatorname{obs}(s''_a) = \operatorname{obs}(s''_b)} |\mathcal{H}_r(s_a, s''_a, s_b, s''_b, i)| \right); \\ \infty \qquad \text{if } \exists s'_a \; \nexists s'_b (s_a \to s'_a) \wedge (s_b \to s'_b) \wedge (\operatorname{obs}(s_a) = \operatorname{obs}(s_b)) \end{cases}$$

$f_r(\langle s_a, s_b, i\rangle$ keeps track of the accumulated lead of player 1 at the rth step, assuming the game started with player 1 being in lead by i units. g_r keeps track of the maximum value of the difference seen upto the rth stage. visited$_r[s_a, s_b, i, k]$ indicates the game starting with s_a, s_b with player 1 having a lead of i has seen an ε of k before r steps (player 1 has lead player 2 by k at some step before the rth stage).

If at some point f_r becomes more than $2 \cdot n_1 \cdot n_2 \cdot W$, then we know the final value cannot be finite by Theorem. 1 , so we set it to ∞ .

For termination, we record the values of the lead/lag seen upto r steps in visited$_r$. When visited$_r$ reaches a fixpoint, no new values of lead/lag can be generated, and hence ε will not increase. Finally, the value of ε for two states s_a, s_b, is the value of the game starting at 0 : $g_r(\langle s_a, s_b, 0\rangle)$.

Suppose $n_1 = n_2 = n$, let there be m edges in each graph. Initializations take time $O(n^6 W^2)$. Each computation of Π take time $O((n^2 + m^2)W)$, so the time taken for each iteration of the repeat loop is dominated by the array assignment $O(n^6 W^2)$. In each iteration, at least one of the elements in the visited array must change, so there can at most be $O(n^6 W^2)$ iterations. Hence the total running time is $O(n^{12} W^4)$.

Timed Automata.

Timed automata provide a syntax for timed transition systems. A *timed automaton* A is a tuple $\langle L, \Sigma, C, \mu, \to, Q_0\rangle$, where

- L is the set of locations.
- Σ is the set of atomic propositions.
- C is a finite set of clocks. A *clock valuation* $v : C \mapsto \mathrm{I\!R}^+$ for a set of clocks C assigns a real value to each clock in C.
- $\mu : L \mapsto 2^\Sigma$ is the observation map (it does not depend on clock values).
- $\rightarrow \subseteq L \times L \times 2^C \times \Phi(C)$ gives the set of transitions, where $\Phi(C)$ is the set of clock constraints generated by $\psi := x \leq d \mid d \leq x \mid \neg\psi \mid \psi_1 \wedge \psi_2$.
- $Q_0 \subseteq L \times \mathrm{I\!R}^{+|C|}$ is the set of initial states.

Each clock increases at rate 1 inside a location. A *clock valuation* is a function $\kappa : C \mapsto \mathrm{I\!R}_{\geq 0}$ that maps every clock to a nonnegative real. The set of all clock valuations for C is denoted by $K(C)$. Given a clock valuation $\kappa \in K(C)$ and a time delay $\Delta \in \mathrm{I\!R}_{\geq 0}$, we write $\kappa + \Delta$ for the clock valuation in $K(C)$ defined by $(\kappa + \Delta)(x) = \kappa(x) + \Delta$ for all clocks $x \in C$. For a subset $\lambda \subseteq C$ of the clocks, we write $\kappa[\lambda := 0]$ for the clock valuation in $K(C)$ defined by $(\kappa[\lambda := 0])(x) = 0$ if $x \in \lambda$, and $(\kappa[\lambda := 0])(x) = \kappa(x)$ if $x \notin \lambda$. A clock valuation $\kappa \in K(C)$ *satisfies* the clock constraint $\theta \in \mathsf{Constr}(C)$, written $\kappa \models \theta$, if the condition θ holds when all clocks in C take on the values specified by κ.

A *state* $s = \langle l, \kappa \rangle$ of the timed automaton game $\mathcal{T}$ is a location $l \in L$ together with a clock valuation $\kappa \in K(C)$ such that the invariant at the location is satisfied, that is, $\kappa \models \gamma(l)$. The set of states is denoted $Q = L \times (\mathrm{I\!R}^+)^{|C|}$. An edge $\langle l, l', \lambda, g \rangle$ represents a transition from location l to location l' when the clock values at l satisfy the constraint g. The set $\lambda \subseteq C$ gives the clocks to be reset with this transition. The semantics of timed automata are given as timed transition systems. This is standard [AD94], and omitted here.

For simplicity, we assume every clock of a timed automaton A stays within $M + 1$, where M is the largest constant in the system.

Region equivalence relation. Algorithms for problems on timed automata typically use the region equivalence relation which induces a time-abstract bisimulation quotient. For a real $t \geq 0$, let $\mathsf{frac}(t) = t - \lfloor t \rfloor$ denote the fractional part of t. Given a timed automaton game $\mathcal{T}$, for each clock $x \in C$, let c_x denote the largest integer constant that appears in any clock constraint involving x in $\mathcal{T}$. Two states $\langle l_1, \kappa_1 \rangle$ and $\langle l_1, \kappa_1 \rangle$ are said to be *region equivalent* if all the following conditions are satisfied:

1. The locations match, that is $l_1 = l_2$.
2. For all clocks x, $\kappa_1(x) \leq c_x$ iff $\kappa_2(x) \leq c_x$.
3. For all clocks x with $\kappa_1(x) \leq c_x$, $\lfloor \kappa_1(x) \rfloor = \lfloor \kappa_2(x) \rfloor$.
4. For all clocks x, y with $\kappa_1(x) \leq c_x$ and $\kappa_1(y) \leq c_y$, $\mathsf{frac}(\kappa_1(x)) \leq \mathsf{frac}(\kappa_1(x))$ iff $\mathsf{frac}(\kappa_2(x)) \leq \mathsf{frac}(\kappa_2(x))$, and
5. For all clocks x with $\kappa_1(x) \leq c_x$, $\mathsf{frac}(\kappa_1(x)) = 0$ iff $\mathsf{frac}(\kappa_2(x)) = 0$.

A *region* is an equivalence class of states with respect to the region equivalence relation. There are finitely many clock regions; more precisely, the number of clock regions is bounded by $|L| \cdot \prod_{x \in C}(c_x + 1) \cdot |C|! \cdot 2^{|C|}$.

A region R of a timed automaton A can be represented as a tuple $\langle l, h, \mathcal{P}(C) \rangle$ where

- l is a location of A.
- h is a function which specifies the integer values of clocks $h : C \rightarrow (\mathbb{N} \cap [0, M])$ (M is the largest constant in A).
- $\mathcal{P}(C)$ is a disjoint partition of the clocks $\{X_0, \ldots X_n \mid \uplus X_i = C, X_i \neq \emptyset \text{ for } i > 0\}$.

We say a state s with clock valuation v is in the region R when,

1. The location of s corresponds to the location of R
2. For all clocks x with $\kappa(x) < M + 1$, $\lfloor \kappa(x) \rfloor = h(x)$.
3. For $\kappa(x) \geq M + 1$, $h(x) = M$. (This is slightly more refined than the standard region partition, we have created more partitions in $[M, M + 1)$, we map clock values which are greater than M into this interval. This is to simplify the proofs.)
4. For any pair of clocks (x, y), $\mathsf{frac}(\kappa(x)) < \mathsf{frac}(\kappa(y))$ iff $x \in X_i$ and $y \in X_j$ with $i < j$ (so, $x, y \in X_k$ implies $\mathsf{frac}(\kappa(x)) = \mathsf{frac}(\kappa(y))$).
5. $\mathsf{frac}(\kappa(x)) = 0$ iff $x \in X_0$.

We now show that given states r, s in a timed automaton A, the values of $\mathcal{S}(r, s, 0)$ and $\mathcal{B}(r, s, 0)$ can be computed to within any desired degree of accuracy. We use a *corner point abstraction* (similar to that in [BBL04]) which can be viewed as a region graph augmented with additional timing information. We show that the corner points are at a close bisimilarity distance from the states inside the corresponding regions. Finally we use Theorem 1 to compute the approximation for $\mathcal{S}(\cdot)$ on the corner point graph.

A *corner point* is a tuple $\langle \alpha, R\rangle$, where $\alpha \in \mathbb{N}^{|C|}$ and R is a region. A region $R = \langle l, h, \{X_0, \dots X_n\}\rangle$ has $n+1$ corner points $\{\langle \alpha_i, R\rangle \mid 0 \leq i \leq n\}$:

$$\alpha_i(x) = \begin{cases} h(x) & : \quad x \in X_j \text{ with } j \leq i \\ h(x)+1 & : \quad x \in X_j \text{ with } j > i \end{cases}$$

Intuitively, corner points denote the boundary points of the region.

Using the corner points, we construct a finite timed graph as follows. The structure is similar to the region graph, only we use corner points, and weights on some of the edges to model the passage of time. For a timed automaton A, the *corner point abstraction* CP(A) has corner points p of A as states. The observation of the state $\langle \alpha, \langle l, h, \mathcal{P}(C)\rangle\rangle$ is $\mu(l)$. The abstraction has the following weighted transitions :

Discrete There is an edge $\langle \alpha, R\rangle \xrightarrow{0} \langle \alpha', R'\rangle$ if A has an edge $\langle l, l', \lambda, g\rangle$ (l, l' are locations of R, R' respectively) such that (1) R satisfies the constraint g, and (2) $R' = R[\lambda \mapsto 0]$, $\alpha' = \alpha[\lambda \mapsto 0]$ (note that corner points are closed under resets).

Timed For corner points $\langle \alpha, R\rangle, \langle \alpha', R\rangle$ such that $\forall x \in C,\ \alpha'(x) = \alpha(x)+1$, we have an edge $\langle \alpha, R\rangle \xrightarrow{1} \langle \alpha', R\rangle$. These are the edges which model the flow of time. Note that for each such edge, there are concrete states in A which are arbitrarily close to the corner points, such that there is a time flow of length arbitrarily close to 1 in between those two states.

Region flow These transitions model the immediate flow transitions in between “adjacent” regions. Suppose $\langle \alpha, R\rangle, \langle \alpha, R'\rangle$ are such that R' is an immediate time successor of R, then we have an edge $\langle \alpha, R\rangle \xrightarrow{0} \langle \alpha, R'\rangle$. If $\langle \alpha+1, R'\rangle$ is also a corner point of R', then we also add the transition $\langle \alpha, R\rangle \xrightarrow{1} \langle \alpha+1, R'\rangle$.

Self loops Each state also has a self loop transition of weight 0.

Transitive closure We transitively close the timed, region flow, and the self loop transitions upto weight M (the subset of the full transitive closure where edges have weight less than or equal to M).

The number of states in the corner point abstraction of a timed automaton A is $O(|L| \cdot |C| \cdot (2M)^{|C|})$, where L is the set of locations in A, C the set of clocks, and M the largest constant in the system.

Lemma 1. *Let s be a state in a timed automaton A, and let p be a corner point of the region R corresponding to s in the corner point abstraction of A. Then s is ε-bisimilar to p for $\varepsilon = |C| + 1$, that is, $\mathcal{S}(s, \mathsf{p}, 0) \leq |C| + 1$, where C is the set of clocks in A.*

Informally, each clock can be the cause of at most 1 unit of time difference, as the time taken to hit a constraint is always of the form $d - \kappa(x)$ for some clock x and integer d. Once a clock is reset, it collapses onto a corner point, and the time taken from that point to reach a constraint controlled by x is the same as that for the corresponding corner point in CP(A).

Using Lemma 1 and Theorem 1, we can "blow" up the time unit for a timed automaton to compute ε-simulation and ε-bisimilarity to within any given degree of accuracy. This gives an EXPTIME algorithm in the size of the timed automaton and the desired accuracy.

Theorem 2. *Given two states r, s in a timed automaton A, and a natural number m, we can compute numbers $\gamma_1, \gamma_2 \in \mathbb{R}$ such that $\mathcal{S}(r, s, 0) \in [\gamma_1 - \frac{1}{m}, \gamma_1 + \frac{1}{m}]$ and $\mathcal{B}(r, s, 0) \in [\gamma_2 - \frac{1}{m}, \gamma_2 + \frac{1}{m}]$ in time polynomial in the number of states of the corner point abstraction and in $m^{|C|}$, where C is the set of clocks of A.*

Proof. Suppose given m, we want to compute $\mathcal{S}(r, s, 0)$ to an accuracy within $1/m$. Multiply the timed automaton by $u = m2(|C| + 1)$, and let r', s' be region equivalent to ru, su (each clock value multiplied by u) in the resulting corner point abstraction. We have $\mathcal{S}(ru, r') \leq |C| + 1, \mathcal{S}(s', su, 0) \leq |C| + 1$.
Also let $\mathcal{S}(r', s', 0) = \eta$ $\mathcal{S}(ru, su, 0) \leq \mathcal{S}(ru, r', 0) + \mathcal{S}(r', s', 0) + \mathcal{S}(s', su, 0) = \eta + 2(|C| + 1)$. So $\mathcal{S}(r, s, 0) \leq \eta/u + 2(|C| + 1)/u = \eta/u + 1/m$. The other direction is similar. □

2.3 Robustness of Timed Computation Tree Logic

TCTL.

Timed computation tree logic (TCTL) [ACD93] is a real time extension of CTL [CES86]. TCTL adds time constraints such as ≤ 5 to CTL formulae for specifying timing requirements. For example, while the CTL formula $\forall\Diamond a$ only requires a to eventually hold on all paths, the TCTL formula $\forall\Diamond_{\leq 5} a$ requires a to hold on all paths before 5 time units.

We will use $\sim$ to mean one of the binary relations $<, \leq, >, \geq$. The formulae of TCTL are given inductively as follows:

$$\varphi \ := \ a \mid \text{FALSE} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \exists(\varphi_1 \mathcal{U}_{\sim d} \varphi_2) \mid \forall(\varphi_1 \mathcal{U}_{\sim d} \varphi_2)$$

where $a \in \Sigma$ and $d \in \mathbb{N}$.

The semantics of TCTL formulas is given over states of timed transition systems. For a state s in a TTS

- $s \models a$ iff $a \in \mu(s)$.
- $s \not\models$ FALSE.
- $s \models \neg\varphi$ iff $s \not\models \varphi$.
- $s \models \varphi_1 \vee \varphi_2$ iff $s \models \varphi_1$ or $s \models \varphi_2$.
- $s \models \varphi_1 \wedge \varphi_2$ iff $s \models \varphi_1$ and $s \models \varphi_2$.
- $s \models \exists(\varphi_1 \mathcal{U}_{\sim d} \varphi_2)$ iff for some run ρ_s starting from s, for some $t \sim d$, the state at time t, $\rho_s(t) \models \varphi_2$, and for all $0 \leq t' < t$, $\rho_s(t') \models \varphi_1$.
- $s \models \forall(\varphi_1 \mathcal{U}_{\sim d} \varphi_2)$ iff for all (infinite) paths ρ_s starting from s, for some $t \sim d$, the state at time t, $\rho_s(t) \models \varphi_2$, and for all $0 \leq t' < t$, $\rho_s(t') \models \varphi_1$.

We define the *waiting-for* operator as $\exists(\varphi_1 \mathcal{W}_{\sim c} \varphi_2) = \neg\forall(\neg\varphi_2 \mathcal{U}_{\sim c} \neg(\varphi_1 \vee \varphi_2))$, $\forall(\varphi_1 \mathcal{W}_{\sim c} \varphi_2) = \neg\exists(\neg\varphi_2 \mathcal{U}_{\sim c} \neg(\varphi_1 \vee \varphi_2))$. The until operator in $\varphi_1 \mathcal{U}_{\sim d} \varphi_2$ requires that φ_2 become true at some time, the waiting-for formula $\varphi_1 \mathcal{W}_{\sim d} \varphi_2$ admits the possibility of φ_1 forever "waiting" for all times $t \sim d$ and φ_2 never being satisfied. Formally, $s \models \forall(\varphi \mathcal{W}_{\sim d} \theta)$ (respectively, $s \models \exists(\varphi \mathcal{W}_{\sim d} \theta)$) iff for all traces (respectively, for some trace) ρ_s from s, either 1) for all times $t \sim d$, $\rho_s(t) \models \varphi$, or 2) at some time t, $\rho_s(t) \models \theta$, and for all $(t' < t) \wedge (t' \sim d)$, $\rho_s(t') \models \varphi$. Using the waiting-for operator and the identities $\neg(\varphi \exists \mathcal{U}_{\sim d} \theta) = (\neg\varphi) \forall \mathcal{W}_{\sim d}(\neg\varphi \wedge \neg\theta)$ and $\neg(\varphi \forall \mathcal{U}_{\sim d} \theta) = (\neg\varphi) \exists \mathcal{W}_{\sim d}(\neg\varphi \wedge \neg\theta)$, we can write each TCTL formula φ in negation normal form by pushing the negation to the atomic propositions.

Lemma 2. *$\neg(\varphi \ddagger \mathcal{U}_{\sim d} \theta) = (\neg\varphi) \bar{\ddagger} \mathcal{W}_{\sim d}(\neg\varphi \wedge \neg\theta)$ where $\ddagger = \forall(\exists)$, $\bar{\ddagger} = \exists(\forall)$ (the corresponding dual to $\ddagger$).*

Proof. We prove the first claim, the other case is similar.
$\Rightarrow$. We try to see if with the given condition $s \not\models \neg(\neg\theta \exists \mathcal{U}_{\sim c} \neg(\varphi \vee \theta))$, ie if $s \models \psi = \neg\theta \exists \mathcal{U}_{\sim c} \neg(\varphi \vee \theta)$.

Suppose for all $t \sim c$, $\rho(t) \models \varphi$, then, $\rho(t) \models \varphi \vee \theta$, so assume at some time t, $\rho(t) \models \theta$, and for $(t' < t) \wedge (t' \sim c)$ $\rho(t') \models \varphi$.

If $\neg(t \sim c)$, then clearly ψ cannot be satisfied, so assume $t \sim c$.

At time t θ is satisfied, so to satisfy ψ, we must have $\neg\varphi \wedge \neg\theta$ satisfied at $(t' < t) \wedge (t' \sim c)$. But this is not possible, for at all $(t' < t) \wedge (t' \sim c)$, we have $\rho(t') \models \varphi$ Thus, there is no trace which can be a witness for satisfying ψ, ie $s \models \neg\psi$, which is the given formula.
$\Leftarrow$. $s \models \neg(\neg\theta \exists \mathcal{U}_{\sim c} \neg(\varphi \vee \theta))$ iff there is no trace ρ such that for some time $t \sim c$, $\rho(t) \models \neg(\varphi \vee \theta)$ and for all $t' < t$ $\rho(t') \models \neg\theta$ iff for all traces either for all $t \sim c$ $\rho(t) \models \varphi \vee \theta$ or if for $t \sim c$, if $\rho(t) \models \neg\varphi \wedge \neg\theta$, then there is some $t' < t$ such that $\rho(t') \models \theta$. We claim this implies the given condition.

Suppose for some trace ρ for all $t \sim c$, $\rho(t) \models \varphi \vee \theta$. Assume at time $t' \sim c$, $\rho(t') \models \theta \wedge \neg\varphi$. For this trace to be a witness to unsatisfiability of the given two conditions, the second clause needs to be violated (we have assumed the violation of the first condition). We also have $\rho(t) \models \theta$. Assume there is no t' such that $\neg(t' \sim c)$ and $\rho(t') \models \theta$. So then, we need that there be some $t' \sim c$ such that $\rho(t') \models \neg\varphi$ and for all $(t'' \leq t') \wedge (t'' \sim c)$, $\rho(t'') \models \neg\theta$. But that means at $\rho(t') \models \neg\varphi \wedge \neg\theta$, contrary to assumption.

Suppose for some trace ρ that there is some $t \sim c$ such that $\rho(t) \models \neg(\theta \vee \varphi)$. Then, we also have that there is some $t' < t$ such that $\rho(t') \models \theta$. This trace violates the first condition, we need to see if it can violate the second one. The second condition can be written as $\exists t[\rho(t) \models \theta \wedge \forall x(((x < t) \wedge (x \sim c)) \rightarrow \rho(x) \models \varphi)] = \exists t[\rho(t) \models \theta \wedge \forall x((x \geq t) \vee \neg(x \sim c) \vee \rho(x) \models \varphi)]$. The negation of the condition is $\forall t[\rho(t) \not\models \theta \vee \exists x((x < t) \wedge (x \sim c) \wedge \rho(x) \not\models \varphi)] = \forall t[\rho(t) \models \theta \rightarrow \exists x((x < t) \wedge (x \sim c) \wedge \rho(x) \not\models \varphi)]$. This condition can be seen to be equivalent to $\forall t[\rho(t) \models \theta \rightarrow \exists x\{((x < t) \wedge (x \sim c) \wedge \rho(x) \not\models \varphi) \wedge \forall y(y \leq x \rightarrow \rho(y) \not\models \theta)\}]$.

But we have that if at a time $t \sim c$ $\rho(t) \models \neg(\theta \vee \varphi)$ then there is always a $t' < t$ such that $\rho(t') \models \theta$. Thus the previous condition is not satisfiable, and hence we must satisfy at least one of the conditions in the lemma.

□

δ-weakened TCTL.

For each TCTL formula φ in negation normal form, and $\delta \in \mathbb{R}^+$, a δ-weakening $\zeta^\delta(\varphi)$ of φ with respect to δ is defined as follows:

- $\zeta^\delta(a) := a$
- $\zeta^\delta(\neg a) := \neg a$
- $\zeta^\delta(\text{FALSE}) := \text{FALSE}$
- $\zeta^\delta(\varphi_1 \vee \varphi_2) = \zeta^\delta(\varphi_1) \vee \zeta^\delta(\varphi_2)$
- $\zeta^\delta(\varphi_1 \wedge \varphi_2) = \zeta^\delta(\varphi_1) \wedge \zeta^\delta(\varphi_2)$
- $\zeta^\delta(\ddagger(\varphi_1 \mathcal{U}_{\sim d} \varphi_2)) = \ddagger(\zeta^\delta(\varphi_1) \mathcal{U}_{\sim \delta(d,\sim)} \zeta^\delta(\varphi_2))$
- $\zeta^\delta(\ddagger(\varphi_1 \mathcal{W}_{\sim d} \varphi_2)) = \ddagger(\zeta^\delta(\varphi_1) \mathcal{W}_{\sim \delta'(d,\sim)} \zeta^\delta(\varphi_2))$

where $\ddagger \in \{\exists, \forall\}$ and

$$\delta(d,\sim) = \begin{cases} d+\delta & \text{if } \sim \in \{<,\leq\} \\ d-\delta & \text{if } \sim \in \{>,\geq\} \end{cases} \qquad \delta'(d,\sim) = \begin{cases} d-\delta & \text{if } \sim \in \{<,\leq\} \\ d+\delta & \text{if } \sim \in \{>,\geq\} \end{cases}$$

The ζ^δ function relaxes the timing constraints by δ. The $\mathcal{U}$ and the $\mathcal{W}$ operators are weakened dually. Note that $\neg\zeta^\delta(\psi) \neq \zeta^\delta(\neg\psi)$. The discrepancy occurs because of the difference in how δ and δ' are defined. Let

$$\delta_2(d,\sim) = \begin{cases} d+2\delta & \text{where } \sim \text{ is } <,\leq \\ d-2\delta & \text{where } \sim \text{ is } >,\geq \end{cases}$$

and

$$\delta_2'(d,\sim) = \begin{cases} d-2\delta & \text{where } \sim \text{ is } <,\leq \\ d+2\delta & \text{where } \sim \text{ is } >,\geq \end{cases}$$

Proposition 3.

- $\neg\zeta^{\delta}(p) = \zeta^{\delta}(\neg p)$
- $\neg\zeta^{\delta}(\text{FALSE}) = \zeta^{\delta}(\neg\text{FALSE})$
- $\neg\zeta^{\delta}(\varphi_1 \vee \varphi_2) = \zeta^{\delta}(\neg\varphi_1) \vee \zeta^{\delta}(\neg\varphi_2)$
- $\neg\zeta^{\delta}(\varphi_1 \wedge \varphi_2) = \zeta^{\delta}(\neg\varphi_1) \wedge \zeta^{\delta}(\neg\varphi_2)$
- $\neg\zeta^{\delta}(\ddagger(\varphi_1 \mathcal{U}_{\sim d}\varphi_2)) = \zeta^{\delta}(\neg \ddagger (\varphi_1 \mathcal{U}_{\sim \delta_2'(d,\sim)}\varphi_2))$
- $\neg\zeta^{\delta}(\ddagger(\varphi_1 \mathcal{W}_{\sim d}\varphi_2)) = \zeta^{\delta_2'(c)}(\neg \ddagger (\varphi_1 \mathcal{W}_{\sim \delta_2(d,\sim)}\varphi_2))$

Proof. Take for instance $\varphi \ddagger \mathcal{U}_{\sim c}\theta$. $\neg\zeta^{\delta}(\varphi \ddagger \mathcal{U}_{\sim c}\theta) = \neg(\zeta^{\delta}(\varphi) \ddagger \mathcal{U}_{\sim\delta(c,\sim)}\zeta^{\delta}(\theta)) = \neg(\zeta^{\delta}(\varphi))\overline{\ddagger}\mathcal{W}_{\sim\delta(c,\sim)}\neg(\zeta^{\delta}(\theta) \vee \zeta^{\delta}(\varphi)) = \zeta^{\delta}(\neg\varphi))\overline{\ddagger}\mathcal{W}_{\sim\delta(c,\sim)}\zeta^{\delta}(\neg(\theta \vee \varphi)) = \zeta^{\delta}(\neg\varphi))\overline{\ddagger}\mathcal{W}_{\sim\delta'(\delta_2(c,\sim),\sim)}\zeta^{\delta}(\neg(\theta \vee \varphi)) = \zeta^{\delta}(\neg\varphi\overline{\ddagger}\mathcal{W}_{\delta_2(c,\sim)}\neg(\theta \vee \varphi)) = \zeta^{\delta}(\neg(\varphi \ddagger \mathcal{U}_{\sim\delta_2(c,\sim)}\theta))$ □

Example 3. *Let a and b be atomic propositions. We have* $\zeta^2(\exists(a\mathcal{U}_{\leq 5}b)) = \exists(a\mathcal{U}_{\leq 7}b)$. *Earlier, a state had to get to b within 5 time units, now it has 7 time units to satisfy the requirement. Similarly,* $\zeta^2(\exists(a\mathcal{W}_{\leq 5}b)) = \exists(a\mathcal{W}_{\leq 3}b))$. *The pre-weakened formula requires that either 1) for all* $t \leq 5$ *the proposition a must hold, or 2) at some time t, b must hold, and for all* $(t' < t) \wedge (t' \leq 5)$ *a must hold. The weakening operator relaxes the requirement on a holding for all times less than or equal to 5 to only being required to hold at times less than or equal to 3 (modulo the* $(t' < t)$ *clause in case 2).*

The next lemma states that the ζ operator is indeed a weakening operator.

Lemma 3. *For all reals* $\delta \geq 0$, TCTL *formulae* φ, *and states s of a* TTS, *if* $s \models \varphi$, *then* $s \models \zeta^{\delta}(\varphi)$.

Proof. The proofs for base case, and the boolean connectives are obvious. Consider the $\forall\mathcal{U}$ case. Suppose $s \models \varphi\forall\mathcal{U}_{\sim c}\theta$. Then for every path ρ starting from s, for some $t \sim c$ $\rho(t) \models \theta$, and for all $t' < t$, $\rho(t') \models \varphi$. The induction hypothesis gives $\rho(t) \models \zeta^{\delta}(\theta)$, and $\rho(t') \models \zeta^{\delta}(\varphi)$. Thus $s \models \zeta^{\delta}(\varphi\forall\mathcal{U}_{\sim c}\theta)$. The other connectives follow a similar outline. □

We now connect the bisimilarity metric with satisfaction of TCTL specifications. Of course, close states may not satisfy the *same* TCTL specifications. Take $\varphi = \forall\Diamond_{=5}a$, it requires a to occur at exactly 5 time units. One state may have traces that satisfy a at exactly 5 time units, another state at $5 + \varepsilon$ for an arbitrarily small ε. The first state will satisfy φ, the second will not. However, two states close in the bisimilarity metric does satisfy "close" TCTL specifications. Theorem 4 makes this precise.

Define state s_0, s_1 to be $[\omega, \varepsilon]$ bisimilar if

1. $|\omega| \leq \varepsilon$.
2. The observations of s_0, s_1 match.
3. If s_i takes a discrete move to s_i^1, then $s_{\bar{i}}$ makes a discrete move to $s_{\bar{i}}^1$, such that $\text{obs}(s_i)^1 = \text{obs}(s_{\bar{i}}^1)$; and s_0^1, s_1^1 are again $(\varepsilon_0, \varepsilon_1)^+$ bisimilar.
4. If s_0 (s_1) takes a time move t (t'), then s_1 (s_0) can take a time move t' (t) such that $s_0 + t, s_1 + t'$ are $[\omega + (t - t'), \varepsilon]$ bisimilar.

ω, keeps track of how much system 1 is "leading" or "lagging", and ε keeps track of the maximum value of ω seen so far.

Lemma 4. *Two states are ε-bisimilar iff they are $[0, \varepsilon]$ bisimilar.*

Theorem 3. *Suppose s_1, s_2 are two (ω, ε) bisimilar states. If $s_1 \models \varphi$, then $s_2 \models \zeta^\delta(\varphi)$, where $\delta = 2\varepsilon$.*

Proof. The base case, and the boolean connectives are again simple. Take the $\forall\mathcal{W}$ case. Suppose $s_1 \models \varphi \forall\mathcal{W}_{\sim c}\theta$ and $s_2 \not\models \zeta^\delta(\varphi)\forall\mathcal{W}_{\sim\delta'(\sim,c)}\zeta^\delta(\theta)$. Since s_2 does not satisfy the formula, it must be that $s_2 \models \neg(\zeta^\delta(\varphi)\forall\mathcal{W}_{\sim\delta'(\sim,c)}\zeta^\delta(\theta))$. Equivalently $s_2 \models \neg\zeta^\delta(\theta)\exists\mathcal{U}_{\sim\delta'(\sim,c)}\neg(\zeta^\delta(\varphi) \vee \zeta^\delta(\theta))$.

Take $\sim$ to be $>$, the other cases are similar. There must be a path ρ_2 starting from s_2 such that 1) there exists a time $t > c + \delta$ such that $\rho_2(t) \models \neg\zeta^\delta(\varphi) \wedge \neg\zeta^\delta(\theta)$, and 2) for all $t' < t$, $\rho_2(t') \models \neg\zeta^\delta(\theta)$. Since s_1, s_2 are $[\omega, \varepsilon]$ bisimilar, there exists a path ρ_1, and a time t_1 corresponding to t such that $\rho_1(t_1), \rho_2(t)$ are $[\omega + t_1 - t, \varepsilon]$ bisimilar. $|\omega + t_1 - t| \leq \varepsilon$, so $-\omega - \varepsilon + t \leq t_1$. $\omega \leq \varepsilon$, and $t > c + 2\varepsilon$, so we get $t_1 > c$.

$s_1 \models \varphi\forall\mathcal{W}_{>c}\theta$, so either for all $t_1' > c$ $\rho_1(t_1') \models \varphi$, or, for some t_1' $\rho_1(t_1') \models \theta$, and for all $c < t_1'' < t_1'$, $\rho_1(t_1'') \models \varphi$. In the first case using $\rho_1(t_1) \models \varphi$, we get $\rho_2(t) \models \zeta^\delta(\varphi)$ by inductive hypothesis - a contradiction. In the second case, suppose there is a $t_1' \leq c$ such that $\rho_1(t_1') \models \theta$. Let t' be such that $\rho_1(t_1), \rho_2(t')$ are $[\omega + t_1 - t', \varepsilon]$ bisimilar. Using $t_1' \leq c, \omega \leq \varepsilon, -(\omega + t_1 - t') \leq \varepsilon$, we get $t' \leq c + 2\varepsilon$, and by inductive hypothesis we have $\rho_2(t') \models \zeta^\delta(\theta)$, again a contradiction.

So, we just need to see if there can be a $t_1' > c \; \rho_1(t_1') \models \theta$, and for all $c < t_1'' < t_1'$, $\rho_1(t_1'') \models \varphi$.

Suppose there is such a $t_1' > t_1$. Since $\rho_1(t_1) \models \varphi$, we get $\rho_2(t) \models \zeta^\delta(\varphi)$ by inductive hypothesis, a contradiction. So if there is a t_1', we must have $t_1' \leq t_1$ Now $\rho_1(t_1'), \rho_2(t')$ are ε bisimilar for some $t' \leq t$. $\rho_1(t_1') \models \theta$, so by inductive hypothesis $\rho_2(t') \models \zeta^\delta(\theta)$, a contradiction.

Thus finally, we must have that $s_2 \models \zeta^\delta(\varphi) \forall \mathcal{W}_{>\delta'(>,c)} \zeta^\delta(\theta) = \zeta^\delta(\varphi \forall \mathcal{W}_{>c} \theta)$. □

Theorem 4. *Let $\varepsilon > 0$. Let r, s be two ε-bisimilar states of a timed transition system, and let φ a TCTL formula in negation normal form. If $r \models \varphi$, then $s \models \zeta^{2\varepsilon}(\varphi)$.*

The proof follows from Lemma 4 and Theorem 3. The crucial point is to note that if r, s are ε-bisimilar, and if, starting from r, s the bisimilarity game arrives at the configuration r_1, s_1, then r_1, s_1 are 2ε-bisimilar. So if $r \overset{t_1}{\rightsquigarrow} r_1 \overset{t_2}{\rightsquigarrow} r_2$, and $s \overset{t_1'}{\rightsquigarrow} s_1 \overset{t_2'}{\rightsquigarrow} s_2$ (with r_i, s_i being the corresponding states), then $|t_2 - t_2'| \leq 2\varepsilon$. The states r_1 and s_1 are *not* ε-bisimilar in general, but the traces originating from the two states are close and remain within 2ε.

2.4 Discounted CTL for Timed Systems

Our next step is to develop a quantitative specification formalism that assigns real numbers in the interval $[0, 1]$ to CTL formulas. A value close to 0 is "bad," a value close to 1 "good." We use time and *discounting* for this quantification. Discounting gives more weight to the near future than to the far away future. The resulting logic is called DCTL.

Syntax, Semantics, and Robustness. We look at a subset of standard boolean CTL, with $\Diamond$ being the only temporal operator. The formulae of DCTL are inductively defined as follows:

$$\varphi \; := \; a \mid \text{FALSE} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists\Diamond\varphi \mid \forall\Diamond\varphi$$

where a ranges over atomic propositions. From this, we derive the formulas: $\exists\Box\varphi = \neg\forall\Diamond\neg\varphi$ and $\forall\Box\varphi = \neg\exists\Diamond\neg\varphi$.

The semantics of DCTL formulas are given as functions from states to the real interval $[0, 1]$. For a *discount parameter* $\beta \in [0, 1]$, and a timed transition system, the value of a formula φ at a state s is defined as follows:

- $[\![a]\!](s) := 1$ if $s \models a$, 0 otherwise.

- $[\![\text{FALSE}]\!](s) := 0$.

- $[\![\neg\varphi]\!](s) := 1 - [\![\varphi]\!](s)$.

- $[\![\varphi_1 \left\{\substack{\vee \\ \wedge}\right\} \varphi_2]\!](s) := \left\{\substack{\max \\ \min}\right\}\{[\![\varphi_1]\!](s), [\![\varphi_2]\!](s)\}$.

- $[\![\left\{\substack{\exists \\ \forall}\right\}\Diamond\varphi]\!](s) := \left\{\substack{\sup \\ \inf}\right\}_{\pi_s} \sup_{t\in\mathbb{R}^+}\{\beta^t([\![\varphi]\!](\pi_s(t)))\}$.

where π_s is an infinite time diverging path starting from state s, and $\pi_s(t)$ is the state on that path at time t. Intuitively, for the $\Diamond$ operator, the quicker we can get to a good state, the better, and the discounted value reflects this fact. The temporal operators can again be seen as playing a game. Environment chooses the path π_s, and we choose the best value on that path. In $\exists\Diamond$ the environment is cooperating and chooses the best path, in $\forall\Diamond$, it plays adversially and takes the worst path. Note that $\beta = 1$ gives us the boolean case.

Example 4. *Consider T_r in Figure 2.1. Assume we cannot stay at a location forever (location invariants can ensure this). The value of $\forall\Diamond b$ at the state $\langle a, x = 0, y = 0\rangle$ is β^6. The automaton must move from a to b within 6 time units, for otherwise it will get stuck at c and not be able to take the transition to d. Similarly, the value at the starting state in T_s is β^7.*

Consider now the formula $\forall\Box(b \Rightarrow \forall\Diamond a) = \neg\exists\Diamond\neg(\neg b \vee \forall\Diamond a) = 1 - \exists\Diamond(\min(b, (1 - \forall\Diamond a)))$. What is its value at the starting state, $\langle a, 0, 0\rangle$, of T_r? The value of $\min(b, \cdot)$ is 0 at states not satisfying b, so we only need look at the b location in the outermost $\exists\Diamond$ clause. T_r needs to move out of b within 9 time units (else it will get stuck at c). Thus we need to look at states $\langle b, 0 \le x \le 9, 0 \le y \le 4\rangle$. On those states, we need the value of $\forall\Diamond a$. Suppose we enter b at time t. Then the b states encountered are $\{\langle b, t + z, z\rangle \mid z \le 4, t + z \le 9\}$. The value of $\forall\Diamond a$ at a state $\langle b, t + z, z\rangle$ is $\beta^{3+9-(t+z)}$ (we exit c at time $9 - (t + z)$, and can avoid a for 3 more time units). Thus the value of $\exists\Diamond(\min(b, (1 - \forall\Diamond a)))$ at the initial state is $\sup_{t,z}\{\beta^{t+z}(1 - \beta^{3+9-(t+z)}) \mid z \le 4, t + z \le 9\}$ (view $t + z$ as the elapsed time; the individual contributions of t and z in the sum depending on the choice of the path). The maximum value occurs when $t + z$ is 0. Thus the value of the sup is $1 - \beta^{12}$. So finally we have the value of $\forall\Box(b \Rightarrow \forall\Diamond a)$ at the starting state to be β^{12}. It turns out that the initial state in T_s has the same value for $\forall\Box(b \Rightarrow \forall\Diamond a)$. Both systems have the same "response" times for an a following a b. □

dCTL is robust with respect to ε-bisimilarity: close states in the bisimilarity metric have close dCTL values. Notice however that the closeness is not uniform and may depend on the nesting depth of temporal operators [dAFH$^+$05].

Theorem 5. *Let k be the number of nested temporal operators in a* dCTL *formula φ, and let β be a real discount factor in $[0,1]$. For all states r,s in a* TTS*, if $|\mathcal{B}(r,s,0)| \leq \varepsilon$, then $|[\![\varphi]\!](r) - [\![\varphi]\!](s)| \leq (k+1)(1-\beta^{2\varepsilon})$.*

Example 5. *Consider $\forall\Diamond b$ at the starting states (which are 1-bisimilar) in T_r, T_s in Fig. 2.1. As shown in Ex. 4, the value in T_r is β^6, and β^7 in T_s. $\beta^6 - \beta^7 = \beta^6(1-\beta) \leq 1-\beta \leq 1-\beta^2$.* □

Model Checking dCTL over Timed Automata. We compute the value of $[\![\varphi]\!](s)$ as follows: for $\varphi = \exists\Diamond\theta$, first recursively obtain $[\![\theta]\!](v)$ for each state v in the TTS. The value of $[\![\varphi]\!](s)$ is then $\sup\{\beta^{t_v}([\![\theta]\!](v))\}$, where t_v is the shortest time to reach state v from state s. For $\varphi = \forall\Diamond\theta$, we need to be a bit more careful. We cannot simply take the longest time to reach states and then have an outermost inf (i.e., dual to the $\exists\Diamond$ case). The reason is that the $\exists\Diamond$ case had $\sup_{\pi_s}\sup_t$, and both the sups can be collapsed into one. The $\forall\Diamond$ case has $\inf_{\pi_s}\sup_t$, and the actual path taken to visit a state matters. For example, it may happen that on the longest path to visit a state v, we encounter a better value of θ before v say at u; and on some other path to v, we never get to see u, and hence get the true value of the inf. The value for a formula at a state in a finite timed graph can be computed using the algorithms in [dAFH$^+$05] (with trivial modifications). Timed automata involve real time and require a different approach. We show how to compute the values for a subset of dCTL on the states of a timed automaton.

Let $F_{\min}(s,Z)$ denote the set of times that must elapse in order for a timed automaton A to hit some configuration in the set of states Z starting from the state s. Then the minimum time to reach the set Z from state s (denoted by $t_{\min}(s,Z)$) is defined to be the inf of the set $F_{\min}(s,Z)$. The maximum time to reach a set of states Z from s for the first time ($t_{\max}(s,Z)$) can be defined dually.

Theorem 6 ([CY92])**.** *(1) For a timed automaton A, the minimum and maximum times required to reach a region R from a state s for the first time ($t_{\min}(s,R), t_{\max}(s,R)$) are computable in time $O(|C|\cdot|G|)$ where C is the set of clocks in A, and G is the region automaton of A. (2) For regions R and R', either there is an integer constant d such that* *for every state $s \in R'$, we have $t_{\min}(s, R) = d$, or there is an integer constant d and a clock x such that for every state $s \in R'$, we have $t_{\min}(s, R) = d - \mathsf{frac}(t_x)$, where t_x is the value of clock x in s; and similarly for $t_{\max}(s, R)$.*

We note that for any state s, we have $[\![P]\!](s)$ is 0 or 1 for a boolean combination of propositions P, and this value is constant over a region. Thus the value of $[\![\exists\Diamond P]\!](s)$ is $\beta^{t_{\min}}$ where $t_{\min}$ is the shortest time to reach a region satisfying P from s. For computing $\forall\Diamond P$, we look at the inf-sup game where the environment chooses a path π_s, and we pick a state $\pi_s(t)$ on that path. The value of the game resulting from these choices is $\beta^t P(\pi_s(t))$. Environment is trying to minimise this value, and we are trying to maximise. Given a path, we will pick the earliest state on that path satisfying P. Thus the environment will pick paths which avoid P the longest. Hence, the value of $[\![\forall\Diamond P]\!](s)$ is $\beta^{t_{\max}}$ where $t_{\max}$ is the maximum time that can be spent avoiding regions satisfying P. The next theorem generalizes Theorem 6 to pairs of states. A state is integer (resp., rational) valued if its clock valuation maps each clock to an integer (resp., rational).

Theorem 7. *(1) Let r be an integer valued state in a timed automaton A. Then $t_{\min}(r, s)$, the minimum time to reach the state s from r is computable in time $O(|C| \cdot |G|)$ where C is the set of clocks in A, and G is the region automaton of A. (2) For a region R', either there is an integer constant d such that for every state $s \in R'$, we have $t_{\min}(r, s) = d$; or there is an integer constant d and a clock x such that $t_{\min}(r, s) = d + \mathsf{frac}(t_x)$, where t_x is the value of clock x in s.*

Theorem 7 is based on the fact that if a timed automaton can take a transition from s to s', then 1) for every state w region equivalent to s, there is a transition $w \to w'$ where w' is region equivalent to s', and 2) for every state w' region equivalent to s', there is a transition $w \to w'$ where w is region equivalent to s. If π_r is a trajectory starting from r and ending at s with minimal delay, then for any other state s' region equivalent to s, there is a corresponding minimum delay trajectory π'_r from r which makes the same transitions as π_r, in the same order, going through the same regions of the region graph (only the timings may be different). Note that an integer valued state constitutes a separate region by itself. Theorem 7 is easily generalised to rational valued initial states using the standard trick of multiplying automata guards with integers.

Theorem 8. *Let φ be a* DCTL *formula with no nested temporal operators. Then $[\![\exists\Diamond\varphi]\!](s)$ (and so $[\![\forall\Box\varphi]\!](s)$) can be computed for all rational-valued states s of a timed automaton.*

Let φ be a boolean combination of formulas of form $[\![\{\substack{\exists\\\forall}\}\Diamond P]\!]$ (P a boolean combination of propositions). We have shown $[\![\varphi]\!](s')$ to be computable for all states (and moreover it to have a simple form over regions). The value of $[\![\exists\Diamond\varphi]\!](s)$ is then $\sup\{\beta^{t_{\min}(s,s')}[\![\varphi]\!](s')\}$. The sup as s' varies over a region is easily computable, as both $[\![\varphi]\!](s')$ and $t_{\min}(s,s')$ have uniform forms over regions. We can then take a max over the regions. Let $|G|$ be the size of the region graph, $|G(Q)|$ the number of regions. Then computation of φ over all regions takes time $O(|G(Q)| \cdot |C| \cdot |G|)$. The computation of the minimum time in Theorem 7 takes $O(|C| \cdot |G| \cdot m^{|C|})$, where m is the least common multiple of the denominators of the rational clock values. Thus, the value of the formula $\exists\Diamond\varphi$ can be computed in time $O(|C|^2 \cdot |G|^2 \cdot |G(Q)| \cdot m^{|C|})$, i.e., polynomial in the size of the region graph and in $m^{|C|}$.

We can also compute the maximum time that can elapse to go from a rational valued state to any (possibly irrational valued) state, but that does not help in the computation in the $\forall\Diamond\varphi$ case, as the actual path taken is important. We can do it for the first temporal operator since then φ is a boolean combination of propositions, and either 0 or 1 on regions. In the general case φ can have some real value in $[0, 1]$, and this boolean approach does not work. Incidentally, note that its not known whether the maximum time problem between two general states is decidable. The minimum time problem *is* decidable for general states via a complicated reduction to the additive theory of real numbers [CJ99]. Whether these techniques may be used to get a model checking algorithm for DCTL is open.

Chapter 3

Timed Automaton Games

3.1 Introduction

Timed automaton games [dAFH$^+$03, AdAF05, CDF$^+$05, FTM02] are used to distinguish between the actions of several players (typically a "controller" and a "plant"). In this chapter, we shall consider two-player timed automaton games with ω-regular objectives specified as *parity conditions*. The class of ω-regular objectives can express all safety and liveness specifications that arise in the synthesis and verification of reactive systems, and parity conditions are a canonical form to express ω-regular objectives [Tho97]. The construction of a winning strategy for player 1 in such games corresponds to the *controller synthesis problem for real-time systems* [DM02, MPS95, PAMS98, WH91] with respect to achieving a desired ω-regular objective.

The issue of *time divergence* is crucial in timed games, as a naive control strategy might simply block time, leading to "zeno" runs. Such invalid solutions have often been avoided by putting strong syntactic constraints on the cycles of timed automaton games [PAMS98, AM99, FTM02, BCFL04], or by semantic conditions that discretize time [HK99]. Other works [MPS95, DM02, BDMP03, CDF$^+$05] have required that time divergence be ensured by the controller —a one-sided, unfair view in settings where the player modeling the plant is allowed to block time. We use the more general, semantic and fully symmetric formalism of [dAFH$^+$03] for dealing with the issue of time divergence. This setting places no syntactic restriction on the game structure, and gives both players equally powerful options for advancing time, but for a player to win, she must not be *responsible* for causing time to converge. We shall show that this is equivalent to requiring that the players are restricted

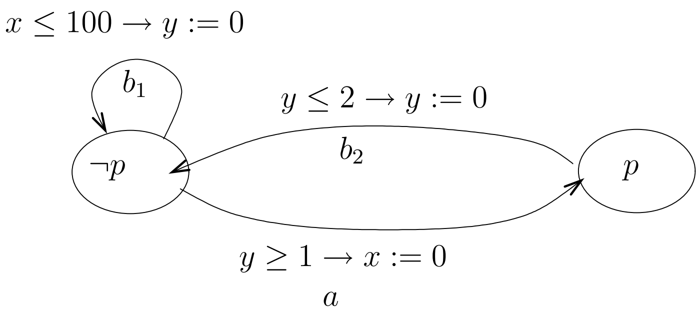

$x \leq 100 \rightarrow y := 0$
b_1
$y \leq 2 \rightarrow y := 0$
b_2
$\neg p$
p
$y \geq 1 \rightarrow x := 0$
a

Figure 3.1: A timed automaton game.

to the use of *receptive* strategies [AH97, SGSAL98], which, while being required to not prevent time from converging, are not required to ensure time divergence. More formally, our timed games proceed in an infinite sequence of rounds. In each round, both players simultaneously propose moves, with each move consisting of an action and a time delay after which the player wants the proposed action to take place. Of the two proposed moves, the move with the shorter time delay "wins" the round and determines the next state of the game. Let a set Φ of runs be the desired objective for player 1. Then player 1 has a *winning* strategy for Φ if she has a strategy to ensure that, no matter what player 2 does, one of the following two conditions hold: (1) time diverges and the resulting run belongs to Φ, or (2) time does not diverge but player-1's moves are chosen only finitely often (and thus she is not to be blamed for the convergence of time).

Example 6. *Consider the game depicted on Figure 3.1. Let edge a be controlled by player 1, the others being controlled by player 2. There are two clocks x and y, and transitions can be taken only if the clock constraints are satisfied. In addition, a transition might also reset some clocks to 0. For example, the transition labeled a has the clock constraint $y \geq 1$, and resets the clock x to 0 when taken. Suppose we want to know if player 1 can reach p starting from $\langle \neg p, x = 0, y = 0 \rangle$. Player 1 is not able to guarantee time divergence as player 2 can keep on taking edge b_1. On the other hand, we also do not want to put any restriction of the number of times player 2 takes edge b_1. The formulation of using only reasonable strategies avoids these unnecessary restrictions and correctly indicates a winning strategy for player 1 to reach p.* □

In this chapter, we present the framework of timed games and show that concurrent timed automaton parity games can be reduced to finite state turn based parity games. The reduction allows us to use the rich literature of algorithms for finite game graphs for solving timed automaton games, and also leads to algorithms with better complexity than the one presented in [dAFH$^+$03]. We note that the restriction to receptive strategies does not fundamentally change the complexity — it only increases the number of indices of the parity function by two.

Outline. In section 3.2 we first introduce timed game structures, runs, strategies in subsection 3.2.1. We then introduce objectives, timed winning conditions and receptiveness in subsection 3.2.2. Timed automaton games are presented in subsection 3.2.3. The construction of [dAFH$^+$03] for solving timed games is briefly reviewed in section 3.3. We improve the complexity by obtaining a reduction to finite state game graphs in section 3.4, from roughly $O\left((M \cdot |C| \cdot |A_1| \cdot |A_2|)^2 \cdot (16 \cdot |S_{\mathsf{Reg}}|)^{d+2}\right)$ to roughly $O\left(M \cdot |C| \cdot |A_2| \cdot (32 \cdot |S_{\mathsf{Reg}}| \cdot M \cdot |C| \cdot |A_1|)^{\frac{d+2}{3}+\frac{3}{2}}\right)$, where M is the maximum constant in the timed automaton, $|C|$ is the number of clocks, $|A_i|$ is the number of player-i edges, $|A_i|^* = \min\{|A_i|, |L| \cdot 2^{|C|}\}$, $|L|$ is the number of of locations, $|S_{\mathsf{Reg}}|$ is the number of states in the region graph (bounded by $|L| \cdot \prod_{x \in C}(c_x + 1) \cdot |C|! \cdot 2^{|C|}$), and d is the number of priorities in the parity index function.

3.2 Timed Games

3.2.1 Timed Game Structures

In this subsection we present the definitions of timed game structures, runs, and strategies of the two players.

Timed game structures. A *timed game structure* is a tuple $\mathcal{G} = \langle S, A_1, A_2, \Gamma_1, \Gamma_2, \delta \rangle$ with the following components:

- S is a set of states.
- A_1 and A_2 are two disjoint sets of actions for players 1 and 2, respectively. We assume that $\perp \notin A_i$, and write $A_i^{\perp}$ for $A_i \cup \{\perp\}$. The set of *moves* for player i is $M_i = \mathbb{R}_{\geq 0} \times A_i^{\perp}$. Intuitively, a move $\langle \Delta, a_i \rangle$ by player i indicates a waiting period of Δ time units followed by a discrete transition labeled with action a_i.
- $\Gamma_i : S \mapsto 2^{M_i} \setminus \emptyset$ are two move assignments. At every state s, the set $\Gamma_i(s)$ contains the moves that are available to player i. We require that $\langle 0, \perp \rangle \in \Gamma_i(s)$ for all states

$s \in S$ and $i \in \{1, 2\}$. Intuitively, $\langle 0, \perp \rangle$ is a time-blocking stutter move.

- $\delta : S \times (M_1 \cup M_2) \mapsto S$ is the transition function. We require that for all time delays $\Delta, \Delta' \in \mathbb{R}_{\geq 0}$ with $\Delta' \leq \Delta$, and all actions $a_i \in A_i^{\perp}$, we have (1) $\langle \Delta, a_i \rangle \in \Gamma_i(s)$ iff both $\langle \Delta', \perp \rangle \in \Gamma_i(s)$ and $\langle \Delta - \Delta', a_i \rangle \in \Gamma_i(\delta(s, \langle \Delta', \perp \rangle))$; and (2) if $\delta(s, \langle \Delta', \perp \rangle) = s'$ and $\delta(s', \langle \Delta - \Delta', a_i \rangle) = s''$, then $\delta(s, \langle \Delta, a_i \rangle) = s''$.

The game proceeds as follows. If the current state of the game is s, then both players simultaneously propose moves $\langle \Delta_1, a_1 \rangle \in \Gamma_1(s)$ and $\langle \Delta_2, a_2 \rangle \in \Gamma_2(s)$. The move with the shorter duration "wins" in determining the next state of the game. If both moves have the same duration, then one of the two moves is chosen nondeterministically. Formally, we define the *joint destination function* $\delta_{\mathsf{jd}} : S \times M_1 \times M_2 \mapsto 2^S$ by

$$\delta_{\mathsf{jd}}(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle) = \begin{cases} \{\delta(s, \langle \Delta_1, a_1 \rangle)\} & \text{if } \Delta_1 < \Delta_2; \\ \{\delta(s, \langle \Delta_2, a_2 \rangle)\} & \text{if } \Delta_2 < \Delta_1; \\ \{\delta(s, \langle \Delta_1, a_1 \rangle), \delta(s, \langle \Delta_2, a_2 \rangle)\} & \text{if } \Delta_1 = \Delta_2. \end{cases}$$

The time elapsed when the moves $m_1 = \langle \Delta_1, a_1 \rangle$ and $m_2 = \langle \Delta_2, a_2 \rangle$ are proposed is given by $\mathsf{delay}(m_1, m_2) = \min(\Delta_1, \Delta_2)$. The boolean predicate $\mathsf{blame}_i(s, m_1, m_2, s')$ indicates whether player i is "responsible" for the state change from s to s' when the moves m_1 and m_2 are proposed. Denoting the opponent of player $i \in \{1, 2\}$ by $\sim\!i = 3 - i$, we define

$$\mathsf{blame}_i(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle, s') \;=\; \big(\Delta_i \leq \Delta_{\sim i} \;\wedge\; \delta(s, \langle \Delta_i, a_i \rangle) = s'\big).$$

Runs.A *run* of the timed game structure $\mathcal{G}$ is an infinite sequence $r = s_0, \langle m_1^0, m_2^0 \rangle, s_1, \langle m_1^1, m_2^1 \rangle, \ldots$ such that $s_k \in S$ and $m_i^k \in \Gamma_i(s_k)$ and $s_{k+1} \in \delta_{\mathsf{jd}}(s_k, m_1^k, m_2^k)$ for all $k \geq 0$ and $i \in 1, 2$. For $k \geq 0$, let $\mathsf{time}(r, k)$ denote the "time" at position k of the run, namely, $\mathsf{time}(r, k) = \sum_{j=0}^{k-1} \mathsf{delay}(m_1^j, m_2^j)$ (we let $\mathsf{time}(r, 0) = 0$). By $r[k]$ we denote the $(k+1)$-th state s_k of r. The run prefix $r[0..k]$ is the finite prefix of the run r that ends in the state s_k; we write $\mathsf{last}(r[0..k])$ for the ending state s_k of the run prefix. Let Runs be the set of all runs of $\mathcal{G}$, and let $\mathsf{FinRuns}$ be the set of run prefixes.

Strategies. A *strategy* π_i for player $i \in \{1, 2\}$ is a function $\pi_i : \mathsf{FinRuns} \mapsto M_i$ that assigns to every run prefix $r[0..k]$ a move to be proposed by player i at the state $\mathsf{last}(r[0..k])$ if the history of the game is $r[0..k]$. We require that $\pi_i(r[0..k]) \in \Gamma_i(\mathsf{last}(r[0..k]))$ for every run prefix $r[0..k]$, so that strategies propose only available moves. The results of this paper are equally valid if strategies do not depend on past moves chosen by the players, but only on the past sequence of states and time delays [dAFH$^+$03]. For $i \in \{1, 2\}$, let Π_i be the set of player-i strategies. Given two strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$, the set of possible *outcomes* of the game starting from a state $s \in S$ is denoted $\mathsf{Outcomes}(s, \pi_1, \pi_2)$: it contains all runs $r = s_0, \langle m_1^0, m_2^0\rangle, s_1, \langle m_1^1, m_2^1\rangle, \ldots$ such that $s_0 = s$ and for all $k \geq 0$ and $i \in \{1, 2\}$, we have $\pi_i(r[0..k]) = m_i^k$.

3.2.2 Timed Winning Conditions

In this subsection we present the definitions of parity objectives, receptiveness, and winning conditions which account for the fact that players might not block time to achieve their objectives.

Objectives. An *objective* for the timed game structure $\mathcal{G}$ is a set $\Phi \subseteq \mathsf{Runs}$ of runs. We will be interested in the classical reachability, safety and parity objectives. Parity objectives are canonical forms for ω-regular properties that can express all commonly used specifications that arise in verification.

- Given a set of states Y, the *reachability* objective $\mathsf{Reach}(Y)$ is defined as the set of runs that visit Y, formally, $\mathsf{Reach}(Y) = \{r \mid \text{ there exists } i \text{ such that } r[i] \in Y\}$.

- Given a set of states Y, the *safety* objective consists of the set of runs that stay within Y, formally, $\mathsf{Safe}(Y) = \{r \mid \text{ for all } i \text{ we have } r[i] \in Y\}$.

- Let $\Omega : S \mapsto \{0, \ldots, k-1\}$ be a parity index function. The parity objective for Ω requires that the maximal index visited infinitely often is even. Formally, let $\mathsf{InfOften}(\Omega(r))$ denote the set of indices visited infinitely often along a run r. Then the parity objective defines the following set of runs: $\mathsf{Parity}(\Omega) = \{r \mid \max(\mathsf{InfOften}(\Omega(r))) \text{ is even }\}$.

A timed game structure $\mathcal{G}$ together with the index function Ω constitute a *parity timed game* (of *order* k) in which the objective of player 1 is $\mathsf{Parity}(\Omega)$. We use similar notations for reachability and safety timed games.

Timed winning conditions. To win an objective Φ, a player must ensure that the possible outcomes of the game satisfy the *winning condition* $\mathsf{WC}(\Phi)$, a different subset of Runs. We distinguish between objectives and winning conditions, because players must win their objectives using only physically meaningful strategies; for example, a player should not satisfy the objective of staying in a safe set by blocking time forever. Formally, player $i \in \{1, 2\}$ *wins* for the objective Φ at a state $s \in S$ if there is a player-i strategy π_i such that for all opposing strategies $\pi_{\sim i}$, we have $\mathsf{Outcomes}(s, \pi_1, \pi_2) \subseteq \mathsf{WC}_i(\Phi)$. In this case, we say that player i has the *winning strategy* π_i. The winning condition is formally defined as

$$\mathsf{WC}_i(\Phi) \;=\; (\mathsf{Timediv} \cap \Phi) \;\cup\; (\mathsf{Blameless}_i \setminus \mathsf{Timediv}),$$

which uses the following two sets of runs:

- $\mathsf{Timediv} \subseteq \mathsf{Runs}$ is the set of all time-divergent runs. A run r is *time-divergent* if $\lim_{k\to\infty} \mathsf{time}(r, k) = \infty$.
- $\mathsf{Blameless}_i \subseteq \mathsf{Runs}$ is the set of runs in which player i is responsible only for finitely many transitions. A run $s_0, \langle m_1^0, m_2^0\rangle, s_1, \langle m_1^1, m_2^1\rangle, \ldots$ belongs to the set $\mathsf{Blameless}_i$, for $i = \{1, 2\}$, if there exists a $k \geq 0$ such that for all $j \geq k$, we have $\neg\, \mathsf{blame}_i(s_j, m_1^j, m_2^j, s_{j+1})$.

Thus a run r belongs to $\mathsf{WC}_i(\Phi)$ if and only if the following conditions hold:

- if $r \in \mathsf{Timediv}$, then $r \in \Phi$;
- if $r \notin \mathsf{Timediv}$, then $r \in \mathsf{Blameless}_i$.

Informally, if time diverges, then the outcome of the game is valid and the objective must be met, and if time does not diverge, then only the opponent should be responsible for preventing time from diverging.

A state $s \in S$ in a timed game structure $\mathcal{G}$ is *well-formed* if both players can win at s for the trivial objective Runs. The timed game structure $\mathcal{G}$ is *well-formed* if all states of $\mathcal{G}$ are well-formed. Structures that are not well-formed are not physically meaningful. We restrict out attention to well-formed timed game structures.

Receptive strategies. A strategy π_i for player $i \in \{1, 2\}$ is *receptive* if for all opposing strategies $\pi_{\sim i}$, all states $s \in S$, and all runs $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)$, either $r \in \mathsf{Timediv}$ or $r \in \mathsf{Blameless}_i$. Thus, no what matter what the opponent does, a receptive player-i strategy should not be responsible for blocking time. Strategies that are not receptive are not physically meaningful. A timed game structure is thus well-formed iff both players have receptive strategies. We now show in Theorem 9 that we can restrict our attention to games which allow only receptive strategies. We first need the following lemma.

Lemma 5. *Consider a timed game structure $\mathcal{G}$ and a state $s \in S$. Let $\pi_1 \in \Pi_1^R$ and $\pi_2^R \in \Pi_2^R$ be player-1 and player-2 receptive strategies, and let $\pi_2 \in \Pi_2$ be any player-2 strategy such that $\mathsf{Outcomes}(s, \pi_1, \pi_2) \cap \mathsf{Timediv} \neq \emptyset$. Let $r^* \in \mathsf{Outcomes}(s, \pi_1, \pi_2) \cap \mathsf{Timediv}$. Consider a player-2 strategy π_2^* be defined as, $\pi_2^*(r[0..k]) = \pi_2(r^*[0..k])$ for all run prefixes $r[0..k]$ of r^*, and $\pi_2^*(r[0..k]) = \pi_2^R(r[k'..k])$ otherwise, where k' is the first position such that $r[0..k']$ is not a run prefix of r^*. Then, π_2^* is a receptive strategy.*

Proof. Intuitively, the strategy π_2^* acts like π_2 on r^* , and like π_2^R otherwise. Consider any player-1 strategy $\pi_1' \in \Pi_1$, and any run $r \in \mathsf{Outcomes}(s, \pi_1', \pi_2^*)$. If $r = r^*$, then $r \in \mathsf{Timediv}$. Suppose $r \neq r^*$. Let $k' \geq 0$ be the first step in the game (with player-2 strategy π_2^*) which witnesses the fact that $r \neq r^*$, that is, 1) we have $r[0..k'-1]$ to be a run prefix of r^*, and 2) $r[0..k']$ to not be a run prefix of r^* Consider the state $s_{k'} = r[k']$. After this point (ie., from $r[0..k']$ onwards), the strategy π_2^* behaves like π_2^R when "started" from $s_{k'}$. Since π_2^R is a receptive player-2 strategy, we have $\mathsf{Outcomes}(s_{k'}, \pi_1', \pi_2^*) \subseteq \mathsf{Timediv} \cup \mathsf{Blameless}_2$. Thus, $r \in \mathsf{Timediv} \cup \mathsf{Blameless}_2$ (finite prefixes of runs do not change membership in these sets). Hence π_2^* is a receptive player-2 strategy. □

Theorem 9. *Let $s \in S$ be a state of a well-formed time game structure $\mathcal{G}$, and let $\Phi \subseteq \mathsf{Runs}$ be an objective.*

1. *Player 1 wins for the objective Φ at the state s iff there is a* receptive *player-1 winning strategy π_1^*, that is, for all player-2 strategies π_2, we have $\mathsf{Outcomes}(s, \pi_1^*, \pi_2) \subseteq \mathsf{WC}(\Phi)$.*

2. *Player 1 does not win for the objective Φ at s using only receptive strategies iff there is a receptive player-2 spoiling strategy π_2^*. Formally, for every receptive player-1 strategy π_1^*, there is a player-2 strategy π_2 such that $\mathsf{Outcomes}(s, \pi_1^*, \pi_2) \not\subseteq \mathsf{WC}(\Phi)$ iff there is a* receptive *player-2 strategy π_2^* such that $\mathsf{Outcomes}(s, \pi_1^*, \pi_2^*) \not\subseteq \mathsf{WC}(\Phi)$.*

The symmetric claims with players 1 and 2 interchanged also hold.

Proof. (1) Let π_1 be the winning strategy for player 1 for objective Φ at state s. Let π_1 be not receptive. Then, by definition, there exists an opposing strategy π_2 such that for some run $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)$, we have both $r \notin \mathsf{Timediv}$ and $r \notin \mathsf{Blameless}_1$. This contradicts the fact that π_1 was a winning strategy.

(2) Let π_1^* be any player-1 receptive strategy. Player 1 loses for the objective Φ from state s, thus there exists a player 2 spoiling strategy π_2 such that $\mathsf{Outcomes}(s, \pi_1^*, \pi_2) \not\subseteq \mathsf{WC}(\Phi)$.

This requires that for some run $r \in \mathsf{Outcomes}(s, \pi_1^*, \pi_2)$, we have either 1) $(r \in \mathsf{Timediv}) \wedge (r \notin \Phi)$ or 2) $(r \notin \mathsf{Timediv}) \wedge (r \notin \mathsf{Blameless}_1)$. We cannot have the second case, for π_1^* is a receptive strategy, thus, the first case must hold. By definition, for every state s' in a well-formed time game structure, there exists a player-2 receptive strategy $\pi_2^{s'}$. Now, let π_2^* be such that its acts like π_2 on the particular run r, and is like π_2^s otherwise, that is $\pi_2^*(r_f) = \pi_2(r_f)$, for all run prefixes r_f of r, and $\pi_2^*(r_f) = \pi_2^s(r_f)$ otherwise. The strategy π_2^* is receptive, as for all strategies π_1', and for every run $r' \in \mathsf{Outcomes}(s, \pi_1', \pi_2^*)$, we have $(r' \in \mathsf{Timediv}) \vee (r' \in \mathsf{Blameless}_2)$. Since π_2^* acts like π_2 on the particular run r, it is also spoiling for the player-1 strategy π_1^*. □

Corollary 1. *For $i = \{1, 2\}$, let $\mathsf{Win}_i(\Phi)$ be the states of a well-formed timed game structure $\mathcal{G}$ at which player i can win for the objective Φ for the winning condition $\mathsf{WC}_i(\Phi)$. Let $\mathsf{Win}_i^*(\Phi)$ be the states at which player i can win for the objective Φ when both players are restricted to use receptive strategies. Then, $\mathsf{Win}_i(\Phi) = \mathsf{Win}_i^*(\Phi)$.*

Note that if π_1^* and π_2^* are player-1 and player-2 receptive strategies, then for every state s and every run $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)$, the run r is non-zeno. Thus, if we restrict our attention to plays in which both players use only receptive strategies, then for every objective Φ, player i wins for the winning condition $\mathsf{WC}(\Phi)$ if and only if she wins for the winning condition Φ. We can hence talk *semantically* about games restricted to receptive player strategies in well-formed timed game structures, without differentiating between objectives and winning conditions. From a *computational* perspective, we allow all strategies, taking care to distinguish between objectives and winning conditions. Theorem 9 indicates both approaches to be equivalent.

3.2.3 Timed Automaton Games

In this section we present timed automaton games which are based on timed automata [AD94] and which give a finite syntax for specifying infinite-state timed game structures.

Timed automaton games. A *timed automaton game* is a tuple $\mathcal{T} = \langle L, \Sigma, \sigma, C, A_1, A_2, E, \gamma \rangle$ with the following components:

- L is a finite set of locations.
- C is a finite set of clocks.

- A_1 and A_2 are two disjoint sets of actions for players 1 and 2, respectively.
- $E \subseteq L \times (A_1 \cup A_2) \times \mathsf{Constr}(C) \times L \times 2^C$ is the edge relation, where the set $\mathsf{Constr}(C)$ of *clock constraints* is generated by the grammar

  $$\theta ::= x \leq d \mid d \leq x \mid \neg\theta \mid \theta_1 \wedge \theta_2$$

  for clock variables $x \in C$ and nonnegative integer constants d. For an edge $e = \langle l, a_i, \theta, l', \lambda \rangle$, the clock constraint θ acts as a guard on the clock values which specifies when the edge e can be taken, and by taking the edge e, the clocks in the set $\lambda \subseteq C \backslash \{z\}$ are reset to 0. We require that for all edges $\langle l, a_i, \theta', l', \lambda' \rangle \neq \langle l, a_i, \theta'', l'', \lambda'' \rangle \in E$, we have $a_i \neq a_i'$. This requirement ensures that a state and a move together uniquely determine a successor state.
- $\gamma : L \mapsto \mathsf{Constr}(C)$ is a function that assigns to every location an invariant for both players. All clocks increase uniformly at the same rate. When at location l, each player i must propose a move out of l before the invariant $\gamma(l)$ expires. Thus, the game can stay at a location only as long as the invariant is satisfied by the clock values.

A *clock valuation* is a function $\kappa : C \mapsto \mathbb{R}_{\geq 0}$ that maps every clock to a nonnegative real. The set of all clock valuations for C is denoted by $K(C)$. Given a clock valuation $\kappa \in K(C)$ and a time delay $\Delta \in \mathbb{R}_{\geq 0}$, we write $\kappa + \Delta$ for the clock valuation in $K(C)$ defined by $(\kappa + \Delta)(x) = \kappa(x) + \Delta$ for all clocks $x \in C$. For a subset $\lambda \subseteq C$ of the clocks, we write $\kappa[\lambda := 0]$ for the clock valuation in $K(C)$ defined by $(\kappa[\lambda := 0])(x) = 0$ if $x \in \lambda$, and $(\kappa[\lambda := 0])(x) = \kappa(x)$ if $x \notin \lambda$. A clock valuation $\kappa \in K(C)$ *satisfies* the clock constraint $\theta \in \mathsf{Constr}(C)$, written $\kappa \models \theta$, if the condition θ holds when all clocks in C take on the values specified by κ.

A *state* $s = \langle l, \kappa \rangle$ of the timed automaton game $\mathcal{T}$ is a location $l \in L$ together with a clock valuation $\kappa \in K(C)$ such that the invariant at the location is satisfied, that is, $\kappa \models \gamma(l)$. Let S be the set of all states of $\mathcal{T}$. In a state, each player i proposes a time delay allowed by the invariant map γ, together either with the action $\perp$, or with an action $a_i \in A_i$ such that an edge labeled a_i is enabled after the proposed time delay. We require that for $i \in \{1, 2\}$ and for all states $s = \langle l, \kappa \rangle$, if $\kappa \models \gamma(l)$, either $\kappa + \Delta \models \gamma(l)$ for all $\Delta \in \mathbb{R}_{\geq 0}$, or there exist a time delay $\Delta \in \mathbb{R}_{\geq 0}$ and an edge $\langle l, a_i, \theta, l', \lambda \rangle \in E$ such that (1) $a_i \in A_i$ and

(2) $\kappa+\Delta \models \theta$ and for all $0 \leq \Delta' \leq \Delta$, we have $\kappa+\Delta' \models \gamma(l)$, and (3) $(\kappa+\Delta)[\lambda := 0] \models \gamma(l')$. This requirement is necessary (but not sufficient) for well-formedness of the game.

The timed automaton game $\mathcal{T}$ defines the following timed game structure $[\![\mathcal{T}]\!] = \langle S, A_1, A_2, \Gamma_1, \Gamma_2, \delta\rangle$:

- S is defined above.
- For $i \in \{1, 2\}$, the set $\Gamma_i(\langle l, \kappa\rangle)$ contains the following elements:
  1. $\langle \Delta, \bot\rangle$ if for all $0 \leq \Delta' \leq \Delta$, we have $\kappa + \Delta' \models \gamma(l)$.
  2. $\langle \Delta, a_i\rangle$ if for all $0 \leq \Delta' \leq \Delta$, we have $\kappa + \Delta' \models \gamma(l)$, and $a_i \in A_i$, and there exists an edge $\langle l, a_i, \theta, l', \lambda\rangle \in E$ such that $\kappa + \Delta \models \theta$.
- $\delta(\langle l, \kappa\rangle, \langle \Delta, \bot\rangle) = \langle l, \kappa+\Delta\rangle$, and $\delta(\langle l, \kappa\rangle, \langle \Delta, a_i\rangle) = \langle l', (\kappa+\Delta)[\lambda := 0]\rangle$ for the unique edge $\langle l, a_i, \theta, l', \lambda\rangle \in E$ with $\kappa + \Delta \models \theta$.

The timed game structure $[\![\mathcal{T}]\!]$ is not necessarily well-formed, because it may contain cycles along which time cannot diverge. We will see below how we can check well-formedness for timed automaton games.

Clock region equivalence. Timed automaton games can be solved using the region construction from the theory of timed automata [AD94], see subsection 2.2.2 of Chapter 2 for the definition of regions. For a state $s \in S$, we write $\mathsf{Reg}(s) \subseteq S$ for the clock region containing s. For a run r, we let the *region sequence* $\mathsf{Reg}(r) = \mathsf{Reg}(r[0]), \mathsf{Reg}(r[1]), \cdots$. Two runs r, r' are region equivalent if their region sequences are the same. An ω-regular objective Φ is a region objective if for all region-equivalent runs r, r', we have $r \in \Phi$ iff $r' \in \Phi$. A strategy π_1 is a *region strategy*, if for all runs r_1 and r_2 and all $k \geq 0$ such that $\mathsf{Reg}(r_1[0..k]) = \mathsf{Reg}(r_2[0..k])$, we have that if $\pi_1(r_1[0..k]) = \langle \Delta, a_1\rangle$, then $\pi_1(r_2[0..k]) = \langle \Delta', a_1\rangle$ with $\mathsf{Reg}(r_1[k] + \Delta) = \mathsf{Reg}(r_2[k] + \Delta')$. The definition for player 2 strategies is analogous. Two region strategies π_1 and π_1' are region-equivalent if for all runs r and all $k \geq 0$ we have that if $\pi_1(r[0..k]) = \langle \Delta, a_1\rangle$, then $\pi_1'(r[0..k]) = \langle \Delta', a_1\rangle$ with $\mathsf{Reg}(r_1[k] + \Delta) = \mathsf{Reg}(r_2[k] + \Delta')$. A parity index function Ω is a region (resp. location) parity index function if $\Omega(s_1) = \Omega(s_2)$ whenever $\mathsf{Reg}(s_1) = \mathsf{Reg}(s_2)$ (resp. s_1, s_2 have the same location). Henceforth, we shall restrict our attention to region and location objectives.

3.3 Solving Timed Automaton Games

In this section we review the μ-calculus formulation for solving timed automaton games as presented in [dAFH$^+$03]. This formulation will be used in the next section to reduce timed automaton games to finite state turn based parity games. We first show how to encode $\mathsf{Timediv}$ and $\mathsf{Blameless}_i$ in terms of observable of the system.

Encoding Time-Divergence by Enlarging the Game Structure. Given a timed automaton game $\mathcal{T}$, consider the enlarged game structure $\widehat{\mathcal{T}}$ with the state space $\widehat{S} \subseteq S \times \mathbb{R}_{[0,1)} \times \{\text{TRUE}, \text{FALSE}\}^2$, and an augmented transition relation $\widehat{\delta} : \widehat{S} \times (M_1 \cup M_2) \mapsto \widehat{S}$. In an augmented state $\langle s, \mathfrak{z}, \mathit{tick}, \mathit{bl}_1\rangle \in \widehat{S}$, the component $s \in S$ is a state of the original game structure $[\![\mathcal{T}]\!]$, $\mathfrak{z}$ is value of a fictitious clock z which gets reset to 0 every time it hits 1, tick is true iff z hit 1 at last transition and bl_1 is true if player 1 is to blame for the last transition. Note that any strategy π_i in $[\![\mathcal{T}]\!]$, can be considered a strategy in $\widehat{\mathcal{T}}$. The values of the clock z, tick and bl_1 correspond to the values each player keeps in memory in constructing his strategy. Any run r in $\mathcal{T}$ has a corresponding unique run $\widehat{r}$ in $\widehat{\mathcal{T}}$ with $\widehat{r}[0] = \langle r[0], 0, \text{FALSE}, \text{FALSE}\rangle$ such that r is a projection of $\widehat{r}$ onto $\mathcal{T}$. For an objective Φ, we can now encode time-divergence as: $\mathsf{TimeDivBl}_1(\Phi) = (\Box\Diamond\, \mathit{tick} \rightarrow \Phi) \;\wedge\; (\neg\Box\Diamond\, \mathit{tick} \rightarrow \Diamond\Box\neg\, \mathit{bl}_1)$. Let $\widehat{\kappa}$ be a valuation for the clocks in $\widehat{C} = C \cup \{z\}$. A state of $\widehat{\mathcal{T}}$ can then be considered as $\langle\langle l, \widehat{\kappa}\rangle, \mathit{tick}, \mathit{bl}_1\rangle$. We extend the clock equivalence relation to these expanded states: $\langle\langle l, \widehat{\kappa}\rangle\, \mathit{tick}, \mathit{bl}_1\rangle \cong \langle\langle l', \widehat{\kappa}'\rangle, \mathit{tick}', \mathit{bl}_1'\rangle$ iff $l = l', \mathit{tick} = \mathit{tick}', \mathit{bl}_1 = \mathit{bl}_1'$ and $\widehat{\kappa} \cong \widehat{\kappa}'$ (we let $c_{\mathfrak{z}} = 1$). Given a location l, and a set $\lambda \subseteq \widehat{C}$, we let $\widehat{R}[\mathsf{loc} := l, \lambda := 0]$ denote the region $\{\langle l, \widehat{\kappa}\rangle \in \widehat{S} \mid$ there exist l' and $\widehat{\kappa}'$ with $\langle l', \widehat{\kappa}'\rangle \in \widehat{R}$ and $\widehat{\kappa}(x) = 0$ if $x \in \lambda, \widehat{\kappa}(x) = \widehat{\kappa}'(x)$ if $x \notin \lambda\}$. For every ω-regular region objective Φ of $\mathcal{T}$, we have $\mathsf{TimeDivBl}(\Phi)$ to be an ω-regular region objective of $\widehat{\mathcal{T}}$.

We first note the classical result of [AD94] that the region equivalence relation induces a time abstract bisimulation on the regions.

Lemma 6 ([AD94]). *Let $\mathcal{T}$ be a timed automaton game and let $\widehat{Y}, \widehat{Y}'$ be regions in the enlarged timed game structure $\widehat{\mathcal{T}}$. Suppose player i has a move from $\widehat{s}_1 \in \widehat{Y}$ to $\widehat{s}_1' \in \widehat{Y}'$, for $i \in \{1, 2\}$. Then, for any $s_2 \in \widehat{Y}$, player i has a move from $\widehat{s}_2$ to some $\widehat{s}_2' \in \widehat{Y}'$.*

Let $\widehat{Y}, \widehat{Y}_1', \widehat{Y}_2'$ be regions of $\widehat{\mathcal{T}}$. We next prove in Lemma 7 that one of the following two conditions hold: (a) for all states in $\widehat{Y}$ there is a move for player 1 with destination in $\widehat{Y}_1'$, such that for all player 2 moves with destination in $\widehat{Y}_2'$, the next state is in $\widehat{Y}_1'$; or (b) for all states in $\widehat{Y}$ for all moves for player 1 with destination in $\widehat{Y}_1'$ there is a move of player 2 to ensure that the next state is in $\widehat{Y}_2'$.

Lemma 7. *Let $\mathcal{T}$ be a timed automaton game and let $\widehat{Y}, \widehat{Y}_1', \widehat{Y}_2'$ be regions in the enlarged timed game structure $\widehat{\mathcal{T}}$. Suppose player i has a pure-time move from $\widehat{s}_1 \in \widehat{Y}$ to $\widehat{s}_1' \in \widehat{Y}_i'$, for $i \in \{1, 2\}$. Then, one of the following cases must hold:*

1. *From all states $\widehat{s} \in \widehat{Y}$, for every player-1 pure-time move $m_1^{\widehat{s}}$ with $\delta(s, m_1^{\widehat{s}}) \in Y_1'$, for all pure-time moves $m_2^{\widehat{s}}$ of player 2 with $\delta(\widehat{s}, m_2^{\widehat{s}}) \in \widehat{Y}_2'$, we have $\mathsf{blame}_1(\widehat{s}, m_1^{\widehat{s}}, m_2^{\widehat{s}}, \delta(\widehat{s}, m_1^{\widehat{s}})) = \text{TRUE}$ and $\mathsf{blame}_2(\widehat{s}, m_1^{\widehat{s}}, m_2^{\widehat{s}}, \delta(\widehat{s}, m_2^{\widehat{s}})) = \text{FALSE}$.*

2. *From all states $\widehat{s} \in \widehat{Y}$, for every player-1 pure-time move $m_1^{\widehat{s}}$ with $\delta(\widehat{s}, m_1^{\widehat{s}}) \in \widehat{Y}_1'$, there exists a pure-time moves $m_2^{\widehat{s}}$ of player 2 with $\delta(\widehat{s}, m_2^{\widehat{s}}) \in \widehat{Y}_2'$, such that $\mathsf{blame}_2(\widehat{s}, m_1^{\widehat{s}}, m_2^{\widehat{s}}, \delta(\widehat{s}, m_2^{\widehat{s}})) = \text{TRUE}$.*

Proof. We first present the proof for the case when $\widehat{Y}_1' \neq \widehat{Y}_2'$. The proof follows from the fact that each region has a unique first *time-successor* region. A region $\widehat{R}'$ is a first time-successor of $\widehat{R} \neq \widehat{R}'$ if for all states $\widehat{s} \in \widehat{R}$, there exists $\Delta > 0$ such that $s + \Delta \in \widehat{R}'$ and for all $\Delta' < \Delta$, we have $s + \Delta' \in \widehat{R} \cup \widehat{R}'$. The time-successor of $\langle l, h, \mathcal{P}(\widehat{C}) \rangle$ is $\langle l, h', \mathcal{P}'(\widehat{C}) \rangle$ when (recall that $c_z = 1$, and that the clock z cycles from 0 to 1, but it never has the value 1):

- $h = h'$, $\mathcal{P}(\widehat{C}) = \langle C_{-1}, C_0 \neq \emptyset, C_1, \ldots, C_n \rangle$, and $\mathcal{P}'(\widehat{C}) = \langle C_{-1}, C_0' = \emptyset, C_1', \ldots, C_{n+1}' \rangle$ where $C_i' = C_{i-1}$, and $h(x) < c_x$ for every $x \in C_0$.

- $h = h'$, $\mathcal{P}(\widehat{C}) = \langle C_{-1}, C_0 \neq \emptyset, C_1, \ldots, C_n \rangle$, and $\mathcal{P}'(\widehat{C}) = \langle C_{-1}' = C_{-1} \cup C_0, C_0' = \emptyset, C_1, \ldots, C_n \rangle$, and $h(x) \geq c_x$ for every $x \in C_0$.

- $h = h'$, $\mathcal{P}(\widehat{C}) = \langle C_{-1}, C_0 \neq \emptyset, C_1, \ldots, C_n \rangle$, and $\mathcal{P}'(\widehat{C}) = \langle C_{-1}', C_0' = \emptyset, C_1', \ldots, C_{n+1}' \rangle$ where $C_i' = C_{i-1}$ for $i \geq 2$, $h(x) < c_x$ for every $x \in C_1' \subseteq C_0$, and $h(x) \geq c_x$ for every $x \in C_0 \setminus C_1'$, and $C_{-1}' = C_{-1} \cup C_0 \setminus C_1'$.

- $\mathcal{P}(\widehat{C}) = \langle C_{-1}, C_0 = \emptyset, C_1, \ldots, C_n \rangle$, $\mathcal{P}'(\widehat{C}) = \langle C_{-1}, C_0' = C_n, C_1, \ldots, C_{n-1} \rangle$, and $h'(x) = h(x) + 1 \leq c_x$ for every $x \in C_n \setminus \{z\}$, and $h'(x) = h(x)$ otherwise.

- $\mathcal{P}(\widehat{C}) = \langle C_{-1}, C_0 = \emptyset, C_1, \ldots, C_n \rangle$, $\mathcal{P}'(\widehat{C}) = \langle C_{-1}' = C_{-1} \cup C_n, C_0, C_1, \ldots, C_{n-1} \rangle$, and $h'(x) = h(x) = c_x$ for every $x \in C_n$, and $h'(x) = h(x)$ otherwise.

- $\mathcal{P}(\widehat{C}) = \langle C_{-1}, C_0 = \emptyset, C_1, \ldots, C_n \rangle$, $\mathcal{P}'(\widehat{C}) = \langle C'_{-1} = C_{-1} \cup C_n \setminus C'_0, C'_0, C_1, \ldots, C_{n-1} \rangle$, and $h'(x) = h(x) + 1 \leq c_x$ for every $x \neq z \in C'_0 \subseteq C_n$, $h'(x) = h(x) = c_x$ for every $x \neq z \in C_n \setminus C'_0$, and $h'(x) = h(x)$ otherwise.

In case $\widehat{Y}'_1 = \widehat{Y}'_2$, then player 2 can pick the same time to elapse as player 1, and ensure that the conditions of the lemma hold. □

Note that the lemma is asymmetric, the asymmetry arises in the case when time delays of the two moves result in the same region. In this case, not all moves of player 2 might work, but some will (e.g., a delay of player 2 that is the same as that for player 1).

Given a parity objective Φ and the corresponding winning condition $\mathsf{TimeDivBl}_1(\Phi)$, the winning set for player 1 can be expressed as the fixpoint of a μ-calculus expression [dAHM01a]. The μ-calculus expression uses *controllable predecessor* operator for player 1, $\mathsf{CPre}_1 : 2^{\widehat{S}} \mapsto 2^{\widehat{S}}$, defined formally by $\widetilde{s} \in \mathsf{CPre}_1(Z)$ iff $\exists m_1 \in \widehat{\Gamma}_1(\widehat{s})\ \forall m_2 \in \widehat{\Gamma}_2(\widehat{s}) \,.\, \widehat{\delta}_{\mathsf{jd}}(\widehat{s}, m_1, m_2) \subseteq Z$. Informally, $\mathsf{CPre}_1(Z)$ consists of the set of states from which player 1 can ensure that the next state will be in Z, no matter what player 2 does. Fox example, the μ-calculus expression for the reachability objective can be expressed as: $\mu Y \nu X \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X)) \right]$.

Lemma 8. *Let $X \subseteq \widehat{S}$ consist of a union of extended regions in a timed game structure $\widehat{[\![\mathcal{T}]\!]}$. Then $\mathsf{CPre}_1(X)$ is again a union of extended regions.*

Proof. Follows from Lemmas 6 and 7 □

Lemma 8 demonstrates that the sets in the fixpoint computation of the μ-calculus iteration always consist of unions of regions of $\widehat{\mathcal{T}}$. Since the number of regions is finite, the termination of the fixpoint iteration follows.

Theorem 10. *Let $\mathcal{T}$ be a timed automaton game and let Φ be an ω-regular region objective of order d. Then the set of states from which player-i can win for Φ can be computed in time $O\left((M \cdot |C| \cdot |A_1| \cdot |A_2|)^2 \cdot (16 \cdot |S_{\mathsf{Reg}}|)^{d+2} \right)$.*

Corollary 2. *The problem of solving a timed automaton game with a parity region objective is* EXPTIME*-complete.*

EXPTIME-hardness follows from the EXPTIME-hardness of alternating reachability on timed automata [HK99].

Solving timed automaton games with ω-regular objectives allows us to check the well-formedness of a timed automaton game $\mathcal{T}$: a state s of the timed automaton game $\mathcal{T}$ is well formed iff both players can win for the objective L^ω. This well-formedness check is the generalization to the game setting of the non-zenoness check for timed automata, which computes the states s such that there exists a time divergent run from s [HNSY94]. If not all states of $\mathcal{T}$ are well-formed, then the location invariants of $\mathcal{T}$ can be strengthened to characterize well-formed states (note that the set of well-formed states consists of a union of regions).

It also follows from Lemmas 6 and 7, the moves of player 1 can always prescribe moves to the same $\widehat{R}'$ from every state of a region $\widehat{R}$. Hence we have the following result.

Lemma 9. *Let $\mathcal{T}$ be a timed automaton game and $\widehat{\mathcal{T}}$ be the corresponding enlarged game structure. Let $\widehat{\Phi}$ be an ω-regular region objective of $\widehat{\mathcal{T}}$. Then the following assertions hold.*

1. *Let π_1 be a region strategy that is winning for $\widehat{\Phi}$ from $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$ and π_1' is a strategy that is region-equivalent to π_1. Then π_1' is a winning strategy for $\widehat{\Phi}$ from $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$.*

2. *Let player 1 have a winning strategy for $\widehat{\Phi}$ in $\widehat{\mathcal{T}}$. Then, player 1 has a finite memory region strategy that is winning.*

Finite memory suffices as player 1 only needs to remember a finite region history; as the regions she proposes can be obtained from the μ-calculus fixpoint iteration algorithm as in [dAHM01b].

3.4 Efficient Solution of Timed Automaton Games

In this section we shall present a reduction of timed automaton games to finite game graphs. The reduction allows us to use the rich literature of algorithms for finite game graphs for solving timed automaton games. It also leads to algorithms with better complexity than the one presented in [dAFH$^+$03]. Let $\mathcal{T}$ be a timed automaton game, and let $\widehat{\mathcal{T}}$ be the corresponding enlarged timed game structure that encodes time divergence. We shall construct a finite state turn based game structure $\mathcal{T}^f$ based on regions of $\widehat{\mathcal{T}}$ which can be used to compute winning states for ω-regular objectives for the timed automaton game $\mathcal{T}$. In this finite state game, first player 1 proposes a destination region $\widehat{R}_1$ together with a discrete action a_1. Intuitively, this can be taken to mean that in the game $\widehat{\mathcal{T}}$, player 1 wants to first let time elapse to get to to the region $\widehat{R}_1$, and then take the discrete action a_1. Let us denote the intermediate state in $\mathcal{T}^f$ by the tuple $\langle \widehat{R}, \widehat{R}_1, a_1 \rangle$. From this state in $\mathcal{T}^f$, player 2 similarly also proposes a move consisting of a region $\widehat{R}_2$ together with a discrete action a_2. These two moves signify that player i proposed a move $\langle \Delta_i, a_i \rangle$ in $\widehat{\mathcal{T}}$ from a state $\widehat{s} \in \widehat{R}$ such that $\widehat{s} + \Delta_i \in \widehat{R}_i$. Lemma 7 indicates that only the regions of $\widehat{s} + \Delta_i$ are important in determining the successor region in $\widehat{\mathcal{T}}$.

Let $\widehat{S}_{\mathsf{Reg}} = \{x \mid X \text{ is a region of } \widehat{\mathcal{T}}\}$. The state space of the finite turn based game will then be $O(|\widehat{S}_{\mathsf{Reg}}|^2 \cdot |L| \cdot 2^{|C|})$ (a discrete action may switch the location, and reset some clocks). We show that it is not required to keep all possible pairs of regions, leading to a reduction in the size of the state space. This is because from a state $\widehat{s} \in R$, it is not possible to get all regions by letting time elapse.

Lemma 10. *Let $\mathcal{T}$ be a timed automaton game, $\widehat{\mathcal{T}}$ the corresponding enlarged game structure, and $\widehat{R} = \langle l_1, tick, bl_1, h, \langle C_{-1}, C_0, \ldots, C_n \rangle \rangle$ a region in $\widehat{\mathcal{T}}$. The number of possible time successor regions of $\widehat{R}$ are at most $2 \cdot \sum_{x \in C} 2(c_x + 1) \leq 4 \cdot (M+1) \cdot (|C|+1)$, where c_x is the largest constant that clock x is compared to in $\widehat{\mathcal{T}}$, $M = \max\{c_x \mid x \in C\}$ and C is the set of clocks in $\mathcal{T}$.*

Proof. When time elapses, the sets $C_0, \ldots, C_n$ move in a cyclical fashion, i.e., $\mod n+1$. The displacement $\mod n+1$ indicates the relative ordering of the fractional sets. A movement of a "full" cycle of the displacements increases the value of the integral values of all the clocks by 1. We also only track the integral value of a clock $x \in C$ upto c_x, after that the clock is placed into the set C_{-1}. Note that the extra clock z introduced in $\widehat{\mathcal{T}}$ is never placed into C_{-1}, and always has a value mod 1. Let us order the clocks in C in order of their increasing c_x values, i.e., $c_{x_1} \leq c_{x_2} \leq \ldots c_{x_N}$ where $N = C$. The most number of time successors are obtained when all clocks have an integral value of 0 to start with. We count the number of time successors in N stages. In the first stage, $C_{-1} = \emptyset$. After at most c_{x_1} full cycles, the clock x_1 gets moved to C_{-1} as its value exceeds the maximum tracked value. For each full cycle, we also have the number of distinct mod classes to be $N+1$ (recall that we also have the extra clock z). We need another factor of 2 to account for the movement which makes all clock values non-integral, e.g., $\langle x = 1, y = 1.2, z = 0.99 \rangle$ to $\langle x = 1.00001, y = 1.20001, z = 0.99001 \rangle$. Thus, before the clock c_{x_1} gets moved to C_{-1}, we can have $2 \cdot (c_{x_1} + 1) \cdot (N+1)$ time successors. In the second stage, we can have at most $c_{x_2} + 1 - c_{x_1}$ before clock c_{x_2} gets placed into C_{-1}. Also, since x_1 is in C_{-1}, we can only have

$N+1-1$ mod classes in the second stage. Thus, the number of time successors added in the second stage is at most $2\cdot(c_{x_2}+1-c_{x_1})\cdot N$. Continuing in this fashion, we obtain the total number of time successors as $2\cdot\big((c_{x_1}+1)\cdot(N+1)+(c_{x_2}+1-c_{x_1})\cdot(N+1-1)+\cdots+(c_{x_N}+1-\sum_{i=1}^{N-1}c_{x_i})\cdot(N+1-(N-1))\big)=4\cdot\sum_{i=1}^{N}(c_{x_i}+1)$. □

A finite state turn based game G consists of the tuple $\langle(S,E),(S_1,S_2)\rangle$, where (S_1,S_2) forms a partition of the finite set S of states, E is the set of edges, S_1 is the set of states from which only player 1 can make a move to choose an outgoing edge, and S_2 is the set of states from which only player 2 can make a move. The game is bipartite if every outgoing edge from a player-1 state leads to a player-2 state and vice-versa. A bipartite turn based finite game $\mathcal{T}^f=\langle(S^f,E^f),(\widehat{S}_{\mathsf{Reg}}\times\{1\},\widehat{S}_{\mathsf{Tup}}\times\{2\})\rangle$ can be constructed to capture the timed game $\mathcal{T}$. The state space S^f equals $\widehat{S}_{\mathsf{Reg}}\times\{1\}\cup\widehat{S}_{\mathsf{Tup}}\times\{2\}$. The game $\mathcal{T}^f$ is such that if Z is a player-i state, then the only outgoing edges are to the other player states. The set $\widehat{S}_{\mathsf{Reg}}$ is the set of regions of $\widehat{\mathcal{T}}$. Each $Z\in\widehat{S}_{\mathsf{Reg}}\times\{1\}$ is indicative of a state in the timed game $\widehat{\mathcal{T}}$ that belongs to the region $\widehat{S}_{\mathsf{Reg}}$. Each $Y\in\widehat{S}_{\mathsf{Tup}}\times\{2\}$ encodes the following information: (a) the previous state (which is a region of $\widehat{\mathcal{T}}$), (b) an intermediate region of $\widehat{\mathcal{T}}$ (representing a time move in $\widehat{\mathcal{T}}$ from the previous region), and (c) the desired discrete action of player 1 to be taken from the intermediate state. An edge from $Z=\langle\widehat{R},1\rangle$ to Y is indicative of the fact that in the timed game $\widehat{\mathcal{T}}$, from every state $\widehat{s}\in\widehat{R}$, player 1 has a move $\langle\Delta,a_1\rangle$ such that $\widehat{s}+\Delta$ is in the intermediate region component $\widehat{R}'$ of Y, with a_1 being the desired discrete action. From the state Y, player 2 has moves to $\widehat{S}_{\mathsf{Reg}}\times\{1\}$ depending on what moves of player 2 in the timed game $\widehat{\mathcal{T}}$ can beat the player-1 moves from $\widehat{R}$ to $\widehat{R}'$ according to Lemma 7.

Each $Z\in S^f$ is itself a tuple, with the first component being a location of $\mathcal{T}$. Given a location parity index function Ω on $\mathcal{T}$, we let Ω^f be the parity index function on $\mathcal{T}^f$ such that $\Omega^f(\langle l,\cdot\rangle)=\Omega(\langle l,\cdot\rangle)$. Another parity index function $\widehat{\Omega}^f$ with two more priorities can be derived from Ω^f to take care of time divergence issues, as described in [dAFH$^+$03]. Given a set $X=X_1\times\{1\}\cup X_2\times\{2\}\subseteq S^f$, we let $\mathsf{RegStates}(X)=\{\widehat{s}\in\widehat{S}\mid\mathsf{Reg}(\widehat{s})\in X_1\}$. We now present the full construction of the reduction.

Construction of the finite turn based game $\mathcal{T}^f$.
The game $\mathcal{T}^f$ consists of a tuple $\langle S^f,E^f,S_1^f,S_2^f\rangle$ where,

- $S^f=S_1^f\cup S_2^f$ is the state space. The states in S_i^f are controlled by player-i for $i\in\{1,2\}$.

- $S_1^f = \widehat{S}_{\mathsf{Reg}} \times \{1\}$, where $\widehat{S}_{\mathsf{Reg}}$ is the set of regions in $\widehat{\mathfrak{T}}$.

- $S_2^f = \widehat{S}_{\mathsf{Tup}} \times \{2\}$.

  The set $\widehat{S}_{\mathsf{Tup}}$ will be described later. Intuitively, a $B \in \widehat{S}_{\mathsf{Tup}}$ represents a 3-tuple $\langle Y_1, Y_2, a_1 \rangle$ where Y_i are regions of $\widehat{\mathfrak{T}}$, such that $\langle \Delta, a_1 \rangle \in \widehat{\Gamma}_1(\widehat{s})$ with $\widehat{s} + \Delta \in Y_2$. The values of Y_2 and a_1 are maintained indirectly.

- $\widehat{S}_{\mathsf{Tup}} = L \times \{\text{TRUE}, \text{FALSE}\}^2 \times H \times \mathcal{P}(\widehat{C}) \times \{0, \ldots, M\} \times \{0, \ldots, |C|+1\} \times \{\text{TRUE}, \text{FALSE}\} \times L \times 2^C \times \{\text{TRUE}, \text{FALSE}\}$, where H is the set of valuations from C to positive integers such that each clock x is mapped to a value less than or equal to c_x where c_x is the largest constant to which clock x is compared to.

  Given $Z = \langle l_1, \mathit{tick}, \mathit{bl}_1, h, \langle C_{-1}, \ldots, C_n \rangle, k, w, \mathit{om}, l_2, \lambda, \mathit{tev} \rangle \in \widehat{S}_{\mathsf{Tup}}$, we let $\mathsf{FirstRegion}(Z)$ denote the region $\langle l_2, \mathit{tick}, \mathit{bl}_1, h, \langle C_{-1}, \ldots, C_n \rangle \rangle \in \widehat{S}_{\mathsf{Reg}}$. Intuitively, $\mathsf{FirstRegion}(Z)$ is the region from which player 1 first proposes a move. The move of player 1 consists of a intermediate region Y, denoting that first time passes to let state change from $\mathsf{FirstRegion}(Z)$ to Y; and a discrete jump action specified by a destination location l_2, together with the clocks to be reset, λ (we observe that the discrete actions may also be directly specified as $a_1 \in A_1$ in case $|A_1| \leq |L| \cdot 2^C$). The variable tev is true iff player 1 proposed any relinquishing time move. The region Y is obtained from Z using the variables $0 \leq k \leq M$, $0 \leq w \leq |C| + 1$, and $\mathit{om} \in \{\text{TRUE}, \text{FALSE}\}$. The integer w indicates the the relative movement of the clock fractional parts $C_0, \ldots C_n$ (note that the movement must occur in a cyclical fashion). The integer k indicates the number of cycles completed. It can be at most M because after that, all clock values become bigger that the maximum constant, and thus need not be tracked. The boolean variable om indicates whether a small ϵ-move has taken place so that no clock value is integral, eg., $\langle x = 1, y = 1.2, z = 0.99 \rangle$ to $\langle x = 1.00001, y = 1.20001, z = 0.99001 \rangle$.

  Formally, $\mathsf{SecondRegion}(Z)$ denotes the region $\langle l_2, \mathit{tick}', \mathit{bl}_1, h', \langle C'_{-1}, \ldots, C'_m \rangle \rangle \in \widehat{S}_{\mathsf{Reg}}$ where

  - $$h'(x) = \begin{cases} h(x) + k & \text{if } h(x) + k \leq c_x \text{ and } x \in C_j \text{ with } j + w \leq n; \\ h(x) + k + 1 & \text{if } h(x) + k + 1 \leq c_x \text{ and } x \in C_j \text{ with } j + w > n; \\ c_x & \text{otherwise.} \end{cases}$$

    The integer k indicates the number of integer boundaries crossed by all the clocks when getting to the new region. Some clocks may cross k integer boundaries, while others may cross $k+1$ integer boundaries.

- $h_{\max}(x) = \begin{cases} h(x)+k & \text{if } x \in C_j \text{ with } j+w \leq n; \\ h(x)+k+1 & \text{if } x \in C_j \text{ with } j+w > n. \end{cases}$

  ($h_{\max}$ will be used later in the definition of $f_{\max}^{h_{\max}}$.)

- $\langle C'_{-1}, \ldots, C'_m \rangle = f_{\mathsf{Compact}} \circ f_{\max}^{h_{\max}} \circ f_{\mathsf{OpenMove}}^{om} \circ f_{\mathsf{Cycle}}^{w}(\langle C_{-1}, \ldots, C_n \rangle)$, where
  - $f_{\mathsf{Cycle}}^{w}(\langle C_{-1}, \ldots, C_n \rangle) = \langle C_{-1}, C'_0, \ldots, C'_n \rangle$ with $C'_{(j+w) \bmod (n+1)} = C_j$.
    This function cycles around the fractional parts by w.
  - $f_{\mathsf{OpenMove}}^{om}(\langle C_{-1}, C_0, \ldots, C_n \rangle) = \begin{cases} \langle C_{-1}, C_0, \ldots, C_n \rangle & \text{if } om = \text{FALSE}; \\ \langle C_{-1}, \emptyset, C_0, \ldots, C_n \rangle & \text{if } om = \text{TRUE}. \end{cases}$
    This function indicates if the current region is such that all the clocks have non-integral values (if $om = \text{TRUE}$).
  - $f_{\max}(\langle C_{-1}, C_0, \ldots, C_n \rangle) = \langle C'_{-1}, C'_0, \ldots, C'_n \rangle$ with $C'_j = C_j \setminus V_j$ for $j \geq 0$ and $C'_{-1} = C_{-1} \cup_{j=0}^{n} V_j$ where (a) $x \in V_0$ iff $x \in C_0$ and $h_{\max}(x) > c_x$; and (b) $x \in V_j$ for $j > 0$ iff $x \in C_j$ and $h'(x) = c_x$.
    When clocks are cycled around, some of them may exceed the maximal tracked values c_x. In that case, they need to be moved to C_{-1}. This function is accomplished by $f_{\max}$.
  - $f_{\mathsf{Compact}}(\langle C_{-1}, C_0, \ldots, C_m \rangle)$ eliminates the empty sets for $j > 0$. It can be obtained by the following procedure:

```
$i := 0, j := 1$
while $j \leq m$ do
  while $j < m$ and $C_j = \emptyset$ do
    $j := j + 1$
  end while
  if $C_j \neq \emptyset$ then
    $C_{i+1} := C_j$
    $i := i + 1, j := j + 1$
  end if
end while
return $\langle C_{-1}, C_0, \ldots, C_i \rangle$
```

- *tick*$' = \text{TRUE}$ iff $k > 0$; or $z \in C_i$ and $w > n - i$.

- The set of edges is specified by a transition relation δ^f, and a set of available moves Γ_i^f. We let A_i^f denote the set of moves for player-i, and $\Gamma_i(X)$ denote the set of moves available to player-i at state $X \in S_i^f$.

- $A_1^f = (\widehat{S}_{\mathsf{Reg}} \times L \times 2^C \cup \{\bot_1\}) \times \{1\}$.
  The component $\widehat{S}_{\mathsf{Reg}}$ denotes the region that player 1 wants to let time elapse to in $\widehat{\mathfrak{T}}$ to before she takes a jump with the destination specified by the location and the set of clocks that are reset. The move $\{\bot_1\} \times \{1\}$ is a relinquishing move, corresponding to a pure time move in $\widehat{\mathfrak{T}}$.

- $A_2^f = \widehat{S}_{\mathsf{Reg}} \times \{1,2\} \times L \times 2^C \times \{2\}$.
  The component $\widehat{S}_{\mathsf{Reg}}$ denotes the region that player 2 wants to let time elapse to in $\widehat{\mathfrak{T}}$ to before she takes a jump with the destination specified by the location and the set of clocks that are reset. The element in $\{1,2\}$ is used in the case player 2 picks the same intermediate region $\widehat{S}_{\mathsf{Reg}}$ as player 1. In this case, player 2 has a choice of letting the move of player 1 win or not, and the number from $\{1,2\}$ indicates which player wins.

- The set of available moves for player 1 at a state $\langle X, 1\rangle$ is given by $\Gamma_1^f(X \times \{1\}) = \{\bot_1\} \times \{1\} \cup$
$$\left\{ \langle Y, l_y, \lambda, 1\rangle \;\middle|\; \begin{array}{l} \exists\, \widehat{s} = \langle l_x, \widehat{\kappa}_x\rangle \in X,\ \exists\langle \Delta, \bot\rangle \in \widehat{\Gamma}_1(\widehat{s}) \text{ such that} \\ \langle l_x, \widehat{\kappa}_x\rangle + \Delta \in Y \text{ and } \exists \widehat{s}' \in Y,\ \exists\langle l_x, a_1, \theta, l_y, \lambda\rangle \in \widehat{\Gamma}_1(\widehat{s}'), \\ \text{such that } \widehat{s}' \models \theta \end{array} \right\}$$

- The set of available moves for player 2 at a state $\langle X, 2\rangle$ is given by $\Gamma_1^f(X \times \{2\}) =$
$$\left\{ \langle Y, i, l_y, \lambda, 2\rangle \;\middle|\; \begin{array}{l} i \in \{1,2\},\ \exists\, \widehat{s} = \langle l_x, \widehat{\kappa}_x\rangle \in \mathsf{FirstRegion}(\langle X, 2\rangle), \\ \exists\langle \Delta, \bot_2\rangle \in \widehat{\Gamma}_2(\widehat{s}) \text{ such that } \langle l_x, \widehat{\kappa}_x\rangle + \Delta \in Y \text{ and} \\ \text{(a) } l_y = l_x \text{ and } \lambda = \emptyset \text{ or,} \\ \text{(b) } \exists \widehat{s}' \in Y\ \ \exists\langle l_x, a_2, \theta, l_y, \lambda\rangle \in \Gamma_2(\widehat{s}') \text{ such that } \widehat{s}' \models \theta \end{array} \right\}$$

- The transition function δ^f is specified by
  - $\delta^f(\langle l, \mathit{tick}, \mathit{bl}_1, h, \langle C_{-1}, \ldots, C_n\rangle, 1\rangle, \langle Y, l_y, \lambda, 1\rangle) =$
    $\langle l, \mathit{tick}, \mathit{bl}_1, h, \langle C_{-1}, \ldots, C_n\rangle, k, w, \mathit{om}, l_y, \lambda, \textsc{false}, 2\rangle$, where $0 \le k \le M, 0 \le w \le |C| + 1, \mathit{om} \in \{\textsc{true}, \textsc{false}\}$ are such that $Y = \mathsf{SecondRegion}(\langle l, \mathit{tick}, \mathit{bl}_1, h, \langle C_{-1}, \ldots, C_n\rangle, k, w, \mathit{om}, l_y, \lambda, \textsc{false}, 2\rangle)$.

- $\delta^f(\langle l, \mathit{tick}, bl_1, h, \langle C_{-1}, \ldots, C_n\rangle, 1\rangle, \langle \bot, 1\rangle) =$
  $\langle l, \mathit{tick}, bl_1, h, \langle C_{-1}, \ldots, C_n\rangle, 0, 0, \textsc{false}, l, \emptyset, \textsc{true}, 2\rangle$.

- Let $\langle Z, 2\rangle = \langle l, \mathit{tick}, bl_1, h, \langle C_{-1}, \ldots, C_n\rangle, k, w, om, l_z, \lambda_z, tev, 2\rangle)$.
  Then, $\delta^f(\langle Z, 2\rangle, \langle Y, 2, l_y, \lambda_y, 2\rangle) =$
  $$\begin{cases} \langle \mathsf{SecondRegion}(Z)[\mathsf{loc} := l_z, \lambda_z := 0, bl_1 := \textsc{true}], 1\rangle & \\ \qquad \text{if } tev = \textsc{false} \text{ and all player 1 moves to } \mathsf{SecondRegion}(Z) & \\ \qquad \text{beat all player 2 moves to } Y & \\ \qquad \text{from the region } \mathsf{FirstRegion}(Z) \text{ according to Lemma 7;} & \\ \langle Y[loc := l_y, \lambda_y := 0, bl_1 = \textsc{false}], 1\rangle & \text{otherwise.} \end{cases}$$

- $\delta^f(\langle Z, 2\rangle, \langle Y, 1, l_y, \lambda_y, 2\rangle) =$
  $$\begin{cases} \langle \mathsf{SecondRegion}(Z)[\mathsf{loc} := l_z, \lambda_z := 0, bl_1 = \textsc{true}], 1\rangle & \\ \qquad \text{if } tev = \textsc{false} \text{ and all player 1 moves to} & \\ \qquad \mathsf{SecondRegion}(Z) \text{ beats all player 2 moves to } Y & \\ \qquad \text{from the region } \mathsf{FirstRegion}(Z) \text{ according to Lemma 7;} & \\ \langle \mathsf{SecondRegion}(Z)[\mathsf{loc} := l_z, \lambda_z := 0, bl_1 := \textsc{true}], 1\rangle & \\ \qquad \text{if } tev = \textsc{false} \text{ and } Y = \mathsf{SecondRegion}(Z) \text{ ie., both} & \\ \qquad \text{players pick the same time delay, (and player 2} & \\ \qquad \text{allows the player 1 move, signified by the 1 in} \langle Y, 1, l_y, \lambda_y, 2\rangle); & \\ \langle Y[loc := l_y, \lambda_y := 0, bl_1 := \textsc{false}], 1\rangle & \text{otherwise..} \end{cases}$$

Note that we change the values of bl_1 and tick only after player-2 moves.

Theorem 11. *Let $\widehat{\mathcal{T}}$ be an enlarged timed game structure, and let $\mathcal{T}^f$ be the corresponding finite game structure. Then, given an ω-regular region objective* $\mathsf{Parity}(\Omega)$*, we have* $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\mathsf{TimeDivBl}_1(\mathsf{Parity}(\Omega))) = \mathsf{RegStates}(\mathsf{Win}_1^{\mathcal{T}^f}(\mathsf{Parity}(\widehat{\Omega}^f)))$.

Proof. A solution for obtaining the set $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\mathsf{TimeDivBl}_1(\mathsf{Parity}(\Omega)))$ has been presented in [dAFH$^+$03] using a μ-calculus formulation. The μ-calculus iteration uses the *controllable predecessor* operator for player 1, $\mathsf{CPre}_1 : 2^{\widehat{S}} \mapsto 2^{\widehat{S}}$, defined formally by $\widetilde{s} \in \mathsf{CPre}_1(Z)$ iff $\exists m_1 \in \widehat{\Gamma}_1(\widehat{s})\ \forall m_2 \in \widehat{\Gamma}_2(\widehat{s}) \,.\, \widehat{\delta}_{\mathsf{jd}}(\widehat{s}, m_1, m_2) \subseteq Z$. Informally, $\mathsf{CPre}_1(Z)$ consists of the set of states from which player 1 can ensure that the next state will be in Z, no matter what player 2 does. It can be shown that CPre_1 preserves regions of $\widehat{\mathcal{T}}$ using Lemma 7. We use the Pre_1 operator in turn based games: $\mathsf{Pre}_1(X) = \{s \in \widehat{S}_{\mathsf{Reg}} \times \{1\} \mid \exists s' \in X \text{ such that } (s, s') \in E^f\} \cup \{s \in \widehat{S}_{\mathsf{Tup}} \times \{2\} \mid \forall (s, s') \in E^f \text{ we have } s' \in X\}$. From the construction of $\mathcal{T}^f$, it also follows that given $X = X_1 \times \{1\} \cup X_2 \times \{2\} \subseteq S^f$, we have

$$\mathsf{RegStates}(\mathsf{Pre}_1^{\mathcal{T}^f}(\mathsf{Pre}_1^{\mathcal{T}^f}(X))) = \mathsf{CPre}_1^{\widehat{\mathcal{T}}}(\mathsf{RegStates}(X)) = \mathsf{CPre}_1^{\widehat{\mathcal{T}}}(X_1) \tag{3.1}$$

Let ϕ_c be the μ-calculus formula using the CPre_1 operator describing the winning set for $\mathsf{Parity}(\widehat{\Omega}) = \mathsf{TimeDivBl}_1(\mathsf{Parity}(\Omega))$. Let ϕ_t be the μ-calculus formula using the Pre_1 operator in a turn based game describing the winning set for $\mathsf{Parity}(\widehat{\Omega})$. The formula ϕ_t can be obtained from ϕ_c by syntactically replacing every CPre_1 by Pre_1. Let the winning set for $\mathsf{Parity}(\widehat{\Omega})$ in $\mathcal{T}^f$ be $W_1 \times \{1\} \cup W_2 \times \{2\}$. It is described by ϕ_t. The game in $\mathcal{T}^f$ proceeds in a bipartite fashion — player 1 and player 2 alternate moves, with the state resulting from the move of player 1 having the same parity index as the originating state. Note that the objective $\mathsf{Parity}(\widehat{\Omega})$ depends only on the infinitely often occurring indices in the trace. Thus, $W_1 \times \{1\}$ can be also be described by the μ-calculus formula ϕ'_t obtained by replacing each Pre_1 in ϕ_t with $\mathsf{Pre}_1 \circ \mathsf{Pre}_1$, and taking states of the form $s \times \{1\}$ in the result. Since we are only interested in the set $W_1 \times \{1\}$, and since we have a bipartite game where the parity index remains the same for every next state of a player-1 state, the set $W_1 \times \{1\}$ can also be described by the μ-calculus formula ϕ''_t obtained from ϕ'_t by intersecting every variable with $\widehat{S}_{\mathsf{Reg}} \times \{1\}$. Now, ϕ''_t can be computed using a finite fixpoint iteration. Using the identity 3.1, we have that the sets in the fixpoint iteration computation of ϕ''_t correspond to the sets in the fixpoint iteration computation of ϕ_c, that is, if $X \times \{1\}$ occurs in the computation of ϕ''_t at stage j, then $\mathsf{RegStates}(X)$ occurs in the computation of ϕ''_t at the same stage j. This implies that the sets are the same on termination for both ϕ''_t and ϕ_c. Thus, $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\mathsf{TimeDivBl}_1(\mathsf{Parity}(\Omega))) = \mathsf{RegStates}(\mathsf{Win}_1^{\mathcal{T}^f}(\mathsf{Parity}(\widehat{\Omega}^f)))$. □

Complexity of reduction. Recall that for a timed automaton game $\mathcal{T}$, A_i is the set of actions for player i, C is the set of clocks and M is the largest constant in $\mathcal{T}$. Let $|A_i|^* = \min\{|A_i|, L \cdot 2^{|C|}\}$ and let $|\mathcal{T}_{\mathsf{Constr}}|$ denote the length of the clock constraints in $\mathcal{T}$. We now show that the size of the state space of $\mathcal{T}^f$ is bounded by $|\widehat{S}_{\mathsf{Reg}}| \cdot (1 + (M+1) \cdot (|C|+2) \cdot 2 \cdot (|A_1|^* + 1))$, where $|\widehat{S}_{\mathsf{Reg}}| \leq 16 \cdot |L| \cdot \prod_{x \in C}(c_x+1) \cdot |C+1|! \cdot 2^{|C|+1}$ is the number of regions of $\widehat{\mathcal{T}}$. We also show that the number of edges in $\mathcal{T}^f$ is bounded by $|\widehat{S}_{\mathsf{Reg}}| \cdot ((M+1) \cdot (|C|+2) \cdot 2) \cdot (|A_1|^* + 1) \left[(1 + (|A_2|^* + 1) \cdot ((M+1) \cdot (|C|+2) \cdot 2)\right]$.

In the construction of $\mathcal{T}^f$, we can keep track of actions, or the locations together with the reset sets depending on whether $|A_i|$ is bigger than $L \cdot 2^{|C|}$ or not, hence we shal use $|A_i|^*$ in our analysis. We have $|S_1^f| = |\widehat{S}_{\mathsf{Reg}}|$, and $|S_2^f| = |\widehat{S}_{\mathsf{Reg}}| \cdot (M+1) \cdot (|C|+2) \cdot 2 \cdot (|A_1|^*+1)$

(we have incorporated a modification where we represent possible actions by $\{\bot_1\} \cup A_1$ instead of $L \times 2^C \times \{\text{TRUE}, \text{FALSE}\}$). Given a state $Z \in S_1^f$, the number player-1 edges from Z is equal to one plus the cardinality of the set of time successors of Z multiplied by player-1 actions. This is equal to $(|A_1| + 1)^* \cdot ((M+1) \cdot (|C|+2) \cdot 2)$ (the $+1$ corresponds to the relinquishing move). Thus the total number of player-1 edges is at most $|\widehat{S}_{\mathsf{Reg}}| \cdot (|A_1|^* + 1) \cdot ((M+1) \cdot (|C|+2) \cdot 2)$. Given a state $X \in S_2^f$, the number player-2 edges from X is equal to $2 \cdot (|A_2|^* + 1)$ multiplied by the cardinality of the set of time successors of $\mathsf{FirstRegion}(X)$ (the plus one arises as player-2 can have a pure time move in addition to actions from A_2). Thus, the number of player-2 edges is at most $|S_2^f| \cdot 2 \cdot (|A_2|^* + 1) \cdot ((M+1) \cdot (|C|+2) \cdot 2)$. Hence, $|E^f| \leq |\widehat{S}_{\mathsf{Reg}}| \cdot ((M+1) \cdot (|C|+2) \cdot 2) \cdot (|A_1|^* + 1) \left[(1 + (|A_2|^* + 1) \cdot ((M+1) \cdot (|C|+2) \cdot 2)\right]$. Let $|\mathcal{T}_{\mathsf{Constr}}|$ denote the length of the clock constraints in $\mathcal{T}$. For our complexity analysis, we assume all clock constraints are in conjunctive normal form. For constructing $\mathcal{T}^f$, we need to check whether regions satisfy clock constraints from $\mathcal{T}$. For this, we build a list of regions with valid invariants together with edge constraints satisfied at the region. This takes $O(|\widehat{S}_{\mathsf{Reg}}| \cdot |\mathcal{T}_{\mathsf{Constr}}|)$ time. We assume a region can be represented in constant space.

Theorem 12. *Let $\mathcal{T}$ be a timed automaton game, and let Ω be a region parity index function of order d. The set $\mathsf{WinTimeDiv}_1^{\mathcal{T}}(\mathsf{Parity}(\Omega))$ can be computed in time*

$$O\left((|\widehat{S}_{\mathsf{Reg}}| \cdot |\mathcal{T}_{\mathsf{Constr}}|) + [M \cdot |C| \cdot |A_2|^*] \cdot \left[2 \cdot |\widehat{S}_{\mathsf{Reg}}| \cdot M \cdot |C| \cdot |A_1|^*\right]^{\frac{d+2}{3} + \frac{3}{2}} \right)$$

where $|\widehat{S}_{\mathsf{Reg}}| \leq 16 \cdot |L| \cdot \prod_{x \in C}(c_x + 1) \cdot |C+1|! \cdot 2^{|C|+1}$, M is the largest constant in $\mathcal{T}$, $|\mathcal{T}_{\mathsf{Constr}}|$ is the length of the clock constraints in $\mathcal{T}$, C is the set of clocks, $|A_i|^* = \min\{|A_i|, L \cdot 2^{|C|}\}$, and $|A_i|$ the number of discrete actions of player i for $i \in \{1, 2\}$.

Proof. From [Sch07], we have that a turn based parity game with m edges, n states and d parity indices can be solved in $O(m \cdot n^{\frac{d}{3} + \frac{1}{2}})$ time. Thus, $\mathsf{WinTimeDiv}_1^{\mathcal{T}}(\mathsf{Parity}(\Omega))$ can be computed in time $O\left((|\widehat{S}_{\mathsf{Reg}}| \cdot |\mathcal{T}_{\mathsf{Constr}}|) + \mathcal{F}_1 \cdot \mathcal{F}_2^{\frac{d+2}{3} + \frac{1}{2}} \right)$, where $\mathcal{F}_1 = |\widehat{S}_{\mathsf{Reg}}| \cdot ((M+1) \cdot (|C|+2) \cdot 2) \cdot (|A_1|^* + 1) \left[(1 + (|A_2|^* + 1) \cdot ((M+1) \cdot (|C|+2) \cdot 2)\right]$, and $\mathcal{F}_2 = |\widehat{S}_{\mathsf{Reg}}| \cdot (1 + (M+1) \cdot (|C|+2) \cdot 2 \cdot (|A_1|^* + 1))$, which is equal to

$$O\left((|\widehat{S}_{\mathsf{Reg}}| \cdot |\mathcal{T}_{\mathsf{Constr}}|) + [M \cdot |C| \cdot |A_2|^*] \cdot \left[2 \cdot |\widehat{S}_{\mathsf{Reg}}| \cdot M \cdot |C| \cdot |A_1|^*\right]^{\frac{d+2}{3} + \frac{3}{2}} \right)$$

□

From Theorem 11, we can solve the finite state game $\mathcal{T}^f$ to compute winning sets for all ω-regular region parity objectives Φ for a timed automaton game $\mathcal{T}$, using *any* algorithm for finite state turn based games, e.g., strategy improvement, small-progress algorithms [VJ00, Jur00]. Note that $\mathcal{T}^f$ does not depend on the parity condition used, and there is a correspondence between the regions repeating infinitely often in $\mathcal{T}$ and $\mathcal{T}^f$. Hence, it is not required to explicitly convert an ω-regular objective Φ to a parity objective to solve using the $\mathcal{T}^f$ construction. We can solve the finite state game $\mathcal{T}^f$ to compute winning sets for all ω-regular region objectives Φ, where Φ is a Muller objective. Since Muller objectives subsume Rabin, Streett (strong fairness objectives), parity objectives as a special case, our result holds more a much richer class of objectives than parity objectives.

Corollary 3. *Let $\widehat{\mathcal{T}}$ be an enlarged timed game structure, and let $\mathcal{T}^f$ be the corresponding finite game structure. Then, given an ω-regular region objective Φ, where Φ is specified as a Muller objective, we have* $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\mathsf{TimeDivBl}_1(\Phi)) = \mathsf{RegStates}(\mathsf{Win}_1^{\mathcal{T}^f}(\mathsf{TimeDivBl}(\Phi)))$.

Chapter 4

Timed-Alternating Time Logic

4.1 Introduction

Temporal logics are a system for qualitatively describing and reasoning about how the truth values of assertions change over time (see [Eme90] for a survey). These logics can reason about properties like "eventually the specified assertion becomes true", or "the specified assertion is true infinitely often". Branching time logics provide explicit quantifications over the set of computations, for example the CTL formula $\forall\varphi$ requires that a state satisfying φ be visited on all paths, and the formula $\exists\varphi$ specifies that a state satisfying φ be visited on some path. Given a state of a system and a temporal logic specification, the model checking problem is to determine whether the state satisfies the logic specification.

In game structures, we want to differentiate between agents in the logic specification. In [AHK02], several alternating-time temporal logics were introduced to specify properties of untimed game structures, including the CTL-like logic ATL, and the CTL*-like logic ATL*. These logics are natural specification languages for multi-component systems, where properties need to be guaranteed by subsets of the components irrespective of the behavior of the other components. Each component represents a player in the game, and sets of players may form teams. For example, the ATL formula $\langle\langle i\rangle\rangle\Diamond p$ is true at a state s iff player i can force the game from s into a state that satisfies the proposition p. We interpret these logics over *timed* game structures, and enrich them by adding *freeze* quantifiers [AH94] for specifying timing constraints. The resulting logics are called TATL and TATL*. The new logic TATL subsumes both the untimed game logic ATL, and the timed non-game logic TCTL [ACD93]. For example, the TATL formula $\langle\langle i \rangle\rangle \Diamond_{\leq d}\, p$ is true at a state s iff player i can force the game from s into a p state in at most d time units. We restrict our attention here to the two-player case (e.g., system vs. environment; or plant vs. controller), but all results can be extended to the multi-player case.

The model checking of these logics requires the solution of timed games. Timed game structures are infinite-state. In order to consider algorithmic solutions, we restrict our attention to timed automaton game structures. For timed systems, we need the players to use only receptive strategies when achieving their objectives and we use the framework presented in Chapter 3. We show that the receptiveness requirement can be encoded within TATL* (but not within TATL). However, solving TATL* games is undecidable, because TATL* subsumes the linear-time logic TPTL [AH94], whose dense-time satisfiability problem is undecidable. We nonetheless establish the decidability of TATL model checking, by carefully analyzing the fragment of TATL* we obtain through the winning condition translation. We show that TATL model checking over timed automaton games is complete for EXPTIME; that is, no harder than the solution of timed automaton games with reachability objectives.

Outline. In Section 4.2 we present the syntax and semantics of the logic TATL, and in Section 4.3 that for the logic TATL*. Model checking of TATL proceed by an encoding to TATL* and is described in Section 4.4.

4.2 TATL Syntax and Semantics

In this chapter we consider a fixed timed game structure together with propositions on states, $\mathcal{G} = \langle S, \Sigma, \sigma, A_1, A_2, \Gamma_1, \Gamma_2, \delta \rangle$ where

- Σ is a finite set of propositions.
- $\sigma : S \mapsto 2^{\Sigma}$ is the observation map, which assigns to every state the set of propositions that are true in that state.
- $S, A_1, A_2, \Gamma_1, \Gamma_2, \delta$ are as defined in Chapter 3.

In this chapter we shall consider ω-objectives Φ over propositions, ie., objectives Φ that are such that there exists an ω-regular set $\Psi \subseteq (2^{\Sigma})^{\omega}$ of infinite sequences of sets of propositions such that a run $r = s_0, \langle m_1^0, m_2^0 \rangle, s_1, \langle m_1^1, m_2^1 \rangle, \ldots$ is in Φ iff the projection $\sigma(r) = \sigma(s_0), \sigma(s_1), \sigma(s_2), \ldots$ is in Ψ.

The temporal logic TATL (Timed Alternating-Time Temporal Logic) is interpreted over the states of $\mathcal{G}$. We use the syntax of *freeze quantification* [AH94] for specifying timing constraints within the logic. The freeze quantifier "$x\cdot$" binds the value of the clock variable x in a formula $\varphi(x)$ to the current time $t \in \mathbb{R}_{\geq 0}$; that is, the constraint $x \cdot \varphi(x)$ holds at time t iff $\varphi(t)$ does. For example, the property that "every p state is followed by a q state within d time units" can be written as: $\forall \Box x \cdot (p \rightarrow \Diamond y \cdot (q \wedge y \leq x + d))$. This formula says that "in every state with time x, if p holds, then there is a later state with time y such that both q and $y \leq x + d$ hold." Formally, given a set D of clock variables, a TATL formula is one of the following:

- $\textsc{true} \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2$, where $p \in \Sigma$ is a proposition, and φ_1, φ_2 are TATL formulae.

- $x + d_1 \leq y + d_2 \mid x \cdot \varphi$, where $x, y \in D$ are clock variables and d_1, d_2 are nonnegative integer constants, and φ is a TATL formula. We refer to the clocks in D as *formula clocks.*

- $\langle\langle \mathfrak{P} \rangle\rangle \Box \varphi \mid \langle\langle \mathfrak{P} \rangle\rangle \varphi_1 \mathcal{U} \varphi_2$, where $\mathfrak{P} \subseteq \{1, 2\}$ is a set of players, and $\varphi, \varphi_1, \varphi_2$ are TATL formulae.

We omit the next operator of ATL, which has no meaning in timed systems. The freeze quantifier $x \cdot \varphi$ binds all free occurrences of the formula clock variable x in the formula φ. A TATL formula is *closed* if it contains no free occurrences of formula clock variables. Without loss of generality, we assume that for every quantified formula $x \cdot \varphi$, if $y \cdot \varphi'$ is a subformula of φ, then x and y are different; that is, there is no nested reuse of formula clocks. When interpreted over the states of a timed automaton game $\mathcal{T}$, a TATL formula may also contain free (unquantified) occurrences of clock variables from $\mathcal{T}$.

There are four possible sets of players (so-called *teams*), which may collaborate to achieve a common goal: we write $\langle\langle\,\rangle\rangle$ for $\langle\langle \emptyset \rangle\rangle$; we write $\langle\langle i \rangle\rangle$ for $\langle\langle \{i\} \rangle\rangle$ with $i \in \{1, 2\}$; and we write $\langle\langle 1, 2 \rangle\rangle$ for $\langle\langle \{1, 2\} \rangle\rangle$. Roughly speaking, a state s satisfies the TATL formula $\langle\langle i \rangle\rangle \varphi$ iff player i can win the game at s for an objective derived from φ. The state s satisfies the formula $\langle\langle\,\rangle\rangle \varphi$ (resp., $\langle\langle 1, 2 \rangle\rangle \varphi$) iff every run (resp., some run) from s is contained in the objective derived from φ. Thus, the team $\emptyset$ corresponds to both players playing adversially, and the team $\{1, 2\}$ corresponds to both players collaborating to achieve a goal. We therefore write $\forall$ short for $\langle\!\langle\, \rangle\!\rangle$, and $\exists$ short for $\langle\!\langle 1, 2\rangle\!\rangle$, as in ATL.

We assign the responsibilities for time divergence to teams as follows: let $\mathsf{Blameless}_{\emptyset} = \mathsf{Runs}$, let $\mathsf{Blameless}_{\{1,2\}} = \emptyset$, and let $\mathsf{Blameless}_{\{i\}} = \mathsf{Blameless}_i$ for $i \in \{1, 2\}$. A strategy $\pi_{\mathfrak{P}}$ for the team $\mathfrak{P}$ consists of a strategy for each player in $\mathfrak{P}$. We denote the "opposing" team by $\sim\!\mathfrak{P} = \{1, 2\} \setminus \mathfrak{P}$. Given a state $s \in S$, a team-$\mathfrak{P}$ strategy $\pi_{\mathfrak{P}}$, and a team-$\sim\!\mathfrak{P}$ strategy $\pi_{\sim\mathfrak{P}}$, we define $\mathsf{Outcomes}(s, \pi_{\mathfrak{P}} \cup \pi_{\sim\mathfrak{P}}) = \mathsf{Outcomes}(s, \pi_1, \pi_2)$ for the player-1 strategy π_1 and the player-2 strategy π_2 in the set $\pi_{\mathfrak{P}} \cup \pi_{\sim\mathfrak{P}}$ of strategies. Given a team-$\mathfrak{P}$ strategy $\pi_{\mathfrak{P}}$, we define the set of possible outcomes from state s by $\mathsf{Outcomes}(s, \pi_{\mathfrak{P}}) = \cup_{\pi_{\sim\mathfrak{P}}} \mathsf{Outcomes}(s, \pi_{\mathfrak{P}} \cup \pi_{\sim\mathfrak{P}})$, where the union is taken over all team-$\sim\!\mathfrak{P}$ strategies $\pi_{\sim\mathfrak{P}}$.

To define the semantics of TATL, we need to distinguish between *physical time* and *game time*. We allow moves with zero time delay, thus a physical time $t \in \mathbb{R}_{\geq 0}$ may correspond to several linearly ordered states, to which we assign the game times $\langle t, 0\rangle, \langle t, 1\rangle, \langle t, 2\rangle, \ldots$ For a run $r \in \mathsf{Runs}$, we define the set of game times as

$$\mathsf{GameTimes}(r) \;=\; \begin{array}{l} \{\langle t, k\rangle \in \mathbb{R}_{\geq 0} \times \mathbb{N} \mid 0 \leq k < |\{j \geq 0 \mid \mathsf{time}(r, j) = t\}|\} \;\cup \\ \{\langle t, 0\rangle \mid \mathsf{time}(r, j) \geq t \text{ for some } j \geq 0\}. \end{array}$$

The state of the run r at a game time $\langle t, k\rangle \in \mathsf{GameTimes}(r)$ is defined as

$$\mathsf{state}(r, \langle t, k\rangle) \;=\; \begin{cases} r[j + k] \;\text{ if } \mathsf{time}(r, j) = t \text{ and for all } j' < j, \mathsf{time}(r, j') < t; \\ \delta(r[j], \langle t - \mathsf{time}(r, j), \bot\rangle) \;\text{ if } \mathsf{time}(r, j) < t < \mathsf{time}(r, j + 1). \end{cases}$$

Note that if r is a run of the timed game structure $\mathcal{G}$, and $\mathsf{time}(r, j) < t < \mathsf{time}(r, j + 1)$, then $\delta(r[j], \langle t - \mathsf{time}(r, j), \bot\rangle)$ is a state in S, namely, the state that results from $r[j]$ by letting time $t - \mathsf{time}(r, j)$ pass. We say that the run r *visits* a proposition $p \in \Sigma$ if there is a $\tau \in \mathsf{GameTimes}(r)$ such that $p \in \sigma(\mathsf{state}(r, \tau))$. We order the game times of a run lexicographically: for all $\langle t, k\rangle, \langle t', k'\rangle \in \mathsf{GameTimes}(r)$, we have $\langle t, k\rangle < \langle t', k'\rangle$ iff either $t < t'$, or $t = t'$ and $k < k'$. For two game times τ and τ', we write $\tau \leq \tau'$ iff either $\tau = \tau'$ or $\tau < \tau'$.

An *environment* $\mathcal{E} : D \mapsto \mathbb{R}_{\geq 0}$ maps every formula clock in D to a nonnegative real. Let $\mathcal{E}[x := t]$ be the environment such that $(\mathcal{E}[x := t])(y) = \mathcal{E}(y)$ if $y \neq x$, and $(\mathcal{E}[x := t])(y) = t$ if $y = x$. For a state $s \in S$, a time $t \in \mathbb{R}_{\geq 0}$, an environment $\mathcal{E}$, and a TATL formula φ, the satisfaction relation $(s, t, \mathcal{E}) \models_{\mathrm{td}} \varphi$ is defined inductively as follows (the subscript td indicates that players may win in only a physically meaningful way):

- $(s, t, \mathcal{E}) \models_{\mathrm{td}} \textsc{true}$.
- $(s, t, \mathcal{E}) \models_{\mathrm{td}} p$, for a proposition p, iff $p \in \sigma(s)$.
- $(s, t, \mathcal{E}) \models_{\mathrm{td}} \neg\varphi$ iff $(s, t, \mathcal{E}) \not\models_{\mathrm{td}} \varphi$.
- $(s, t, \mathcal{E}) \models_{\mathrm{td}} \varphi_1 \wedge \varphi_2$ iff $(s, t, \mathcal{E}) \models_{\mathrm{td}} \varphi_1$ and $(s, t, \mathcal{E}) \models_{\mathrm{td}} \varphi_2$.
- $(s, t, \mathcal{E}) \models_{\mathrm{td}} x + d_1 \leq y + d_2$ iff $\mathcal{E}(x) + d_1 \leq \mathcal{E}(y) + d_2$.
- $(s, t, \mathcal{E}) \models_{\mathrm{td}} x \cdot \varphi$ iff $(s, t, \mathcal{E}[x := t]) \models_{\mathrm{td}} \varphi$.
- $(s, t, \mathcal{E}) \models_{\mathrm{td}} \langle\langle \mathfrak{P} \rangle\rangle \Box \varphi$ iff there is a team-$\mathfrak{P}$ strategy $\pi_{\mathfrak{P}}$ such that for all runs $r \in \mathsf{Outcomes}(s, \pi_{\mathfrak{P}})$, the following conditions hold:

  > If $r \in \mathsf{Timediv}$, then for all $\langle u, k\rangle \in \mathsf{GameTimes}(r)$, we have $(\mathsf{state}(r, \langle u, k\rangle), t + u, \mathcal{E}) \models_{\mathrm{td}} \varphi$. If $r \notin \mathsf{Timediv}$, then $r \in \mathsf{Blameless}_{\mathfrak{P}}$.

- $(s, t, \mathcal{E}) \models_{\mathrm{td}} \langle\langle \mathfrak{P} \rangle\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there is a team-$\mathfrak{P}$ strategy $\pi_{\mathfrak{P}}$ such that for all runs $r \in \mathsf{Outcomes}(s, \pi_{\mathfrak{P}})$, the following conditions hold:

  > If $r \in \mathsf{Timediv}$, then there is a $\langle u, k\rangle \in \mathsf{GameTimes}(r)$ such that $(\mathsf{state}(r, \langle u, k\rangle), t + u, \mathcal{E}) \models_{\mathrm{td}} \varphi_2$, and for all $\langle u', k'\rangle \in \mathsf{GameTimes}(r)$ with $\langle u', k'\rangle < \langle u, k\rangle$, we have $(\mathsf{state}(r, \langle u', k'\rangle), t + u', \mathcal{E}) \models_{\mathrm{td}} \varphi_1$. If $r \notin \mathsf{Timediv}$, then $r \in \mathsf{Blameless}_{\mathfrak{P}}$.

Note that for an $\exists$ formula to hold, we require time divergence (as $\mathsf{Blameless}_{\{1,2\}} = \emptyset$). Also note that for a closed formula, the value of the environment is irrelevant in the satisfaction relation. A state s of the timed game structure $\mathcal{G}$ *satisfies* a closed formula φ of TATL, denoted $s \models_{\mathrm{td}} \varphi$, if $(s, 0, \mathcal{E}) \models_{\mathrm{td}} \varphi$ for any environment $\mathcal{E}$.

We use the following abbreviations. We write $\langle\langle \mathfrak{P} \rangle\rangle \varphi_1 \mathcal{U}_{\sim d}\, \varphi_2$ for $x \cdot \langle\langle \mathfrak{P} \rangle\rangle \varphi_1 \mathcal{U}\, y \cdot (\varphi_2 \wedge y \sim x + d)$, where $\sim$ is one of $<$, $\leq$, $=$, $\geq$, or $>$. Interval constraints can also be encoded in TATL; for example, $\langle\langle \mathfrak{P} \rangle\rangle \varphi_1 \mathcal{U}_{(d_1, d_2]}\, \varphi_2$ stands for $x \cdot \langle\langle \mathfrak{P} \rangle\rangle \varphi_1 \mathcal{U}\, y \cdot (\varphi_2 \wedge y > x + d_1 \wedge y \leq x + d_2)$. We write $\Diamond \varphi$ for $\textsc{true} \mathcal{U} \varphi$ as usual, and therefore $\langle\langle \mathfrak{P} \rangle\rangle \Diamond_{\sim d}\, \varphi$ stands for $x \cdot \langle\langle \mathfrak{P} \rangle\rangle \Diamond y \cdot (\varphi \wedge y \sim x + d)$.

4.3 TATL*

TATL is a fragment of the more expressive logic called TATL*. There are two types of formulae in TATL*: *state formulae*, whose satisfaction is related to a particular state, and *path formulae*, whose satisfaction is related to a specific run. Formally, a TATL* state formula is one of the following:

(S1) TRUE or p for propositions $p \in \Sigma$.

(S2) $\neg\varphi$ or $\varphi_1 \wedge \varphi_2$ for TATL* state formulae φ, φ_1, and φ_2.

(S3) $x + d_1 \leq y + d_2$ for clocks $x, y \in D$ and nonnegative integer constants d_1, d_2.

(S4) $\langle\langle \mathfrak{P} \rangle\rangle \psi$ for $\mathfrak{P} \subseteq \{1, 2\}$ and TATL* path formulae ψ.

A TATL* path formula is one of the following:

(P1) A TATL* state formula.

(P2) $\neg\psi$ or $\psi_1 \wedge \psi_2$ for TATL* path formulae ψ, ψ_1, and ψ_2.

(P3) $x \cdot \psi$ for formula clocks $x \in D$ and TATL* path formulae ψ.

(P4) $\psi_1 \mathcal{U} \psi_2$ for TATL* path formulae ψ_1, ψ_2.

The logic TATL* consists of the formulae generated by the rules S1–S4. As in TATL, we assume that there is no nested reuse of formula clocks. Additional temporal operators are defined as usual; for example, $\Diamond\varphi$ stands for TRUE$\mathcal{U}\varphi$, and $\Box\varphi$ stands for $\neg\Diamond\neg\varphi$. The logic TATL can be viewed as a fragment of TATL* consisting of formulae in which every $\mathcal{U}$ operator is immediately preceeded by a $\langle\langle \mathfrak{P} \rangle\rangle$ operator, possibly with an intermittent negation symbol [AHK02].

The semantics of TATL* formulae are defined with respect to an environment $\mathcal{E} : D \mapsto \mathbb{R}_{\geq 0}$. We write $(s, t, \mathcal{E}) \models \varphi$ to indicate that the state s of the timed game structure $\mathcal{G}$ satisfies the TATL* state formula φ at time $t \in \mathbb{R}_{\geq 0}$; and $(r, \tau, t, \mathcal{E}) \models \psi$ to indicate that the suffix of the run r of $\mathcal{G}$ which starts from game time $\tau \in \mathsf{GameTimes}(r)$ satisfies the TATL* path formula ψ, provided the time at the initial state of r is t. Unlike TATL, we allow all strategies for both players (including non-receptive strategies), because we will see that the use of receptive strategies can be enforced within TATL* by certain path formulae. Formally, the satisfaction relation $\models$ is defined inductively as follows. For state formulae φ,

- $(s, t, \mathcal{E}) \models$ TRUE.

- $(s, t, \mathcal{E}) \models p$, for a proposition p, iff $p \in \sigma(s)$.
- $(s, t, \mathcal{E}) \models \neg\varphi$ iff $(s, t, \mathcal{E}) \not\models \varphi$.
- $(s, t, \mathcal{E}) \models \varphi_1 \wedge \varphi_2$ iff $(s, t, \mathcal{E}) \models \varphi_1$ and $(s, t, \mathcal{E}) \models \varphi_2$.
- $(s, t, \mathcal{E}) \models x + d_1 \leq y + d_2$ iff $\mathcal{E}(x) + d_1 \leq \mathcal{E}(y) + d_2$.
- $(s, t, \mathcal{E}) \models \langle\langle \mathfrak{P} \rangle\rangle \psi$ iff there is a team-$\mathfrak{P}$ strategy $\pi_{\mathfrak{P}}$ such that for all runs $r \in \mathsf{Outcomes}(s, \pi_{\mathfrak{P}})$, we have $(r, \langle 0, 0\rangle, t, \mathcal{E}) \models \psi$.

For path formulae ψ,

- $(r, \langle u, k\rangle, t, \mathcal{E}) \models \varphi$, for a state formula φ, iff $(\mathsf{state}(r, \langle u, k\rangle), t + u, \mathcal{E}) \models \varphi$.
- $(r, \tau, t, \mathcal{E}) \models \neg\psi$ iff $(r, \tau, t, \mathcal{E}) \not\models \psi$.
- $(r, \tau, t, \mathcal{E}) \models \psi_1 \wedge \psi_2$ iff $(r, \tau, t, \mathcal{E}) \models \psi_1$ and $(r, \tau, t, \mathcal{E}) \models \psi_2$.
- $(r, \langle u, k\rangle, t, \mathcal{E}) \models x{\cdot}\psi$ iff $(r, \langle u, k\rangle, t, \mathcal{E}[x := t + u]) \models \psi$.
- $(r, \tau, t, \mathcal{E}) \models \psi_1 \mathcal{U} \psi_2$ iff there is a $\tau' \in \mathsf{GameTimes}(r)$ such that $\tau \leq \tau'$ and $(r, \tau', t, \mathcal{E}) \models \psi_2$, and for all $\tau'' \in \mathsf{GameTimes}(r)$ with $\tau \leq \tau'' < \tau'$, we have $(r, \tau'', t, \mathcal{E}) \models \psi_1$.

A state s of the timed game structure $\mathcal{G}$ *satisfies* a closed formula φ of TATL*, denoted $s \models \varphi$, if $(s, 0, \mathcal{E}) \models \varphi$ for any environment $\mathcal{E}$.

4.4 Model Checking TATL

We restrict our attention to timed automaton games. Given a closed TATL (resp. TATL*) formula φ, a timed automaton game $\mathcal{T}$, and a state s of the timed game structure $[\![\mathcal{T}]\!]$, the *model-checking problem* is to determine whether $s \models_{\mathrm{td}} \varphi$ (resp., $s \models \varphi$). The alternating-time logic TATL* subsumes the linear-time logic TPTL [AH94]. Thus the model-checking problem for TATL* is undecidable. On the other hand, we now solve the model-checking problem for TATL by reducing it to a special kind of TATL* problem, which turns out to be decidable.

Given a TATL formula φ over the set D of formula clocks, and a timed automaton game $\mathcal{T}$, we look at the timed automaton game $\mathcal{T}_\varphi$ with the set $C_\varphi = C \uplus D$ of clocks (we assume $C \cap D = \emptyset$). Let c_x be the largest constant to which the formula variable x is compared in φ. We pick an invariant $\gamma(l)$ in $\mathcal{T}$ and modify it to $\gamma(l)' = \gamma(l) \wedge (x \leq c_x \vee x \geq c_x)$ in $\mathcal{T}_\varphi$ for every formula clock $x \in D$ (this is to inject the proper constants in the region equivalence relation). Thus, $\mathcal{T}_\varphi$ acts exactly like $\mathcal{T}$ except that it contains some extra clocks which are never used.

As in Chapter 3, we represent the sets $\mathsf{Timediv}$ and $\mathsf{Blameless}_i$ using ω-regular conditions. Since both players appear in TATL objectives, we need the variable bl_2 in addition to bl_1. Thus, we look at the enlarged automaton game structure $\widehat{\llbracket \mathcal{T}_\varphi \rrbracket}$ with the state space $\widehat{S} = S_\varphi \times \{T, F\}^3$, and an augmented transition relation $\widehat{\delta}_{\mathsf{jd}} : \widehat{S} \times M_1 \times M_2 \mapsto 2^{\widehat{S}}$. In an augmented state $\langle s, \mathit{tick}, bl_1, bl_2 \rangle \in \widehat{S}$, the component $s \in S_\varphi$ is a state of the original game structure $\llbracket \mathcal{T}_\varphi \rrbracket$, tick is true if the global clock z has crossed an integer boundary in the last transition, and bl_i is true if player i is to blame for the last transition. It can be seen that a run is in $\mathsf{Timediv}$ iff tick is true infinitely often, and that the set $\mathsf{Blameless}_i$ corresponds to runs along which bl_i is true only finitely often. We extend the clock equivalence relation to these expanded states: $\langle \langle l, \kappa \rangle, \mathit{tick}, bl_1, bl_2 \rangle \cong \langle \langle l', \kappa' \rangle, \mathit{tick}', bl_1', bl_2' \rangle$ iff $l = l', \mathit{tick} = \mathit{tick}', bl_1 = bl_1', bl_2 = bl_2'$ and $\kappa \cong \kappa'$. Finally, we extend bl to teams: $bl_\emptyset = \text{FALSE}, bl_{\{1,2\}} = \text{TRUE}, bl_{\{i\}} = bl_i$.

We will use the algorithms of Chapter 3 which compute winning sets for timed automaton games with untimed ω-regular objectives. We first consider the subset of TATL in which formulae are clock variable free. Using the encoding for time divergence and blame predicates, we can embed the notion of receptive winning strategies into TATL* formulae.

Lemma 11. *A state s in a timed game structure $\llbracket \mathcal{T}_\varphi \rrbracket$ satisfies a formula clock variable free* TATL *formula φ in a meaningful way, denoted $s \models_{\text{td}} \varphi$, iff the state $\widehat{s} = \langle s, \text{FALSE}, \text{FALSE}, \text{FALSE} \rangle$ in the expanded game structure $\widehat{\llbracket \mathcal{T}_\varphi \rrbracket}$ satisfies the* TATL* *formula* $\mathsf{atlstar}(\varphi)$*, that is, iff $\widehat{s} \models \mathsf{atlstar}(\varphi)$ where* $\mathsf{atlstar}$ *is a partial mapping from* TATL *to* TATL**, defined inductively as follows:*

$$
\begin{array}{l}
\mathsf{atlstar}(\text{TRUE}) = \text{TRUE} \\
\mathsf{atlstar}(p) = p \\
\mathsf{atlstar}(\neg\varphi) = \neg\, \mathsf{atlstar}(\varphi); \qquad \mathsf{atlstar}(\varphi_1 \wedge \varphi_2) = \mathsf{atlstar}(\varphi_1) \wedge \mathsf{atlstar}(\varphi_2) \\
\mathsf{atlstar}(\langle\langle \mathfrak{P} \rangle\rangle \Box \varphi) = \langle\langle \mathfrak{P} \rangle\rangle \left((\Box\Diamond\, \mathit{tick} \rightarrow \Box\, \mathsf{atlstar}(\varphi)) \wedge (\Diamond\Box\neg\, \mathit{tick} \rightarrow \Diamond\Box\neg\, bl_{\mathfrak{P}}) \right) \\
\mathsf{atlstar}(\langle\langle \mathfrak{P} \rangle\rangle \varphi_1 \mathcal{U} \varphi_2) = \langle\langle \mathfrak{P} \rangle\rangle \left(\begin{array}{c} (\Box\Diamond\, \mathit{tick} \rightarrow \mathsf{atlstar}(\varphi_1)\, \mathcal{U}\, \mathsf{atlstar}(\varphi_2)) \wedge \\ (\Diamond\Box\neg\, \mathit{tick} \rightarrow \Diamond\Box\neg\, bl_{\mathfrak{P}}) \end{array} \right)
\end{array}
$$

Now, for φ a clock variable free TATL formula, $\mathsf{atlstar}(\varphi)$ is actually an ATL* formula. Thus, the untimed ω-regular model checking algorithm of [dAFH$^+$03] can be used to (recursively) model check $\mathsf{atlstar}(\varphi)$. As we are working in the continuous domain, we need to ensure that for an until formula $\langle\langle\mathfrak{P}\rangle\rangle\varphi_1\mathcal{U}\varphi_2$, team $\mathfrak{P}$ does not "jump" over a time at which $\neg(\varphi_1 \vee \varphi_2)$ holds. This can be handled by introducing another player in the opposing team $\sim\mathfrak{P}$, the *observer*, who can only take pure time moves. The observer entails the opposing team to observe *all* time points. The observer is necessary only when $\mathfrak{P} = \{1, 2\}$. We omit the details.

A naive extension of the above approach to full TATL does not immediately work, for then we get TATL* formulae which are not in ATL* (model checking for TATL* is undecidable). We do the following: for each formula clock constraint $x + d_1 \leq y + d_2$ appearing in the formula φ, let there be a new proposition p_α for $\alpha = x + d_1 \leq y + d_2$. We denote the set of all such formula clock constraint propositions by Λ. A state $\langle l, \kappa\rangle$ in the timed automaton game $\mathcal{T}_\varphi$ satisfies p_α for $\alpha = x + d_1 \leq y + d_2$ iff $\kappa(x) + d_1 \leq \kappa(y) + d_2$. The propositions p_α are invariant over regions, maintaining the region-invariance of sets in the algorithms of Chapter 3. We note that in applying the reduction of Section 3.4 of Chapter 3, we need to construct a separate finite state game graph for each team $\mathfrak{P}$.

Lemma 12. *For a* TATL *formula* φ*, let* φ^Λ *be obtained from* φ *by replacing all formula variable constraints* $x + d_1 \leq y + d_2$ *with equivalent propositions* $p_\alpha \in \Lambda$*. Let* $[\![\mathcal{T}_\varphi]\!]^\Lambda$ *denote the timed game structure* $[\![\mathcal{T}_\varphi]\!]$ *together with the propositions from* Λ*. Then,*

1. *We have* $s \models_{\mathrm{td}} \varphi$ *for a state* s *in the timed game structure* $[\![\mathcal{T}_\varphi]\!]$ *iff the state* $s \models_{\mathrm{td}} \varphi^\Lambda$ *in* $[\![\mathcal{T}_\varphi]\!]^\Lambda$*.*

2. *Let* $\varphi^\Lambda = w{\cdot}\psi^\Lambda$*. Then, in the structure* $[\![\mathcal{T}_\varphi]\!]^\Lambda$ *the state* $s \models_{\mathrm{td}} \varphi^\Lambda$ *iff* $s[w := 0] \models_{\mathrm{td}} \psi^\Lambda$*.*

3. *Let* $\varphi = \langle\langle\mathfrak{P}\rangle\rangle\Box p \mid \langle\langle\mathfrak{P}\rangle\rangle p_1\mathcal{U}p_2$*, where* p, p_1, p_2 *are propositions that are invariant over states of regions in* $\mathcal{T}_\varphi$*. Then for* $s \cong s'$ *in* $\mathcal{T}_\varphi$*, we have* $s \models_{\mathrm{td}} \varphi$ *iff* $s' \models_{\mathrm{td}} \varphi$*.*

Lemmas 11 and 12 together with the EXPTIME algorithm for timed automaton games with untimed ω-regular region objectives give us a recursive model-checking algorithm for TATL.

Theorem 13. *The model-checking problem for* TATL *(over timed automaton games) is* EXPTIME*-complete.*

EXPTIME-hardness follows from the EXPTIME-hardness of alternating reachability on timed automata [HK99].

Chapter 5

Minimum-Time Reachability in Timed Games

5.1 Introduction

In this chapter we consider the problem of *minimum-time reachability* in timed games, where we ask what is the earliest time at which player 1 is able to guarantee the satisfaction of a proposition. This is the quantitative version of the classical reachability query and is useful in competitive optimization problems. We work in the framework of Chapter 3 where both players must only use receptive strategies in the timed game. We illustrate the problem with the following example.

Example 7. *Consider the game depicted in Figure 5.1. Let edge a be controlled by player-1; the others being controlled by player-2. Suppose we want to know what is the earliest time that player-1 can reach p starting from the state $\langle \neg p, x = 0, y = 0\rangle$ (i.e., the initial values of both clocks x and y are 0). Player-1 is not able to guarantee time divergence, as player-2 can keep on choosing the edge b_1. On the other hand, we do not want to put any restriction of the number of times that player-2 chooses b_1. Requiring that the players use only receptive strategies avoids such unnecessary restrictions, and gives the correct minimum time for player-1 to reach p, namely, 101 time units.*

We present an EXPTIME algorithm to compute the minimum time needed by player-1 to force the game into a target location, with both players restricted to using only receptive strategies (note that reachability in timed automaton games is EXPTIME-

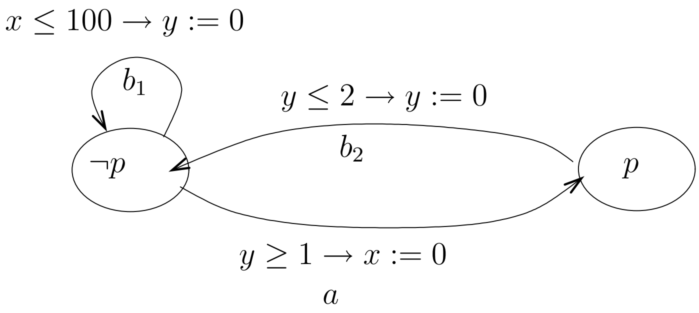

$x \leq 100 \rightarrow y := 0$
b_1
$y \leq 2 \rightarrow y := 0$
b_2
$\neg p$
p
$y \geq 1 \rightarrow x := 0$
a

Figure 5.1: A timed automaton game.

complete [HK99]). We first show that the minimum time can be obtained by solving a certain μ-calculus fixpoint equation. We then give a proof of termination for the fixpoint evaluation. This requires an important new ingredient: an extension of the clock-region equivalence [AD94] for timed automata. We show our extended region equivalence classes to be stable with respect to the monotone functions used in the fixpoint equation.

We note that standard clock regions do not suffice for the solution. The minimum-time reachability game has two components: a reachability part that can be handled by discrete arguments based on the clock-region graph; and a minimum-time part that requires minimization within clock regions (cf. [CY92]). Unfortunately, both arguments are intertwined and cannot be considered in isolation. Our extended regions decouple the two parts in the proofs. We also note that region sequences that correspond to time-minimal runs may in general be required to contain region cycles in which time does not progress by an integer amount; thus a reduction to a loop-free region game, as in [AdAF05], is not possible.

Outline. The minimum-time reachability problem is defined in Section 5.2. The problem is reduced to finding the winning set for simple reachability in Section 5.3. We show in Section 5.4 that reachability problem can solved using the classical μ-calculus algorithm. The algorithm runs in time exponential in the number of clocks and the size of clock constraints.

5.2 The Minimum-Time Reachability Problem

Let $\mathcal{G}$ be a well-formed timed game with propositions on locations as described in Chapters 3 and 4. The *minimum-time reachability problem* is to determine the minimal time in which a player can force the game into a set of target states, using only receptive strategies. Formally, given a timed game $\mathcal{G}$, a target proposition $p \in \Sigma$, and a run r of $\mathcal{T}$, let

$$T_{\text{visit}}(\mathcal{G}, r, p) = \begin{cases} \infty \;\; \text{if } r \text{ does not visit } p; \\ \inf \{t \in \mathbb{R}_{\geq 0} \mid p \in \sigma(\mathsf{state}(r, \langle t, k \rangle)) \text{ for some } k\} \text{ otherwise.} \end{cases}$$

The *minimal time* for player-1 to force the game from a start state $s \in S$ to a visit to p is then

$$T_{\min}(\mathcal{G}, s, p) = \inf_{\pi_1 \in \Pi_1^R} \sup_{\pi_2 \in \Pi_2^R} \sup_{r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)} T_{\text{visit}}(\mathcal{T}, r, p)$$

We omit $\mathcal{G}$ when clear from the context. We restrict our attention to well-formed timed automaton games. The definition of $T_{\min}$ quantifies strategies over the set of receptive strategies. Our algorithm will instead work over the set of *all* strategies. Theorem 14 presents this reduction. We will then present a game structure for the timed automaton game $\mathcal{T}$ in which $\mathsf{Timediv}$ and $\mathsf{Blameless}_1$ can be represented using Büchi and co-Büchi constraints as in Chapter 3. In addition, our game structure will also have a backwards running clock, which will be used in the computation of the minimum time, using a μ-calculus algorithm on *extended regions.*

Allowing Players to Use All Strategies. To allow quantification over all strategies, we first modify the payoff function T_{visit}, so that players are maximally penalized on zeno runs:

$$T^{\text{UR}}_{\text{visit}}(r, p) = \begin{cases} \infty \;\; \text{if } r \notin \mathsf{Timediv} \text{ and } r \notin \mathsf{Blameless}_i; \\ \infty \;\; \text{if } r \in \mathsf{Timediv} \text{ and } r \text{ does not visit } p; \\ 0 \;\; \text{if } r \notin \mathsf{Timediv} \text{ and } r \in \mathsf{Blameless}_i; \\ \inf \{t \in \mathbb{R}_{\geq 0} \mid p \in \sigma(\mathsf{state}(r, \langle t, k \rangle)) \text{ for some } k\} \;\; \text{otherwise.} \end{cases}$$

It turns out that penalizing on zeno-runs is equivalent to penalizing on non-receptive strategies:

Theorem 14. *Let s be a state and p a proposition in a well-formed timed game structure $\mathcal{G}$. Then:*

$$T_{\min}(s, p) = \inf_{\pi_1 \in \Pi_1} \sup_{\pi_2 \in \Pi_2} \sup_{r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)} T^{\text{UR}}_{\text{visit}}(r, p)$$

Proof. We restrict our attention to strategies for plays starting from state s. The proof of the theorem relies on Lemmas 13,14 and 15 which we present next. □

Lemma 13. *Consider a timed game structure $\mathcal{G}$ and a state $s \in S$. Let $\pi_1 \in \Pi_1^R$ and $\pi_2^R \in \Pi_2^R$ be player-1 and player-2 receptive strategies, and let $\pi_2 \in \Pi_2$ be any player-2 strategy such that $\mathsf{Outcomes}(s, \pi_1, \pi_2) \cap \mathsf{Timediv} \neq \emptyset$. Let $r^* \in \mathsf{Outcomes}(s, \pi_1, \pi_2) \cap \mathsf{Timediv}$. Consider a player-2 strategy π_2^* be defined as, $\pi_2^*(r[0..k]) = \pi_2(r^*[0..k])$ for all run prefixes $r[0..k]$ of r^*, and $\pi_2^*(r[0..k]) = \pi_2^R(r[k'..k])$ otherwise, where k' is the first position such that $r[0..k']$ is not a run prefix of r^*. Then, π_2^* is a receptive strategy.*

Proof. Intuitively, the strategy π_2^* acts like π_2 on r^* , and like π_2^R otherwise. Consider any player-1 strategy $\pi_1' \in \Pi_1$, and any run $r \in \mathsf{Outcomes}(s, \pi_1', \pi_2^*)$. If $r = r^*$, then $r \in \mathsf{Timediv}$. Suppose $r \neq r^*$. Let $k' \geq 0$ be the first step in the game (with player-2 strategy π_2^*) which witnesses the fact that $r \neq r^*$, that is, 1) we have $r[0..k'-1]$ to be a run prefix of r^*, and 2) $r[0..k']$ to not be a run prefix of r^* Consider the state $s_{k'} = r[k']$. After this point (ie., from $r[0..k']$ onwards), the strategy π_2^* behaves like π_2^R when "started" from $s_{k'}$. Since π_2^R is a receptive player-2 strategy, we have $\mathsf{Outcomes}(s_{k'}, \pi_1', \pi_2^*) \subseteq \mathsf{Timediv} \cup \mathsf{Blameless}_2$. Thus, $r \in \mathsf{Timediv} \cup \mathsf{Blameless}_2$ (finite prefixes of runs do not change membership in these sets). Hence π_2^* is a receptive player-2 strategy. □

Lemma 14. *Consider a timed game structure $\mathcal{G}$ and a state $s \in S$. We have,*

$$\inf_{\pi_1 \in \Pi_1} \sup_{\pi_2 \in \Pi_2} \sup_{r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)} T_{\mathsf{visit}}^{\mathrm{UR}}(r, p) = \inf_{\pi_1 \in \Pi_1^R} \sup_{\pi_2 \in \Pi_2} \sup_{r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)} T_{\mathsf{visit}}^{\mathrm{UR}}(r, p)$$

Proof. Consider any $\pi_1 \in \Pi_1 \setminus \Pi_1^R$. There exists $\pi_2 \in \Pi_2$ such that $\mathsf{Outcomes}(s, \pi_1, \pi_2) \not\subseteq \mathsf{Timediv} \cup \mathsf{Blameless}_1$. Thus, $\inf_{\pi_1 \in \Pi_1 \setminus \Pi_1^R} \sup_{\pi_2 \in \Pi_2} \sup_{r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)} T_{\mathsf{visit}}^{\mathrm{UR}}(r, p) = \infty$. □

Lemma 15. *Consider a timed game structure $\mathcal{G}$ and a state $s \in S$. For every player-1 receptive strategy $\pi_1 \in \Pi_1^R$, we have $\sup_{\pi_2 \in \Pi_2} \sup_{r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)} T_{\mathsf{visit}}^{\mathrm{UR}}(r, p) = \sup_{\pi_2 \in \Pi_2^R} \sup_{r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)} T_{\mathsf{visit}}^{\mathrm{UR}}(r, p)$.*

Proof. Let $\pi_2 \in \Pi_2$.

Consider $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)$. Since π_1 is receptive, we cannot have $r \notin \mathsf{Timediv}$ and $r \notin \mathsf{Blameless}_1$.

Suppose $r \notin \mathsf{Timediv}$. Then $r \in \mathsf{Blameless}_1$. In this case, $0 = T_{\mathsf{visit}}^{\mathrm{UR}}(r, p) \leq T_{\mathsf{visit}}^{\mathrm{UR}}(r', p)$ for any $r' \in \mathsf{Outcomes}(s, \pi_1, \pi_2^R)$ and π_2^R any player-2 receptive strategy (as we have a well-formed time game structure, there exists some receptive strategy π_2^R).

Suppose $r \in \mathsf{Timediv}$ and r does not visit p. Consider the strategy π_2^* which acts like π_2 on r, and like π_2^R otherwise, as formally defined in Lemma 13. We have π_2^* to be receptive. Clearly $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2^*)$ does not visit p, and hence $\sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2)} T_{\text{visit}}^{\text{UR}}(r,p) = \sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2^*)} T_{\text{visit}}^{\text{UR}}(r,p) = \infty$.

Finally, let r visit p and be in $\mathsf{Timediv}$. Let π_2^* be a player-2 receptive strategy as in Lemma 13. We again have $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2^*)$, and hence $\sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2)} T_{\text{visit}}^{\text{UR}}(r,p) \leq \sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2^*)} T_{\text{visit}}^{\text{UR}}(r,p)$.

Thus, $\sup_{\pi_2 \in \Pi_2} \sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2)} T_{\text{visit}}^{\text{UR}}(r,p) = \sup_{\pi_2 \in \Pi_2^R} \sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2)} T_{\text{visit}}^{\text{UR}}(r,p)$. □

Lemmas 14 and 15 together imply

$$\inf_{\pi_1 \in \Pi_1} \sup_{\pi_2 \in \Pi_2} \sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2)} T_{\text{visit}}^{\text{UR}}(r,p) = \inf_{\pi_1 \in \Pi_1^R} \sup_{\pi_2 \in \Pi_2^R} \sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2)} T_{\text{visit}}^{\text{UR}}(r,p)$$

Theorem 14 follows from the fact that for $\pi_1 \in \Pi_1^R, \pi_2 \in \Pi_2^R$ and $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)$, we have $T_{\text{visit}}^{\text{UR}}(r,p) = T_{\text{visit}}(r,p)$.

5.3 Reduction to Reachability with Büchi and co-Büchi Constraints

We now decouple reachability from optimizing for minimal time, and show how reachability with time divergence can be solved for, using an appropriately chosen μ-calculus fixpoint.

Lemma 16. *Given a state s, and a proposition p of a well-formed timed automaton game $\mathcal{T}$, 1)we can determine if $T_{\min}(s,p) < \infty$, and 2) if $T_{\min}(s,p) < \infty$, then $T_{\min}(s,p) < M = 8|L| \cdot \prod_{x \in C}(c_x + 1) \cdot |C + 1|! \cdot 2^{|C|}$. This upper bound is the same for all $s' \cong s$.*

Proof. 1. $T_{\min}(s,p) < \infty$ iff player 1 has a strategy to reach p from state s, and this can be determined using the algorithms of Chapter 3. (here we have used the syntax of TATL from Chapter 4).

2. Suppose $T_{\min}(s,p) < \infty$. This means there is a player-1 strategy π_1 such that for all opposing strategies π_2 of player 2, and for all runs $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)$ we have that, 1) if time diverges in run r then r contains a state satisfying p, and 2) if time does not diverge in r, then player 1 is blameless. Suppose that for all $d > 0$ we have $s \not\models_{\text{td}} \langle\langle 1 \rangle\rangle \Diamond_{\leq d}\, p$. We have that player 1 cannot win for his objective of $\Diamond_{\leq d}\, p$,

in particular, π_1 is not a winning strategy for this new objective. Hence, there is a player-2 strategy π_2^d such that for some run $r_d \in \mathsf{Outcomes}(s, \pi_1, \pi_2^d)$ either 1) time converges and player 1 is to blame or 2) time diverges in run r_d and r_d contains a location satisfying p, but not before time d. Player 1 does not have anything to gain by blocking time, so assume time diverges in run r_d (or equivalently, assume π_1 to be a receptive strategy). The only way strategies π_2^d and runs r_d can exist for every $d > 0$ is if player 2 can *force* the game (while avoiding p) so that a portion of the run lies in a region cycle $R_{k_1}, \ldots R_{k_m}$, with *tick* being true in one of the regions of the cycle (note that a system may stay in a region for at most one time unit). Now, if a player can control the game from state s so that the next state lies in region R, then he can do the same from any state s' such that $s' \cong s$. Thus, it must be that player 2 has a strategy π_2^* such that a run in $\mathsf{Outcomes}(s, \pi_1, \pi_2^*)$ corresponds to the region sequence $R_0, \ldots, R_k, (R_{k_1}, \ldots R_{k_m})^\omega$, with none of the regions satisfying p. Time diverges in this run as *tick* is infinitely often true due to the repeating region cycle. This contradicts the fact the π_1 was a winning strategy for player 1 for $\langle\langle 1 \rangle\rangle \Diamond p$. Thus, it cannot be that for all $d > 0$, player 2 has a strategy π_2^d such that for some run $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2^d)$, time diverges in run r and r contains a state satisfying p, but not before time d.

□

Let M be the upper bound on $T_{\min}(s, p)$ as in Lemma 16 if $T_{\min}(s, p) < \infty$, and $M = 1$ otherwise. For a number N, let $\mathbb{R}_{[0,N]}$ and $\mathbb{R}_{[0,N)}$ denote $\mathbb{R} \cap [0, N]$ and $\mathbb{R} \cap [0, N)$ respectively. We first look at the enlarged game structure $\widehat{[\![\mathcal{T}]\!]}$ with the state space $\widehat{S} = S \times \mathbb{R}_{[0,1)} \times (\mathbb{R}_{[0,M]} \cup \{\bot\}) \times \{\text{TRUE}, \text{FALSE}\}^2$, and an augmented transition relation $\widehat{\delta} : \widehat{S} \times (M_1 \cup M_2) \mapsto \widehat{S}$. In an augmented state $\langle s, \mathfrak{z}, \beta, \mathit{tick}, \mathit{bl}_1 \rangle \in \widehat{S}$, the component $s \in S$ is a state of the original game structure $[\![\mathcal{T}]\!]$, $\mathfrak{z}$ is value of a fictitious clock z which gets reset every time it hits 1, β is the value of a fictitious clock which is running *backwards*, *tick* is true iff the last transition resulted in the clock z hitting 1 (so *tick* is true iff the last transition resulted in $\mathfrak{z} = 0$), and bl_1 is true if player-1 is to blame for the last transition.

Formally, $\langle s', \mathfrak{z}', \beta', \mathit{tick}', \mathit{bl}_1' \rangle = \widehat{\delta}(\langle s, \mathfrak{z}, \beta, \mathit{tick}, \mathit{bl}_1 \rangle, \langle \Delta, a_i \rangle)$ iff

1. $s' = \delta(s, \langle \Delta, a_i \rangle)$

2. $\mathfrak{z}' = (\mathfrak{z} + \Delta) \bmod 1$;

3. $\beta' = \beta \ominus \Delta$, where we define $\beta \ominus \Delta$ as $\beta - \Delta$ if $\beta \neq \bot$ and $\beta - \Delta \geq 0$, and $\bot$ otherwise ($\bot$ is an absorbing value for β).

4. $tick' = \text{TRUE}$ if $\mathfrak{z} + \Delta \geq 1$, and FALSE otherwise

5. $bl_1 = \text{TRUE}$ if $a_i \in A_1^{\bot}$ and FALSE otherwise.

Each run r of $[\![\mathfrak{T}]\!]$, and values $\mathfrak{z} \in \mathbb{R}_{\geq 0}, \beta \leq M$ can be mapped to a corresponding unique run $\widehat{r}_{\mathfrak{z},\beta}$ in $\widehat{[\![\mathfrak{T}]\!]}$, with $\widehat{r}_{\mathfrak{z},\beta}[0] = \langle r[0], \mathfrak{z}, \beta, \text{FALSE}, \text{FALSE}\rangle$. Similarly, each run $\widehat{r}$ of $\widehat{[\![\mathfrak{T}]\!]}$ can be projected to a unique run $\widehat{r} \downarrow \mathfrak{T}$ of $[\![\mathfrak{T}]\!]$. It can be seen that the run r is in $\mathsf{Timediv}$ iff *tick* is true infinitely often in $\widehat{r}_{\mathfrak{z},\beta}$, and that the set $\mathsf{Blameless}_1$ corresponds to runs along which bl_1 is true only finitely often.

Proposition 4. *Consider the set S_p for a proposition p in a timed game structure $[\![\mathfrak{T}]\!]$.*

1. *If a run r of $[\![\mathfrak{T}]\!]$ visits S_p at time $t \leq M$, then, the run $\widehat{r}_{0,\beta}$ visits $S_p \times \mathbb{R}_{[0,1)} \times \{0\} \times \{\text{TRUE}, \text{FALSE}\}^2$, for $\beta = t$.*

2. *If for some $\beta \in \mathbb{R}$, a run $\widehat{r}$ of $\widehat{[\![\mathfrak{T}]\!]}$ with $\widehat{r}[0] = \langle s, 0, \beta, \text{FALSE}, \text{FALSE}\rangle$ visits $S_p \times \mathbb{R}_{[0,1)} \times \{0\} \times \{\text{TRUE}, \text{FALSE}\}^2$, then the corresponding run $r = \widehat{r} \downarrow \mathfrak{T}$ of $[\![\mathfrak{T}]\!]$ visits S_p at time $t = \beta$.*

Proposition 4 is a straightforward result of the fact that β is kept decrementing at rate -1 till it hits 0.

Lemma 17. *Given a timed game structure $[\![\mathfrak{T}]\!]$, let $\widehat{X}_p = S_p \times \mathbb{R}_{[0,1)} \times \{0\} \times \{\text{TRUE}, \text{FALSE}\}^2$.*

1. *For a run r of the timed game structure $[\![\mathfrak{T}]\!]$, let $T_{\text{visit}}(r,p) < \infty$. Then, $T_{\text{visit}}(r,p) = \inf\{\beta \mid \beta \in \mathbb{R}_{[0,M]}$ and $\widehat{r}_{0,\beta}$ visits the set $\widehat{X}_p\}$.*

2. *Let $T_{\min}(s,p) < \infty$. Then,*
$$T_{\min}(s,p) = \inf\left\{ \ \beta \mid \beta \in \mathbb{R}_{[0,M]} \text{ and } \langle s, 0, \beta, \text{FALSE}, \text{FALSE}\rangle \in \langle\langle 1\rangle\rangle \Diamond \widehat{X}_p \ \right\}$$

3. *If $T_{\min}(s,p) = \infty$, then for all β, we have $\langle s, 0, \beta, \text{FALSE}, \text{FALSE}\rangle \notin \langle\langle 1\rangle\rangle \Diamond \widehat{X}_p$.*

Proof. 1. The first claim is a corollary of Proposition 4.

2. The second claim of Lemma 17 essentially follows from the fact that the additional components in the states do not help the players in creating more powerful strategies. $T_{\min}(s,p)$

$$= \inf_{\pi_1 \in \Pi_1^R} \sup_{\pi_2 \in \Pi_2^R} \sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2)} T_{\text{visit}}([\![\mathcal{T}]\!], r, p)$$

$$= \inf_{\pi_1 \in \Pi_1^R} \sup_{\pi_2 \in \Pi_2^R} \sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2)} \left\{ \begin{array}{l} \infty \text{ if } r \text{ does not visit } p; \\ \inf \left\{ \begin{array}{l} \beta \mid \beta \in \mathbb{R}_{[0,M]} \text{ and} \\ \widehat{r}_{0,\beta} \text{ visits the set } \widehat{X}_p \end{array} \right\} \text{ o.w.} \end{array} \right\}$$

$$= \inf_{\pi_1 \in \Pi_1^R} \sup_{\pi_2 \in \Pi_2^R} \sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2)} \inf_{\beta \in \mathbb{R}_{[0,M]}} \left\{ g(r,\beta) \,\middle|\, g(r,\beta) = \infty \text{ if } \widehat{r}_{0,\beta} \text{ does not visit } \widehat{X}_p;\ \beta \text{ otherwise } \right\}$$

$$= \inf_{\beta \in \mathbb{R}_{[0,M]}} \inf_{\pi_1 \in \Pi_1^R} \sup_{\pi_2 \in \Pi_2^R} \sup_{r \in \mathsf{Outcomes}(s,\pi_1,\pi_2)} \left\{ g(r,\beta) \,\middle|\, g(r,\beta) = \infty \text{ if } \widehat{r}_{0,\beta} \text{ does not visit } \widehat{X}_p;\ \beta \text{ otherwise } \right\}$$

Now, considering plays in $\widehat{[\![\mathcal{T}]\!]}$ which start from state $\widehat{s} = \langle s, z, \beta, \mathit{tick}, \mathit{bl}_1 \rangle$, every strategy $\widehat{\pi_i} \in \widehat{\Pi_i}$ is equivalent to a strategy $\pi_i \in \Pi_i$ in which player-i "guesses" the values of $z, \beta, \mathit{tick}, \mathit{bl}_1$. Once these initial values have been guessed, each player can keep on deterministically updating the values at each step. Hence observation of the additional components in states of $\widehat{[\![\mathcal{T}]\!]}$ do not help the players in their strategies. Therefore,

$$T_{\min}(s,p) = \inf_{\beta \in \mathbb{R}_{[0,M]}} \inf_{\widehat{\pi_1} \in \widehat{\Pi_1}^R} \sup_{\widehat{\pi_2} \in \widehat{\Pi_2}^R} \sup_{\widehat{r}_{0,\beta} \in \mathsf{Outcomes}(s,\widehat{\pi_1},\widehat{\pi_2})} \left\{ g(r,\beta) \,\middle|\, g(r,\beta) = \infty \text{ if } \widehat{r}_{0,\beta} \text{ does not visit } \widehat{X}_p;\ \beta \text{ otherwise } \right\}$$

3. The values of $\mathfrak{z}, \beta, \mathit{tick}$ and bl_1 do not control transitions, and hence are irrelevant in determining whether the target proposition is reached or not.

□

The reachability objective can be reduced to a parity game: each state in $\widehat{S}$ is assigned an index $\Omega : \widehat{S} \mapsto \{0,1\}$, with $\Omega(\widehat{s}) = 1$ iff $\widehat{s} \notin \widehat{X_p}$; and $\mathit{tick} \vee \mathit{bl}_1 = \text{TRUE}$. We also modify the game structure so that the states in $\widehat{X_p}$ are absorbing.

Lemma 18. *For the timed game $\widehat{[\![\mathcal{T}]\!]}$ with the reachability objective $\widehat{X}_p$, the state $\widehat{s} = \langle s, 0, \beta, \text{FALSE}, \text{FALSE} \rangle \in \langle\!\langle 1 \rangle\!\rangle \Diamond \widehat{X}_p$ iff player-1 has a strategy π_1 such that for all strategies π_2 of player-2, and all runs $\widehat{r}_{0,\beta} \in \mathsf{Outcomes}(\widehat{s}, \pi_1, \pi_2)$, the index 1 does not occur infinitely often in $\widehat{r}_{0,\beta}$.*

Proof. We first note that the states in $\widehat{X}_p$ can be absorbing as $\widehat{[\![\mathcal{G}]\!]}$ is a well-formed time game structure, and hence player-1 has a receptive strategy which does not block time when the game starts at state $\widehat{s}$ for every state $\widehat{s} \in \widehat{X}_p$. Consider a run $\widehat{r}$ such that $\widehat{r}$ visits $\widehat{X}_p$. We can assume without loss of generality that either time diverges in $\widehat{r}$, or time converges but player-1 is not to blame (player-1 can play a receptive strategy upon reaching $\widehat{X}_p$).

Thus this run satisfies the winning condition for player-1. And since $\widehat{X}_p$ is absorbing in our parity game, we see 1 only finitely often.

Consider a run $\widehat{r}$ such that $\widehat{r}$ does not visit $\widehat{X}_p$. Let time diverge in this run. This run violates the winning condition for player-1, and correspondingly we also see the index 1 infinitely often (due to *tick* being true infinitely often). Now let time converge in this run (so *tick* is true only finitely often). If player-1 is to blame for blocking time, then the index 1 will again be true infinitely often. If player-1 is not to blame, then bl_1 will only be true finitely often in this run, and hence we will see the index 1 only finitely often.

□

The fixpoint formula for solving the parity game in Lemma 18 is given by (as in [dAHM01a]),

$$Y = \mu Y \nu Z \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(Z)) \right]$$

The fixpoint expression uses the variables $Y, Z \subseteq \widehat{S}$ and the *controllable predecessor operator*, $\mathsf{CPre}_1 : 2^{\widehat{S}} \mapsto 2^{\widehat{S}}$, defined formally by $\mathsf{CPre}_1(X) \equiv \{\widehat{s} \mid \exists m_1 \in \Gamma_1(\widehat{s})\ \forall m_2 \in \Gamma_2(\widehat{s})\ (\widehat{\delta}_{\mathsf{jd}}(\widehat{s}, m_1, m_2) \subseteq X)\}$. Intuitively, $\widehat{s} \in \mathsf{CPre}_1(X)$ iff player 1 can force the augmented game from $\widehat{s}$ into X in one move.

5.4 Termination of the Fixpoint Iteration

We prove termination of the μ-calculus fixpoint iteration by demonstrating that we can work on a finite partition of the state space. Let an equivalence relation $\cong_e$ on the states in $\widehat{S}$ be defined as: $\langle\langle l^1, \kappa^1\rangle, \mathfrak{z}^1, \beta^1, tick^1, bl_1^1\rangle \cong_e \langle\langle l^2, \kappa^2\rangle, \mathfrak{z}^2, \beta^2, tick^2, bl_1^2\rangle$ iff

1. $l^1 = l^2, tick^1 = tick^2$, and $bl^1 = bl^2$.

2. $\widehat{\kappa^1} \cong \widehat{\kappa^2}$ where $\widehat{\kappa^i} : C \cup \{z\} \mapsto \mathbb{R}_{\geq 0}$ is a clock valuation such that $\widehat{\kappa^i}(c) = \kappa^i(c)$ for $c \in C$, $\widehat{\kappa^i}(z) = \mathfrak{z}^i$,and $c_z = 1$ (c_z is the maximum value of the clock z in the definition of $\cong$) for $i \in \{1, 2\}$.

3. $\beta^1 = \bot$ iff $\beta^2 = \bot$.

4. If $\beta^1 \neq \bot, \beta^2 \neq \bot$ then

   - $\lfloor \beta^1 \rfloor = \lfloor \beta^2 \rfloor$

- $\mathsf{frac}(\beta^1) = 0$ iff $\mathsf{frac}(\beta^2) = 0$.
- For each clock $x \in C \cup \{z\}$ with $\kappa^1(x) \leq c_x$ and $\kappa^2(x) \leq c_x$, we have $\mathsf{frac}(\kappa^1(x)) + \mathsf{frac}(\beta^1) \sim 1$ iff $\mathsf{frac}(\kappa^2(x)) + \mathsf{frac}(\beta^2) \sim 1$ with $\sim \in \{<, =, >\}$.

The number of equivalence classes induced by $\cong_e$ is again finite $(O\left((|L| \cdot \prod_{x \in C}(c_x + 1) \cdot |C + 1|! \cdot 2^{|C|})^2 \cdot |C|\right))$. We call each equivalence class an *extended region*. An extended region Y of $\widehat{[\![\mathcal{T}]\!]}$ can be specified by the tuple $\langle l, \mathit{tick}, \mathit{bl}_1, h, \mathcal{P}, \beta_i, \beta_f, C_<, C_=, C_> \rangle$ where for a state $\widehat{s} = \langle \langle l, \kappa \rangle, \mathfrak{z}, \beta, \mathit{tick}, \mathit{bl}_1 \rangle$,

- $l, \mathit{tick}, \mathit{bl}_1$ correspond to $l, \mathit{tick}, \mathit{bl}_1$ in $\widehat{s}$.
- h is a function which specifies the integer values of clocks: $h(x) = \lfloor \kappa(x) \rfloor$ if $\kappa(x) < C_x + 1$, and $h(x) = C_x + 1$ otherwise.
- $\mathcal{P} \subseteq 2^{C \cup \{z\}}$ is a partition of the clocks $\{C_0, \ldots, C_n \mid \uplus C_i = C \cup \{z\}, C_i \neq \emptyset \text{ for } i > 0\}$, such that 1)for any pair of clocks x, y, we have $\mathsf{frac}(\kappa(x)) < \mathsf{frac}(\kappa(y))$ iff $x \in C_j, y \in C_k$ for $j < k$; and 2)$x \in C_0$ iff $\mathsf{frac}(\kappa(x)) = 0$.
- $\beta_i \in \mathbb{N} \cap \{0, \ldots, M\} \cup \{\bot\}$ indicates the integral value of β.
- $\beta_f \in \{\textsc{true}, \textsc{false}\}$ indicates whether the fractional value of β is greater than 0, $\beta_f = \textsc{true}$ iff $\beta \neq \bot$ and $\mathsf{frac}(\beta) > 0$.
- For a clock $x \in C \cup \{z\}$ and $\beta \neq \bot$, we have $\mathsf{frac}(\kappa(x)) + \mathsf{frac}(\beta) \sim 1$ iff $x \in C_\sim$ for $\sim \in \{<, =, >\}$.

Pictorially, the relationship between $\widehat{\kappa}$ and β can be visualized as in Fig. 5.2. The figure depicts an extended region for $C_0 = \emptyset, \beta_i \in \mathbb{N} \cap \{0, \ldots, M\}, \beta_f = \textsc{true}, C_< = C \cup \{z\}, C_= = \emptyset, C_> = \emptyset$. The vertical axis is used for the fractional value of β. The horizontal axis is used for the fractional values of the clocks in C_i. Thus, given a disjoint partition $\{C_0, \ldots, C_n\}$ of the clocks, we pick $n+1$ points on a line parallel to the horizontal axis, $\{\langle C_0^f, \mathsf{frac}(\beta) \rangle, \ldots, \langle C_n^f, \mathsf{frac}(\beta) \rangle\}$, with C_i^f being the fractional value of the clocks in the set C_i at $\widehat{\kappa}$.

We now show that the extended regions induce a backward stable bisimulation quotient.

Lemma 19. *Let Y, Y' be extended regions in a timed game structure $\widehat{[\![\mathcal{T}]\!]}$. Consider a state $\widehat{s} \in Y$ and $t \in \mathbb{R}_{>0}$. Suppose $(0, t] = T^Y \cup T^{Y'}$, such that for all $\tau \in T^Y$ we have $\widehat{s} + \tau \in Y$,*

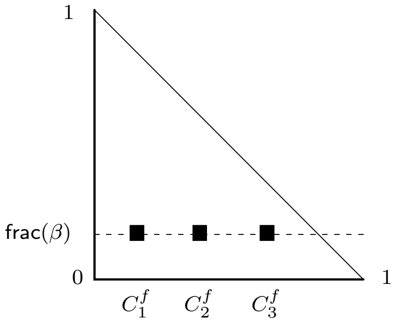


Figure 5.2: An extended region with $C_< = C \cup \{z\}, C_= = \emptyset, C_> = \emptyset$

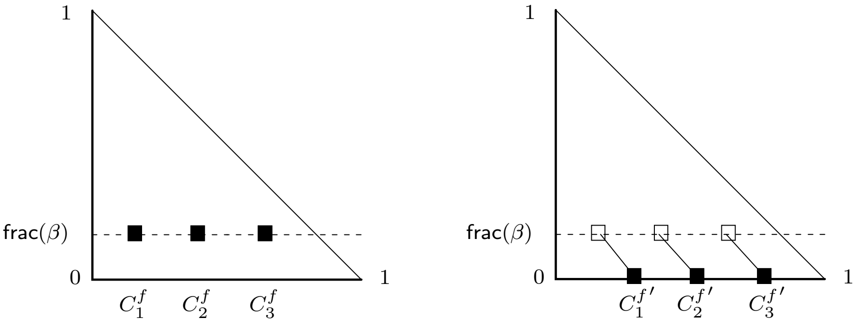


Figure 5.3: An extended region with $C_< = C \cup \{z\}, C_= = \emptyset$ and its time successor.

and for all $\tau \in T^{Y'}$ we have $\widehat{s} + \tau \in Y'$ ($Y \to Y'$ is the first extended region change due to the passage of time). Then, for all states $\widehat{s}_2 \in Y$, there exists $t_2 \in \mathbb{R}_{>0}$ such that for some $T_2^Y, {T^{Y'}}_2$ with $(0, t_2] = T_2^Y \cup {T^{Y'}}_2$, for all $\tau_2 \in T_2^Y$ we have $\widehat{s}_2 + \tau_2 \in Y$, and for all $\tau_2 \in {T^{Y'}}_2$ we have $\widehat{s}_2 + \tau_2 \in Y'$.

Proof. We outline a sketch of the proof. For simplicity, consider the values of each clock x to be less than $C_x + 1$. We look at the time successors of states $\widehat{s}$ in Y. The following cases for $Y = \langle l, tick, bl_1, h, \mathcal{P} = \{C_0, \ldots, C_n\}, \beta_i, \beta_f, C_<, C_=, C_> \rangle$ can arise:

Case 1 $C_0 = \emptyset, \beta_i \in \mathbb{N} \cap \{0, \ldots, M\}, \beta_f = \textbf{true}, C_< = C \cup \{z\}, C_= = \emptyset, C_> = \emptyset$.

For any state in Y, the next extended region Y' can *only* be $\langle l, tick, bl_1, h, \mathcal{P}, \beta_i, \beta'_f = \textsc{false}, C_<, C_=, C_> \rangle$, which is hit after a time of $\mathsf{frac}(\beta_f)$ (note that $C_n^f + \mathsf{frac}(\beta) < 1$ implies $\mathcal{P}$ is going to be unchanged in the time successor extended region).

Case 2 $C_0 = \emptyset, \beta_i \in \mathbb{N} \cap \{0, \ldots, M\}, \beta_f = \textbf{true}, C_< \neq \emptyset, C_= \neq \emptyset, C_> \neq \emptyset$.

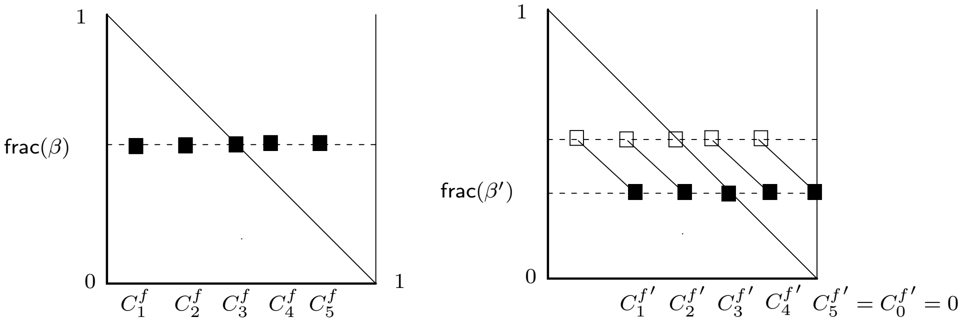

1
frac(β)
0
1
C_1^f C_2^f C_3^f C_4^f C_5^f
1
frac(β')
0
$C_1^{f'}$ $C_2^{f'}$ $C_3^{f'}$ $C_4^{f'}$ $C_5^{f'} = C_0^{f'} = 0$

Figure 5.4: An extended region with $C_< \neq \emptyset, C_= \neq \emptyset, C_> \neq \emptyset$ and its time successor.

Pictorially, this can be depicted as in Fig. 5.4.

Consider any state in Y. The extended region changes after a time of $1 - C_n^f$. The new state then lies in an extended region such that $C'_i = C_i$ for $0 < i < n$, and $C'_0 = C_n$. Also, $C_i^{f'} = C_i^f + (1 - C_n^f)$ for $0 < i < n$, and $\mathsf{frac}\,\beta' = \mathsf{frac}(\beta) - (1 - C_n^f)$. We also have that if $C_i^f + \mathsf{frac}(\beta) \sim 1$, then $C_i^{f'} + \mathsf{frac}\,\beta' = C_i^f + \mathsf{frac}(\beta) \sim 1$ for $\sim \in \{<, =, >\}, 0 < i < n$. Thus the new state lies in the region $\langle l, \mathit{tick}', bl_1, h', \mathcal{P}' = \{C'_0, \ldots C'_{n-1} \mid C'_i = C_i \text{ for } 0 < i < n, C'_0 = C_n\}, \beta_i, \beta_f, C'_< = C_< \cup C_n, C'_= = C_=, C'_> = C_> \setminus C_n\rangle$, with $\mathit{tick}' = \textsc{true}$ iff $z \in C_n$, and h' is h with the integer values for clocks in $C_n \setminus \{z\}$ incremented by 1. This analysis holds for *all* the states in Y. Thus the extended region Y' following Y is unique.

Case 3 $C_0 \neq \emptyset, \beta_i \in \mathbb{N} \cap \{0, \ldots, M\}, \beta_f = \textbf{true}$

All the states in Y then move to $\langle l, \mathit{tick}, bl_1, h, \mathcal{P}' = \{C'_0, \ldots, C'_{n+1} \mid C'_0 = \emptyset \text{ and } C'_{i+1} = C_i, 0 \leq i \leq n\}, \beta_i, \beta_f, C_<, C_=, C_>\rangle$.

Case 4 $C_0 \neq \emptyset, \beta_i \in \mathbb{N} \cap \{1, \ldots, M\}, \beta_f = \textbf{false}$

The time successor in this case is $\langle l, \mathit{tick}, bl_1, \mathcal{P}' = \{C'_0, \ldots, C'_{n+1} \mid C'_0 = \emptyset \text{ and } C'_{i+1} = C_i, 0 \leq i \leq n\}, \beta'_i = \beta_i - 1, \beta'_f = \textsc{true}, C'_<, C'_=, C'_>\rangle$. We show $C'_<, C'_=, C'_>$ to be unique as follows: the new state $\widehat{s} + t$ has the constraints 1)$\mathsf{frac}(\beta') = 1 - t$ and 2)${C_{i+1}^f}' = C_i^f + t$ for $i \leq n$. Thus, $\mathsf{frac}(\beta') + {C_{i+1}^f}' = (1 - t) + C_i^f + t = 1 + C_i^f$. Hence, $C'_< = \emptyset$ and $C'_= = C'_1 = C_0$ (the other clocks belong in $C'_>$..

Case 5 $\beta_i = 0, \beta_f = \textbf{false}$

We get $\beta' = \perp$ in the next state (and hence $C_< = C_= = \emptyset, \beta_i = \perp, \beta_f = \textsc{false}$). The rest of the components of the extended region have a unique value as in the time successors of standard regions.

Case 6 $\beta_i = \bot$

The value of $\mathcal{P}'$ gets updated as in the time successors of standard regions.

The analysis of the remaining cases proceeds in a similar vein to the above cases.

□

Lemma 19 has the following corollary, which states that the equivalence relation $\cong_e$ induces a time-abstract bisimulation.

Corollary 4. *Let Y, Y' be extended regions in a timed game structure $\widehat{[\![\mathcal{T}]\!]}$. Suppose player-i has a move from $s_1 \in Y$ to $s_1' \in Y'$, for $i \in \{1,2\}$. Then, for any $s_2 \in Y$, player-i has a move from s_2 to some $s_2' \in Y'$.*

Let Y, Y_1', Y_2' be extended regions. We have that from a state in Y, for every move of player-2 to the extended region Y_2', either player-1 can force the game in one step so that the next state lies in Y_1', or player-2 can always foil player-1 from going to the extended region Y_1'. Thus moves to some extended regions always "beat" moves to other extended regions.

Lemma 20. *Let Y, Y_1', Y_2' be extended regions in a timed game structure $\widehat{[\![\mathcal{T}]\!]}$. Suppose player-i has a move from $s_1 \in Y$ to $s_1' \in Y'$, for $i \in \{1,2\}$. Then, one of the following cases must hold:*

1. *From all states $\widehat{s} \in Y$, player-1 has some move $m_1^{\widehat{s}}$ with $\widehat{\delta}(\widehat{s}, m_1^{\widehat{s}}) \in Y_1'$ such that for all moves $m_2^{\widehat{s}}$ of player-2 with $\widehat{\delta}(\widehat{s}, m_2^{\widehat{s}}) \in Y_2'$, we have $\mathsf{blame}_1(\widehat{s}, m_1^{\widehat{s}}, m_2^{\widehat{s}}, \widehat{\delta}(\widehat{s}, m_1^{\widehat{s}})) = \text{TRUE}$ and $\mathsf{blame}_2(\widehat{s}, m_1^{\widehat{s}}, m_2^{\widehat{s}}, \widehat{\delta}(\widehat{s}, m_2^{\widehat{s}})) = \text{FALSE}$.*

2. *From all states $\widehat{s} \in Y$, for all moves $m_1^{\widehat{s}}$ of player-1 with $\widehat{\delta}(\widehat{s}, m_1^{\widehat{s}}) \in Y_1'$, player-2 has some move $m_2^{\widehat{s}}$ with $\widehat{\delta}(\widehat{s}, m_2^{\widehat{s}}) \in Y_2'$ such that $\mathsf{blame}_2(\widehat{s}, m_1^{\widehat{s}}, m_2^{\widehat{s}}, \widehat{\delta}(\widehat{s}, m_2^{\widehat{s}})) = \text{TRUE}$.*

Lemma 21. *Let $X \subseteq \widehat{S}$ consist of a union of extended regions in a timed game structure $\widehat{[\![\mathcal{T}]\!]}$. Then $\mathsf{CPre}_1(X)$ is again a union of extended regions.*

Proof. The lemma is essentially a corollary of Lemma 20. □

Lemma 21 demonstrates that the sets in the fixpoint computation of the μ-calculus algorithm which computes winning states for player-1 for the reachability objective $\widehat{X}_p$

consist of unions of extended regions. Since the number of extended regions is finite, the algorithm terminates.

Theorem 15. *For a state s and a proposition p in a timed automaton game $\mathcal{T}$,*

1. *The minimum time for player-1 to visit p starting from s (denoted $T_{\min}(s,p)$) is computable in time $O\Big((|L| \cdot \prod_{x\in C}(c_x+1)\cdot |C+1|! \cdot 2^{|C|})^7 \cdot |C|^5 \cdot |A_1|^{*2} \cdot |A_2|^*\Big)$, where $|C|$ is the number of clocks, $|A_i|$ is the number of player-i edges, $|A_i|^* = \min\{|A_i|, |L| \cdot 2^{|C|}\}$, and $|L|$ is the number of of locations.*

2. *For every region R of $[\![\mathcal{T}]\!]$, either there is a constant $d_R \in \mathbb{N} \cup \{\infty\}$ such that for every state $s \in R$, we have $T_{\min}(s,p) = d_R$, or there is an integer constant d_R and a clock $x \in C$ such that for every state $s \in R$, we have $T_{\min}(s,p) = d_R - \mathsf{frac}(\kappa(x))$, where $\kappa(x)$ is the value of the clock x in s.*

Proof. 1. Let M be the upper bound on $T_{\min}(s,p)$ as in Lemma 16 if $T_{\min}(s,p) < \infty$, and $M = 1$ otherwise. We have $M = O\left(|L| \cdot \prod_{x\in C}(c_x+1) \cdot |C+1|! \cdot 2^{|C|}\right)$ from Lemma 16. The number of equivalence classes in the enlarged game structure $\widehat{[\![\mathcal{T}]\!]}$ is $N = O\left((|L| \cdot \prod_{x\in C}(c_x+1) \cdot |C+1|! \cdot 2^{|C|})^2 \cdot |C|\right)$. Similar to the construction presented in Section 3.4 of Chapter 3, we can construct an equivalent turn based parity game for $\widehat{[\![\mathcal{T}]\!]}$, with $n = O\left(M \cdot |C| \cdot |A_1|^* \cdot N\right)$ vertices, and $m = O(N \cdot M^2 \cdot |C|^2 \cdot |A_1|^* \cdot |A_2|^*)$ edges. A parity game with 2 priorities on a graph with m edges and n vertices can be solved in $O(n \cdot m)$ time. Hence, the minimum time required to visit p can be computed in $O(N^2 \cdot M^3 \cdot |C|^3 \cdot |A_1|^{*2} \cdot |A_2|^*) = O\Big((|L| \cdot \prod_{x\in C}(c_x+1) \cdot |C+1|! \cdot 2^{|C|})^7 \cdot |C|^5 \cdot |A_1|^{*2} \cdot |A_2|^*\Big)$ time.

2. From the comments after Lemma 8, the states in $\widehat{S}$ from which player-1 has a winning strategy for reaching $\widehat{X}_p$ are computable, and consist of a union of extended regions $\cup_{k=1}^n Y_k$. Suppose this union is non-empty. Using Lemma 17, the minimum time for player-1 to reach p from s is then $\min_k \left\{\inf\left\{\beta \mid \beta \in \mathbb{R}_{[0,M]} \text{ and } \langle s, 0, \beta, \textsc{false}, \textsc{false}\rangle \in Y_k\right\}\right\}$. Note that $s = \langle l, \kappa\rangle$ is fixed here, only β can be varied. We also have that $\inf\{\beta \mid \beta \in \mathbb{R}_{[0,M]} \text{ and } \langle\langle l,\kappa\rangle, 0, \beta, \textsc{false}, \textsc{false}\rangle \in Y_k\}$ is equal to (letting $Y_k = \langle l, \textsc{false}, \textsc{false}, h, \mathcal{P}, \beta_i, \beta_f, C_<, C_=, C_>\rangle$):

   (a) An integer when $C_> = C_= = \emptyset$ or when $\beta_f = \textsc{false}$. The infimum value for β is reached when $\beta_f = \text{FALSE}$ (for then the set of β's is a singleton). Thus, player-1 has an optimal strategy when $\beta_f = \text{FALSE}$.

(b) $d_k - \mathsf{frac}(\kappa(x))$ when $C_= = C_j \neq \emptyset$, and where $x \in C_j$. The infimum value is actually attained by player-1 with some strategy π_1 in this case.

(c) $d_k - \mathsf{frac}(\kappa(x))$ when $C_= = \emptyset, C_> \neq \emptyset$, where $x \in C_j$ for $C_> = \{C_j, \ldots, C_n\}$. The infimum value is not attained by player-1 in this case – he can only get arbitrarily close to the optimum.

Note that $z \in C_0$ in every Y_k (for, $\widehat{\kappa}(z) = 0$). Finally, $\min_k \{e_k \mid e_k = d_k \text{ or } d_k - x_k\}$ is again an expression of the form d_r or $d_r - x$ over a region.

□

Chapter 6

Trading Memory for Randomness in Timed Games

6.1 Introduction

The winning strategies constructed in Chapter 3 for timed automaton games assume the presence of an infinitely precise global clock to measure the progress of time, and the strategies crucially depend on the value of this global clock. Since the value of this clock needs to be kept in memory, the constructed strategies require *infinite memory.* In fact, the following example (Example 9) shows that infinite memory is necessary for winning with respect to reachability objectives. Besides the infinite-memory requirement, the strategies constructed in Chapter 3 are structurally complicated, and it would be difficult to implement the synthesized controllers in practice. Before offering a novel solution to this problem, we illustrate the problem with an example of a simple timed game whose solution requires infinite memory.

Example 8 (Signaling hub)**.** *Consider a signaling hub that both sends and receives signals at the same port. At any time the port can either receive or send a signal, but it cannot do both. Moreover, the hub must accept all signals sent to it. If both the input and the output signals arrive at the same time, then the output signal of the hub is discarded. The input signals are generated by other processes, and infinitely many signals cannot be generated in a finite amount of time. The time between input signals is not known a priori. The system may be modeled by the timed automaton game shown in Figure 6.1. The actions* b_1 *and* b_2

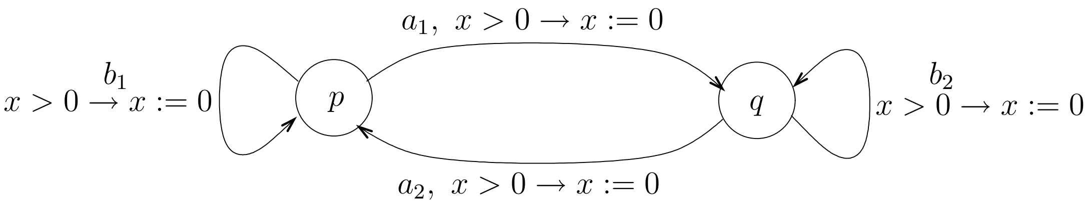


Figure 6.1: A timed automaton game.

correspond to input signals, and a_1 and a_2 to output signals. The actions b_i are controlled by the environment and denote input signals; the actions a_i are controlled by the hub and denote signals sent by the hub. The clock x models the time delay between signals: all signals reset this clock, and signals can arrive or be sent provided the value of x is greater than 0, ensuring that there is a positive delay between signals. The objective of the hub controller is to keep sending its own signals, which can be modeled as the generalized Büchi condition of switching infinitely often between the locations p and q (ie., the LTL *objective* $\Box(\Diamond p \land \Diamond q)$*).* □

Example 9 (Winning requires infinite memory)**.** *Consider the timed game of Figure 6.1. We let κ denote the valuation of the clock x. We let the special "action" $\bot$ denote a time move (representing time passage without an action). The objective of player 1 is to reach q starting from $s_0 = \langle p, x = 0\rangle$ (and similarly, to reach p from q). We let π_1 denote the strategy of player 1 which prescribes moves based on the history $r[0..k]$ of the game at stage k. Suppose player 1 uses only finite memory. Then player 1 can propose only moves from a finite set when at s_0. Since a zero time move keeps the game at p, we may assume that player 1 does not choose such moves. Let $\Delta > 0$ be the least time delay of these finitely many moves of player 1. Then player 2 can always propose a move $\langle \Delta/2, b\rangle$ when at s_0. This strategy will prevent player 1 from reaching q, and yet time diverges. Hence player 1 cannot win with finite memory; that is, there is no hub controller that uses only finite memory. However, player 1 has a winning strategy with infinite memory. For example, consider the player 1 strategy π_2 such that $\pi_2(r[0..k]) = \langle 1/2^{k+2}, a_1\rangle$ if $r[k] = \langle p, \kappa\rangle$. and $\pi_2(r[0..k]) = \langle 1, \bot\rangle$ otherwise.* □

In this chapter we observe that the infinite-memory requirement of Example 6.1 is due to the determinism of the permissible strategies: a strategy is *deterministic* (or *pure*) if in each round of the game, it proposes a unique move (i.e., action and time delay). A more general class of strategies are the randomized strategies: a *randomized* strategy may propose, in each round, a probability distribution of moves. We now show that in the game of Example 9 finite-memory randomized winning strategies do exist. Indeed, the needed randomization has a particularly simple form: player 1 proposes a unique action together with a time interval from which the time delay is chosen uniformly at random. Such a strategy can be implemented as a controller that has the ability to wait for a randomly chosen amount of time.

Example 10 (Randomization instead of infinite memory)**.** *Recall the game in Figure 6.1. Player 1 can play a randomized memoryless strategy π_3 such that $\pi_3(\langle p, \kappa \rangle) = \langle \mathsf{Uniform}((0, 1-\kappa(x))), a_i \rangle$; that is, the action a_i is proposed to take place at a time chosen uniformly at random in the interval $(0, 1-\kappa(x))$. Suppose player 2 always proposes the action b_i with varying time delays Δ_j at round j. Then the probability of player-1's move being never chosen is $\prod_{j=1}^{\infty}(1-\Delta_j)$, which is 0 if $\sum_{j=1}^{\infty}\Delta_j = \infty$ (by Lemma 26). Interrupting moves with pure time moves does not help player 2, as $1 - \frac{\Delta_j}{1-\kappa(x)} < 1 - \Delta_j$. Thus the simple randomized strategy π_3 is winning for player 1 with probability 1.* □

Previously, only deterministic strategies were studied for timed games; here, for the first time, we study randomized strategies. We show that randomized strategies are not more powerful than deterministic strategies in the sense that if player 1 can win with a randomized strategy, then she can also win with a deterministic strategy. However, as the example illustrated, randomization can lead to a reduction in the memory required for winning, and to a significant simplification in the structure of winning strategies. Randomization is therefore not only of theoretical interest, but can improve the implementability of synthesized controllers. It is for this reason that we set out, in this paper, to systematically analyze the trade-off between randomization requirements (no randomization; uniform randomization; general randomization), memory requirements (finite memory and infinite memory) and the presence of extra "controller clocks" for various classes of ω-regular objectives (safety; reachability; parity objectives).

Our results in this chapter are as follows. First, we show that for safety objectives pure (no randomization) finite-memory winning strategies exist. Next, for reachability objectives, we show that pure (no randomization) strategies require infinite memory for winning, whereas uniform randomized finite-memory winning strategies exist. We then use the results for reachability and safety objectives in an inductive argument to show that uniform randomized finite-memory strategies suffice for all parity objectives, for which pure strategies require infinite memory (because reachability is a special case of parity). In all our uses of randomization, we only use uniform randomization over time, and more general forms of randomization (nonuniform distributions; randomized actions) are not required. This shows that in timed games, infinite memory can be traded against uniform randomness. Finally, we show that while randomization helps in simplifying winning strategies, and thus allows the construction of simpler controllers, randomization does not help a player in winning at more states, and thus does not allow the construction of more powerful controllers. In other words, the case for randomness rests in the simplicity of the synthesized real-time controllers, not in their expressiveness.

We note that in our setting, player 1 (i.e., the controller) can trade infinite memory also against finite memory together with an extra clock. We assume that the values of all clocks of the plant are observable. For an ω-regular objective Φ, we define the following winning sets depending on the power given to player 1: let $[\![\Phi]\!]_1$ be the set of states from which player 1 can win using any strategy (finite or infinite memory; pure or randomized) and any number of infinitely precise clocks; in $[\![\Phi]\!]_2$ player 1 can win using a pure finite-memory strategy and only one extra clock; in $[\![\Phi]\!]_3$ player 1 can win using a pure finite-memory strategy and no extra clock; and in $[\![\Phi]\!]_4$ player 1 can win using a randomized finite-memory strategy and no extra clock. Then, for every timed automaton game, we have $[\![\Phi]\!]_1 = [\![\Phi]\!]_2 = [\![\Phi]\!]_4$. We also have $[\![\Phi]\!]_3 \subseteq [\![\Phi]\!]_1$, with the subset inclusion being in general strict. It can be shown that at least one bit of memory is required for winning of reachability objectives despite player 1 being allowed randomized strategies. We do not know whether memory is required for winning safety objectives (even in the case of pure strategies).

We note that removing the global clock from winning strategies is nontrivial. The algorithms of Chapter 3 use such a global clock to construct winning strategies. Without a global clock, time cannot be measured directly, and we need to argue about other properties of runs which ensure time divergence. For safety objectives, we construct a formula that depends only on clock resets and on particular region valuations, and we argue that the satisfaction of that formula is both necessary and sufficient for winning. This allows us to construct pure finite-memory winning strategies for safety objectives. For reachability objectives, we construct "ranks" for sets of states of a μ-calculus formula, and use these ranked sets to obtain a randomized finite-memory strategy for winning. The proof requires special care, because our winning strategies are required to be invariant over the values of the global clock. Finally, we show that if player 1 does not have a pure (possibly infinite-memory) winning strategy from a state, then for every $\varepsilon > 0$ and for every randomized strategy of player 1, player 2 has a pure counter strategy that can ensure with probability at least $1 - \varepsilon$ that player 1 does not win. This shows that randomization does not help in winning at more states. We note that in this chapter, we assume that player 2 is playing with randomized strategies. It turns out that randomization is not of help to her in preventing player 1 from winning.

Outline. We start off by introducing the notions of randomized strategies and sure and almost sure winning sets in Section 6.2. We also show that randomized strategies do not change winning sets. In Section 6.3 we show that pure finite memory strategies suffice for winning safety objectives. In Section 6.4 we show that player 1 needs only finite memory for winning reachability objectives provided that she can use randomization in proposing moves. Finally in Section 6.5 we show how the safety and reachability strategies can be combined by player 1 to obtained a finite memory randomized strategy for winning all parity objectives.

6.2 Randomized Strategies in Timed Games

In this section we first present the definitions of objectives, randomized strategies and the notions of sure and almost-sure winning in timed game structures. We then show that sure winning sets do not change in the presence of randomization.

Objectives. An *objective* for the timed game structure $\mathcal{G}$ is a set $\Phi \subseteq \mathsf{Runs}$ of runs. We will be interested in the classical reachability, safety and parity objectives. Parity objectives are canonical forms for ω-regular properties that can express all commonly used specifications that arise in verification.

- Given a set of states Y, the *reachability* objective $\mathsf{Reach}(Y)$ is defined as the set of runs that visit Y, formally, $\mathsf{Reach}(Y) = \{r \mid \text{ there exists } i \text{ such that } r[i] \in Y\}$.

- Given a set of states Y, the *safety* objective consists of the set of runs that stay within Y, formally, $\mathsf{Safe}(Y) = \{r \mid \text{ for all } i \text{ we have } r[i] \in Y\}$.

- Let $\Omega : S \mapsto \{0, \ldots, k-1\}$ be a parity index function. The parity objective for Ω requires that the maximal index visited infinitely often is even. Formally, let $\mathsf{InfOften}(\Omega(r))$ denote the set of indices visited infinitely often along a run r. Then the parity objective defines the following set of runs: $\mathsf{Parity}(\Omega) = \{r \mid \max(\mathsf{InfOften}(\Omega(r)))$ is even $\}$.

A timed game structure $\mathcal{G}$ together with the index function Ω constitute a *parity timed game* (of index k) in which the objective of player 1 is $\mathsf{Parity}(\Omega)$. We use similar notations for reachability and safety timed games.

Strategies. A *strategy* for a player is a recipe that specifies how to extend a run. Formally, a *probabilistic strategy* π_i for player $i \in \{1, 2\}$ is a function π_i that assigns to every run prefix $r[0..k]$ a probability distribution $\mathcal{D}_i(r[0..k])$ over $\Gamma_i(r[k])$, the set of moves available to player i at the state $r[k]$. *Pure strategies* are strategies for which the state space of the probability distribution of $\mathcal{D}_i(r[0..k])$ is a singleton set for every run r and all k. We let Π_i^{pure} denote the set of pure strategies for player i, with $i \in \{1, 2\}$. For $i \in \{1, 2\}$, let Π_i be the set of strategies for player i. If both both players propose the same time delay, then the tie is broken by a *scheduler*. Let $\mathsf{TieBreak}$ be the set of functions from $\mathbb{R}_{\geq 0}$ to $\{1, 2\}$. A *scheduler strategy* π_{sched} is a mapping from $\mathsf{FinRuns}$ to $\mathsf{TieBreak}$. If $\pi_{\mathsf{sched}}(r[0..k]) = h$, then the resulting state given player 1 and player 2 moves $\langle \Delta, a_1 \rangle$ and $\langle \Delta, a_2 \rangle$ respectively, is determined by the move of player $h(\Delta)$. We denote the set of all scheduler strategies by Π_{sched}. Given two strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$, the set of possible *outcomes* of the game starting from a state $s \in S$ is denoted $\mathsf{Outcomes}(s, \pi_1, \pi_2)$. Given strategies π_1 and π_2, for player 1 and player 2, respectively, a scheduler strategy π_{sched} and a starting state s we denote by $\Pr_s^{\pi_1, \pi_2, \pi_{\mathsf{sched}}}(\cdot)$ the probability space given the strategies and the initial state s.

Receptive strategies. A strategy π_i is *receptive* if for all strategies $\pi_{\sim i}$, all states $s \in S$, and all runs $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)$, either $r \in \mathsf{Timediv}$ or $r \in \mathsf{Blameless}_i$. We denote Π_i^R to be the set of receptive strategies for player i. Note that for $\pi_1 \in \Pi_1^R, \pi_2 \in \Pi_2^R$, we have $\mathsf{Outcomes}(s, \pi_1, \pi_2) \subseteq \mathsf{Timediv}$.

Sure and almost-sure winning modes. Let $\underline{\mathsf{PureWin}}_1(\psi)$ denote the winning set for player 1 when both players are forced to use only pure (possibly non-receptive strategies). Let $\mathsf{SureWin}_1^{\mathcal{G}}(\Phi)$ (resp. $\mathsf{AlmostSureWin}_1^{\mathcal{G}}(\Phi)$) be the set of states s in $\mathcal{G}$ such that player 1 has a receptive strategy $\pi_1 \in \Pi_1^R$ such that for all scheduler strategies $\pi_{\mathsf{sched}} \in \Pi_{\mathsf{sched}}$ and for all player-2 receptive strategies $\pi_2 \in \Pi_2^R$, we have $\mathsf{Outcomes}(s, \pi_1, \pi_2) \subseteq \Phi$ (resp. $\Pr_s^{\pi_1, \pi_2, \pi_{\mathsf{sched}}}(\Phi) = 1$). Such a winning strategy is said to be a sure (resp. almost sure) winning receptive strategy. In computing the winning sets, we shall quantify over *all* strategies, but modify the objective to take care of time divergence. Given an objective Φ, let $\mathsf{TimeDivBl}_1(\Phi) = \mathsf{Win}_1(\Phi) = (\mathsf{Timediv} \cap \Phi) \cup (\mathsf{Blameless}_1 \setminus \mathsf{Timediv})$, i.e., $\mathsf{TimeDivBl}_1(\Phi)$ denotes the set of paths such that either time diverges and Φ holds, or else time converges and player 1 is not responsible for time to converge. Let $\underline{\mathsf{SureWin}}_1^{\mathcal{G}}(\Phi)$ (resp. $\underline{\mathsf{AlmostSureWin}}_1^{\mathcal{G}}(\Phi)$) be the set of states in $\mathcal{G}$ such that for all $s \in \underline{\mathsf{SureWin}}_1^{\mathcal{G}}(\Phi)$ (resp. $\underline{\mathsf{AlmostSureWin}}_1^{\mathcal{G}}(\Phi)$), player 1 has a strategy $\pi_1 \in \Pi_1$ such that for all strategies for all scheduler strategies $\pi_{\mathsf{sched}} \in \Pi_{\mathsf{sched}}$ and for all player-2 strategies $\pi_2 \in \Pi_2$, we have $\mathsf{Outcomes}(s, \pi_1, \pi_2) \subseteq \Phi$ (resp. $\Pr_s^{\pi_1,\pi_2,\pi_{\mathsf{sched}}}(\Phi) = 1$). Such a winning strategy is said to be a sure (resp. almost sure) winning for the non-receptive game. The following result establishes the connection between $\mathsf{SureWin}$ and $\underline{\mathsf{SureWin}}$ sets.

Theorem 16. *For all well-formed timed game structures $\mathcal{G}$, and for all ω-regular objectives Φ, we have $\underline{\mathsf{SureWin}}_1^{\mathcal{G}}(\mathsf{TimeDivBl}_1(\Phi)) = \mathsf{SureWin}_1^{\mathcal{G}}(\Phi)$.*

Proof. Since we are interested in the sure winning set for player 1, we can restrict our attention to pure strategies of player 1. The rest of the proof is similar to that for Theorem 9. □

Region equivalence. For a state $s \in S$, we write $\mathsf{Reg}(s) \subseteq S$ for the clock region containing s. For a run r, we let the *region sequence* $\mathsf{Reg}(r) = \mathsf{Reg}(r[0]), \mathsf{Reg}(r[1]), \cdots$. Two runs r, r' are region equivalent if their region sequences are the same. Given a distribution $\mathcal{D}_{\mathsf{states}}$ over states, we obtain a corresponding distribution $\mathcal{D}_{\mathsf{reg}} = \mathsf{Reg}_d(\mathcal{D}_{\mathsf{states}})$ over regions as follows: for a region R we have $\mathcal{D}_{\mathsf{reg}}(R) = \mathcal{D}_{\mathsf{states}}(\{s \mid s \in R\})$. An ω-regular objective Φ is a region objective if for all region-equivalent runs r, r', we have $r \in \Phi$ iff $r' \in \Phi$. A strategy π_1 is a *region strategy*, if for all prefixes r_1 and r_2 such that $\mathsf{Reg}(r_1) = \mathsf{Reg}(r_2)$, we have $\mathsf{Reg}_d(\pi_1(r_1)) = \mathsf{Reg}_d(\pi_1(r_2))$. The definition for player 2 strategies is analogous. Two region strategies π_1 and π_1' are region-equivalent if for all prefixes r we have $\mathsf{Reg}_d(\pi_1(r)) = \mathsf{Reg}_d(\pi_1'(r))$. A parity index function Ω is a region parity index function if $\Omega(s_1) = \Omega(s_2)$ whenever $s_1 \cong s_2$. Henceforth, we shall restrict our attention to region objectives. As in Chapter 3, we let $\widehat{\mathcal{T}}$ be the enlarged game structure with the global clock z.

Winning sets with Randomization. We now show that the sure winning sets for player 1 remains unchanged in the presence of randomized player-1 or player-2 strategies. First, we show that randomization does not help player 2 in spoiling player 1 from winning.

Lemma 22. *Let $\mathcal{T}$ be a timed automaton game and $\widehat{\mathcal{T}}$ be the corresponding enlarged game structure. Let $\widehat{\Phi}$ be an ω-regular region objective of $\widehat{\mathcal{T}}$. Then, $\underline{\mathsf{PureWin}}_1^{\widehat{\mathcal{T}}}(\Phi) \subseteq \underline{\mathsf{SureWin}}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$.*

Proof. Let $\widehat{s} \in \underline{\mathsf{PureWin}}_1^{\widehat{\mathcal{T}}}(\Phi)$ be a state. Player 1 has a pure winning strategy π_1^{pure} which wins against all possible pure strategies of player 2 from $\widehat{s}$. That is, for all strategies π_2^{pure} of player 2 we have $\mathsf{Outcomes}(\widehat{s}, \pi_1\mathsf{pure}, \pi_2^{\mathsf{pure}}) \subseteq \Phi$. This means in particular that $\bigcup_{\pi_2^{\mathsf{pure}} \in \Pi_2^{\mathsf{pure}}} \mathsf{Outcomes}(\widehat{s}, \pi_1^{\mathsf{pure}}, \pi_2^{\mathsf{pure}}) \subseteq \widehat{\Phi}$. Let π_2 be any randomized strategy of player 2. Then, we have $\mathsf{Outcomes}(\widehat{s}, \pi_1^{\mathsf{pure}}, \pi_2) \subseteq \bigcup_{\pi_2^{\mathsf{pure}} \in \Pi_2^{\mathsf{pure}}} \mathsf{Outcomes}(\widehat{s}, \pi_1^{\mathsf{pure}}, \pi_2^{\mathsf{pure}}) \subseteq \widehat{\Phi}$. Thus, $\underline{\mathsf{PureWin}}_1^{\widehat{\mathcal{T}}}(\Phi) \subseteq \underline{\mathsf{SureWin}}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$. □

We now show that randomization does not help player 1 in winning at more states.

Theorem 17. *Consider a timed automaton game $\mathcal{T}$ with an ω-regular objective Φ. For all $s \in S \setminus \mathsf{SureWin}_1^{\mathcal{T}}(\Phi)$, for every $\varepsilon > 0$, for every randomized strategy $\pi_1 \in \Pi_1$ of player 1, there is a player 2 pure strategy $\pi_2 \in \Pi_2^{\mathsf{pure}}$ and a scheduler strategy $\pi_{\mathsf{sched}} \in \Pi_{\mathsf{sched}}$ such that $\Pr_s^{\pi_1, \pi_2, \pi_{\mathsf{sched}}}(\mathsf{TimeDivBl}_1(\Phi)) \leq \varepsilon$.*

Proof. Let $\mathcal{T}$ be a timed automaton game with an ω-regular region objective Φ. Suppose $\widehat{s}$ is a not a sure winning state for player 1, i.e., $\widehat{s} \in S \setminus \mathsf{SureWin}_1^{\mathcal{T}}(\Phi)$. We show that for all randomized strategies π_1, for all $\varepsilon > 0$, there exists a pure region strategy π_2 for player 2 and a strategy π_{sched} for the scheduler such that $\Pr_{\widehat{s}}^{\pi_1, \pi_2, \pi_{\mathsf{sched}}}(\mathsf{TimeDivBl}_1(\Phi)) \leq \varepsilon$. Consider the finite turn based graph $\mathcal{T}^f$ from Section 3.4 of Chapter 3. In $\mathcal{T}^f$, essentially player 1 first selects a destination region, then player 2 picks a counter-move to specify another destination region. Since $\mathsf{TimeDivBl}_1(\Phi)$ is an ω-regular region objective in $\mathcal{T}^f$, if player 1 cannot win surely, then there is a pure region spoiling strategy π_2^* for player 2 that works against all player 1 strategies in $\mathcal{T}^f$.

Fix some $\varepsilon > 0$, and a sequence $(\varepsilon_i)_{i \geq 0}$ such that $\varepsilon_i > 0$, for all $i \geq 0$, and $\sum_{i \geq 0} \varepsilon_i \leq \varepsilon$. Consider a randomized strategy π_1 of player 1 in $\widehat{\mathcal{T}}$. We will construct a counter strategy π_2 for player 2 to π_1. If player 1 proposes a pure move, then the counter move of player 2 can be derived from the strategy π_2^* in $\mathcal{T}^f$. Suppose player 1 proposes a randomized move of the form $\langle \mathcal{D}^{(\alpha,\beta)}, a_1^j \rangle$ (the case where the move is of the form $\langle \mathcal{D}^{[\alpha,\beta)}, a_1^j \rangle$, $\langle \mathcal{D}^{[\alpha,\beta]}, a_1^j \rangle$, $\langle \mathcal{D}^{(\alpha,\beta]}, a_1^j \rangle$ is similar) at a state $\widehat{s}_j$ in the j-th step. The interval (α, β) can be decomposed into $2k+1$ intervals $(\beta_0, \beta_1), \{\beta_1\}, (\beta_1, \beta_2), \{\beta_2\}, \ldots, \{\beta_k\}, (\beta_k, \beta_{k+1})$, with $\beta_0 = \alpha$ and $\beta_{k+1} = \beta$, such that for all $0 \leq i \leq k$, the set $H_i = \{\widehat{s}_j + \Delta \mid \beta_i < \Delta < \beta_{i+1}\}$ is a subset of a region $\widehat{R}_i$, and $\widehat{R}_i \neq \widehat{R}_j$, for $i \neq j$, and similar result hold for the singletons. Consider the counter strategy π_2^* of player 2 in the region game graph for the player 1 moves to $\widehat{R}_1, \ldots, \widehat{R}_{2k+1}$. The counter strategy π_2 at the j-th step is as follows.

- Suppose the strategy π_2^* allows player 1 moves to all $\widehat{R}_1, \ldots, \widehat{R}_{2k+1}$. Then the strategy π_2 picks a move in a region $\widehat{R}'$ such that $\widehat{R}'$ is a counter move of player 2 against $\widehat{R}_{2k+1}$ in π_2^*.
- Suppose the strategy π_2^* allows player 1 moves to $\widehat{R}_1, \ldots \widehat{R}_m$, and not to $\widehat{R}_{m+1}$. Let the counter strategy π_2^* pick some region $\widehat{R}'$ (together with some action a_2) against the player 1 move to $\widehat{R}_{m+1}$. The strategy π_2 is specified considering the following cases.
  1. Suppose $\widehat{R}'$ is a closed region, then from $\widehat{s}_j$ there is an unique time move Δ_j such that $\widehat{s}_j + \Delta_j \in \widehat{R}'$, and the strategy π_2 of player 2 picks $\langle \Delta_j, a_2 \rangle$ such that $\widehat{s} + \Delta_j \in \widehat{R}'$.
  2. Suppose $\widehat{R}'$ is an open region. If $\widehat{R}'$ lies "before" $\widehat{R}_1$, then π_2 picks any move to $\widehat{R}'$. Otherwise, let $\widehat{R}' = \widehat{R}_{2l+1}$ for some l with $2l + 1 \leq m + 1$. Then, player 2 has some move $\langle \Delta_j, a_2 \rangle$, such that $\langle \Delta_j, a_2 \rangle$ will "beat" player 1 moves to $\widehat{R}_{m+1}, \cdots, \widehat{R}_{2k+1}$ with probability greater than $1 - \varepsilon_j$ and $\widehat{s}_j + \Delta_j \in \widehat{R}'$, and π_2 picks the move (Δ_j, a_2).

The player 2 strategy π_2 ensures that some desired region sequence (complementary to player 1's objective) is followed with probability at least $1 - \varepsilon$ for some strategy of the scheduler. This gives us the desired result. □

Lemma 22 and Theorem 17 together imply that the sure winning set remains unchanged in the presence of randomization.

Theorem 18. *Let $\mathcal{T}$ be a timed automaton game and $\widehat{\mathcal{T}}$ be the corresponding enlarged game structure. Let $\widehat{\Phi}$ be an ω-regular region objective of $\widehat{\mathcal{T}}$. Then,* $\underline{\mathsf{PureWin}}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi}) = \underline{\mathsf{SureWin}}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi}) = \underline{\mathsf{AlmostSureWin}}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$.

We now present a lemma that states for region ω-regular objectives region winning strategies exist, and all strategies region-equivalent to a region winning strategy are also winning.

Lemma 23. *Let $\mathfrak{T}$ be a timed automaton game and $\widehat{\mathfrak{T}}$ be the corresponding enlarged game structure. Let $\widehat{\Phi}$ be an ω-regular region objective of $\widehat{\mathfrak{T}}$. Then the following assertions hold.*

1. *There is a pure finite-memory region strategy π_1 that is sure winning for $\widehat{\Phi}$ from the states in $\underline{\mathsf{SureWin}}_1^{\widehat{\mathfrak{T}}}(\widehat{\Phi})$.*

2. *If π_1 is a pure region strategy that is sure winning for $\widehat{\Phi}$ from $\underline{\mathsf{SureWin}}_1^{\widehat{\mathfrak{T}}}(\widehat{\Phi})$ and π_1' is a pure strategy that is region-equivalent to π_1, then π_1' is a sure winning strategy for $\widehat{\Phi}$ from $\underline{\mathsf{SureWin}}_1^{\widehat{\mathfrak{T}}}(\widehat{\Phi})$.*

3. *If π_1 is a pure sure winning region strategy from $\underline{\mathsf{SureWin}}_1^{\mathfrak{T}}(\widehat{\Phi})$ and π_1' is a strategy surely (or almost surely) region-equivalent to π_1, then π_1' is a sure (resp. almost sure) winning strategy for $\widehat{\Phi}$ from $\underline{\mathsf{SureWin}}_1^{\mathfrak{T}}(\widehat{\Phi})$.*

Proof. Let π_1^{pure} be any pure winning strategy for player 1 when player 2 is also restricted to pure strategies. Then, π_1^{pure} is also a winning strategy against all strategies of player 2 (see the proof of Lemma 22). The first two results then follow from Lemma 9.

We now prove the third part of the Lemma. Let π_1 be a pure region sure winning strategy for player 1 from $\underline{\mathsf{SureWin}}_1^{\widehat{\mathfrak{T}}}(\widehat{\Phi})$, and let π_1' be surely region equivalent to π_1. Given a probability distribution $\mathcal{D}$, let $\mathsf{DistSpace}(\mathcal{D})$ denote the state space of the distribution. Consider any strategy π_2 of player 2. We have $\mathsf{Outcomes}(s, \pi_1', \pi_2) = \{r \mid \forall\, k \geq 0\, \exists\, m_1^k \in \mathsf{DistSpace}(\pi_1'(r[0..k]))$ and $r[k+1] = \widehat{\delta}_{\mathsf{jd}}(r[k], m_1^k, \pi_2(r[0..k]))\}$. We then have $\mathsf{Outcomes}(s, \pi_1', \pi_2) = \{r \mid \exists \pi_1'' \in \Pi_1^{\mathsf{pure}}\, \forall k \geq 0\, \pi_1''(r[0..k]) \in \mathsf{DistSpace}(\pi_1'(r[0..k]))$ and $r[k+1] = \widehat{\delta}_{\mathsf{jd}}(r[k], \pi_1''(r[0..k]), \pi_2(r[0..k]))$ and π_1'' behaves like π_1 on other runs.$\}$. Now in the above set, each π_1'' is region equivalent to π_1, and hence is a winning strategy for player 1. Thus, in particular, $\mathsf{Outcomes}(s, \pi_1'', \pi_2) \subseteq \mathsf{TimeDivBl}_1(\widehat{\Phi})$. Taking the union over all π_1'', we have that $\mathsf{Outcomes}(s, \pi_1', \pi_2)$ is surely a subset of $\underline{\mathsf{SureWin}}_1^{\widehat{\mathfrak{T}}}(\widehat{\Phi})$.

Now we prove the result for almost sure equivalence. Let π_1' be almost surely region equivalent to π_1. Consider any strategy π_2 of player 2. We have $\mathsf{Outcomes}(s, \pi_1', \pi_2) = \{r \mid \forall\, k \geq 0\, \exists\, m_1^k \in \mathsf{DistSpace}(\pi_1'(r[0..k]))$ and $r[k+1] = \widehat{\delta}_{\mathsf{jd}}(r[k], m_1^k, \pi_2(r[0..k]))\}$. Consider the strategy π_1^* which is such that for the run r, for all $k \geq 0$ we have $\mathsf{DistSpace}(\pi_1^*(r[0..k]) = \mathsf{DistSpace}(\pi_1(r[0..k]) \cap \mathsf{DistSpace}(\pi_1'(r[0..k])$, and otherwise the measure behaves as π_1', that is for all sets $A \subseteq \mathsf{DistSpace}(\pi_1(r[0..k])$, we have $\pi_1^*(r[0..k])(A) = \pi_1'(r[0..k])(A)$. On runs other than r, π_1^* behaves like π_1. Note that $\pi_1^*(r[0..k])$ is a probability measure as we have only taken out a null set. Now, π_1^* is surely region equivalent to π_1. Thus, $\mathsf{Outcomes}(s, \pi_1^*, \pi_2) \subseteq \widehat{\Phi}$. Observe that the measure of the set $\mathsf{Outcomes}(s, \pi_1', \pi_2) \setminus \mathsf{Outcomes}(s, \pi_1^*, \pi_2)$ is 0. Hence, we have that $\mathsf{Outcomes}(s, \pi_1^*, \pi_2) \subseteq \widehat{\Phi}$ almost surely.

□

Note that there is an infinitely precise global clock z in the enlarged game structure $\widehat{\mathcal{T}}$. If $\mathcal{T}$ does not have such a global clock, then strategies in $\widehat{\mathcal{T}}$ correspond to strategies in $\mathcal{T}$ where player 1 (and player 2) maintain the value of the infinitely precise global clock in memory (requiring infinite memory).

6.3 Safety Objectives: Pure Finite-memory Receptive Strategies Suffice

In this section we show the existence of pure finite-memory sure winning strategies for safety objectives in timed automaton games. Given a timed automaton game $\mathcal{T}$, we define two functions $P_{>0} : C \mapsto \{\text{TRUE}, \text{FALSE}\}$ and $P_{\geq 1} : C \mapsto \{\text{TRUE}, \text{FALSE}\}$. For a clock x, the values of $P_{>0}(x)$ and $P_{\geq 1}(x)$ indicate if the clock x was greater than 0 or greater than or equal to 1 respectively, during the last transition (excluding the originating state). Consider the enlarged game structure $\widetilde{\mathcal{T}}$ with the state space $\widetilde{S} = S \times \{\text{TRUE}, \text{FALSE}\} \times \{\text{TRUE}, \text{FALSE}\}^C \times \{\text{TRUE}, \text{FALSE}\}^C$ and an augmented transition relation $\widetilde{\delta}$. A state of $\widetilde{\mathcal{T}}$ is a tuple $\langle s, bl_1, P_{>0}, P_{\geq 1}\rangle$, where s is a state of $\mathcal{T}$, the component bl_1 is TRUE iff player 1 is to be blamed for the last transition, and $P_{>0}, P_{\geq 1}$ are as defined earlier. The clock equivalence relation can be lifted to states of $\widetilde{\mathcal{T}}$: $\langle s, bl_1, P_{>0}, P_{\geq 1}\rangle \cong_{\widetilde{A}} \langle s', bl_1', P_{>0}', P_{\geq 1}'\rangle$ iff $s \cong_{\mathcal{T}} s'$, $bl_1 = bl_1'$, $P_{>0} = P_{>0}'$ and $P_{\geq 1} = P_{\geq 1}'$.

Lemma 24. *Let $\mathcal{T}$ be a timed automaton game in which all clocks are bounded (i.e., for all clocks x we have $x \leq c_x$, for a constant c_x). Let $\widetilde{\mathcal{T}}$ be the enlarged game structure obtained from $\mathcal{T}$. Then player 1 has a receptive strategy from a state s iff $\langle s, \cdot\rangle \in \underline{\mathsf{SureWin}}_1^{\widetilde{\mathcal{T}}}(\Phi)$, where*

$$\Phi = \Box\Diamond(bl_1 = \text{TRUE}) \rightarrow \left(\left(\bigwedge_{x\in C} \Box\Diamond(x = 0)\right) \wedge \left(\begin{array}{c} \left(\bigvee_{x\in C} \Box\Diamond((P_{>0}(x) = \text{TRUE}) \wedge (bl_1 = \text{TRUE}))\right) \\ \vee \\ \left(\bigvee_{x\in C} \Box\Diamond((P_{\geq 1}(x) = \text{TRUE}) \wedge (bl_1 = \text{FALSE}))\right) \end{array} \right) \right).$$

Proof. We prove inclusion in both directions.

1. ($\Leftarrow$). For a state $\widetilde{s} \in \underline{\mathsf{SureWin}}_1^{\widetilde{\mathfrak{T}}}(\Phi)$, we show that player 1 has a receptive strategy from $\widetilde{s}$. Let π_1 be a pure sure winning strategy: since Φ is an ω-regular region objective such a strategy exists by Lemma 23. Consider a strategy π_1' for player 1 that is region-equivalent to π_1 such that whenever from a state $\widetilde{s}'$ the strategy π_1 proposes a move $\langle \Delta, a_1 \rangle$ such that $\widetilde{s}' + \Delta$ satisfies $(x > 0)$, then π_1' proposes the move $\langle \Delta', a_1 \rangle$ such that $\mathsf{Reg}(\widetilde{s}' + \Delta) = \mathsf{Reg}(\widetilde{s}' + \Delta')$ and $\widetilde{s}' + \Delta'$ satisfies $(x > 0) \wedge (\vee_{y \in C}\ y > 1/2)$. Such a move always exists; this is because, if there exists Δ such that $\widetilde{s} + \Delta \in R \subseteq (x > 0)$, then there exists Δ' such that $\widetilde{s} + \Delta' \in R \cap ((x > 0) \wedge (\vee_{y \in C}\ y > 1/2))$. Intuitively, player 1 jumps near the endpoint of R. By Lemma 23, π_1' is also sure-winning for Φ. The strategy π_1' ensures that in all resulting runs, if player 1 is not blameless, then all clocks are 0 infinitely often (since for all clocks $\Box\Diamond(x = 0)$), and that some clock has value more than 1/2 infinitely often. This implies time divergence. Hence player 1 has a receptive winning strategy from $\widetilde{s}$.

2. ($\Rightarrow$). For a state $\widetilde{s} \notin \underline{\mathsf{SureWin}}_1^{\widetilde{\mathfrak{T}}}(\Phi)$, we show that player 1 does not have any receptive strategy starting from state $\widetilde{s}$. Let $\neg\Phi =$

$$(\Box\Diamond(bl_1 = \text{TRUE})) \wedge$$
$$\left(\left(\bigvee_{x \in C} \Diamond\Box(x > 0)\right) \vee \left(\bigwedge_{x \in C} \Diamond\Box \left(\begin{array}{c} (bl_1 = \text{TRUE} \rightarrow (P_{>0}(x) = \text{FALSE})) \\ \wedge \\ (bl_1 = \text{FALSE} \rightarrow (P_{\geq 1}(x) = \text{FALSE})) \end{array} \right) \right) \right).$$

The objective of player 2 is $\neg\Phi$. Consider a state $\widetilde{s}'$ of $\widetilde{\mathfrak{T}}$. Suppose player 2 has some move from $\widetilde{s}'$ to a region R'', against a move of player 1 to a region R', then (by Lemma 7) it follows that from all states in $\mathsf{Reg}(\widetilde{s}')$, for each move of player 1 to R', player 2 has some move to R''. Since the objective $\neg\Phi$ is a region objective, only the region trace is relevant. Thus, for obtaining spoiling strategies of player 2, we may construct a finite-state region graph game, where the the states are the regions of the game, and edges specifies transitions across regions. Note that for a concrete move m_1 of player 1, if player 2 has a concrete move $m_2 = (\Delta_2, a_2)$ with a desired successor region R, then for any move $m_2' = (\Delta_2', a_2)$ with $\Delta_2' < \Delta_2$, the destination is R against the move m_1. The objective $\neg\Phi$ can be expressed as a disjunction of conjunction of Büchi and coBüchi objectives, and hence is a Rabin-objective. Then there exists a pure memoryless region-strategy for player 2 in the region-based game graph. In our original game, for all player 1 strategies π_1 there exists a player 2 strategy

π_2 such that from every region the strategy π_2 specifies a destination region, and $\mathsf{Outcomes}(\widetilde{s}, \pi_1, \pi_2) \cap \neg\Phi \neq \emptyset$. Consider a player 1 strategy π_1 and the counter strategy π_2 satisfying the above conditions. Consider a run $r \in \mathsf{Outcomes}(\widetilde{s}, \pi_1, \pi_2) \cap \neg\Phi$. If for some clock, we have $\Diamond\Box(x > 0)$, then time converges (as all clocks are bounded in $\mathcal{T}$), and thus π_1 is not a receptive strategy. Suppose we have $\wedge_{x\in C}\ \Box\Diamond(x = 0)$, then $\bigwedge_{x\in C}\ \Diamond\Box\left((bl_1 = \text{TRUE} \rightarrow (P_{>0}(x) = \text{FALSE})) \wedge (bl_1 = \text{FALSE} \rightarrow (P_{\geq 1}(x) = \text{FALSE}))\right)$ holds. This means that after some point in the run, player 1 is only allowed to take moves which result in all the clock values being 0 throughout the move, this implies she can only take moves of time 0. Also, if player 2's move is chosen, then all the clock values are less than 1. Recall that in each step of the game, player 2 has a specific region he wants to go to. Consider a region equivalent strategy π_2' to the original player 2 spoiling strategy in which player 2 takes smaller and smaller times to get into a region R. If the new state is to have $\wedge_{x\in C}\ (P_{\geq 1}(x) = \text{FALSE})$, then player 2 gets there by choosing a time move smaller than $1/2^j$ in the j-th step. Since the destination regions are the same, and since smaller moves are always better, π_2' is also a spoiling strategy for player 2 against π_1. Moreover, time converges in the run where player 2 plays with π_2'. Thus, if a state $\widetilde{s} \notin \underline{\mathsf{SureWin}}_1^{\widetilde{\mathcal{T}}}(\Phi)$, then player 1 does not have a receptive strategy from $\widetilde{s}$.

□

Lemma 24 is generalized to all timed automaton games in the following lemma. Theorem 19 follows from Lemma 25

Lemma 25. *Let $\mathcal{T}$ be a timed automaton game, and $\widetilde{\mathcal{T}}$ be the corresponding enlarged game. Then player 1 has a receptive strategy from a state s iff $\langle s, \cdot\rangle \in \underline{\mathsf{SureWin}}_1^{\widetilde{\mathcal{T}}}(\Phi^*)$, where $\Phi^* = \Box\Diamond(bl_1 = \text{TRUE}) \rightarrow \bigvee_{X\subseteq C}\ \phi_X$, and $\phi_X =$*

$$\left(\bigwedge_{x\in X} \Diamond\Box(x > c_x)\right) \wedge \left(\left(\bigwedge_{x\in C\setminus X} \Box\Diamond(x = 0)\right) \wedge \left(\begin{array}{c} \left(\bigvee_{x\in C\setminus X} \Box\Diamond((P_{>0}(x) = \text{TRUE}) \wedge (bl_1 = \text{TRUE}))\right) \\ \vee \\ \left(\bigvee_{x\in C\setminus X} \Box\Diamond((P_{\geq 1}(x) = \text{TRUE}) \wedge (bl_1 = \text{FALSE}))\right) \end{array} \right) \right).$$

Proof. We give a proof sketch. This result is a generalization of Lemma 24. Note that once a clock x becomes more than c_x, then its actual value can be considered irrelevant in determining regions. If only the clocks in $X \subseteq C$ have escaped beyond their maximum tracked values, the rest of the clocks still need to be tracked, and this gives rise to a subconstraint ϕ_X for every $X \subseteq C$. The rest of the proof is similar to that for Lemma 24. □

Theorem 19. *Let $\mathcal{T}$ be a timed automaton game and $\widetilde{\mathcal{T}}$ be the corresponding enlarged game. Let Y be a union of regions of $\mathcal{T}$. Then the following assertions hold.*

1. $\mathsf{SureWin}_1^{\mathcal{T}}(\Box Y) = \underline{\mathsf{SureWin}}_1^{\widetilde{\mathcal{T}}}((\Box Y) \wedge \Phi^*)$, *where Φ^* is as defined in Lemma 25.*

2. *Player 1 has a pure, finite-memory, receptive, region strategy that is sure winning for the safety objective* $\mathsf{Safe}(Y)$ *at every state in* $\mathsf{SureWin}_1^{\mathcal{T}}(\Box Y)$.

Proof. We prove for the general case (where clocks might not be bounded).

1. If a state $\widetilde{s} \in \underline{\mathsf{SureWin}}_1^{\widetilde{\mathcal{T}}}(\Box Y \wedge \Phi^*)$, then as in Lemma 25, there exists a receptive region strategy for player 1, and moreover this strategy ensures that the game stays in Y.

   If $s \notin \widetilde{s} \in \underline{\mathsf{SureWin}}_1^{\widetilde{\mathcal{T}}}(\Box Y \wedge \Phi^*)$, then for every player-1 strategy π_1, there exists a player-2 strategy π_2 such that one of the resulting runs either violates $\Box Y$, or Φ^*. If Φ^* is violated, then π_1 is not a receptive strategy. If $\Box Y$ is violated, then player 2 can switch over to a receptive strategy as soon as the game gets outside Y. Thus, in both cases $s \notin \mathsf{SureWin}_1^{\mathcal{T}}(\Box Y)$.

2. Result similar to lemma 23 holds for the structure $\widetilde{\mathcal{T}}$. Since the objective Φ^* can be expressed as a Streett (strong fairness) objectives, it follows that player 1 has a pure finite-memory sure winning strategy for every state in $\underline{\mathsf{SureWin}}_1^{\widetilde{\mathcal{T}}}((\Box Y) \wedge \Phi^*)$. The desired result then follows using the first part of the theorem.

□

6.4 Reachability Objectives: Randomized Finite-memory Receptive Strategies Suffice

We have seen in Example 9 that pure sure winning strategies require infinite memory in general for reachability objectives. In this section, we shall show that uniform randomized almost-sure winning strategies with finite memory exist. This shows that we can trade-off infinite memory with uniform randomness.

We shall need the following Lemma from analysis.

Lemma 26 ([Pug02]). *Let $1 \geq \Delta_j \geq 0$ for each j. Then, $\lim_{n\to\infty} \prod_{j=1}^{n}(1-\Delta_j) = 0$ if $\lim_{n\to\infty} \sum_{j=1}^{n} \Delta_j = \infty$.*

Proof. Suppose $\Delta_j = 1$ for some j. Then, clearly $\lim_{n\to\infty} \prod_{j=1}^{n}(1-\Delta_j) = 0$. Suppose $\Delta_j < 1$ for all j. We then have $\prod_{j=1}^{n}(1-\Delta_j) > 0$ for all n. Consider $\ln\left(\prod_{j=1}^{n}(1-\Delta_j)\right) = \sum_{j=1}^{n} \ln(1-\Delta_j)$. Let $g(x) = x + \ln(1-x)$. We have $g(0) = 0$ and $\frac{dg}{dx} = 1 - \frac{1}{1-x} = \frac{-x}{1-x} \leq 0$ for all $1 > x \geq 0$. Thus, $g(x) \leq 0$ for all $1 > x \geq 0$. Hence, $0 \leq \Delta_j < -\ln(1-\Delta_j)$ for every j. Since $\lim_{n\to\infty} \sum_{j=1}^{n} \Delta_j = \infty$, we must have $\lim_{n\to\infty} \sum_{j=1}^{n}(-\ln(1-\Delta_j)) = \infty$, which means $\lim_{n\to\infty} \sum_{j=1}^{n} \ln(1-\Delta_j) = -\infty$. This in turn implies that $\lim_{n\to\infty} \prod_{j=1}^{n}(1-\Delta_j) = 0$. □

Let S_R be the destination set of states that player 1 wants to reach. We only consider S_R such that S_R is a union of regions of $\mathcal{T}$. For the timed automaton $\mathcal{T}$, consider the enlarged game structure of $\mathcal{T}$. We let $\widehat{S_R} = S_R \times \mathbb{R}_{[0,1]} \times \{\text{TRUE}, \text{FALSE}\}^2$. From the reachability objective (denoted $\mathsf{Reach}(S_R)$) we obtain the reachability parity objective with index function Ω_R as follows: $\Omega_R(\langle s, \mathfrak{z}, \mathit{tick}, \mathit{bl}_1\rangle) = 1$ if $\mathit{tick} \vee \mathit{bl}_1 = \text{TRUE}$ and $s \notin S_R$ (0 otherwise). We assume the states in S_R are absorbing. We let $\widehat{S}_R = S_R \times \mathbb{R}_{[0,1]} \times \{\text{TRUE}, \text{FALSE}\}^2$.

Lemma 27. *For a timed automaton game $\mathcal{T}$, with the reachability objective S_R, consider the enlarged game structure $\widehat{\mathcal{T}}$, and the corresponding reachability parity function Ω_R. Then we have that $\underline{\mathsf{SureWin}}_1(\mathsf{TimeDivBl}(\mathsf{Reach}(S_R))) = \underline{\mathsf{SureWin}}_1(\mathsf{Parity}(\Omega_R)) = \mu Y.\ \nu X.\ \left[(\Omega_R^{-1}(1) \cap \mathsf{CPre}_1(Y)) \cup (\Omega_R^{-1}(0) \cap \mathsf{CPre}_1(X))\right]$.*

Proof. We present the result that $\underline{\mathsf{SureWin}}_1(\mathsf{TimeDivBl}(\mathsf{Reach}(S_R))) = \underline{\mathsf{SureWin}}_1(\mathsf{Parity}(\Omega_R))$. The characterization of the winning set by the μ-calculus formula is a classical result. To show $\underline{\mathsf{SureWin}}_1(\mathsf{TimeDivBl}(\mathsf{Reach}(S_R))) = \underline{\mathsf{SureWin}}_1(\mathsf{Parity}(\Omega_R))$, we prove inclusion in both directions.

1. Suppose player 1 can win for the reachability objective S_R. Let π_1 be the winning strategy. Consider any player-2 strategy π_2, and any run $\widehat{r} \in \mathsf{Outcomes}(\langle s, 0, \text{FALSE}, \text{FALSE}\rangle, \pi_1, \pi_2)$. Suppose $\widehat{r}$ visits $\widehat{S_R}$. Then since S_R is absorbing, and all states in $\widehat{S_R}$ have index 0, only the index 0 is seen from some point on.

2. Suppose $\widehat{r}$ does not visit $\widehat{S_R}$, and let $\widehat{r}$ be time-diverging. If the moves of player 1 are chosen infinitely often in $\widehat{r}$, then the index 1 is visited infinitely often. If the moves of

player 1 are chosen only finitely often, then from some point on, the clock z is reset only when it hits 1, and thus since time diverges, *tick* is true infinitely often. The index 1 is again visited infinitely often in this case.

Suppose $\widehat{r}$ does not visit $\widehat{S_R}$, and let $\widehat{r}$ be time-converging. If the moves of player 1 are chosen infinitely often in $\widehat{r}$, then player 1 is to blame for blocking time. In this case 1 is visited infinitely often. If the moves of player 1 are only chosen finitely often, then again from some point on, the clock z is reset only when it hits 1. Since time does not diverge, *tick* is true only finitely often. Thus after some point, only the index 0 is seen, in agreement with the fact that player 1 is blameless.

□

We first present a μ-calculus characterization for the sure winning set (using only pure strategies) for player 1 for reachability objectives. The *controllable predecessor* operator for player 1, $\mathsf{CPre}_1 : 2^{\widehat{S}} \mapsto 2^{\widehat{S}}$, defined formally by $\widetilde{s} \in \mathsf{CPre}_1(Z)$ iff $\exists m_1 \in \widehat{\Gamma}_1(\widehat{s})\ \forall m_2 \in \widehat{\Gamma}_2(\widehat{s})\,.\,\widehat{\delta}_{\mathsf{jd}}(\widehat{s}, m_1, m_2) \subseteq Z$. Informally, $\mathsf{CPre}_1(Z)$ consists of the set of states from which player 1 can ensure that the next state will be in Z, no matter what player 2 does. From Lemma 27 it follows that the sure winning set can be described as the μ-calculus formula: $\mu Y \nu X \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))\right]$. The winning set can then be computed as a fixpoint iteration on regions of $\widehat{\mathfrak{T}}$. We can also obtain a pure winning strategy π_{pure} of player 1 as in [dAHM01b]. Note that this strategy π_{pure} corresponds to an infinite-memory strategy of player 1 in the timed automaton game $\mathfrak{T}$, as she needs to maintain the value of the clock z in memory.

To compute randomized finite-memory almost-sure winning strategies, we will use the structure of the μ-calculus formula. Let $Y^* = \mu Y \nu X \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))\right]$. The iterative fixpoint procedure computes $Y_0 = \emptyset \subseteq Y_1 \subseteq \cdots \subseteq Y_n = Y^*$, where $Y_{i+1} = \nu X \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y_i)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))\right]$. We can consider the states in $Y_i \setminus Y_{i-1}$ as being added in two steps, T_{2i-1} and $T_{2i} (= Y_i)$ as follows:

1. $T_{2i-1} = \Omega^{-1}(1) \cap \mathsf{CPre}_1(Y_{i-1})$. T_{2i-1} is clearly a subset of Y_i.
2. $T_{2i} = \nu X \left[T_{2i-1} \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))\right]$. Note that $(T_{2i} \setminus T_{2i-1}) \cap \Omega^{-1}(1) = \emptyset$.

Thus, in the odd stages, we add states with index 1, and in even stages, we add states with index 0. The *rank* of a state $\widehat{s} \in Y^*$ is j if $\widehat{s} \in T_j \setminus \cup_{k=0}^{j-1} T_k$. For a state of even rank j, we have that player 1 can ensure that she has a move such that against all moves of player 2, the next state either (a) has index 0 and belongs to the same rank or less, or (b) the next state has index 1 and belongs to rank smaller than j. For a state of odd rank j, we have that player 1 can ensure that she has a move such that against all moves of player 2, the next state belongs to a lower rank (and has index either 1 or 0).

We now consider the rank sets for the reachability fixpoint in more detail. We have that S_R is a union of regions of $\mathfrak{T}$. $T_0 = T_1 = \emptyset$, and T_2 consists of all the states in $\widehat{S}_R$ together with the states where $\mathit{tick} = \mathit{bl}_1 = \text{FALSE}$, and from where player 1 can ensure that the next state is either in $\widehat{S}_R$, or the next state continues to have $\mathit{tick} = \mathit{bl}_1 = \text{FALSE}$; formally $T_2 = \nu X(\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))$. Henceforth, when we refer to a region R of $\mathfrak{T}$, we shall mean the states $R \times \mathbb{R}_{[0,1]} \times \{\text{TRUE}, \text{FALSE}\}^2$ of $\widehat{\mathfrak{T}}$.

Lemma 28. *Let* $T_2 = \nu X(\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))$. *Then player 1 has a (randomized) memoryless strategy* π_{rand} *such that she can ensure reaching* $\widehat{S}_R \subseteq \Omega^{-1}(0)$ *with probability 1 against all receptive strategies of player 2 and all strategies of the scheduler from all states* $\widehat{s}$ *of a region* R *such that* $R \cap T_2 \neq \emptyset$. *Moreover,* π_{rand} *is independent of the values of the global clock, tick and* bl_1.

We break the proof of Lemma 28 into several parts. For a set T of states, we shall denote by $\mathsf{Reg}(T)$ the set of states that are region equivalent in $\mathfrak{T}$ to some state in T. We let π_{pure} be the pure infinite-memory winning strategy of player 1 to reach $\widehat{S}_R$. First we prove the following result.

Lemma 29. *Let* $T_2 = \nu X(\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))$. *Then, for every state in* $\mathsf{Reg}(T_2)$*, player 1 has a move to* $\widehat{S}_R$.

Proof. Suppose $T_2 \neq \widehat{S}_R$ (the other case is trivial). Then player 1 must have a move from every state in $T_2 \setminus \widehat{S}_R$ to $\widehat{S}_R$ in $\widehat{\mathfrak{T}}$, for otherwise, for any state in $T_2 \setminus \widehat{S}_R$, player 2 (with cooperation from the scheduler) can allow player 1 to pick any move, which will result in an index of 1 in the next state, contradicting the fact that player 1 had a strategy to stay in T_2 forever (note all the states in T_2 have index 0). Moreover, since $\widehat{S}_R$ is a union of regions of $\mathfrak{T}$, we have that the states in T_2 from which player 1 has a move to $\widehat{S}_R$, consist of a union of sets of the form $T_2 \cap R$ for R a region of $\mathfrak{T}$. This implies that player 1 has a move to $\widehat{S}_R$ from all states in $\mathsf{Reg}(T_2)$ (Lemma 6). □

If at any time player 1's move is chosen, then player 1 comes to $\widehat{S}_R$, and from there plays a receptive strategy. We show that player 1 has a randomized memoryless strategy such that the probability of player 1's move being never chosen against a receptive strategy of player 2 is 0. This strategy will be pure on target left-closed regions, and a uniformly distributed strategy on target left-open regions. We now describe the randomized strategy.

Consider a state $\widehat{s}$ in some region $R' \subseteq \mathsf{Reg}(T_2 \setminus \widehat{S}_R)$ of $\mathcal{T}$. Now consider the set of times at which moves can be taken so that the state changes from $\widehat{s}$ to $\widehat{S}_R$. This set consists of a finite union of sets I_k of the form $(\alpha_l, \alpha_r), [\alpha_l, \alpha_r), (\alpha_l, \alpha_r]$, or $[\alpha_l, \alpha_r]$ where α_l, α_r are of the form d or $d - x$ for d some integer constant, and x some clock in C (this clock x is the same for all the states in R'). Furthermore, these intervals have the property that $\{\widehat{s} + \Delta \mid \Delta \in I_k\} \subseteq R_k$ for some region R_k, with $R_l \cap R_j = \emptyset$ for $j \neq l$. From a state $\widehat{s}$, consider the "earliest" interval contained in this union: the interval I such that the left endpoint is the infimum of the times at which player 1 can move to $\widehat{S}_R$. We have that $\{\widehat{s} + \Delta \mid \Delta \in I\} \subseteq R_1$. Consider any state $\widehat{s}' \in R'$. Then from $\widehat{s}'$, the earliest interval in the times required to get to $\widehat{S}_R$ is also of the form I. Note that in allowing time to pass to get to R_1, we may possibly go outside T_2 (recall that T_2 is not a union of regions of $\mathcal{T}$).

If this earliest interval I is left closed, then player 1 has a "shortest" move to $\widehat{S}_R$. Then this is the best move for player 1, and she will always propose this move. We call these regions *target left-closed*. If the target interval is left open, we call the region *target left-open*. Let the left and the right endpoints of target intervals be α_l, α_r respectively. Then let player 1 play a probabilistic strategy with time distributed uniformly at random over $(\alpha_l, (\alpha_l + \alpha_r)/2]$ on these target left-open regions. Let us denote this player-1 strategy by π_{rand}.

Lemma 30. *Let* $T_2 = \nu X(\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))$. *Then, for every state in* $\mathsf{Reg}(T_2)$ *the strategy* π_{rand} *as described above ensures that player 1 stays inside* $\mathsf{Reg}(T_2)$ *surely.*

Proof. Consider a state $\widehat{s}$ in $T_2 \setminus \widehat{S}_R$. Since $\{\widehat{s} + t \mid t \in I\}$ is a subset of a single region of $\mathcal{T}$, no new discrete actions become enabled due to the randomized strategy of player 1. If player 2 can foil player 1 by taking a move to a region R'' for the player 1 randomized strategy, she can do so against any pure (infinite-memory) strategy of player 1. No matter what player 2 proposes at each step, player 1's strategy is such that the next state (against any player 2's moves) lies in a region R'' (of $\mathcal{T}$) such that $R'' \cap T_2 \neq \emptyset$. Because of this, player 1 can always play the above mentioned strategy at each step of the game, and ensure that she stays inside $\mathsf{Reg}(T_2)$ (until the destination $\widehat{S}_R$ is reached). □

Lemma 31. *Consider the the player 1 strategy π_{rand} and any receptive strategy π_2 of player 2. Let $r \in \mathsf{Outcomes}(s, \pi_{\mathsf{rand}}, \pi_2)$ be a run with $s \in \mathsf{Reg}(T_2)$. If there exists $m \geq 0$ such that $\pi_{\mathsf{rand}}(r[0..j])$ is left-closed for all $j \geq m$, then we have that r visits $\widehat{S}_R$.*

Proof. Consider a run r for the player 1 strategy π_{rand} against any strategy π_2 of player 2. Note that we must have $\mathsf{Reg}(r[k]) \subseteq \mathsf{Reg}(T_2)$ for every k by Lemma 30. Let $r[m] = \widehat{s}' = \langle s', \mathfrak{z}, tick, bl_1\rangle \in R'$. Consider the pure winning strategy π_{pure} from a state $\widehat{s}'' = \langle s', \mathfrak{z}', tick', bl_1'\rangle \in R' \cap T_2$ (such a state must exist). The state $\widehat{s}''$ differs from $\widehat{s}'$ only in the values of the clock z, and the boolean variables $tick$ and bl_1. The new values do not affect the moves available to either player. Consider $\widehat{s}''$ as the starting state. The strategy π_{pure} cannot propose shorter moves to $\widehat{S}_R$, since π_{rand} proposes the earliest move to $\widehat{S}_R$. Hence, if a receptive player 2 strategy π_2 can prevent π_{rand} from reaching $\widehat{S}_R$ from $\widehat{s}'$, then it can also prevent π_{pure} from reaching $\widehat{S}_R$ from $\widehat{s}''$, a contradiction. □

Lemma 32. *Consider the the player 1 strategy π_{rand} and any receptive strategy π_2 of player 2. Let $r \in \mathsf{Outcomes}(s, \pi_{\mathsf{rand}}, \pi_2)$ be a run with $s \in \mathsf{Reg}(T_2)$. There exists $m \geq 0$ such that for all $j \geq m$, if $\pi_{\mathsf{rand}}(r[0..j])$ is left-open, then the left endpoint is $\alpha_l = 0$.*

Proof. Let α_l correspond to the left endpoints for one of the infinitely occurring target left-open regions R.

1. We show that we cannot have α_l to be of the form d for some integer $d > 0$.

   We prove by contradiction. Suppose α_l is of the form $d > 0$. Then player 2 could always propose a time blocking move of duration d, this would mean that if the scheduler picks the move of player 2 (as both have the same delay), the next state would have $tick = \text{TRUE}$, no matter what the starting value of the clock z in R, contradicting the fact that $R \cap T_2 \neq \emptyset$ ($T_2 = \nu X(\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))$). We have a contradiction, as player 1 had a pure winning strategy π_{pure} from every state in T_2. Take any $\widehat{s} \in R \cap T_2$. Then π_{pure} must have proposed some move to $\widehat{S}_R$, such that all the intermediate states (before the move time) had $tick = \text{FALSE}$. The strategy π_{rand} picks the earliest left most endpoint to get to $\widehat{S}_R$. This means that π_{pure} must also propose a time which is greater than or equal to the move proposed by π_{rand}. Hence α_l cannot be d for $d > 0$ (otherwise the player 2 counter strategy to π_{pure} can take the game out of T_2 by making $tick = \text{TRUE}$).

2. We show that we cannot have α_l to be of the form $d - x$ for some integer $d > 0$.

   We prove by contradiction. Suppose $\alpha_l = d - x$ for some some clock x for the target constraint. Let player 2 counter with any strategy.

   Suppose clock x is not reset infinitely often in the run r. Then the fact that the clock x has not progressed beyond d at any time in the run without being reset implies time is convergent, contradicting the fact that player 2 is playing with a receptive strategy (note that only player 2's moves are being chosen). Thus, this situation cannot arise.

   Suppose x is reset infinitely often. Then between a reset of x, and the time at which player 1 can jump to $\widehat{S}_R$, we must have a time distance of more than d. Suppose R' is one of the infinitely occurring regions in the run with the value of x being 0 in it. So player 2 has a strategy against our player 1 strategy such that one of the resulting runs contains a region subsequence $R' \rightsquigarrow R$. If this is so, then she would have a strategy which could do the same from every state in $R' \cap T_2$ against the pure winning strategy of player 1 (since the randomized strategy π_{rand} does not enable player 2 to go to more regions than against π_{pure}, as π_{rand} proposes moves to the earliest region in $\widehat{S}_R$). But, if so, we have that *tick* will be true no matter what the starting value of z in $R' \cap T_2$, before player 1 can take a jump to $\widehat{S}_R$ from $R \cap T_2$, taking the game outside of T_2. Since player 1 can stay inside T_2 at each step with the infinite memory strategy π_{pure}, this cannot be so, that is, we cannot observe the region subsequence $R' \rightsquigarrow R$ for the randomized strategy of player 1. Thus the case of $\alpha_l = d - x$ cannot arise infinitely often.

The only remaining option is $\alpha_l = 0$, and we must have that the only randomized moves player 1 proposes after a while are of the form $(0, \alpha_r/2]$. □

Lemma 33. *Consider runs r with $r[0] \in \mathsf{Reg}(T_2)$ for the player 1 strategy π_{rand} against any receptive strategy π_2 of player 2 and a scheduler strategy π_{sched}. Let $\mathcal{E}$ be the set of runs such that for all $m \geq 0$ there exists $j \geq m$ such that $\pi_{\mathsf{rand}}(r[0..j])$ is left-open, with the the left endpoint being $\alpha_l = 0$. Then, we have $\Pr_{r[0]}^{\pi_{\mathsf{rand}},\pi_2,\pi_{\mathsf{sched}}}(\mathsf{Reach}(S_R) \,|\mathcal{E}) = 1$.*

Proof. Let one of the infinitely often occurring player 1 left-open moves be to the region R. Player 1 proposes a uniformly distributed move over $(0, \alpha_r/2]$ to R. Let β_i be the duration of player 2's move for the ith visit to R Suppose $\alpha_r = d$. Then the probability of player 1's move being never chosen is less than $\prod_{i=1}^{\infty}(1 - \frac{2\beta_i}{d})$, which is 0 if $\sum_{i=1}^{\infty} \beta_i = \infty$

by Lemma 26. A similar analysis holds if player 2 proposes randomized moves with a time distribution $\mathcal{D}^{(\beta_i,\cdot]}, \mathcal{D}^{[\beta_i,\cdot]}, \mathcal{D}^{(\beta_i,\cdot)}$ or $\mathcal{D}^{[\beta_i,\cdot)}$. Suppose $\alpha_r = d - x$. Again, the probability of player 1's move being never chosen is less than $\prod_{i=1}^{\infty}(1 - \frac{2\beta_i}{(d-\kappa_i(x))})$, and since $\frac{\beta_i}{(d-\kappa_i(x))} > \frac{\beta_i}{d}$, this also is 0 if $\sum_{i=1}^{\infty}\beta_i = \infty$ by Lemma 26 . Finally, we note that if player 2 does not block time from T_2, then for at least one region, she must propose a β_i sequence such that $\sum_{i=1}^{\infty}\beta_i = \infty$, and we will have that for this region, player 1's move will be chosen eventually with probability 1. □

Proof of Lemma 28. Lemmas 29, 30, 31, 32 and 33 together imply that using the randomized memoryless strategy π_{rand}, player 1 can ensure going from *any* region R of $\mathcal{T}$ such that $R \cap T_2 \neq \emptyset$ to $\widehat{S}_R$ with probability 1, without maintaining the infinitely precise value of the global clock. □

The following lemma states that if for some state $s \in \mathcal{T}$, we have $(s, \mathfrak{z}, \mathit{tick}, \mathit{bl}_1) \in T_{2i+1}$, for some i, then for some $\mathfrak{z}', \mathit{tick}', \mathit{bl}_1'$ we have $(s, \mathfrak{z}', \mathit{tick}', \mathit{bl}_1') \in T_{2i}$. Then in Lemma 35 we present the inductive case of Lemma 28. The proof of Lemma 35 is similar to the base case i.e., Lemma 28. The proofs can be found in the appendix.

Lemma 34. *Let R be a region of $\mathcal{T}$ such that $R \cap T_{2i+1} \neq \emptyset$. Then $R \cap T_{2i} \neq \emptyset$.*

Proof. Consider a state $\langle s, \mathfrak{z}, \mathit{tick}, \mathit{bl}_1\rangle \in T_{2i+1}$ and let $s \in R$. All the states in T_{2i+1} have the property that player-1 can always guarantee that the next state has a lower rank, no matter what the move of player 2. Consider the player-2 move of $\langle 0, \bot\rangle$ at state $\langle s, \mathfrak{z}, \mathit{tick}, \mathit{bl}_1\rangle \in T_{2i+1}$. The next state is then going to be $\langle s, \mathfrak{z}, \mathit{tick}' = \text{FALSE}, \mathit{bl}_1' = \text{FALSE}\rangle$. Since $\mathit{tick} \vee \mathit{bl}_1 = \text{FALSE}$, the index of $\langle s, \mathfrak{z}, \mathit{tick}' = \text{FALSE}, \mathit{bl}_1' = \text{FALSE}\rangle$ is 0, and hence it must belong to an even rank which is lower than $2i + 1$. Finally, we note that $\cup_{k=0}^{2i-1} T_k \subset T_{2i}$. □

Lemma 35. *Let R be a region of $\mathcal{T}$ such that $R \cap T_{2i} \neq \emptyset$, and $R \cap T_j = \emptyset$ for all $2 \leq j < 2i$. Then player 1 has a (randomized) memoryless strategy π_{rand} to go from R to some R' such that $R' \cap T_j \neq \emptyset$ for some $j < 2i$ with probability 1 against all receptive strategies of player 2 and all strategies of the scheduler. Moreover, π_{rand} is independent of the values of the global clock, tick and bl_1.*

Proof. The proof follows along similar line to that of Lemma 28. Let $\mathcal{A} = \{\widehat{s'} \mid \widehat{s'} \in R'$ and $R' \cap T_j \neq \emptyset$ for some $j < 2i\}$. Note that $\mathcal{A} \subseteq \mathsf{Reg}(T_{2i})$. We show player 1 can reach $\mathcal{A}$, without encountering a region R' such that $R' \cap (T_{2i} \cup \mathcal{A}) = \emptyset$. Let $\widehat{s} \in R$, with

$R \cap T_{2i} \neq \emptyset$, and $R \cap T_j = \emptyset$ for all $2 \leq j < 2i$. The result follows from Lemmas 36, 37, 38, 39, and 40. □

Lemma 36. *Let R be a region of $\mathcal{T}$ such that $R \cap T_{2i} \neq \emptyset$, and $R \cap T_j = \emptyset$ for all $2 \leq j < 2i$. Then, player 1 has a move from every state in R to $\mathcal{A}$.*

Proof. Note that according to π_{pure}, player 1 always propose a move from $T_{2i} \cap R$ to $\mathcal{A}$ as the destination of the move of player 1 must be in rank $2i - 1$ or lower (note that a move of player 1 being chosen makes the index 1). Thus, since player 1 has a move from $T_{2i} \cap R$ to $\mathcal{A}$ according to π_{pure}, he must have a move from every $\widehat{s} \in R$ to $\mathcal{A}$ by Lemma 6. □

Consider a state $\widehat{s}$ in some region $R' \subseteq \mathsf{Reg}(T_{2i})$ of $\mathcal{T}$. Now consider the set of times at which moves can be taken so that the state changes from $\widehat{s}$ to $\mathcal{A}$. This set consists of a finite union of sets I_k of the form $(\alpha_l, \alpha_r), [\alpha_l, \alpha_r), (\alpha_l, \alpha_r]$, or $[\alpha_l, \alpha_r]$ where α_l, α_r are of the form d or $d - x$ for d some integer constant, and x some clock in C (this clock x is the same for all the states in R'). Furthermore, these intervals have the property that $\{\widehat{s} + \Delta \mid \Delta \in I_k\} \subseteq R_k$ for some region R_k, with $R_l \cap R_j = \emptyset$ for $j \neq l$. From a state $\widehat{s}$, consider the "earliest" interval contained in this union: the interval I such that the left endpoint is the infimum of the times at which player 1 can move to $\mathcal{A}$. We have that $\{\widehat{s} + \Delta \mid \Delta \in I\} \subseteq R_1$. Consider any state $\widehat{s}' \in R'$. Then from s', the earliest interval in the times required to get to $\mathcal{A}$ is also of the form I. Note that in allowing time to pass to get to R_1, we may possibly go outside T_{2i} (recall that T_{2i} is not a union of regions of $\mathcal{T}$).

If this earliest interval is left closed, then player 1 has a "shortest" move to $\mathcal{A}$. Then, this is the best move in our strategy for player 1, and she will always propose this move. Let the left and the right endpoints of target intervals be α_l, α_r respectively. Then, if the target interval is left open, let player 1 play a probabilistic strategy with time distributed uniformly at random over $(\alpha_l, (\alpha_l + \alpha_r)/2]$. Let us denote this player-1 strategy by π_{rand}. Also note that the z, *tick* and the bl_1 components play no role in determining the availability of moves.

Lemma 37. *Let R be a region of $\mathcal{T}$ such that $R \cap T_{2i} \neq \emptyset$, and $R \cap T_j = \emptyset$ for all $2 \leq j < 2i$. Then, the strategy π_{rand} ensures that from any state in R, the game stays in $\mathsf{Reg}(T_{2i})$ surely till $\mathcal{A}$ is visited.*

Proof. Let R be a region of $\mathcal{T}$ such that $R \cap T_{2i} \neq \emptyset$, and $R \cap T_j = \emptyset$ for all $2 \leq j < 2i$. Consider a state $\widehat{s} \in R \cap T_{2i}$. In π_{pure}, player 1 proposes a move to $\mathcal{A}$ from each state in $R \cap T_{2i}$. By Lemma 7, we have a unique set $M_2^{2i} = \{R' \mid$ player-2 moves to R' from R beat player-1 moves to $\mathcal{A}\}$. Since $\{\widehat{s}+t \mid t \in I\}$ constitutes a single region of $\mathcal{T}$, and I is the earliest interval that can land player 1 in $\mathcal{A}$, no new discrete actions become enabled due to the randomized strategy of player 1 — if player 2 can foil the randomized strategy of player 1 by taking a move to a region R' such that $R' \cap \mathsf{Reg}(T_{2i}) = \emptyset$, she can do so against π_{pure}. Thus, by induction using Lemma 7, we have that player 1 can guarantee with the randomized strategy that the game will stay in $\mathsf{Reg}(T_{2i})$ starting from a state in $R \cap T_{2i}$. Since the values z, *tick* and the bl_1 components play no role in determining the availability of moves, player 1 can ensure that the game states within $\mathsf{Reg}(T_{2i})$ starting from any state in a region R such that $R \cap T_{2i} \neq \emptyset$, and $R \cap T_j = \emptyset$ for all $2 \leq j < 2i$ till $\mathcal{A}$ is visited. □

If at any time the move of player 1 is chosen, then player 1 comes to $\mathcal{A}$. We show that when player 1 uses the randomized memoryless strategy π_{rand}, the probability of the move of player 1 being never chosen against a receptive strategy of player 2 is 0.

Lemma 38. *Let R be a region of $\mathcal{T}$ such that $R \cap T_{2i} \neq \emptyset$, and $R \cap T_j = \emptyset$ for all $2 \leq j < 2i$. Consider any receptive strategy π_2 of player 2, and a run $r \in \mathsf{Outcomes}(s, \pi_{\mathsf{rand}}, \pi_2)$ with $s \in R$. Suppose there exists $m \geq 0$ such that for all $k \geq m$, if $r[0..k]$ has not visited $\mathcal{A}$, then we have $\pi_{\mathsf{rand}}(r[0..k])$ to be left-closed. Then, we have that r visits $\mathcal{A}$.*

Proof. Note that if a move of player 1 is chosen at any point, then $\mathcal{A}$ is visited. Suppose the moves of player 1 are never chosen. Consider a run r against any strategy of player 2. Let us consider the run from $r[m]$ onwards. Only target left-closed regions occur form this point on. Let $r[m] = \widehat{s}' = \langle s', \mathfrak{z}, tick, bl_1 \rangle \in R'$. Consider the pure winning strategy π_{pure} from a state $\widehat{s}'' = \langle s', \mathfrak{z}', tick', bl_1' \rangle \in R' \cap T_{2i}$ (such a state must exist). The state $\widehat{s}''$ differs from $\widehat{s}'$ only in the values of the clock z, and the boolean variables *tick* and bl_1. The new values do not affect the moves available to either player. Consider $\widehat{s}''$ as the starting state. The strategy π_{pure} cannot propose shorter moves to $\mathcal{A} \cap (\cup_{i=2}^{2i-1} T_j)$, since π_{rand} proposes the earliest move to $\mathcal{A}$. Hence, if a receptive player-2 strategy π_2 can prevent π_{rand} from reaching $\mathcal{A}$ from $\widehat{s}'$, then it can also prevent π_{pure} from reaching $\mathcal{A} \cap (\cup_{i=2}^{2i-1} T_j)$ from $\widehat{s}''$, a contradiction. □

Lemma 39. *Let R be a region of $\mathcal{T}$ such that $R \cap T_{2i} \neq \emptyset$, and $R \cap T_j = \emptyset$ for all $2 \leq j < 2i$. Consider any receptive strategy π_2 of player 2, and a run $r \in \mathsf{Outcomes}(s, \pi_{\mathsf{rand}}, \pi_2)$ with*

$s \in R$. There exists $m \geq 0$ such that for all $k \geq m$ if (a) $r[0..k]$ has not visited $\mathcal{A}$, and (b) $\pi_{\mathsf{rand}}(r[0..k])$ is left-open with left-endpoint being α_l, then we have $\alpha_l = 0$.

Proof. Let α_l correspond to the left endpoint for one of the infinitely often occurring target left-open interval region R'.

1. We show that we cannot have α_l to be of the form d for some integer $d > 0$.

   We prove by contradiction. Suppose α_l is of the form d for some integer $d > 0$ for a region R'. Then, player 2 can always propose a time blocking move of d, this would mean that if the scheduler picks the move of player 2 (as both have the same delay), the next state will have *tick* true, no matter what the starting value of the clock z is. Now consider any state in $R' \cap T_{2i}$. The strategy π_{pure} always proposes some move to $\mathcal{A}$, and the time duration must be greater than d. Because of the d time-blocking move of player 2 new state will then be not in $\mathcal{A}$, and have $tick = \text{TRUE}$, hence, it will actually have an index of more than $2i$, contradicting the fact that π_{pure} ensured that the rank never decreased. Thus, $d > 0$ can never arise.

2. We show that we cannot have α_l to be of the form $d - x$ for some integer $d > 0$ and clock x.

   We prove by contradiction. Suppose clock x is not reset infinitely often in the run r. Then, the fact that the clock x has not progressed beyond d after some point in the run without being reset implies time is convergent, contradicting the fact that player 2 is playing with a receptive strategy (note that only moves of player 2 are being chosen). Thus, this situation cannot arise. Suppose x is reset infinitely often. Then, between a reset of x, and the time at which player 1 can jump to $\mathcal{A}$, we must have a time distance of more than d. Suppose R'' is one of the infinitely occurring regions in the run with the value of x being 0 in it. So player 2 has a strategy against our player-1 strategy such that one of the resulting runs contains a region subsequence $R'' \rightsquigarrow R'$. If this is so, then she would have a strategy which could do the same from every state in $R'' \cap T_{2i}$ against the pure winning strategy of player 1 (since the randomized strategy π_{rand} does not enable player 2 to go to more regions than against π_{pure}, as π_{rand} proposes moves to the earliest region in $\mathcal{A}$). But, if so, we have that *tick* will be true no matter what the starting value of z in $R'' \cap T_{2i}$, before player 1 can take a jump to $\mathcal{A}$ from $R' \cap T_{2i}$, taking the game outside of $\mathcal{A} \cup T_{2i}$. Since player 1 can stay inside T_{2i}, or visit $\mathcal{A}$ at each step with the infinite memory strategy π_{pure}, this cannot be so, that is, we cannot observe the region subsequence $R'' \rightsquigarrow R'$ for the player-1 randomized strategy. Hence the case of $\alpha_l = d - x$ cannot arise infinitely often.

The only remaining option is $\alpha_l = 0$, and we must have that the only randomized moves player 1 proposes after a while are of the form $(0, \alpha_r/2]$. □

Lemma 40. *Let R be a region of $\mathfrak{T}$ such that $R \cap T_{2i} \neq \emptyset$, and $R \cap T_j = \emptyset$ for all $2 \leq j < 2i$. Consider any receptive strategy π_2 of player 2, and a strategy π_{sched} of the scheduler. Let $\mathcal{E}$ denote the set of runs containing runs $r \in \mathsf{Outcomes}(s, \pi_{\mathsf{rand}}, \pi_2)$ with $s \in R$. such that there exists $m \geq 0$ and for all $k \geq m$ (a) $r[0..k]$ has not visited $\mathcal{A}$, and (b) $\pi_{\mathsf{rand}}(r[0..k])$ is left-open with left-endpoint being $\alpha_l = 0$. Then, we have $\Pr_{r[0]}^{\pi_{\mathsf{rand}}, \pi_2, \pi_{\mathsf{sched}}}(\mathsf{Reach}(\mathcal{A}) \,|\mathcal{E}) = 1$.*

Proof. Let R' be one of the infinitely often occurring regions in r with the target left-endpoint being $\alpha_l = 0$. Let β_i be the duration of the move of player 2 for the ith visit to R' Suppose $\alpha_r = d$. Then the probability of a move of player 1 being never chosen is less than $\prod_{i=1}^{\infty}(1 - \frac{2\beta_i}{d})$, which is 0 if $\sum_{i=1}^{\infty} \beta_i = \infty$ by Lemma 26. A similar analysis holds if player 2 proposes randomized moves with a time distribution $\mathcal{D}^{(\beta_i,\cdot]}, \mathcal{D}^{[\beta_i,\cdot]}, \mathcal{D}^{(\beta_i,\cdot)}$ or $\mathcal{D}^{[\beta_i,\cdot)}$. Suppose $\alpha_r = d - x$. Suppose $\alpha_r = d - x$. Again, the probability of a move of player 1 being never chosen is less than $\prod_{i=1}^{\infty}(1 - \frac{2\beta_i}{(d - \kappa_i(x))})$, and since $\frac{\beta_i}{(d - \kappa_i(x))} > \frac{\beta_i}{d}$, this also is 0 if $\sum_{i=1}^{\infty} \beta_i = \infty$ by Lemma 26. Finally, we note that if player 2 does not block time from T_{2i}, then for at least one region, she must propose a β_i sequence such that $\sum_{i=1}^{\infty} \beta_i = \infty$, and we will have that for this region, a move of player 1 will be chosen eventually with probability 1. □

Once player 1 reaches the target set, she can switch over to the finite-memory receptive strategy of Lemma 25. Thus, using Lemmas 25, 28, 34, and 35 we have the following theorem.

Theorem 20. *Let $\mathfrak{T}$ be a timed automaton game, and let S_R be a union of regions of $\mathfrak{T}$. Player 1 has a randomized, finite-memory, receptive, region strategy π_1 such that for all states $s \in \mathsf{SureWin}_1(\mathsf{Reach}(S_R))$, and for all scheduler strategies π_{sched}, the following assertions hold: (a) for all receptive strategies π_2 of player 2 we have $\Pr_s^{\pi_1, \pi_2, \pi_{\mathsf{sched}}}(\mathsf{Reach}(S_R)) = 1$; and (b) for all strategies π_2 of player 2 we have $\Pr_s^{\pi_1, \pi_2, \pi_{\mathsf{sched}}}(\mathsf{TimeDivBl}_1(\mathsf{Reach}(S_R))) = 1$.*

6.5 Parity Objectives: Randomized Finite-memory Receptive Strategies Suffice

In this section we show that randomized finite-memory almost-sure strategies exist for parity objectives. Let $\Omega : S \mapsto \{0, \ldots, k\}$ be the parity index function. We consider the case when $k = 2d$ for some d, and the case when $k = 2d-1$, for some d can be proved using similar arguments. If $k = 2d-1$, then we will will look at the dual odd parity objective: $\mathsf{Parity}_{\mathsf{odd}}(\Omega') = \{r \mid \max(\mathsf{InfOften}(r)) \text{ is odd }\}$, with $\Omega' = \Omega + 1 : S \mapsto \{1, \ldots, 2d\}$. If we get an odd parity objective with $\Omega' : S \mapsto \{1, \ldots, 2d-1\}$, then we can map it back to an even parity objective with $\Omega = \Omega' - 1$.

Given a timed game structure $\mathcal{T}$, a set $X \subsetneq S$, and a parity function $\Omega : S \mapsto \{0, \ldots, 2d\}$, with $d > 0$, let $\langle \mathcal{T}', \Omega' \rangle = \mathsf{ModifyEven}(\mathcal{T}, \Omega, X)$ be defined as follows: (a) the state space S' of $\mathcal{T}'$ is $\{s^\perp\} \cup S \setminus X$, where $s^\perp \notin S$; (b) $\Omega'(s^\perp) = 2d-2$, and $\Omega' = \Omega$ otherwise; (c) $\Gamma'_i(s) = \Gamma_i(s)$ for $s \in S \setminus X$, and $\Gamma'_i(s^\perp) = \Gamma_i(s^\perp) = \mathbb{R}_{\geq 0} \times \perp$; and (d) $\delta'(s, m) = \delta(s, m)$ if $\delta(s, m) \in S \setminus X$, and $\delta'(s, m) = s^\perp$ otherwise. We will use the function $\mathsf{ModifyEven}$ to play timed games on a subset of the original structure. The extra state, and the modified transition function are to ensure well-formedness of the reduced structure. We will now obtain receptive strategies for player 1 for the objective $\mathsf{Parity}(\Omega)$ using winning strategies for reachability and safety objectives. We consider the following procedure.

1. $i := 0$, and $\mathcal{T}_i = \mathcal{T}$.

2. Compute $X_i = \mathsf{SureWin}_1^{\mathcal{T}_i}(\Diamond(\Omega^{-1}(2d)))$.

3. Let $\langle \mathcal{T}'_i, \Omega' \rangle = \mathsf{ModifyEven}(\mathcal{T}_i, \Omega, X_i)$; and let $Y_i = \mathsf{SureWin}_1^{\mathcal{T}'_i}(\mathsf{Parity}(\Omega'))$. Let $L_i = S_i \setminus Y_i$, where S_i is the set of states of $\mathcal{T}_i$.

4. Compute $Z_i = \mathsf{SureWin}_1^{\mathcal{T}_i}(\Box(S_i \setminus L_i))$.

5. Let $(\mathcal{T}_{i+1}, \Omega) = \mathsf{ModifyEven}(\mathcal{T}, \Omega, S \setminus Z_i)$ and $i := i + 1$.

6. Go to step 2, unless $Z_{i-1} = S_i$.

Consider the sets $S \setminus Z_i$ that are removed in each iteration. For every L_i, the probability of player 1 winning in $\mathcal{T}$ is 0. This is because, from L_i, player 1 cannot visit the index $2d$ with positive probability, thus we can restrict our attention to $\mathcal{T}'$, and in this structure, L_i is not winning for player 1 almost surely. This in turn implies that

$S \setminus \mathsf{SureWin}_1^{\mathcal{T}_i}(\Box(S \setminus L_i))$ is a losing set for player 1 almost surely in the structure $\mathcal{T}$. Thus, at the end of the iterations, we have $\mathsf{SureWin}_1^{\mathcal{T}}(\mathsf{Parity}(\Omega)) \subseteq Z_i$. Hence, we have $(S \setminus Z_i) \cap \mathsf{SureWin}_1^{\mathcal{T}}(\mathsf{Parity}(\Omega)) = \emptyset$. We now exhibit randomized, finite-memory, receptive, region almost-sure winning strategies to show that the set Z_i is almost-sure winning.

The set Z_i on termination has two subsets: (a) $X_i = \mathsf{SureWin}_1^{\mathcal{T}_i}(\Diamond(\Omega^{-1}(2d)))$; and (b) $Y_i = S_i \setminus X_i$ such that player 1 wins in the structure $\mathcal{T}_i'$ for the parity objective $\mathsf{Parity}(\Omega)$. Let π^Y be a randomized, finite-memory, receptive, region almost-sure winning strategy for player 1 in $\mathcal{T}_i'$; since the range of Ω $\mathcal{T}_i'$ is $\{0, 1, \ldots, 2d-1\}$, by inductive hypothesis such a strategy exists. Consider any receptive strategy of player 2. If the game is in Y_i, then player 1 use the strategy π^Y, using the the run suffix r^Y, where r^Y is the largest suffix of the run such that all the states of r^Y belong to Y_i . Moreover, player 1 is never to blame if time converges (since π^Y is a receptive strategy). Suppose the game hits X_i. Then, player 1 uses a randomized, finite-memory, receptive, region almost-sure winning strategy π^X to visit the index $2d$, and as soon as $2d$ is visited, she switches over to a pure, finite-memory, receptive, region safety strategy for the objective $\Box(Z_i)$ to allow a fixed amount of time $\Delta > 0$ to pass. This can be done similar to the receptive strategies of Theorem 19 with an imprecise clock (in the imprecise clock the time elapse between any two ticks is at least Δ). Once time more than Δ has passed, player 1 switches over to π^X or π^Y, depending on whether the current state is in X_i or Y_i, respectively, and repeats the process. This is a receptive strategy which ensures that the maximal priority that is visited infinitely often is even almost-surely. The strategy also requires only a finite amount of memory.

Theorem 21. *Let $\mathcal{T}$ be a timed automaton game, and let Ω be a region parity index function. Suppose that player 1 has access to imprecise clock events such that between any two events, some time more than Δ passes for a fixed real $\Delta > 0$. Then, player 1 has a randomized, finite-memory, receptive, region strategy π_1 such that for all states $s \in \mathsf{SureWin}_1(\mathsf{Parity}(\Omega))$, and for all scheduler strategies π_{sched}, the following assertions hold: (a) for all receptive strategies π_2 of player 2 we have $\Pr_s^{\pi_1,\pi_2,\pi_{\mathsf{sched}}}(\mathsf{Parity}(\Omega)) = 1$; and (b) for all strategies π_2 of player 2 we have $\Pr_s^{\pi_1,\pi_2,\pi_{\mathsf{sched}}}(\mathsf{TimeDivBl}_1(\mathsf{Parity}(\Omega))) = 1$.*

Chapter 7

Robust Winning of Timed Games

7.1 Introduction

In the winning strategies presented in Chapter 3 for timed automaton games, there are cases where a player can win by proposing a certain strategy of moves, but where moves that deviate in the timing by an arbitrarily small amount from the winning strategy moves lead to her losing. If this is the case, then the synthesized controller needs to work with infinite precision in order to achieve the control objective. As this requirement is unrealistic, we propose two notions of *robust winning strategies*. In the first robust model, each move of player 1 (the "controller") must allow some jitter in when the action of the move is taken. The jitter may be arbitrarily small, but it must be greater than 0. We call such strategies *limit-robust*. In the second robust model, we give a lower bound on the jitter, i.e., every move of player 1 must allow for a fixed jitter, which is specified as a parameter for the game. In addition, the game specifies a nonzero lower bound on the response time, which is the minimal time between a discrete transition and an action of player 1. We call these strategies *bounded-robust*. The strategies of player 2 (the "plant") are left unrestricted (apart from being receptive). We show that these types of strategies are in strict decreasing order in terms of power: general strategies are strictly more powerful than limit-robust strategies; and limit-robust strategies are strictly more powerful than bounded-robust strategies for *any* lower bound on the jitter, i.e., there are games in which player 1 can win with a limit-robust strategy, but there does not exist any nonzero bound on the jitter for which player 1 can win with a bounded-robust strategy. The following example illustrates this issue.

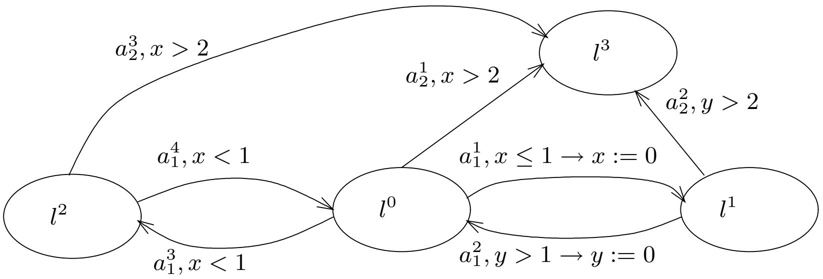


Figure 7.1: A timed automaton game $\mathcal{T}$.

Example 11. *Consider the timed automaton $\mathcal{T}$ in Fig. 7.1. The edges denoted a_1^k for $k \in \{1, 2, 3, 4\}$ are controlled by player 1 and edges denoted a_2^j for $j \in \{1, 2, 3\}$ are controlled by player 2. The objective of player 1 is $\Box(\neg l^3)$, ie., to avoid l^3. The important part of the automaton is the cycle l^0, l^1. The only way to avoid l^3 in a time divergent run is to cycle in between l^0 and l^1 infinitely often. In addition player 1 may choose to also cycle in between l^0 and l^2, but that does not help (or harm) her. Due to strategies being receptive, player 1 cannot just cycle in between l^0 and l^2 forever, she must also cycle in between l^0 and l^1; that is, to satisfy $\Box(\neg l^3)$ player 1 must ensure $(\Box\Diamond l^0) \wedge (\Box\Diamond l^1)$, where $\Box\Diamond$ denotes "infinitely often". But note that player 1 may cycle in between l^0 and l^2 as many (finite) number of times as she wants in between an l^0, l^1 cycle.*

In our analysis below, we omit such l^0, l^2 cycles for simplicity. Let the game start from the location l^0 at time 0, and let l^1 be visited at time t^0 for the first time. Also, let t^j denote the difference between times when l^1 is visited for the $j+1$-th time, and when l^0 is visited for the j-th time. We can have at most 1 time unit between two successive visits to l^0, and we must have strictly more than 1 time unit elapse between two successive visits to l^1. Thus, t^j must be in a strictly decreasing sequence. Also, for player 1 to cycle around l^0 and l^1 infinitely often, we must have that all $t^j \geq 0$. Consider any bounded-robust strategy. Since the jitter is some fixed ε_j, for any strategy of player 1 which tries to cycle in between l^0 and l^1, there will be executions where the transition labeled a_1^1 will be taken when x is less than or equal to $1 - \varepsilon_\mathrm{j}$, and the transition labeled a_1^2 will be taken when y is greater than $1 - \varepsilon_\mathrm{j}$. This means that there are executions where t^j decreases by at least $2 \cdot \varepsilon_\mathrm{j}$ in each cycle. But, this implies that we cannot having an infinite decreasing sequence of t^j's for any ε_j and for any starting value of t^0.

With a limit-robust strategy however, player 1 can cycle in between the two locations *infinitely often, provided that the starting value of x is strictly less than 1. This is because at each step of the game, player 1 can pick moves that are such that the clocks x and y are closer and closer to 1 respectively. A general strategy allows player 1 to win even when the starting value of x is 1. The details will be presented later in Example 13 in Section 7.3.*

In this chapter, we show that timed automaton games with limit-robust and bounded-robust strategies can be solved by reductions to general timed automaton games (with exact strategies). The reductions differentiate between whether the jitter is controlled by player 1 (in the limit-robust case), or by player 2 (in the bounded robust case). This is done by changing the winning condition in the limit-robust case, and by a syntactic transformation in the bounded-robust case. These reductions provide algorithms for synthesizing robust controllers for real-time systems, where the controller is guaranteed to achieve the control objective even if its time delays are subject to jitter. We also demonstrate that limit-robust strategies suffice for winning the special case of timed-automaton games where all guards and invariants are strict (i.e., open). The question of the *existence* of a lower bound on the jitter for which a game can be won with a bounded-robust strategy remains open.

Outline. In Section 7.2, we obtain robust winning sets for player 1 in the presence of non-zero jitter (which are assumed to be arbitrarily small) for each of her proposed moves. In Section 7.3, we assume the the jitter to be some fixed $\varepsilon_{\mathsf{j}} \geq 0$ for every move that is known. The strategies of player 2 are left unrestricted. In the case of lower-bounded jitter, we also introduce a *response time* for player-1 strategies. The response time is the minimum delay between a discrete action, and a discrete action of the controller. We note that the set of player-1 strategies with a jitter of $\varepsilon_{\mathsf{j}} > 0$ contains the set of player-1 strategies with a jitter of $\varepsilon_{\mathsf{j}}/2$ and a response time of $\varepsilon_{\mathsf{j}}/2$. Thus, the strategies of Section 7.2 automatically have a response time greater than 0. The winning sets in both sections are hence robust towards the presence of jitter and response times. In both sections, we show how the winning sets can be obtained by reductions to general timed automaton games. The results of Chapter 3 can then be used to obtain algorithms for computing the robust winning sets and strategies.

7.2 Robust Winning of Timed Parity Games

There is inherent uncertainty in real-time systems. In a physical system, an action may be prescribed by a controller, but the controller can never prescribe a single timepoint where that action will be taken with probability 1. There is usually some *jitter* when the specified action is taken, the jitter being non-deterministic. The model of general timed automaton games, where player 1 can specify exact moves of the form $\langle \Delta, a_1 \rangle$ consisting of an action together with a delay, assume that the jitter is 0. In this section, we model games where the jitter is assumed to be greater than 0, but arbitrarily small in each round of the game for player 1. The strategies of player 2 are left unrestricted. For ease of modeling, we also allow player 1 to relinquish control in a round of the game to player 2. We do this by letting the move of player 2 determine the next state whenever player 1 proposes a simple timed move. Formally, we define the *joint destination function* $\delta_{\mathsf{jd}} : S \times M_1 \times M_2 \mapsto 2^S$ by

$$\delta_{\mathsf{jd}}(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle) = \begin{cases} \{\delta(s, \langle \Delta_1, a_1 \rangle)\} & \text{if } \Delta_1 < \Delta_2 \text{ and } a_1 \neq \bot_1; \\ \{\delta(s, \langle \Delta_2, a_2 \rangle)\} & \text{if } \Delta_2 < \Delta_1 \text{ or } a_1 = \bot_1; \\ \{\delta(s, \langle \Delta_2, a_2 \rangle), \delta(s, \langle \Delta_1, a_1 \rangle)\} & \text{if } \Delta_2 = \Delta_1 \text{ and } a_1 \neq \bot_1. \end{cases}$$

We give this special power to player 1 as the controller always has the option of letting the state evolve in a controller-plant framework, without always having to provide inputs to the plant. We also need to modify the boolean predicate $\mathsf{blame}_i(s, m_1, m_2, s')$ indicates whether player i is "responsible" for the state change from s to s' when the moves m_1 and m_2 are proposed. The time elapsed when the moves $m_1 = \langle \Delta_1, a_1 \rangle$ and $m_2 = \langle \Delta_2, a_2 \rangle$ are proposed is given by $\mathsf{delay}(m_1, m_2) = \min(\Delta_1, \Delta_2)$. Denoting the opponent of player i by $\sim\!i = 3 - i$, for $i \in \{1, 2\}$, we define

$$\mathsf{blame}_i(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle, s') \;=\; \big(\Delta_i \leq \Delta_{\sim i} \;\wedge\; \delta(s, \langle \Delta_i, a_i \rangle) = s'\big) \wedge (i = 1 \rightarrow a_1 \neq \bot_1)\,.$$

These modifications are not necessary, but they are useful as they lead to a reduction in model size, and the formulae to be model checked.

Given a state s, a *limit-robust move* for player 1 is either the move $\langle \Delta, \bot_1 \rangle$ with $\langle \Delta, \bot_1 \rangle \in \Gamma_1(s)$; or it is a tuple $\langle [\alpha, \beta], a_1 \rangle$ for some $\alpha < \beta$ such that for every $\Delta \in [\alpha, \beta]$ we have $\langle \Delta, a_1 \rangle \in \Gamma_1(s)$. [1] Note that a time move $\langle \Delta, \bot_1 \rangle$ for player 1 implies that she is relinquishing the current round to player 2, as the move of player 2

[1] We can alternatively have an open, or semi-open time interval, the results do not change.

will always be chosen, and hence we allow a singleton time move. Given a limit-robust move $mrob_1$ for player 1, and a move m_2 for player 2, the set of possible outcomes is the set $\{\delta_{\mathsf{jd}}(s, m_1, m_2) \mid$ either (a) $mrob_1 = \langle \Delta, \bot_1 \rangle$ and$m_1 = mrob_1$; or (b) $mrob_1 = \langle [\alpha, \beta], a_1 \rangle$ and $m_1 = \langle \Delta, a_1 \rangle$ with $\Delta \in [\alpha, \beta]\}$. A *limit-robust strategy* π_1^{rob} for player 1 prescribes limit-robust moves to finite run prefixes. We let Π_1^{rob} denote the set of limit-robust strategies for player-1. Given an objective Φ, let $\mathsf{RobWinTimeDiv}_1^{\mathcal{T}}(\Phi)$ denote the set of states s in $\mathcal{T}$ such that player 1 has a limit-robust receptive strategy $\pi_1^{\mathsf{rob}} \in \Pi_1^R$ such that for all receptive strategies $\pi_2 \in \Pi_2^R$, we have $\mathsf{Outcomes}(s, \pi_1^{\mathsf{rob}}, \pi_2) \subseteq \Phi$. We say a limit-robust strategy π_1^{rob} is region equivalent to a strategy π_1 if for all runs r and for all $k \geq 0$, the following conditions hold: (a) if $\pi_1(r[0..k]) = \langle \Delta, \bot_1 \rangle$, then $\pi_1^{\mathsf{rob}}(r[0..k]) = \langle \Delta', \bot_1 \rangle$ with $\mathsf{Reg}(r[k] + \Delta) = \mathsf{Reg}(r[k] + \Delta')$; and (b) if $\pi_1(r[0..k]) = \langle \Delta, a_1 \rangle$ with $a_1 \neq \bot_1$, then $\pi_1^{\mathsf{rob}}(r[0..k]) = \langle [\alpha, \beta], a_1 \rangle$ with $\mathsf{Reg}(r[k] + \Delta) = \mathsf{Reg}(r[k] + \Delta')$ for all $\Delta' \in [\alpha, \beta]$. Note that for any limit-robust move $\langle [\alpha, \beta], a_1 \rangle$ with $a_1 \neq \bot_1$ from a state s, we must have that the set $\{s + \Delta \mid \Delta \in [\alpha, \beta]\}$ contains an open region of $\mathcal{T}$.

We now show how to compute the set $\mathsf{RobWinTimeDiv}_1^{\mathcal{T}}(\Phi)$. Given a timed automaton game $\mathcal{T}$, we have the corresponding enlarged game structure $\widehat{\mathcal{T}}$ which encodes time-divergence as presented in Chapter 3. We add another boolean variable to $\widehat{\mathcal{T}}$ to obtain another game structure $\widehat{\mathcal{T}}_{\mathsf{rob}}$. The state space of $\widehat{\mathcal{T}}_{\mathsf{rob}}$ is $\widehat{S} \times \{\text{TRUE}, \text{FALSE}\}$. The transition relation $\widehat{\delta}_{\mathsf{rob}}$ is such that $\widehat{\delta}_{\mathsf{rob}}(\langle \widehat{s}, rb_1 \rangle, \langle \Delta, a_i \rangle) = \langle \widehat{\delta}(\widehat{s}, \langle \Delta, a_i \rangle), rb_1' \rangle$, where $rb_1' = \text{TRUE}$ iff $rb_1 = \text{TRUE}$ and one of the following hold: (a) $a_i \in A_2^{\bot}$; or (b) $a_i = \bot_1$; or (c) $a_i \in A_1$ and $s + \Delta$ belongs to an open region of $\widehat{\mathcal{T}}$.

We first need the following Lemma.

Lemma 41. *Let $\mathcal{T}$ be a timed automaton game and $\widehat{\mathcal{T}}$ be the corresponding enlarged game structure. Let $\widehat{\Phi}$ be an ω-regular region objective of $\widehat{\mathcal{T}}$. If π_1 is a region strategy that is winning for $\widehat{\Phi}$ from $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$ and π_1^{rob} is a robust strategy that is region-equivalent to π_1, then π_1^{rob} is a winning strategy for $\widehat{\Phi}$ from $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$.*

Proof. Consider any strategy π_2 for player 2, and a state $\widehat{s} \in \mathsf{Win}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$. We have $\mathsf{Outcomes}(s, \pi_1^{\mathsf{rob}}, \pi_2)$ to be the set of runs r such that for all $k \geq 0$, either a) $\pi_1^{\mathsf{rob}}(r[0..k]) = \langle \Delta, \bot_1 \rangle$ and $r[k+1] = \widehat{\delta}_{\mathsf{jd}}(r[k], \langle \Delta, \bot_1 \rangle, \pi_2(r[0..k]))$ or,

$\pi_1^{\mathsf{rob}}(r[0..k]) = \langle [\alpha, \beta], a_1 \rangle$ and $r[k+1] = \widehat{\delta}_{\mathsf{jd}}(r[k], \langle \Delta, a_1 \rangle, \pi_2(r[0..k]))$ for some $\Delta \in [\alpha, \beta]$. It can be observed that $\mathsf{Outcomes}(s, \pi_1^{\mathsf{rob}}, \pi_2) = \bigcup_{\pi_1'} \mathsf{Outcomes}(\widehat{s}, \pi_1', \pi_2)$ where π_1' ranges over (non-robust) player-1 strategies such that for runs $r \in \mathsf{Outcomes}(\widehat{s}, \pi_1', \pi_2)$

and for all $k \geq 0$ we have $\pi_1'(r[0..k]) = \langle \Delta, \bot_1 \rangle$ if $\pi_1^{\mathsf{rob}}(r[0..k]) = \langle \Delta, \bot_1 \rangle$, and $\pi_1'(r[0..k]) = \langle \Delta, a_1 \rangle$ if $\pi_1^{\mathsf{rob}}(r[0..k]) = \langle [\alpha, \beta], a_1 \rangle$ for some $\Delta \in [\alpha, \beta]$; and π_1' acts like π_1 otherwise (note that the runs r and the strategies π_1' are defined inductively with respect to k, with $r[0] = \widehat{s}$). Each player-1 strategy π_1' in the preceeding union is region equivalent to π_1 since π_1^{rob} is region equivalent to π_1 and hence each π_1' is a winning strategy for player 1 by Lemma 9. Thus, $\mathsf{Outcomes}(s, \pi_1^{\mathsf{rob}}, \pi_2) = \bigcup_{\pi_1'} \mathsf{Outcomes}(\widehat{s}, \pi_1', \pi_2)$ is a subset of $\widehat{\Phi}$, and hence π_1^{rob} is a winning strategy for player 1. □

Theorem 22. *Given a state s in a timed automaton game $\mathcal{T}$ and an ω-regular region objective Φ, we have $s \in \mathsf{RobWinTimeDiv}_1^{\mathcal{T}}(\Phi)$ iff $\langle s, \cdot, \cdot, \cdot, \text{TRUE} \rangle \in \mathsf{Win}_1^{\widehat{\mathcal{T}}^{\mathsf{rob}}}(\Phi \wedge \Box(rb_1 = \text{TRUE}) \wedge (\Diamond\Box(tick = \text{FALSE}) \to (\Diamond\Box(bl_1 = \text{FALSE}))))$.*

Proof. 1. ($\Rightarrow$) Suppose player-1 has a limit-robust receptive strategy winning strategy π_1 for Φ. starting from a state s in $\mathcal{T}$. we show $\langle s, \cdot, \cdot, \cdot, \text{TRUE} \rangle \in \mathsf{Win}_1^{\widehat{\mathcal{T}}^{\mathsf{rob}}}(\Phi \wedge \Box(rb_1 = \text{TRUE}) \wedge (\Diamond\Box(tick = \text{FALSE}) \to (\Diamond\Box(bl_1 = \text{FALSE}))))$.

We may consider π_1 to be a strategy in $\widehat{\mathcal{T}}$. Since π_1 is a limit-robust strategy, player-1 proposes limit-robust moves at each step of the game. Given a state $\widehat{s}$, and a limit-robust move $\langle [\alpha, \beta], a_1 \rangle$, there always exists $\alpha < \alpha' < \beta' < \beta$ such that for every $\Delta \in [\alpha', \beta']$, we have that $\widehat{s} + \Delta$ belongs to an open region of $\widehat{\mathcal{T}}$. Thus, given any limit-robust strategy π_1, we can obtain another limit-robust strategy π_1' in $\widehat{\mathcal{T}}$, such that for every k, (a) if $\pi_1(r[k]) = \langle \Delta, \bot_1 \rangle$, then $\pi_1'(r[k]) = \pi_1(r[k])$; and (b) if $\pi_1(r[k]) = \langle [\alpha, \beta], a_1 \rangle$, then $\pi_1'(r[k]) = \langle (\Delta, a_1 \rangle$ with $\Delta \in [\alpha', \beta'] \subseteq [\alpha, \beta]$, and $\{r[k] + \Delta' \mid \Delta' \in [\alpha', \beta']\}$ being a subset of an open region of $\widehat{\mathcal{T}}$. Thus for any strategy π_2 of player-2, and for any run $r \in \mathsf{Outcomes}(\langle s, \cdot, \cdot, \cdot, \text{TRUE} \rangle, \pi_1', \pi_2)$, we have that r satisfies $\Box(rb_1 = \text{TRUE})$. Since π_1 was a receptive winning strategy for Φ, π_1' is also a receptive winning strategy for Φ. Hence, r also satisfies $\Phi \wedge \Diamond\Box(tick = \text{FALSE}) \to (\Diamond\Box(bl_1 = \text{FALSE})$.

2. ($\Leftarrow$) Suppose $\langle s, \cdot, \cdot, \cdot, \text{TRUE} \rangle \in \mathsf{Win}_1^{\widehat{\mathcal{T}}^{\mathsf{rob}}}(\Phi \wedge \Box(rb_1 = \text{TRUE}) \wedge (\Diamond\Box(tick = \text{FALSE}) \to (\Diamond\Box(bl_1 = \text{FALSE}))))$. We show that player-1 has a limit-robust receptive winning strategy from state s. Let π_1 be a winning region winning strategy for player-1 for the objective $\Phi \wedge \Box(rb_1 = \text{TRUE}) \wedge (\Diamond\Box(tick = \text{FALSE}) \to (\Diamond\Box(bl_1 = \text{FALSE})))$. For every run r starting from state $\langle s, \cdot, \cdot, \cdot, \text{TRUE} \rangle$, the strategy π_1 is such that $\pi_1(r[0..k]) = \langle \Delta^k, a_1^k \rangle$ such that either $a_1^k = \bot_1$, or $r[k] + \Delta^k$ belongs to an open region $\widehat{R}$ of $\widehat{S}$ Since R is an open region, there always exists some $\alpha < \beta$ such that for every $\Delta \in [\alpha, \beta]$,

we have $r[k] + \Delta \in R$. Consider the strategy π_1^{rob} that prescribes a limit-robust move $\langle [\alpha, \beta], a_1^k \rangle$ for the history $r[0..k]$ if $\pi_1(r[0..k]) = \langle \Delta^k, a_1^k \rangle$ with $a_1^k \neq \bot_1$, and $\pi_1^{\mathsf{rob}}(r[0..k]) = \pi_1(r[0..k])$ otherwise. The strategy π_1^{rob} is region-equivalent to π_1, and hence is also winning for player-1 by Lemma 41. Since it only prescribes limit-robust moves, it is a limit-robust strategy. And since it ensures $\Diamond\Box(\mathit{tick} = \textsc{false}) \rightarrow (\Diamond\Box(\mathit{bl}_1 = \textsc{false})$, it is a receptive strategy.

□

We say a timed automaton $\mathcal{T}$ is *open* if all the guards and invariants in $\mathcal{T}$ are open. Note that even though all the guards and invariants are open, a player might still propose moves to closed regions, e.g., consider an edge between two locations l_1 and l_2 with the guard $0 < x < 2$; a player might propose a move from $\langle l_1, x = 0.2 \rangle$ to $\langle l_2, x = 1 \rangle$. The next theorem shows that this is not required of player 1 in general, that is, to win for an ω-regular location objective, player 1 only needs to propose moves to open regions of $\mathcal{T}$. Let $\mathsf{Constr}^*(C)$ be the set of clock constraints generated by the grammar

$$\theta ::= x < d \mid x > d \mid x \geq 0 \mid x < y \mid \theta_1 \wedge \theta_2$$

for clock variables $x, y \in C$ and nonnegative integer constants d. An *open polytope* of $\mathcal{T}$ is set of states X such that $X = \{\langle l, \kappa \rangle \in S \mid \kappa \models \theta\}$ for some $\theta \in \mathsf{Constr}^*(C)$. An open polytope X is hence a union of regions of $\mathcal{T}$. Note that it may contain open as well as closed regions. We say a parity objective $\mathsf{Parity}(\Omega)$ is an open polytope objective if $\Omega^{-1}(j)$ is an open polytope for every $j \geq 0$.

Theorem 23. *Let $\mathcal{T}$ be an open timed automaton game and let $\Phi = \mathsf{Parity}(\Omega)$ be an ω-regular location objective. Then,* $\mathsf{WinTimeDiv}_1^{\mathcal{T}}(\Phi) = \mathsf{RobWinTimeDiv}_1^{\mathcal{T}}(\Phi)$.

Proof. We present a sketch of the proof. We shall work on the expanded game structure $\widehat{\mathcal{T}}^{\mathsf{rob}}$, and prove that $\langle s, \cdot, \cdot, \cdot, \textsc{true} \rangle \in \mathsf{Win}_1^{\widehat{\mathcal{T}}^{\mathsf{rob}}}(\Phi \wedge \Box(\mathit{rb}_1 = \textsc{true}) \wedge (\Diamond\Box(\mathit{tick} = \textsc{false}) \rightarrow (\Diamond\Box(\mathit{bl}_1 = \textsc{false})))$ iff $\langle s, \cdot, \cdot, \cdot, \textsc{true} \rangle \in \mathsf{Win}_1^{\widehat{\mathcal{T}}^{\mathsf{rob}}}(\Phi \wedge (\Diamond\Box(\mathit{tick} = \textsc{false}) \rightarrow (\Diamond\Box(\mathit{bl}_1 = \textsc{false})))$. The desired result will then follow from Theorem 22.

Consider the objective $\mathsf{TimeDivBl}_1(\Phi) = \Phi \wedge (\Diamond\Box(\mathit{tick} = \textsc{false}) \rightarrow (\Diamond\Box(\mathit{bl}_1 = \textsc{false}))$. Let $\widehat{\Omega}$ be the parity index function such that $\mathsf{Parity}(\widehat{\Omega}) = \mathsf{TimeDivBl}_1(\Phi)$. Since Φ is a location objective, and all invariants are open, we have $\widehat{\Omega}^{-1}(j)$ to be an open polytope of $\widehat{\mathcal{T}}^{\mathsf{rob}}$ for all indices $j \geq 0$ (recall that a legal state of $\mathcal{T}$ must satisfy the invariant of the location it is in).

The winning set for a parity objective $\mathsf{Parity}(\widehat{\Omega})$ can be described by a μ-calculus formula, we illustrate the case for when $\widehat{\Omega}$ has only two priorities. The μ-calculus formula is then: $\mu Y \nu X \left[(\widehat{\Omega}^{-1}(1) \cap \mathsf{CPre}_1(Y)) \cup (\widehat{\Omega}^{-1}(0) \cap \mathsf{CPre}_1(X))\right]$. This set can be computed from a (finite) iterative fixpoint procedure. Let $Y^* = \mu Y \nu X \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))\right]$. The iterative fixpoint procedure computes $Y_0 = \emptyset \subseteq Y_1 \subseteq \cdots \subseteq Y_n = Y^*$, where $Y_{i+1} = \nu X \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y_i)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X))\right]$. We claim that each Y_i for $i > 0$ is a union of open polytopes of $\widehat{\mathcal{T}}^{\mathsf{rob}}$. This is because (a) the union and intersection of a union of open polytopes is again a union of open polytopes, and (b) $\nu X(A \cup (B \cap \mathsf{CPre}_1(X)))$ is an open polytope provided A, B are open polytopes, and $\mathcal{T}$ is an open timed automaton game. We can consider the states in $Y_i \setminus Y_{i-1}$ as being added in two steps, T_{2i-1} and $T_{2i}(= Y_i)$ as follows:

1. $T_{2i-1} = \widehat{\Omega}^{-1}(1) \cap \mathsf{CPre}_1(Y_{i-1})$. T_{2i-1} is clearly a subset of Y_i.
2. $T_{2i} = \nu X \left[T_{2i-1} \cup (\widehat{\Omega}^{-1}(0) \cap \mathsf{CPre}_1(X))\right]$. Note $(T_{2i} \setminus T_{2i-1}) \cap \widehat{\Omega}^{-1}(1) = \emptyset$.

Thus, in odd stages we add states with index 1, and in even stages we add states with index 0. The *rank* of a state $\widehat{s} \in Y^*$ is j if $\widehat{s} \in T_j \setminus \cup_{k=0}^{j-1} T_k$. Each rank thus consists of states forming an open polytope. A winning strategy for player 1 can also be obtained based on the fixpoint iteration. The requirements on a strategy to be a winning strategy based on the fixpoint schema are:

1. For a state of even rank j, the strategy for player 1 must ensure that she has a move such that against all moves of player 2, the next state either (a) has index 0 and belongs to the same rank or less, or (b) the next state has index 1 and belongs to rank smaller than j.
2. For a state of odd rank j, the strategy for player 1 must ensures that she has a move such that against all moves of player 2, the next state belongs to a lower rank.

Since the rank sets are all open polytopes, and $\mathcal{T}$ is an open timed automaton, we have that there exists a winning strategy which from every state in a region $\widehat{R}$, either proposes a pure time move, or proposes a move to an open region (as every open polytope must contain an open region). Hence, this particular winning strategy also ensures that $\Box(rb_1 = \text{TRUE})$ holds. Thus, this strategy ensures $\mathsf{TimeDivBl}_1(\Phi) \wedge \Box(rb_1 = \text{TRUE})$. The general case of an index function of order greater than two can be proved by an inductive argument. □

7.3 Winning with Bounded Jitter and Response Time

The limit-robust winning strategies described in Section 7.2 did not have a lower bound on the jitter: player 1 could propose a move $\langle[\alpha, \alpha+\varepsilon], a_1\rangle$ for arbitrarily small α and ε. In some cases, the controller may be required to work with a known jitter, and also a finite *response time*. Intuitively, the response time is the minimum delay between a discrete action and a discrete action of the controller. We note that the set of player-1 strategies with a jitter of $\varepsilon_{\mathsf{j}} > 0$ contains the set of player-1 strategies with a jitter of $\varepsilon_{\mathsf{j}}/2$ and a response time of $\varepsilon_{\mathsf{j}}/2$. Thus, the strategies of Section 7.2 automatically have a response time greater than 0. The winning sets in both sections are hence robust towards the presence of jitter and response times. We model these known jitter and response times by allowing player 1 to propose moves with a single time point, but we make the jitter and the response time explicit and modify the semantics as follows. Player 1 can propose exact moves (with a delay greater than the response time), but the actual delay in the game will be controlled by player 2 and will be in a jitter interval around the proposed player-1 delay. The moves and strategies of player 2 are again left unrestricted.

Given a finite run $r[0..k] = s_0, \langle m_1^0, m_2^0\rangle, s_1, \langle m_1^1, m_2^1\rangle, \ldots, s_k$, let $\mathsf{TimeElapse}(r[0..k]) = \sum_{j=p}^{k-1} \mathsf{delay}(m_1^j, m_2^j)$ where p is the least integer greater than or equal to 0 such that for all $k > j \geq p$ we have $m_2^j = \langle\Delta_2^j, \bot_2\rangle$ and $\mathsf{blame}_2(s_j, m_1^j, m_2^j, s_{j+1}) = \text{TRUE}$ (we take $\mathsf{TimeElapse}(r[0..k]) = 0$ if $p = k$). Intuitively, $\mathsf{TimeElapse}(r[0..k])$ denotes the time that has passed due to a sequence of contiguous pure time moves leading upto s_k in the run $r[0..k]$. Let $\varepsilon_{\mathsf{j}} \geq 0$ and $\varepsilon_{\mathsf{r}} \geq 0$ be given bounded jitter and response time (we assume both are rational). Since a pure time move of player 1 is a relinquishing move, we place no restriction on it. Player 2 can also propose moves such that only time advances, without any discrete action being taken. in this case, we need to adjust the remaining response time. Formally, an ε_{j}*-jitter* ε_{r}*-response bounded-robust strategy* π_1 of player 1 proposes a move $\pi_1(r[0..k]) = m_1^k$ such that either

- $m_1^k = \langle\Delta^k, \bot_1\rangle$ with $\langle\Delta, \bot_1\rangle \in \Gamma_1(S)$, or,
- $m_1^k = \langle\Delta^k, a_1\rangle$ such that the following two conditions hold:
  - $\Delta^k \geq \max(0, \varepsilon_{\mathsf{r}} - \mathsf{TimeElapse}(r[0..k]))$, and,
  - $\langle\Delta', a_1\rangle \in \Gamma_1(s)$ for all $\Delta' \in [\Delta^k, \Delta^k + \varepsilon_{\mathsf{j}}]$.

Given a move $m_1 = \langle \Delta, a_1 \rangle$ of player 1 and a move m_2 of player 2, the set of resulting states is given by $\delta_{\mathsf{jd}}(s, m_1, m_2)$ if $a_1 = \bot_1$, and by $\{\delta_{\mathsf{jd}}(s, m_1 + \epsilon, m_2) \mid \epsilon \in [0, \varepsilon_{\mathsf{j}}]\}$ otherwise. Given an ε_{j}-jitter ε_{r}-response bounded-robust strategy π_1 of player 1, and a strategy π_2 of player 2, the set of possible outcomes in the present semantics is denoted by $\mathsf{Outcomes}_{jr}(s, \pi_1, \pi_2)$. We denote the winning set for player 1 for an objective Φ given finite ε_{j} and ε_{r} by $\mathsf{JRWinTimeDiv}_1^{\mathcal{T},\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}(\Phi)$. We now show that $\mathsf{JRWinTimeDiv}_1^{\mathcal{T},\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}(\Phi)$ can be computed by obtaining a timed automaton $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$ from $\mathcal{T}$ such that $\mathsf{WinTimeDiv}_1^{\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}}(\Phi) = \mathsf{JRWinTimeDiv}_1^{\mathcal{T},\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}(\Phi)$.

Given a clock constraint φ we make the clocks appearing in φ explicit by denoting the constraint as $\varphi(\overrightarrow{x})$ for $\overrightarrow{x} = [x_1, \ldots, x_n]$. Given a real number δ, we let $\varphi(\overrightarrow{x} + \delta)$ denote the clock constraint φ' where φ' is obtained from φ by syntactically substituting $x_j + \delta$ for every occurrence of x_j in φ. Let $f^{\varepsilon_{\mathsf{j}}} : \mathsf{Constr}(C) \mapsto \mathsf{Constr}(C)$ be a function defined by $f^{\varepsilon_{\mathsf{j}}}(\varphi(\overrightarrow{x})) = \mathsf{ElimQuant}\left(\forall \delta \; (0 \leq \delta \leq \varepsilon_{\mathsf{j}} \rightarrow \varphi(\overrightarrow{x} + \delta))\right)$, where $\mathsf{ElimQuant}$ is a function that eliminates quantifiers (this function exists as we are working in the theory of reals with addition [FC75], which admits quantifier elimination). The formula $f^{\varepsilon_{\mathsf{j}}}(\varphi)$ ensures that φ holds at all the points in $\{\overrightarrow{x} + \Delta \mid \Delta \leq \varepsilon_{\mathsf{j}}\}$.

We now describe the timed automaton $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$. The automaton has an extra clock z. The set of actions for player 1 is $\{\langle 1, e \rangle \mid e \text{ is a player-1 edge in } \mathcal{T}\}$ and for player 2 is $A_2 \cup \{\langle a_2, e \rangle \mid a_2 \in A_2 \text{ and } e \text{ is a player-1 edge in } \mathcal{T}\} \cup \{\langle 2, e \rangle \mid e \text{ is a player-1 edge in } \mathcal{T}\}$ (we assume the unions are disjoint). For each location l of $\mathcal{T}$ with the outgoing player-1 edges $e_1^1, \ldots, e_1^m$, the automaton $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$ has $m + 1$ locations: $l, l_{e_1^1}, \ldots, l_{e_1^m}$. Every edge of $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$ includes z in its reset set. The invariant for l is the same as the invariant for l in $\mathcal{T}$. All player-2 edges of $\mathcal{T}$ are also player-2 edges in $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$ (with the reset set being expanded to include z). The invariant for l_{e_j} is $z \leq \varepsilon_{\mathsf{j}}$. If $\langle l, a_2, \varphi, l', \lambda \rangle$ is an edge of $\mathcal{T}$ with $a_2 \in A_2$, then then $\langle l_{e_j}, \langle a_2, e_j \rangle, \varphi, l', \lambda \cup \{z\} \rangle$ is a player-2 edge of $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$ for every player-1 edge e_j of $\mathcal{T}$. For every player-1 edge $e_j = \langle l, a_1^j, \varphi, l', \lambda \rangle$ of $\mathcal{T}$, the location l of $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$ has the outgoing player-1 edge $\langle l, \langle 1, e_j \rangle, f^{\varepsilon_{\mathsf{j}}}\left(\gamma^{\mathcal{T}}(l)\right) \wedge (z \geq \varepsilon_{\mathsf{r}}) \wedge f^{\varepsilon_{\mathsf{j}}}(\varphi), l_{e_j}, \lambda \cup \{z\} \rangle$. The location l_{e_j} also has an additional outgoing *player-2* edge $\langle l_{e_j}, \langle 2, e_j \rangle, \varphi, l', \lambda \cup \{z\} \rangle$. The automaton $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$ as described contains the rational constants ε_{r} and ε_{j}. We can change the timescale by multiplying every constant by the least common multiple of the denominators of ε_{r} and ε_{j} to get a timed automaton with only integer constants. Intuitively, in the game $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$, player 1 moving from l to l_{e_j} with the edge $\langle 1, e_j \rangle$ indicates the desire of player 1 to pick the edge e_j from location l in the game $\mathcal{T}$. This is possible in $\mathcal{T}$ iff the (a) more that ε_{r} time has passed since the last discrete action, (b) the edge e_j is enabled for at least ε_{j} more time units, and (c) the invariant of l is satisfied for at least ε_{j} more time units. These three requirements are captured by the new guard in $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$, namely $f^{\varepsilon_{\mathsf{j}}}\left(\gamma^{\mathcal{T}}(l)\right) \wedge (z \geq \varepsilon_{\mathsf{r}}) \wedge f^{\varepsilon_{\mathsf{j}}}(\varphi)$. The presence of jitter in $\mathcal{T}$ causes uncertainty in when exactly the edge e_j is taken. This is modeled in $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$ by having the location l_{e_j} be controlled entirely by player 2 for a duration of ε_{j} time units. Within ε_{j} time units, player 2 must either propose a move $\langle a_2, e_j\rangle$ (corresponding to one of its own moves a_2 in $\mathcal{T}$, or allow the action $\langle 2, e_j\rangle$ (corresponding to the original player-1 edge e_j) to be taken. Given a parity function $\Omega^{\mathcal{T}}$ on $\mathcal{T}$, the parity function $\Omega^{\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}}$ is given by $\Omega^{\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}}(l) = \Omega^{\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}}(l_{e_j}) = \Omega^{\mathcal{T}}(l)$ for every player-1 edge e_j of $\mathcal{T}$. In computing the winning set for player 1, we need to modify blame_1 for technical reasons. Whenever an action of the form $\langle 1, e_j\rangle$ is taken, we blame player 2 (even though the action is controlled by player 1); and whenever an action of the form $\langle 2, e_j\rangle$ is taken, we blame player 1 (even though the action is controlled by player 2). Player 2 is blamed as usual for the actions $\langle a_2, e_j\rangle$. This modification is needed because player 1 taking the edge e_j in $\mathcal{T}$ is broken down into two stages in $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$. If player 1 to be blamed for the edge $\langle 1, e_j\rangle$, then the following could happen: (a) player 1 takes the edge $\langle 1, e_j\rangle$ in $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$ corresponding to her intention to take the edge e_j in $\mathcal{T}$ (b) player 2 then proposes her own move $\langle a_2, e_j\rangle$ from l_{e_j}, corresponding to her blocking the move e_j by a_2 in $\mathcal{T}$. If the preceeding scenario happens infinitely often, player 1 gets blamed infinitely often even though all she has done is signal her intentions infinitely often, but her actions have not been chosen. Hence player 2 is blamed for the edge $\langle 1, e_j\rangle$. If player 2 allows the intended player 1 edge by taking $\langle 2, e_j\rangle$, then we must blame player 1. We note that this modification is not required if $\varepsilon_{\mathsf{r}} > 0$.

Example 12 (Construction of $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$). *An example of the construction is given in Figure 7.2, corresponding to the timed automaton of Figure 7.1. The location l^3 is an absorbing location — it only has self-loops (we omit these self loops in the figures for simplicity). For the automaton $\mathcal{T}$, we have $A_1 = \{a_1^1, a_1^2, a_1^3, a_1^4\}$ and $A_2 = \{a_2^1, a_2^2, a_2^3\}$. The invariants of the locations of $\mathcal{T}$ are all* TRUE. *Since $\mathcal{T}$ at most a single edge from any location l^j to l^k, all edges can be denoted in the form e_{jk}. The set of player-1 edges is then $\{e_{01}, e_{02}, e_{20}, e_{10}\}$. The location l^3 has been replicated for ease of drawing in $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$. Observe that $f^{\varepsilon_{\mathsf{j}}}(x \leq 1) = x \leq 1 - \varepsilon_{\mathsf{j}}$ and $f^{\varepsilon_{\mathsf{j}}}(y > 1) = y > 1 - \varepsilon_{\mathsf{j}}$.* □

The construction of $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$ can be simplified if $\varepsilon_{\mathsf{j}} = 0$ (then we do not need locations of the form l_{e_j}). Given a set of states $\widetilde{S}$ of $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$, let $\mathsf{JStates}(\widetilde{S})$ denote the projection of

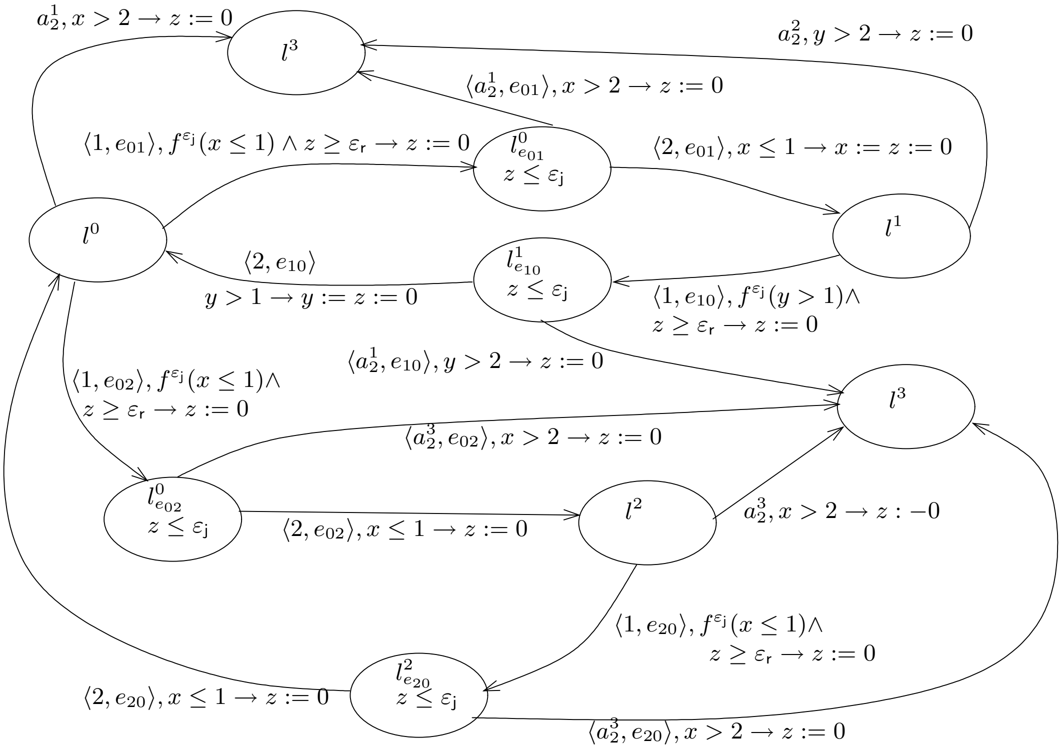

$a_2^1, x > 2 \to z := 0$
l^3
$a_2^2, y > 2 \to z := 0$
$\langle a_2^1, e_{01}\rangle, x > 2 \to z := 0$
$\langle 1, e_{01}\rangle, f^{\varepsilon_j}(x \le 1) \wedge z \ge \varepsilon_r \to z := 0$
$l^0_{e_{01}}$
$z \le \varepsilon_j$
$\langle 2, e_{01}\rangle, x \le 1 \to x := z := 0$
l^0
l^1
$\langle 2, e_{10}\rangle$
$y > 1 \to y := z := 0$
$l^1_{e_{10}}$
$z \le \varepsilon_j$
$\langle 1, e_{10}\rangle, f^{\varepsilon_j}(y > 1) \wedge$
$z \ge \varepsilon_r \to z := 0$
$\langle a_2^1, e_{10}\rangle, y > 2 \to z := 0$
$\langle 1, e_{02}\rangle, f^{\varepsilon_j}(x \le 1) \wedge$
$z \ge \varepsilon_r \to z := 0$
l^3
$\langle a_2^3, e_{02}\rangle, x > 2 \to z := 0$
$l^0_{e_{02}}$
$z \le \varepsilon_j$
$\langle 2, e_{02}\rangle, x \le 1 \to z := 0$
l^2
$a_2^3, x > 2 \to z : -0$
$\langle 1, e_{20}\rangle, f^{\varepsilon_j}(x \le 1) \wedge$
$z \ge \varepsilon_r \to z := 0$
$\langle 2, e_{20}\rangle, x \le 1 \to z := 0$
$l^2_{e_{20}}$
$z \le \varepsilon_j$
$\langle a_2^3, e_{20}\rangle, x > 2 \to z := 0$

Figure 7.2: The timed automaton game $\mathcal{T}^{\varepsilon_j,\varepsilon_r}$ obtained from $\mathcal{T}$.

states to $\mathcal{T}$, defined formally by $\mathsf{JStates}(\widetilde{S}) = \{\langle l, \kappa\rangle \in S \mid \langle l, \widetilde{\kappa}\rangle \in \widetilde{S}$ such that $\kappa(x) = \widetilde{\kappa}(x)$ for all $x \in C\}$, where S is the state space and C the set of clocks of $\mathcal{T}$.

Theorem 24. *Let $\mathcal{T}$ be a timed automaton game, $\varepsilon_r \ge 0$ the response time of player 1, and $\varepsilon_j \ge 0$ the jitter of player 1 actions such that both ε_r and ε_j are rational constants. Then, for any ω-regular location objective $\mathsf{Parity}(\Omega^{\mathcal{T}})$ of $\mathcal{T}$, we have $\mathsf{JStates}\left(\llbracket z = 0\rrbracket \cap \mathsf{WinTimeDiv}_1^{\mathcal{T}^{\varepsilon_j,\varepsilon_r}}(\mathsf{Parity}(\Omega^{\mathcal{T}^{\varepsilon_j,\varepsilon_r}}))\right) = \mathsf{JRWinTimeDiv}_1^{\mathcal{T},\varepsilon_j,\varepsilon_r}(\mathsf{Parity}(\Omega^{\mathcal{T}}))$, where $\mathsf{JRWinTimeDiv}_1^{\mathcal{T},\varepsilon_j,\varepsilon_r}(\Phi)$ is the winning set in the jitter-response semantics, $\mathcal{T}^{\varepsilon_j,\varepsilon_r}$ is the timed automaton with the parity function $\Omega^{\mathcal{T}^{\varepsilon_j,\varepsilon_r}}$ described above,and $\llbracket z = 0\rrbracket$ is the set of states of $\mathcal{T}^{\varepsilon_j,\varepsilon_r}$ with $\widetilde{\kappa}(z) = 0$.*

Example 13 (Differences between various winning modes)**.** *Consider the timed automaton $\mathcal{T}$ in Fig. 7.1. Let the objective of player 1 be $\Box(\neg l^3)$, ie., to avoid l^3. The important part of the automaton is the cycle l^0, l^1. The only way to avoid l^3 in a time divergent run is to cycle in between l^0 and l^1 infinitely often. In additional player 1 may choose to also cycle in between l^0 and l^2, but that does not help (or harm) her. In our analysis, we omit such*

l^0, l^2 cycles. Let the game start from the location l^0. In a run r, let t_1^j and t_2^j be the times when the a_1^1-th transition and the a_1^2-th transitions respectively are taken for the j-th time. The constraints are $t_1^j - t_1^{j-1} \leq 1$ and $t_2^j - t_2^{j-1} > 1$. If the game cycles infinitely often in between l^0 and l^1 we must also have that for all $j \geq 0$, $t_1^{j+1} \geq t_2^j \geq t_1^j$. we also have that if this condition holds then we can construct an infinite time divergent cycle of l^0, l^1 for some suitable initial clock values. Observe that $t_i^j = t_i^0 + (t_i^1 - t_i^0) + (t_i^2 - t_i^1) + \cdots + (t_i^j - t_i^{j-1})$ for $i \in \{1, 2\}$. We need $t_1^{m+1} - t_2^m = (t_1^{m+1} - t_1^m) + \sum_{j=1}^m \left\{(t_1^j - t_1^{j-1}) - (t_2^j - t_2^{j-1})\right\} + (t_1^0 - t_2^0) \geq 0$ for all $m \geq 0$. Rearranging, we get the requirement $\sum_{j=1}^m \left\{(t_2^j - t_2^{j-1}) - (t_1^j - t_1^{j-1})\right\} \leq (t_1^{m+1} - t_1^m) + (t_1^0 - t_2^0)$. Consider the initial state $\langle l^0, x = y = 0\rangle$. Let $t_1^0 = 1, t_2^0 = 1.1, t_1^j - t_1^{j-1} = 1, t_2^j - t_2^{j-1} = 1 + 10^{-(j+1)}$. We have $\sum_{j=1}^m \left\{(t_2^j - t_2^{j-1}) - (t_1^j - t_1^{j-1})\right\} \leq \sum_{j=1}^\infty 10^{-(j+1)} = 10^{-2} * \frac{1}{0.9} \leq 1 - 0.1 = (t_1^{m+1} - t_1^m) + (t_1^0 - t_2^0)$. Thus, we have an infinite time divergent trace with the given values. Hence $\langle l^0, x = y = 0\rangle \in \mathsf{WinTimeDiv}_1^{\mathfrak{T}}(\Box(\neg l^3))$. It can also be similarly seen that $\langle l^0, x = y = 1\rangle \in \mathsf{WinTimeDiv}_1(\Box(\neg l^3))$ (taking $t_1^0 = 0$ and $t_2^0 = 0.1$).

We now show $\langle l^0, x = y = 0\rangle \in \mathsf{RobWinTimeDiv}_1(\Box(\neg l^3))$. Consider $t_1^0 \in [0.9, 1], t_1^j - t_1^{j-1} \in [1 - 10^{-(j+1)}, 1], t_2^0 \in [1.05, 1.1], t_2^j - t_2^{j-1} \in [1 + 0.5 * 10^{-(j+1)}, 1 + 10^{-(j+1)}]$. We have $\sum_{j=1}^m \left\{(t_2^j - t_2^{j-1}) - (t_1^j - t_1^{j-1})\right\} \leq \sum_{j=1}^m 10^{-(j+1)} - (-10^{-(j+1)}) \leq 2 * \sum_{j=1}^\infty 10^{-(j+1)} = 2 * 10^{-2} * \frac{1}{0.9}$. We also have $(t_1^{m+1} - t_1^m) + (t_1^0 - t_2^0) \geq 1 - 10^{-(m+2)} + (0.9 - 1.1) \geq 0.7$. Thus, we have $\sum_{j=1}^m \left\{(t_2^j - t_2^{j-1}) - (t_1^j - t_1^{j-1})\right\} < 2 * 10^{-2} * \frac{1}{0.9} < 0.7 \leq (t_1^{m+1} - t_1^m) + (t_1^0 - t_2^0)$. This shows that we can construct an infinite cycle in between l^0 and l^1 for all the values in our chosen intervals, and hence that $\langle l^0, x = y = 0\rangle \in \mathsf{RobWinTimeDiv}_1(\Box(\neg l^3))$. Observe that $\langle l^0, x = y = 1\rangle \notin \mathsf{RobWinTimeDiv}_1(\Box(\neg l^3))$

We next show that $\langle l^0, x = y = 0\rangle \notin \mathsf{JRWinTimeDiv}_1^{\varepsilon_\mathsf{j}, \varepsilon_\mathsf{r}}(\Box(\neg l^3))$ for any $\varepsilon_\mathsf{j} > 0$. Observe that for any objective Φ, we have $\mathsf{JRWinTimeDiv}_1^{\varepsilon_\mathsf{j}, \varepsilon_\mathsf{r}}(\Phi) \subseteq \mathsf{JRWinTimeDiv}_1^{\varepsilon_\mathsf{j}, 0}(\Phi)$. Let $\varepsilon_\mathsf{j} = \epsilon$ and let $\varepsilon_\mathsf{r} = 0$. Consider any player-1 ϵ-jitter 0-response time strategy π_1 that makes the game cycle in between l^0 and l^1. Player 2 then has a strategy which "jitters" the player-1 moves by ϵ. Thus, the player-1 strategy π_1 can only propose a_1^1 moves with the value of x being less than or equal to $1 - \epsilon$ (else the jitter would make the move invalid). Thus, player 2 can ensure that $t_1^j - t_1^{j-1} \leq 1 - \epsilon$ for all j for some run (since x has the value $t_1^j - t_1^{j-1}$ when a_1^1 is taken for the j-th time for $j > 0$). We then have that for any player-1 ϵ-jitter 0-response time strategy, player 2 has a strategy such that for some resulting run, we have $t_1^j - t_1^{j-1} \leq 1 - \epsilon$ and $t_2^j - t_2^{j-1} > 1$. Thus, $\sum_{j=1}^m \left\{(t_2^j - t_2^{j-1}) - (t_1^j - t_1^{j-1})\right\} > m * \epsilon$, *which can be made arbitrarily large for a sufficiently large* m *for any* ϵ *and hence greater than* $(t_1^{m+1} - t_1^m) + (t_1^0 - t_2^0) \leq 1 + (t_1^0 - t_2^0)$ *for any initial values of* t_1^0 *and* t_2^0*. This violates the requirement for an infinite* l^0, l^1 *cycle. Thus,* $\langle l^0, x = y = 0\rangle \notin \mathsf{JRWinTimeDiv}_1^{\epsilon,0}(\Box(\neg l^3))$ *for any* $\epsilon > 0$*.*

Theorem 25. *Let* $\mathcal{T}$ *be a timed automaton and* Φ *an objective. For all* $\varepsilon_{\mathsf{j}} > 0$ *and* $\varepsilon_{\mathsf{r}} \geq 0$*, we have* $\mathsf{JRWinTimeDiv}_1^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}(\Phi) \subseteq \mathsf{RobWinTimeDiv}_1(\Phi) \subseteq \mathsf{WinTimeDiv}_1(\Phi)$*. All the subset inclusions are strict in general.*

Sampling semantics. Instead of having a response time for actions of player 1, we can have a model where player 1 is only able to take actions in an ε_{j} interval around sampling times, with a given time period $\varepsilon_{\mathsf{sample}}$. A timed automaton can be constructed along similar lines to that of $\mathcal{T}^{\varepsilon_{\mathsf{j}},\varepsilon_{\mathsf{r}}}$ to obtain the winning set.

Chapter 8

Conclusions

This thesis has presented solutions for certain problems on timed automata from a game theoretic viewpoint. In Chapter 2 we computed similarity and bisimilarity metrics by considering a game between two players. We showed that these metrics could be computed to within any desired degree of accuracy for timed automata. The problem of computing the exact metrics remains open. We note that if we consider the similarity and bisimilarity games, we can define k-step bounded similarity and bisimilarity metrics, where we only consider games upto k steps, where the goal of player 1 is to match the moves of the opponent only upto k steps. For each k, we can define k-step bounded similarity and bisimilarity metrics in the theory of reals with addition [FC75], and hence the bounded metrics are computable. We have however not been able to show that the metric functions reach a fixpoint. We suspect that they do. Note that we can detect if the value of a metric is going to escape to infinity using Theorem 1 and Lemma 1 of Chapter 2. We showed that these metrics provide a robust refinement theory for timed systems by relating the metrics to TCTL specifications. We also defined the quantitative discounted logic dCTL and provided a model checking algorithm for a subset of the logic. The problem in obtaining a model checking algorithm for the full logic is that the value of a formula depends on the the (uncountable) set of paths that exist, and all the states in that path. We observe here that it is not even known whether the maximum time that can elapse while avoiding a state s' starting from a state s can be computed. If this maximum time is t, then the value of the dCTL formula $\forall\Diamond s'$ is β^t where β is the discount factor. In general, for computing $\forall\Diamond\varphi$, where φ is another dCTL formula, we need to be able to obtain the maximum time that can be spent avoiding visiting states s', which leads to problems. Note that the maximum time problem is typically presented (as in Chapter 5) as the maximum time possible avoiding a *region* R starting from a state s, and this maximum time can be computed. The computation of dual minimum time problem for two exact states is known to be decidable via a complicated reduction to the additive theory of real numbers [CJ99]. That paper in fact shows that even the simple binary reachability problem for two states of a timed automaton is highly nontrivial.

In Chapter 3, we presented improved algorithms for solving timed games with ω-regular objectives in the framework of [dAFH$^+$03] where we do not need to put any syntactic restriction on the structure of the game to ensure time divergence in the resulting plays. There is nothing unsound about working with a syntactic restriction, such as the strong non-zeno hypothesis which ensures that an integer amount of time passes in every cycle of the region graph, but the problem arises when systems do not satisfy this restriction. For example, we may be given a system where the lower bound on two successive input events may not be known. Our generalized approach is able to handle such cases. This generalization has a cost: (a) the number of parities in the corresponding finite state games increases by two, and (b) the semi-concurrent nature of the games leads to more complicated algorithms with a higher computational complexity than other algorithms in the literature which ensure that the timed games are inherently non-zeno by various syntactic restrictions. While the increase in the number of parities is unavoidable, we believe that further improvements in algorithms for solving timed games are possible which do not incur any penalty due to the concurrent nature of our games. This is because we use only a very restricted form of concurrency, which is used only to determine which player gets to determine a state in a round by proposing a move first.

In Chapter 4 we defined the game logics TATL and TATL*, which extend the untimed game logics ATL and ATL*, and showed that while model checking for TATL* was undecidable, model checking for TATL (in the timed game setting of Chapter 3) can be done in EXPTIME. These game logics capture the fact that control modules must achieve their objectives irrespective of the behavior of the environment. The undecidability of model checking for TATL* is due to the presence of timing constraints in path formulae. We still have decidability for classes of logics that subsume TATL but which do not add any new timing constraints in path formulae. For example, the work in [BLMO07] presents a model checking algorithm for a logic that consists of TATL together with ATL*.

In Chapter 5 we showed that the minimum time required by player 1 to satisfy a proposition irrespective of what player 2 does is computable in EXPTIME, and moreover that this time is given by a simple function over regions. The dual problem of the maximum time that player 1 can spend avoiding a particular proposition can also be solved in a similar fashion, with the maximum time having the same functional form over regions.

In Chapter 6 we introduced randomization to reduce the memory requirements for controllers. We showed that we have to be careful when introducing extra clocks in the system for solving games, for that equates in some cases to giving the controller infinite memory. We used uniform randomization, but many other probability distributions can be used. The controller need not even know the exact probability distribution, all that is required is a known bound that is greater than zero for the probability distribution function. We note that if there is no such bound, the strategy presented in the chapter for player 1 might in fact fail. For example, if the probability distribution function is triangular, and starts from zero, then it can be shown that player 2 has a receptive strategy that wins against the presented player 1 strategy with probability $1 - \varepsilon$ for every $\varepsilon > 0$. We also showed that pure finite memory strategies suffice for player 1 for safety objectives. We conjecture that in fact *memoryless* pure strategies suffice for safety objectives. We observe here that our solution to model checking TATL in chapter 4 proceeded by introducing extra clocks (to measure global time and to measure time till the timing constraints expire) in an enlarged game structure. This reduction suffers from the same problem, and a future direction is to explore model checking TATL and other game logics when player 1 is restricted to use only finite memory. The solution presented in Chapter 5 to determine the minimum time required by player 1 to satisfy a proposition suffers from the same shortcoming.

In Chapter 7 we presented two models for robust winning in timed games. In the first model, each move of player 1 must allow some jitter in when her proposed move is taken. The jitter may be arbitrarily small, but it must be greater than 0. We called such strategies *limit-robust*. In the second robust model, we are given a lower bound on the jitter, i.e., every move of player 1 must allow for a fixed jitter, which is specified as a parameter for the game. We called these strategies *bounded-robust*. We showed that winning sets under both models are computable, and that limit-robust strategies are strictly more powerful than bounded-robust strategies. Limit-robust strategies are of practical interest in addition to being of theoretical concern because of computational reasons. To compute ε bounded-robust winning sets (where ε is a rational constant), we first constructed another timed game where ε appeared as an explicit constant in the game, and used the standard trick of multiplying every constant in the system by the denominator of ε to get an integer valued timed automaton. If the denominator of ε is even moderately large, then the size of the region graph blows up (by a factor of $d^{|C|}$, where d is the denominator of ε) from the original region graph, making the algorithms on the graph intractable. The solution under limit-robust strategies does not involve this multiplication process, and hence may be used to compute an over-approximation of ε bounded-robust winning sets in such cases.

We just scratched the surface of robust timed games in Chapter 7. The question of *existence* of winning bounded-robust strategies in timed games remains open. Note that the union of bounded-robust player 1 strategies is strictly contained in the union of limit-robust strategies. This is because player 1 chooses the jitter in each round of the game in limit-robust strategies, and this jitter might converge to 0 over the sequence of proposed moves; and in bounded-robust strategies, the sequence of jitters must be bounded from below by some constant greater than 0. The problem of existence of winning limit-robust strategies is the dual problem (in a game theoretic framework) to the work in [Pur98, WDMR04, Dim07] which explore the set of reachable states (in the one player case) when (roughly) the constants to which clocks are compared to are increased by ε for *some* ε (which remains fixed for the game). In our case, we (roughly) work on the problem where the constants are *decreased* by ε. Formally, let $\mathsf{Reach}_{\varepsilon^+}$ denote the set of states that are reachable in a timed automaton when the constants to which clocks are compared to are increased by ε; and let $\mathsf{Win}_{\varepsilon^-}(\Phi)$ denote the winning set when player 1 uses ε bounded-robust strategies (for the objective Φ). Then, the works of [Pur98, WDMR04, Dim07] compute $\bigcup_{\varepsilon>0} \mathsf{Reach}_{\varepsilon^+}$; and we are interested in $\bigcap_{\varepsilon>0} \mathsf{Win}_{\varepsilon^-}(\Phi)$. A better approximation to bounded-robust winning sets is provided by $\bigcap_{\varepsilon>0} \mathsf{Win}_{\varepsilon^-}(\Phi)$ than by the limit-robust winning sets. The work in [Pur98, WDMR04, Dim07] also relates the presence of jitters in the *clock rates* (where clocks may increase at rates other than one) to increasing the system constants. We did not explore this relationship in timed games. A robust winning strategy needs to be robust towards (1) jitters in proposed player 1 delays, (2) jitters in clock rates, (3) observation delays, (4) finite precision in observations of clock values, (5) delays in observations, and (6) jitters in the constants to which the clocks are compared. It turns out that many of robustness factors are reducible to one another. The work of [Pur98, WDMR04] explores the interrelationships between 1-4 in the single player case in determining reachable sets. The discrete time behavior of hybrid automata with observation delays, finite precision and action delays was explored in [AT04, AT05]. Controlled systems are also typically sampled, with the controller only being able to observe the plant state at the sampled time-points. It has been shown in [CHR02] that the problem of determining the existence of a sampling controller for *some* sampling rate is undecidable in general. It however remains to be seem if the problem is still undecidable when the controller must also take into account the robustness factors mentioned above. The work in Chapters 4, 5 and 6 can also be redone in a robust framework.

We have not explored *weighted* timed games where each location is given a cost rate together with a discrete cost on transitions in the thesis. For examples, the objective of player 1 might be to minimize the cost incurred in reaching a particular location. This problem is decidable under the strong non-zenoness assumption [BCFL04, BBL04], but undecidable in the general case [BBBR07]. The proof for the undecidability of the problem uses a very precise reduction. Two directions to explore are (1) whether an approximation to the desired value is computable in weighted timed games in the general case, and (2) whether the values can be computed when player 1 is restricted to use robust receptive strategies.

Bibliography

[ACD93] R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking in dense real-time. *Inf. Comput.*, 104(1):2–34, 1993.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[AdAF05] B. Adler, L. de Alfaro, and M. Faella. Average reward timed games. In *FORMATS: Formal Modeling and Analysis of Timed Systems*, Lecture Notes in Computer Science 3829, pages 65–80. Springer, 2005.

[AH94] R. Alur and T. A. Henzinger. A really temporal logic. *Journal of the ACM*, 41:181–204, 1994.

[AH97] R. Alur and T. A. Henzinger. Modularity for timed and hybrid systems. In *CONCUR: Concurrency Theory*, Lecture Notes in Computer Science 1243, pages 74–88. Springer, 1997.

[AHK02] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

[ALW89] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In *ICALP: Automata, Languages, and Programming*, Lecture Notes in Computer Science 372, pages 1–17. Springer, 1989.

[AM99] E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *HSCC: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1569, pages 19–30. Springer, 1999.

[AM04] R. Alur and P. Madhusudan. Decision problems for timed automata: A survey. In *SFM*, pages 1–24, 2004.

[AT04] M. Agrawal and P. S. Thiagarajan. Lazy rectangular hybrid automata. In *HSCC: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 2993, pages 1–15. Springer, 2004.

[AT05] M. Agrawal and P. S. Thiagarajan. The discrete time behavior of lazy linear hybrid automata. In *HSCC: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 3414, pages 55–69. Springer, 2005.

[ATM05] R. Alur, S. L. Torre, and P. Madhusudan. Perturbed timed automata. In *HSCC: Hybrid Systems—Computation and Control*, 3414, pages 70–85, 2005.

[BBBR07] P. Bouyer, T. Brihaye, V. Bruyère, and J. F. Raskin. On the optimal reachability problem of weighted timed automata. *Formal Methods in System Design*, 31(2):135–175, 2007.

[BBL04] P. Bouyer, E. Brinksma, and K. G. Larsen. Staying alive as cheaply as possible. In *HSCC: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 2993, pages 203–218. Springer, 2004.

[BCFL04] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, Lecture Notes in Computer Science 3328, pages 148–160. Springer, 2004.

[BDMP03] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. In *CAV: Computer-Aided Verification*, Lecture Notes in Computer Science 2725, pages 180–192. Springer, 2003.

[BGNV05] A. Blass, Y. Gurevich, L. Nachmanson, and M. Veanes. Play to test. In *FATES: Formal Approaches to Testing of Software*, 2005.

[BHPR07a] T. Brihaye, T. A. Henzinger, V. S. Prabhu, and J.-F. Raskin. Minimum-time reachability in timed games. In *ICALP: Automata, Languages, and Programming*, Lecture Notes in Computer Science 4596, pages 825–837. Springer, 2007.

[BHPR07b] T. Brihaye, T. A. Henzinger, V. S. Prabhu, and J.F. Raskin. Minimum-time reachability in timed games. *UC Berkeley Tech. Report, UCB/EECS-2007-47*, 2007.

[BLMO07] T. Brihaye, F. Laroussinie, N. Markey, and G. Oreiby. Timed concurrent game structures. In *CONCUR: Concurrency Theory*, Lecture Notes in Computer Science 4703, pages 445–459. Springer, 2007.

[BMR06] P. Bouyer, N. Markey, and P. A. Reynier. Robust model-checking of linear-time properties in timed automata. In *LATIN: Theoretical Informatics*, Lecture Notes in Computer Science 3887, pages 238–249. Springer, 2006.

[BMR08] P. Bouyer, N. Markey, and P. A. Reynier. Robust analysis of timed automata *via* channel machines. In *FoSSaCS: Foundations of Software Science and Computation Structures*, LNCS 4962, pages 157–171. Springer, 2008.

[Büc62] J. R. Büchi. On a decision method in restricted second-order arithmetic. In E. Nagel, P. Suppes, and A. Tarski, editors, *Proceedings of the First International Congress on Logic, Methodology, and Philosophy of Science 1960*, pages 1–11. Stanford University Press, 1962.

[CB02] P. Caspi and A. Benveniste. Toward an approximation theory for computerised control. In *EMSOFT: Embedded Software*, Lecture Notes in Computer Science 2491, pages 294–304. Springer, 2002.

[CDF$^+$05] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR: Concurrency Theory*, Lecture Notes in Computer Science 3653, pages 66–80. Springer, 2005.

[Cer92] K. Cerans. Decidability of bisimulation equivalences for parallel timer processes. In *CAV: Computer-Aided Verification*, Lecture Notes in Computer Science 663, pages 302–315. Springer, 1992.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986.

[CGP00] E. M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking.* The MIT Press, 2000.

[Cha07] K. Chatterjee. *Stochastic Omega-Regular Games.* PhD thesis, EECS Department, University of California, Berkeley, Oct 2007.

[CHP08a] K. Chatterjee, T. A. Henzinger, and V. S. Prabhu. Timed parity games: Complexity and robustness. In *FORMATS: Formal Modeling and Analysis of Timed Systems*, Lecture Notes in Computer Science. Springer, 2008.

[CHP08b] K. Chatterjee, T. A. Henzinger, and V. S. Prabhu. Timed parity games: Complexity and robustness. *CoRR*, abs/0805.4167, 2008.

[CHP08c] K. Chatterjee, T. A. Henzinger, and V. S. Prabhu. Trading infinite memory for uniform randomness in timed games. In *HSCC: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 4981, pages 87–100. Springer, 2008.

[CHP08d] K. Chatterjee, T. A. Henzinger, and V. S. Prabhu. Trading infinite memory for uniform randomness in timed games. Technical Report UCB/EECS-2008-4, EECS Department, University of California, Berkeley, Jan 2008.

[CHR02] F. Cassez, T. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *HSCC: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 2289, pages 134–148. Springer, 2002.

[Chu62] A. Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35. Institut Mittag-Leffler, 1962.

[CJ99] H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *CONCUR: Concurrency Theory*, Lecture Notes in Computer Science 1664, pages 242–257. Springer, 1999.

[CY92] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1(4):385–415, 1992.

[dAFH$^+$03] L. de Alfaro, M. Faella, T A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR: Concurrency Theory*, Lecture Notes in Computer Science 2761, pages 144–158. Springer, 2003.

[dAFH$^+$05] L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. *Theoretical Computer Science*, 345:139–170, 2005.

[dAFS04] L. de Alfaro, M. Faella, and M. Stoelinga. Linear and branching metrics for quantitative transition systems. In *ICALP: Automata, Languages, and Programming*, Lecture Notes in Computer Science3142, pages 97–109, 2004.

[dAH01] L. de Alfaro and T. A. Henzinger. Interface theories for component-based design. In *EMSOFT: Embedded Software*, Lecture Notes in Computer Science 2211, pages 148–165. Springer, 2001.

[dAHM00] L. de Alfaro, T. A. Henzinger, and F. Y. C. Mang. Detecting errors before reaching them. In *CAV: Computer-Aided Verification*, Lecture Notes in Computer Science 1855, pages 186–201. Springer, 2000.

[dAHM01a] L. de Alfaro, T. A. Henzinger, and R. Majumdar. From verification to control: Dynamic programs for omega-regular objectives. In *LICS: Logic in Computer Science*, pages 279–290. IEEE Computer Society Press, 2001.

[dAHM01b] L. de Alfaro, T. A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *CONCUR: Concurrency Theory*, Lecture Notes in Computer Science 2154, pages 536–550. Springer, 2001.

[dAHM03] L. de Alfaro, T. A. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *ICALP: Automata, Languages, and Programming*, Lecture Notes in Computer Science 2719, pages 1022–1037. Springer, 2003.

[DGJP04] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labelled markov processes. *Theor. Comput. Sci.*, 318(3):323–354, 2004.

[Dil89] D. L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-independent Circuits.* The MIT Press, 1989.

[Dim07] C. Dima. Dynamical properties of timed automata revisited. In *FORMATS: Formal Modeling and Analysis of Timed Systems*, Lecture Notes in Computer Science 4763, pages 130–146. Springer, 2007.

[DM02] D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *STACS: Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 2285, pages 571–582. Springer, 2002.

[Eme90] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. Elsevier, 1990.

[FC75] J. Ferrante and C.Rackoff. A decision procedure for the first order theory on real addition with order. *SIAM Journal of Computing*, 4(1):69–76, 1975.

[Frä99] M. Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In *CSL: Computer Science Logic*, Lecture Notes in Computer Science 1683, pages 126–140. Springer, 1999.

[Fre05] G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *HSCC: Hybrid Systems—Computation and Control*, pages 258–273, 2005.

[FTM02] M. Faella, S. La Torre, and A. Murano. Dense real-time games. In *LICS: Logic in Computer Science*, pages 167–176. IEEE Computer Society, 2002.

[GHJ97] V. Gupta, T. A. Henzinger, and R. Jagadeesan. Robust timed automata. In *HART: Hybrid and Real-Time Systems*, Lecture Notes in Computer Science 1201, pages 331–345. Springer, 1997.

[GJP06] V. Gupta, R. Jagadeesan, and P. Panangaden. Approximate reasoning for real-time probabilistic processes. *Logical Methods in Computer Science*, 2(1), 2006.

[GJP08] A. Girard, A. A. Julius, and G. Pappas. Approximate simulation relations for hybrid systems. *Discrete Event dynamic Systems*, 18(2):163–179, 2008.

[GP07a] A. Girard and G. Pappas. Approximate bisimulation relations for constrained linear systems. *Automatica*, 43(8):1307–1317, 2007.

[GP07b] A. Girard and G. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5):782–798, 2007.

[HHK95] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 453–462. IEEE Computer Society Press, 1995.

[HHWT95] T. A. Henzinger, P .H. Ho, and H. Wong-Toi. A user guide to HyTech. In *TACAS: Tools and Algorithms for the Construction and Analysis of Systems*, volume 1019 of *Lecture Notes in Computer Science*, pages 41–71. Springer-Verlag, 1995.

[HK99] T. A. Henzinger and P. W. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221:369–392, 1999.

[HKR02] T. A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. *Information and Computation*, 173:64–81, 2002.

[HMP92] T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *ICALP: Automata, Languages, and Programming*, Lecture Notes in Computer Science 623, pages 545–558. Springer, 1992.

[HMP05] T. A. Henzinger, R. Majumdar, and V. S. Prabhu. Quantifying similarities between timed systems. In *FORMATS: Formal Modeling and Analysis of Timed Systems*, Lecture Notes in Computer Science 3829, pages 226–241. Springer, 2005.

[HNSY94] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.

[HP06] T. A. Henzinger and V. S. Prabhu. Timed alternating-time temporal logic. In *FORMATS: Formal Modeling and Analysis of Timed Systems*, Lecture Notes in Computer Science 4202, pages 1–17. Springer, 2006.

[HR00] T. A. Henzinger and J.-F. Raskin. Robust undecidability of timed and hybrid systems. In *HSCC: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1790, pages 145–159. Springer, 2000.

[HVG03] J. Huang, J. Voeten, and M. Geilen. Real-time property preservation in approximations of timed systems. In *MEMOCODE: Formal Methods and Models for Codesign*, pages 163–171, 2003.

[HVG04] J. Huang, J. Voeten, and M. Geilen. Real-time property preservation in concurrent real-time systems. In *RTCSA: Embedded and Real-Time Computing Systems*. Springer, 2004.

[Jur00] M. Jurdzinski. Small progress measures for solving parity games. In *STACS: Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 1770, pages 290–301. Springer, 2000.

[LPY97] K. G. Larsen, P. Pettersson, and W. Yi. Uppaal: Status & developments. In *CAV: Computer-Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 456–459. Springer, 1997.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *STACS: Theoretical Aspects of Computer Science*, pages 229–242, 1995.

[PAMS98] A. Pnueli, E. Asarin, O. Maler, and J. Sifakis. Controller synthesis for timed automata. In *Proc. System Structure and Control*. Elsevier, 1998.

[Pug02] C. C. Pugh. *Real Analysis*. Springer, 2002.

[Pur98] A. Puri. Dynamical properties of timed automata. In *FTRTFT: Formal Techniques in Real-Time and Fault-Tolerant Systems*, Lecture Notes in Computer Science 1486, pages 210–227. Springer, 1998.

[Sch04] K. Schneider. *Verification of Reactive Systems*. Springer, 2004.

[Sch07] S. Schewe. Solving parity games in big steps. In *Proc. FST TCS*. Springer-Verlag, 2007.

[SGSAL98] R. Segala, R. Gawlick, J.F. Søgaard-Andersen, and N. A. Lynch. Liveness in timed and untimed systems. *Inf. Comput.*, 141(2):119–171, 1998.

[Tas98] S. Tasiran. *Compositional and Hierarchical Techniques for the Formal Verification of Real-Time Systems.* Dissertation, University of California, Berkeley, USA, 1998.

[Tho97] W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer, 1997.

[VJ00] J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *CAV: Computer-Aided Verification*, Lecture Notes in Computer Science 1855, pages 202–215. Springer, 2000.

[WDMR04] M. D. Wulf, L. Doyen, N. Markey, and J.F. Raskin. Robustness and implementability of timed automata. In *FORMATS: Formal Modeling and Analysis of Timed Systems*, pages 118–133, 2004.

[WH91] H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *Proc. of 30th Conf. Decision and Control*, pages 1527–1528, 1991.

[WLR05] M. D. Wulf, L.Doyen, and J. F. Raskin. Almost asap semantics: from timed models to timed implementations. *Formal Asp. Comput.*, 17(3):319–341, 2005.