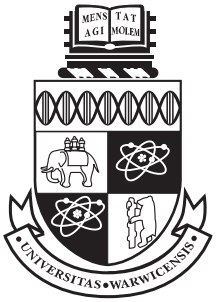
Competitive Optimisation on Timed Automata

By

Ashutosh Trivedi

A thesis submitted in partial fulfilment of the
requirements for the degree
of
Doctor of Philosophy in Computer Science



University of Warwick, Department of Computer Science

April 2009

To Tiziana, for maximising happiness and minimising troubles, and
to Miralisa, for playing two-player zero-sum games with me.

Contents

List of Figures

List of Tables

Acknowledgement

First of all, I wish to express my gratitude to my supervisor *Dr. Marcin Jurdziński*. His passion for mathematical rigour and elegance is the inspiration behind every significant discovery made during the research presented in this thesis. His patience, his timely advice, his encouragements, his belief in my abilities, and his continuous support through some of the difficult moments of my life, helped me and this thesis in uncountably infinite ways.

I am thankful to my adviser *Dr. Ranko Lazić* for ensuring safety, liveness, and fairness properties of my Ph.D. programme.

I would like to acknowledge my Ph.D. examiners *Dr. Joel Ouaknine* and *Dr. Ranko Lazić* for insightful comments on my thesis.

Sincere gratitude is extended to *Prof. Doron Peled* for his invaluable time and guidance during my first year at Warwick. I also thank him for his help during the admission process and for the financial support for my Ph.D. studies.

I would like to thank the support staff at the department of computer science and at the library services of the University of Warwick.

I wish to thank *Prof. Marta Kwiatkowska* for her help and support during my transition from a Ph.D. student to a postdoc researcher.

Special thanks to my friend *Nick Papanikolaou* for his constructive criticism of my ideas, and for various other discussions on some undecidable problems. I thank following colleagues for providing a stimulating and enjoyable atmosphere in the department of computer science: *Haris Aziz, Timothy Davidson, Alexander Dimovski, John Fearnly, Antony Holmes, Ritesh Krishna, Hongyang Qu, Michał Rutkowski*, and *Daniel Valdes-Amaro*.

I would like to thank all my friends for such an enjoyable time at Warwick. I would particularly like to mention: *Diana Apiyo, Rajesh S. Balakrishnan, Massoumeh Dashti, Estelle Guyez/Picard, Samuel Lelièvre, Eleftheria Papoulia, Serena Pattaro, Tina Philip, Jothi Philip, Cyril Picard, Leslie Robinson,* and *Agnieszka Rutkowska*. Also, thanks to my friends, *Amol Patil* and *Monica Mantri*, for some great time at Oxford.

I am fortunate to have a loving, caring, and ever supporting family. This acknowledgement is incomplete without thanking all of them: papa (*Suresh Chandra Trivedi*), mummy (*Bharti Trivedi*), didi (*Bhawana Katyayan*), jijaji (*Amogh Katyayan*), dada (*Vibhor Vibhu Trivedi*), bhabhiji (*Anjali Trivedi*), and Richa (*Richa Trivedi*). Special thanks to my mother-in-law *Franca Beatrice Girardi* for her love and support.

Finally I would like to dedicate this thesis to my wife *Tiziana*, for maximising happiness and minimising troubles, and to my daughter *Miralisa*, for playing two-player zero-sum games with me.

Declaration

This thesis is presented in accordance with the regulations for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself under the supervision of Dr. Marcin Jurdziński.

Although the key results of the thesis have been announced in proceedings of some international conferences, this thesis presents the results with complete proofs and complements them with illustrations.

Chapter 4 (Noncompetitive Optimisation) is an extended version of the paper *Concavely-Priced Timed Automata* [**JT08b**], which appeared in the proceedings of the conference on *formal modelling and analysis of timed systems (FORMATS)* 2008.

Chapter 5 (Reachability-Time Games) is an extended version of the paper *Reachability-Time Games on Timed Automata* [**JT07**], which appeared in the proceedings of *international colloquium on automata, languages and programming (ICALP)* 2007.

Chapter 6 (Average-Time Games) is an extended version of the paper *Average-Time Games* [**JT08a**], which appeared in the proceedings of the *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)* 2008.

Abstract

Timed automata are finite automata accompanied by a finite set of real-valued variables called *clocks*. Optimisation problems on timed automata are fundamental to the *verification* of properties of real-time systems modelled as timed automata, while the *control-program synthesis* problem of such systems can be modelled as a two-player game. This thesis presents a study of optimisation problems and two-player games on timed automata under a general heading of *competitive optimisation on timed automata*.

This thesis views competitive optimisation on timed automata as a multi-stage decision process, where one or two players are confronted with the problem of choosing a sequence of timed moves—a time delay and an action—in order to optimise their objectives. A solution of such problems consists of the "optimal" value of the objective and an "optimal" strategy for each player. This thesis introduces a novel class of strategies, called *boundary strategies*, that suggest to a player a symbolic timed move of the form (b, c, a)— "wait until the value of the clock c is in very close proximity of the integer b, and then execute a transition labelled with the action a". A distinctive feature of the competitive optimisation problems discussed in this thesis is the existence of optimal *boundary strategies*. Surprisingly perhaps, many competitive optimisation problems on timed automata of practical interest admit optimal boundary strategies. For example, optimisation problems with reachability price, discounted price, and average-price objectives, and two-player turn-based games with reachability time and average time objectives.

The existence of optimal boundary strategies allows one to work with a novel abstraction of timed automata, called a *boundary region graph*, where players can use only boundary strategies. An interesting property of a boundary region graph is that, for every state, the set of reachable states is finite. Hence, the existence of optimal boundary strategies permits us to reduce competitive optimisation problem on a timed automaton to the corresponding competitive optimisation problem on a finite graph.

1

Introduction

> Pessimism is, in brief, playing the sure game. You cannot lose at it; you may gain. It is the only view of life in which you can never be disappointed. Having reckoned what to do in the worst possible circumstances, when better arise, as they may, life becomes child's play.
>
> Thomas Hardy

Timed automata are finite automata accompanied by a finite set of real-valued variables. Optimisation problems on timed automata are fundamental to the *verification* of (quantitative timing) properties of systems modelled as timed automata. On the other hand, the problem of *control-program synthesis* of systems modelled as timed automata can be cast as a two-player game—where the two players correspond to the "controller" and the "environment"—and the control-program synthesis corresponds to computing winning (or optimal) strategies for the controller. We study optimisation problems and two-player games on timed automata under a general heading of *competitive optimisation on timed automata*. The theory of *dynamic programming* provides concepts and algorithms to analyse optimisation problems on multi-stage decision processes. We argue that dynamic programming is an effective tool to design and analyse algorithms for competitive optimisation on timed automata.

1.1. Motivation

This research has theoretical as well as practical motivations: algorithms for the competitive optimisation on timed automata provide upper bounds on the computational complexity of such problems. On the practical side, these algorithms are useful for the verification and controller synthesis of real-time open systems. In this section, with the help of a simple example, we demonstrate how certain two-player games on timed automata can be used to model some controller-synthesis problems of real-time systems.

By a real-time open system, we mean a computer system which interacts with its environment (open), and whose correctness depends critically on the time (real time) in which it performs some of its actions. Real-time open systems are prevalent parts of safety-critical systems, where a bug in their design can be catastrophic. Some examples [**Wik09**] of safety-critical real-time systems are artificial pacemaker, robotic surgery machine, nuclear reactor system, railway signalling and control system, aircraft-landing scheduling system, air-bag control system and satellite-launching system.

Given the safety-critical nature of the applications of real-time open systems, it is of paramount importance to ensure their correctness.

As an example of a safety-critical real-time open system, let us consider the following description of an artificial pacemaker.

The pacemaker leads detect the heart's own electrical activity (in the right atrium and right ventricle,) and transmit that information to the pacemaker generator. The generator—which, again, is a computer—analyses the heart's electrical signals, and uses that information to decide whether, when, and where to pace. If the heart rate becomes too slow, the generator transmits a tiny electrical signal to the heart, thus stimulating the heart muscle to contract. (This is called pacing.)

Pacemakers that have two leads not only keep the heart rate from dropping too low, they can also maintain the optimal coordination between the atria and the ventricles (by pacing the atrium and the ventricle in sequence.)

Thus, pacemakers do not take over the work of the heart—the heart still does its own beating—but instead, pacemakers merely help to regulate the timing of the heart beat. (Richard N. Fogoros, M.D., Cardiology, Pittsburgh, PA)

Observe that the main function of a pacemaker is to maintain the rate of heart beats when the natural pacemaker of the heart is either too slow or not working due to some problem. Artificial pacemaker is an example of real-time open systems as it interacts with its environment (heart's own electrical activity) and the correctness of its operations critically depends on the timing of its actions.

An elementary model of the working of an artificial pacemaker is shown in Figure 1.1 as a timed automaton: a finite state transition system with a finite set of continuous variables. These continuous variables are called clocks in timed automata parlance. Clocks can appear as guards on the transitions where they can be compared against rational numbers and after taking a transition it is possible to reset them to 0. In the figure, we have annotated a transition with the triplet: (clock guard, action name, set of clocks to be reset). All the clock variables evolve continuously with uniform rate. Since it is possible to reset clock variables after taking a transition, timed automata can express complex timing behaviours of the real-time systems (e.g., a clock can remember the time since a particular action was fired).

Let us read the timed automaton in Figure 1.1. According to this model, the pacemaker system can be in one of the two states [1] : ℓ_1, where artificial pacemaker is giving the pace signals to the heart; or ℓ_2, where heart's natural pacemaker is functioning normally. Let us say that the artificial pacemaker is controlling the state ℓ_1 and the environment is controlling the state ℓ_2. Notice that in this model every transition is associated with a heart beat. The

[1] As we shall see later, in the context of timed automata, these states are called *locations*, while the noun "state" commonly refer to a valid configuration—a location and a value assignment to the clocks—of the system.

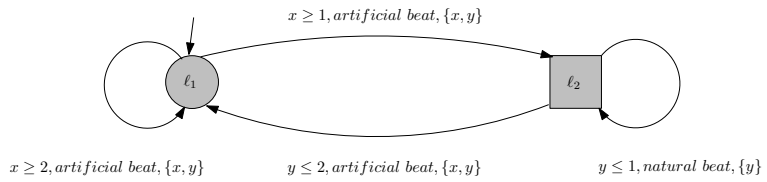


FIGURE 1.1. An artificial heart-pacemaker.

clock x measures the time since the last artificial heart beat and the clock y measures the time since last heart beat (both natural and artificial). When the system is in state ℓ_1, the controller can either choose to induce the artificial beats every 2 time units, or after 1 time unit from the previous artificial beat, it can induce another beat and choose to sense whether the natural pacemaker is working correctly, by changing the state of the system to ℓ_2. When the system is in state ℓ_2 (controlled by environment), the heart beats can be 1 time units apart, if the heart if working normally. However, if there is no heart beat for more than 2 time units, the state of the system is changed to ℓ_1 (controlled by pacemaker) and artificial pacemaker sends a pace signal.

Suppose that we wish to design an optimal controller for artificial pacemaker using this timed automaton as specification. Also suppose that optimisation objective of the controller is to ensure minimum time between the heart beats irrespective of the condition of the heart. Since in this example every transition is associated with a heart beat, minimising time between heart beats is same as minimising time per transition of the timed automaton. Hence the problem of designing an optimal controller can be reduced to the problem of finding a strategy of the controller (minimiser, player Min) which ensures minimum time per transition, assuming an adversarial environment (maximiser, player Max) which tries to maximise time per transition.

1.2. Preliminaries: Dynamic Programming and Game Theory

Before we delve into the discussion on competitive optimisation on timed automata, we present two concepts central to the study of competitive optimisation: *dynamic programming* and *game theory*. Dynamic programming provides theoretical tools and algorithms to analyse problems concerning optimal decision making. On the other hand, game theory is the study of optimal decision making problems of interacting rational agents with conflicting interests. In this section we present a few relevant concepts from these two domains.

1.2.1. Dynamic Programming

In the early 1950's, Richard Bellman [**Bel57**] systematically studied and developed a mathematical theory of multi-stage decision processes which, for historical reasons [**Bel84**], is known as *dynamic programming*. We begin the introduction to the theory by defining a multi-stage decision process.

1.2.1.1. Multi-stage Decision Processes

Multi-stage decision processes [**Bel57, How60, Ber95**] are the systems where a decision maker or a *player* is confronted with a problem of making sequential decisions of choosing actions in order to optimise a certain objective. We assume that every action is associated with a fixed price—a real number. Starting with the initial state of the system, at the first stage the player chooses an action as a result of which the state of the system is modified. In the next stage, the player makes the decision from the resulting state and the state of the system is modified accordingly. The total number of stages, also called the *horizon*, can be finite or infinite. The total cost of the sequence of actions over the horizon is a function of the prices of the individual actions. We refer to this function as the *cost function*. The goal of the player is to make decisions in such a way that it minimises (or maximises) the cost function. An important characteristic of such processes is that the decision made at any stage not only affects the immediate price, but also affects the future states and hence individual decisions can not be viewed in isolation.

Various multi-stage decision processes can succinctly be specified as optimisation problems on mathematical formalisms such as graphs, Markov decision processes [**Put94**], timed automata [**AD90, CY92, BBL04, BBBR07**], etc.

1.2.1.2. Strategies and Value

The optimisation problem for a multi-stage decision processes is to suggest a rule of making "allowable decisions" to the player so that no other sequence of allowable decisions yields a better value of the cost function. Such rule often takes the form of a function from the set of histories of decisions (sequence of visited states and executed actions) to the set of actions, and we call such a function a *policy* or a *strategy*.

Let S be the set of states and let A be the set of actions of a multi-stage decision process M. For a state s, let $A(s)$ be the set of allowable actions from the state s. Let $\pi : S \times A \to \mathbb{R}$ be the price function such that for every state $s \in S$ and action $a \in A(s)$, $\pi(s, a)$ gives the fixed-price of taking action a from the state s.

A strategy σ is then a function $(S \times A)^* \times S \to A$ such that for every history $h = \langle s_1, a_2, s_2, \ldots, a_n, s_n \rangle \in (S \times A)^* \times S$ we have $\sigma(h) \in A(s_n)$. Let us write Σ for the set of strategies. For a given strategy σ, we define its value $\mathsf{Val}(\sigma) : S \to \mathbb{R}$ such that $\mathsf{Val}(\sigma)(s)$ is the value of the cost function if the initial state is s and the decisions are made according to the strategy σ.

We say that a strategy σ_* is optimal if for every strategy $\sigma \in \Sigma$ we have that $\mathsf{Val}(\sigma_*)(s) \leq \mathsf{Val}(\sigma)(s)$, for every state s. We define the *optimum value* as the value of the optimal strategy.

Given a multi-stage decision process, the optimisation problem is to find an optimal policy and the optimum value. The main tool in dynamic programming is to characterise the optimum value by a set of functional equation called *optimality equations*.

1.2.1.3. Optimality Equations

Bellman observed [**Bel57**] that optimal policies (strategies) for a number of multi-stage decision processes follow the following principle.

The Principle of Optimality: *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

This principle states that the global optimality has certain local behaviour. This fact is mathematically captured by designing a set of functional equations which connect the optimum value of a state with the optimum values of its successor states. We call these equations *optimality equations* [**Ber95, Bel57**]. Other popular names for these equations are *Bellman equations*, functional equations, and dynamic programming equations.

To solve an optimisation problem of multi-stage decision processes using dynamic programming [**Put94**] we need to: a) design a set of optimality equations, b) formally prove that given a solution of such equations one can easily compute an optimal policy and the optimum value, and c) design an algorithm to solve these equations.

Once we have the right optimality equations, the next step is to prove the *existence theorem* (i.e., that there exists a solution of these equations). A constructive proof of the existence theorem can be given by writing an algorithm to solve optimality equations. Sometimes it is also possible to prove the *uniqueness theorem* (i.e., that the solution of the optimality equations is unique).

1.2.1.4. Solving Optimality Equations

The optimality equations for a multi-stage decision process can be solved using iterative methods yielding a solution (or an approximation) of the multi-stage decision process. The iterative methods employed to solve optimality equations fall broadly into two categories: *value iteration* (or approximation in value space) and *policy iteration* (or approximation in policy space). To demonstrate the structure of value iteration and policy iteration methods we need to introduce some notations. We call a function of the type $F : S \to \mathbb{R}$ a value function and we write $\mathcal{S}_{\mathsf{val}}$ for the set of value functions. Let $F_1, F_2 \in \mathcal{S}_{\mathsf{val}}$ be two value functions. We say that $F_1 \preceq F_2$ if for all $s \in S$, we have $F_1(s) \leq F_2(s)$. We say that $F_1 \equiv F_2$ if for all $s \in S$, we have $F_1(s) = F_2(s)$. A value improvement function $\mathrm{Improve}_{\mathsf{val}} : \mathcal{S}_{\mathsf{val}} \to \mathcal{S}_{\mathsf{val}}$ is such that $\mathrm{Improve}(F) \preceq F$ for every $F \in \mathcal{S}_{\mathsf{val}}$.

Similarly we call a function of the type $F : (S \times A)^* \times S \to A$ a policy function if for all $h = \langle s_0, a_1, s_1, \ldots, s_n \rangle \in (S \times A)^* \times S$ we have $F(h) \in A(s_n)$. We further define the policy space $\mathcal{S}_{\mathsf{pol}}$ as the set of all policy functions. Let $\mathsf{Val} : \mathcal{S}_{\mathsf{pol}} \to \mathcal{S}_{\mathsf{val}}$ be a function which gives the value of a policy. Let $F_1, F_2 \in \mathcal{S}_{\mathsf{pol}}$ be two policy functions. We say that $F_1 \preceq F_2$ if $\mathsf{Val}(F_1) \preceq \mathsf{Val}(F_2)$. Similarly we say that $F_1 \equiv F_2$ if $\mathsf{Val}(F_1) \equiv \mathsf{Val}(F_2)$. Let $\mathrm{Improve}_{\mathsf{pol}} : \mathcal{S}_{\mathsf{pol}} \to \mathcal{S}_{\mathsf{pol}}$ be a function such that $\mathrm{Improve}(F) \preceq F$, for all $F \in \mathcal{S}_{\mathsf{pol}}$.

The pseudocode of a value iteration method and a policy iteration method is presented as Algorithm 1.2 and Algorithm 1.3, respectively.

Remark. We sometimes write *strategy improvement* instead of "policy iteration" to emphasise the fact that every iteration of the method gives an improved strategy.

EXAMPLE 1.2.1 (Babylonians' method)**.** The problem of computing square root of a positive integer is, as such, not an optimisation problem. However, the solution approach using *Babylonians' method* has certain flavour of the dynamic programming paradigm: the solution

Input: Optimality equations
Output: A solution of the optimality equations
begin
  Let $V_0 \in \mathcal{S}_{\mathsf{val}}$ be an arbitrary value function;
  Set i to 0;
  repeat
    Set $i := i + 1$;
    $V_i := \mathrm{Improve}_{\mathsf{val}}(V_{i-1})$;
  until $V_i \equiv V_{i-1}$;
  return V_i;
end

FIGURE 1.2. Pseudocode of a Value Iteration Method.

Input: Optimality equations
Output: A solution of the optimality equations
begin
  Let $P_0 \in \mathcal{S}_{\mathsf{pol}}$ be an arbitrary policy;
  Set i to 0;
  repeat
    Set $i := i + 1$;
    $P_i := \mathrm{Improve}_{\mathsf{pol}}(P_{i-1})$;
  until $P_i \equiv P_{i-1}$;
  return $\mathsf{Val}(P_i)$;
end

FIGURE 1.3. Pseudocode of a Policy Iteration Method.

is characterised using equations and then a value iteration algorithm is used to give an approximation of the solution.

Given a positive integer $S \in \mathbb{R}_{\oplus}$ we wish to compute its square root to some precision $\varepsilon > 0$. Let us consider the following equation $\mathrm{Opt}_{\surd}(S)$ involving the variable $\mathcal{V}$:

$$\mathcal{V} = \frac{1}{2}\left(\mathcal{V} + \frac{S}{\mathcal{V}}\right).$$

We say that a number $s \in [1, \infty)$ is a solution to $\mathrm{Opt}_{\surd}(S)$, and we write $s \models \mathrm{Opt}_{\surd}(S)$, if the previous equation holds for the valuation $\mathcal{V} \mapsto s$. The following, easy to verify, proposition shows that a solution of this equation gives the positive square root of S.

PROPOSITION 1.2.2. If $s \in [1, \infty)$ is such that $s \models \mathrm{Opt}_{\surd}(S)$ then $s = \sqrt{S}$.

Based on the equation $\mathrm{Opt}_{\surd}(S)$ we can write a straightforward value improvement function Improve : $\mathbb{R} \to \mathbb{R}$ defined as $v \mapsto (v + S/v)/2$. Since it is not always possible to compute the square root of every number, for any given $\varepsilon > 0$ the *value iteration* algorithm (see, Algorithm 1.4) returns an ε-approximation of $\sqrt{S}$. The following, easy to verify, proposition implies quadratic convergence of the Algorithm 1.4.

PROPOSITION 1.2.3 (Quadratic convergence **[Roh99]**)**.** Let $\langle v_0, v_1, v_2, \ldots \rangle$ be a sequence such that $v_0 = 1$ and for every $n \geq 1$ we have $v_n = \mathrm{Improve}(v_{n-1})$ then the following hold:

$$|v_{n+1} - \sqrt{S}| \leq \frac{(v_n - \sqrt{S})^2}{2}$$

Input: The number S and an acceptable precision $\varepsilon > 0$
Output: ε-approximation of $\sqrt{S}$.

```
1 begin
2     Let v_0 = 1;
3     Set i to 0;
4     repeat
5         Set i := i + 1;
6         v_i := Improve(v_{i-1});
7     until |v_i − v_{i-1}| ≤ ε ;
8     return v_i;
9 end
```

FIGURE 1.4. Value Iteration Algorithm to Solve $\mathrm{Opt}_{\surd}(S)$.

So far, we gave a short introduction to the theory of dynamic programming to solve multi-stage decision processes in the presence of one player. Dynamic programming is also useful in analysing *games* (multi-stage decision processes in presence of more than one player with conflicting interests). In the next subsection we give an introduction to game theory, and comment on their connections with dynamic programming.

1.2.2. Game Theory

Game theory is a mathematical theory that studies the interaction among several rational agents (players) in competitive or co-operative environments. We focus our attention to a subclass of games known as two-player zero-sum games, that are competitive games between two players—we call them Min and Max—such that the sum of their winnings is zero, i.e., a player wins whatever his opponent loses. Often we specify the payoff of the game as the winnings of player Max. Hence the goal of player Max is to maximise the payoff, while the goal of player Min is to minimise the payoff.

To formally introduce the two-player zero-sum games, let us consider players Max and Min along with their pure strategy (action) sets Σ_{Max} and Σ_{Min}, and a real-valued payoff

function $\mathbb{P} : \Sigma_{\text{Min}} \times \Sigma_{\text{Max}} \to \mathbb{R}$. If player Max chooses the action $\chi \in \Sigma_{\text{Max}}$ and player Min chooses the action $\mu \in \Sigma_{\text{Min}}$ then we say that the player Max wins the value $\mathbb{P}(\mu, \chi)$ and player Min looses the value $\mathbb{P}(\mu, \chi)$. The goal of player Max is to choose his actions to maximise his winnings and the goal of the player Min is to choose her actions to minimise her losses.

Observe that player Max can choose his actions to win at least an amount arbitrarily close to $\sup_{\chi \in \Sigma_{\text{Max}}} \inf_{\mu \in \Sigma_{\text{Min}}} \mathbb{P}(\mu, \chi)$. This is called the lower value $\underline{\mathsf{Val}}$ of the game:

$$\underline{\mathsf{Val}} = \sup_{\chi \in \Sigma_{\text{Max}}} \inf_{\mu \in \Sigma_{\text{Min}}} \mathbb{P}(\mu, \chi). \tag{1.2.1}$$

Similarly, player Min can choose her actions to lose at most an amount arbitrarily close to $\inf_{\mu \in \Sigma_{\text{Min}}} \sup_{\chi \in \Sigma_{\text{Max}}} \mathbb{P}(\mu, \chi)$. This is called the upper value $\overline{\mathsf{Val}}$ of the game:

$$\overline{\mathsf{Val}} = \inf_{\mu \in \Sigma_{\text{Min}}} \sup_{\chi \in \Sigma_{\text{Max}}} \mathbb{P}(\mu, \chi). \tag{1.2.2}$$

The following proposition states that the lower value is not more than the upper value.

PROPOSITION 1.2.4. For every two-player zero-sum game, it is always the case that $\underline{\mathsf{Val}} \leq \overline{\mathsf{Val}}$.

PROOF. Observe that for every $\mu' \in \Sigma_{\text{Min}}$ and $\chi' \in \Sigma_{\text{Max}}$ we have

$$\mathbb{P}(\mu', \chi') \leq \sup_{\chi \in \Sigma_{\text{Max}}} \mathbb{P}(\mu', \chi).$$

Taking infimum over all strategies of Min we get that for every $\chi' \in \Sigma_{\text{Max}}$ we have

$$\inf_{\mu \in \Sigma_{\text{Min}}} \mathbb{P}(\mu, \chi') \leq \inf_{\mu \in \Sigma_{\text{Min}}} \sup_{\chi \in \Sigma_{\text{Max}}} \mathbb{P}(\mu, \chi).$$

It easily follows that

$$\sup_{\chi \in \Sigma_{\text{Max}}} \inf_{\mu \in \Sigma_{\text{Min}}} \mathbb{P}(\mu, \chi) \leq \inf_{\mu \in \Sigma_{\text{Min}}} \sup_{\chi \in \Sigma_{\text{Max}}} \mathbb{P}(\mu, \chi),$$

which by definition of $\underline{\mathsf{Val}}$ and $\overline{\mathsf{Val}}$ implies that $\underline{\mathsf{Val}} \leq \overline{\mathsf{Val}}$. ∎

We say that a game is *determined* if $\underline{\mathsf{Val}} = \overline{\mathsf{Val}}$. If a game is determined then we say that the value of the game Val exists and $\mathsf{Val} = \overline{\mathsf{Val}} = \underline{\mathsf{Val}}$.

If the game is determined and the suprema and infima are attained in (1.2.1) and (1.2.2) then optimal strategies χ_* for player Max and μ_* player Min are such that the following holds:

$$\inf_{\mu \in \Sigma_{\text{Min}}} \mathbb{P}(\mu, \chi_*) = \mathsf{Val} = \sup_{\chi \in \Sigma_{\text{Max}}} \mathbb{P}(\mu_*, \chi).$$

However, if $\overline{\mathsf{Val}} = \underline{\mathsf{Val}}$ but the suprema and infima are not attained in (1.2.1) and (1.2.2) then for every $\varepsilon > 0$ both players have the so-called ε-optimal strategies. For a given $\varepsilon > 0$, a strategy χ_ε of player Max is called ε-optimal if $\inf_{\mu \in \Sigma_{\text{Min}}} \mathbb{P}(\mu, \chi_\varepsilon) \geq \mathsf{Val} - \varepsilon$. Similarly for a given $\varepsilon > 0$, a strategy μ_ε of player Min is called ε-optimal if $\sup_{\chi \in \Sigma_{\text{Max}}} \mathbb{P}(\mu_\varepsilon, \chi) \leq \mathsf{Val} + \varepsilon$.

Observe that not all games are determined. The following example presents a game that is not determined.

EXAMPLE 1.2.5. Let us consider a continuous payoff function $f : [0,1] \times [0,1] \to [0,\infty)$ given by $f(a,b) = (a-b)^2$. A play of the game is as follows: at the same time players Max and Min pick the first argument a and the second argument b, resp., and as a result player Max wins the amount $f(a,b)$ from player Min.

Let us analyse what the minimum win player Max can secure. Notice that whatever player Max plays, player Min can play the same and pay nothing to Max. Hence the lower value $\underline{\mathsf{Val}}$ of this game is $\sup_{a\in[0,1]} \inf_{b\in[0,1]} f(a,b) = 0$.

Now let us analyse the maximum loss that player Min can guarantee. Notice that if $0 \le b \le 1/2$ then $\sup_{a\in[0,1]} f(a,b) = (1-b)^2$, and if $1/2 \le b \le 1$ then $\sup_{a\in[0,1]} f(a,b) = b^2$. Hence the upper value of the game is $\inf_{b\in[0,1]} \sup_{a\in[0,1]} f(a,b) = 1/4$.

It shall be clear that this game is not determined as $\underline{\mathsf{Val}} \neq \overline{\mathsf{Val}}$. In other words this game cannot be solved up to the satisfaction of both players: for any suggestion $(a,b) \in [0,1] \times [0,1]$ made to them, both players find it advantageous to change their strategy.

In Appendix C, we review the classic results of von Neumann (on the determinacy of *matrix games* [**vN28**]) and Shapley (on the determinacy of *stochastic games* [**Sha53**]) to show that classic proofs of determinacy of these games use concepts central to dynamic programming. We hope that such connections help the reader to digest the rest of the thesis, where we apply dynamic programming techniques to prove determinacy of several two-player zero-sum games on finite graphs and timed automata.

1.3. Literature Review

Competitive optimisation on timed automata has been studied since the introduction of timed automata in the late 1980's. In this section, we review some of the important results in this area. The review of results presented in this section is by no means exhaustive, and is biased towards the relevancy of the results to the ones solved in this thesis. Some good references related to timed automata are [**AM04**], [**Bou06**],and [**UPP**].

1.3.1. Timed Automata

Timed automata were originally introduced by Alur and Dill [**AD90, Alu91**] as a formalism to model asynchronous real-time systems. A timed automaton is a finite automaton—a finite set of *locations* and *transitions*—coupled with a finite set of real-valued variables called *clocks*. Clock variables evolve continuously at unit rate, and may appear in guards of transitions in timed automata. The syntax of timed automata also allows clock values to be reset after executing a transition. A state of timed automaton is a pair of location and clock valuation. A state is called a *corner state* if its clock valuation assigns integer values to all clock variables. The formal definition of timed automata allows one to specify a special set of states called the set of *final states*. Throughout this section, we write S for the set of states, A for the set of actions, and F for the set of final states.

A timed move is a pair (t,a) which represents the action: wait for the time duration t before executing the transition with label a. A *run* of a timed automaton is an alternating sequence of states and timed moves $\langle s_0, (t_1,a_1), s_1, \ldots, s_n\rangle$ such that for every positive integer $i \le n$, the timed move (t_i,a_i) is enabled in the state s_{i-1} and the state s_i is reached

after executing the timed move (t_i, a_i). An *infinite run* $r = \langle s_0, (t_1, a_1), \ldots \rangle$ is defined analogously. We say that a run $\langle s_0, (t_1, a_1), s_1, \ldots, s_n \rangle$ is *accepting* if the state $s_n \in F$. The language recognised by a timed automaton is the set of accepting runs. In their seminal paper, Alur and Dill [**AD90**] showed that languages recognised by timed automata are closed under union and intersection, but not under complementation.

1.3.2. Noncompetitive Optimisation Problems

1.3.2.1. Reachability Problem

The *reachability problem* of a timed automaton is: given an initial state $s \in S$, decide whether a final state is reachable from the initial state, i.e., whether there exists an accepting run starting from s.

Alur and Dill [**AD90**] showed that the reachability problem for timed automata is decidable and that it is PSPACE-complete. They established PSPACE-membership of the reachability problem using a finitary abstraction—the so-called region graph—of timed automata, whose size is exponential in the size of the timed automaton. For the PSPACE-hardness result they showed that for a linear-space Turing machine M and an input word w of length n, there exists a timed automaton $\mathcal{T}$ with $2n + 1$ clocks, such that the language of $\mathcal{T}$ is nonempty iff the machine M accepts w. Courcoubetis and Yannakakis [**CY92**] later tightened the PSPACE-hardness result by showing a reduction from the acceptance problem for linear-space Turing machine to the reachability problem of timed automata with only three clocks.

THEOREM 1.3.1 ([**AD94, CY92**])**.** The reachability problem is PSPACE-complete for timed automata with at least three clocks.

Laroussinie, Markey and Schnoebelen [**LMS04**] proposed a succinct "region" graph abstraction for one-clock timed automata, and using that they proved that the reachability problem for one-clock timed automata is NLOGSPACE-complete.

THEOREM 1.3.2 ([**LMS04**])**.** The reachability problem is NLOGSPACE-complete for timed automata with one clock.

The exact complexity of the reachability problem for timed automata with two clocks remains an open question.

1.3.2.2. Optimal Reachability-Time Problem

A natural optimisation problem on the timed automata is to optimise (minimise or maximise) reachability time, i.e., total time to reach a final state. Given a timed automaton $\mathcal{T}$, an initial state s, and a number $D \in \mathbb{R}_{\oplus}$, the minimum reachability time problem is to decide whether there exists an accepting run starting from s with total time at most D. Maximum reachability-time problem is defined analogously.

Minimum and maximum reachability-time problems were shown to be decidable by Courcoubetis and Yannakakis [**CY92**]. It was shown to be PSPACE-complete by Alur, Courcoubetis, and Henzinger [**ACH97**] and Kesten et al. [**KPSS99**]. An efficient algorithm

to solve the minimum reachability-time problem on timed automata appeared in [**NTY00**], where the initial state was restricted to corner states.

1.3.2.3. Optimal Reachability-Price Problem

A generalisation of timed automata to priced timed automata [**Bou06**]—also known as weighted timed automata—allows a rich variety of applications, e.g., to optimal scheduling [**BFH$^+$01, AM01**]. Priced timed automata are timed automata with a price function $\pi : S \times \mathbb{R}_{\oplus} \times A \to \mathbb{R}$ such that $\pi(s,(t,a))$ gives the price of taking a timed move (t,a) from a state s. The price function can be extended to give the reachability-price $\mathsf{RP}(r)$ of an infinite run $r = \langle s_0,(t_1,a_1),s_1,\ldots\rangle$, the sum of the prices of the timed moves before reaching a final state, in the following manner:

$$\mathsf{RP}(r) = \begin{cases} \sum_{i=1}^{\mathsf{Stop}(r)} \pi(s_{i-1},(t_i,a_i)), & \text{if } \mathsf{Stop}(r) < +\infty, \\ \infty, & \text{otherwise,} \end{cases}$$

where $\mathsf{Stop}(r)$ denotes the index of the first final state in the run r.

Linearly-priced timed automata, a sub-class of priced timed automata, augment the timed automata with price information, such that the price of waiting in a state is directly proportional to the waiting time. A natural generalisation of the minimum reachability-time problem for the timed automata is the minimum reachability-price problem for the priced timed automata.

Given a priced timed automaton $\mathcal{T}$, an initial state s, and a number D, the minimum reachability-price problem is to decide whether there exists an infinite run r starting from s such that $\mathsf{RP}(r) \leq D$.

For a linearly-priced timed automaton, Larsen et al. [**LBB$^+$01, BFH$^+$01, LBB$^+$04**] proposed symbolic algorithms to solve the reachability-price problem, with some restrictions on the initial state (corner state with all clocks set to zero). These symbolic algorithms are also implemented as a part of UPPAAL [**BFHL01**] (a toolkit for the verification of real-time systems).

In [**ALTP04**] Alur et al. proposed an EXPTIME algorithm to solve the reachability-price problem for linearly-priced timed automata using a non-trivial extension of the region graph.

Bouyer et al. [**BBBR07**] showed that the reachability-price problem for the linearly-priced timed automata is PSPACE-complete, given the initial state is a corner state. The PSPACE-hardness result for this problem comes from a straightforward reduction from the reachability problem for timed automata. To show PSPACE-membership, Bouyer et al. argued that the minimum reachability-price for a linearly-priced timed automaton can be computed using a finitary abstraction of the timed automata, if the initial state is a corner state. The abstraction used by them is called the *corner-point abstraction*. Their main observation was that if the initial state is a corner state, then optimal (nearly optimal) reachability-price runs go through (or very close to) corner states.

Remark. Notice that PSPACE-completeness result of Bouyer et al. [**BBBR07**] for the optimum reachability-price problem is restricted to linearly-priced timed automata and initial states that are corner states.

Observe that PSPACE-completeness results, for reachability-time and reachability-price problems, hold for timed automata with at least three clocks. For timed automata with one clock, reachability-time and reachability-price problems are known to be NLOGSPACE-complete [**LMS04**]; while the complexity of these problems for two-clock timed automata remains an open problem.

1.3.2.4. Optimal Average-Price Problem

Bouyer, Brinksma, and Larsen [**BBL04**] considered the problem of finding optimal infinite runs in so-called doubly-priced timed automata. Doubly-priced timed automata are linearly-priced timed automata with two functions: *price* (called cost in [**BBL04**]) and *reward*. Let $\pi : S \times \mathbb{R}_{\oplus} \times A \to \mathbb{R}$ and $\varrho : S \times \mathbb{R}_{\oplus} \times A \to \mathbb{R}$ be the price and reward functions. Optimality criteria considered in [**BBL04**] is the *price-per-reward average* (called ratio price in [**BBL04**]), i.e., limit ratio of accumulated prices and rewards. The price-per-reward average of a run $r = \langle s_0, (t_1, a_1), s_1, \ldots \rangle$ is defined in the following way:

$$\mathsf{PRAvg}(r) = \liminf_{n \to \infty} \frac{\sum_{i=1}^{n} \pi(s_{i-1}, (t_i, a_i))}{\sum_{i=1}^{n} \varrho(s_{i-1}, (t_i, a_i))}.$$

The minimum price-per-reward average problem can be described as follows: given a priced timed automaton $\mathcal{T}$, an initial state s, and a number $D \in \mathbb{R}_{\oplus}$, decide whether there exists an infinite run r starting from s such that $\mathsf{PRAvg}(r) \leq D$.

Using the corner-point abstraction, Bouyer, Brinksma, and Larsen [**BBL04**] showed that the minimum price-per-reward average problem for linearly priced timed automata is PSPACE-complete, if the initial state is a corner state. Again, their main observation was that if the initial state is a corner state then optimal price-per-reward average runs go through corner states.

Remark. Notice that PSPACE-completeness result of Bouyer et al. [**BBL04**] for the optimum price-per-reward average problem is restricted to linearly-priced (doubly priced) timed automata and initial states that are corner states.

1.3.3. Two-Player Games

1.3.3.1. Two-Player Games for Controller Synthesis

Computer scientists [**Tho95, GTW02, dA03**] and control theoreticians [**RW89, FV97**] have been interested in two-player zero-sum games as a model for synthesising optimal controllers. In these games the two players correspond to the controller and the (adversarial) environment [2] and, given some optimality criterion, controller synthesis corresponds to computing optimal strategies for the controller. Two player games on finite automata as a mechanism for supervisory controller synthesis of discrete event systems were introduced by Ramadge and Wonham [**RW89**].

Games considered by Ramadge and Wonham are played on finite automata whose set of actions (alphabets) is partitioned into the sets of controllable actions and uncontrollable

[2] "CS people call *environment* everything that lies outside the computer" — a control theoretician [**AMP95**].

actions. The objective considered by them is safety (or equivalently reachability), i.e., "to avoid bad states". A *strategy* of the controller is a function from the set of runs to the set of controllable actions, which—based on the history of the run—suggests the controller to select a controllable event in order to optimise the objective, i.e., to avoid reaching a bad state. The solution proposed by Ramadge and Wonham is based on computing greatest fixed point of certain controllable predecessor functions.

1.3.3.2. Reachability Games

Hoffman and Wong-Toi [**WTH92, HWT92**] were the first ones to define and solve optimal controller synthesis problem for timed automata. They proved the decidability of reachability games on timed automata by extending the method of Ramadge and Wonham to the finitary region graph abstraction of timed automata.

Asarin, Maler and Pnueli [**AMP95**] give a symbolic algorithm for optimal controller synthesis of timed automata. Their work is closely related to the work by Hoffman and Wong-Toi. However, instead of explicitly constructing the region graph, they use symbolic methods, representing the set of states using arbitrary linear inequalities, to solve the reachability game on timed automata.

Henzinger and Kopke [**HK99**] show that the complexity of solving reachability games on timed automata is EXPTIME-complete. Jurdziński and I [**JT07**] improved their results by showing that reachability games are EXPTIME-complete for timed automata with at least two clocks.

For a detailed introduction to the topic of qualitative games on timed automata, we recommend papers by Asarin et al. [**AMPS98, AMP95**].

1.3.3.3. Reachability-Time Games

Asarin and Maler [**AM99**] initiated the study of quantitative competitive optimisation on timed automata. They considered reachability-time games where controller (player Min in our terminology) is interested in reaching a final state as soon as possible. The symbolic algorithm presented in [**AM99**] is a value iteration algorithm, which solves a set of optimality equations characterising minimum time to reach a final state. They give a *uniform solution*, i.e., their algorithm computes a function which, given an initial state, returns the upper-value of the reachability-time game starting from that state. The result of Asarin and Maler does not give any upper complexity bounds and is restricted to a subclass: the structurally non-Zeno timed automata.

Brihaye et al. [**BHPR07**] and Jurdziński and I [**JT07**] studied reachability-time games for slightly different models of timed games, and showed that the decision version of the reachability-time game is EXPTIME-complete for timed automata with at least two clocks.

Brihaye et al. [**BHPR07**] studied the problem on the "element-of-surprise" timed game model introduced in [**dAFH$^+$03**]. In these models players can take each other by surprise as both players suggest a timed move concurrently, and the timed move with shorter time delay is executed. They also introduced a nice concept of a *receptive* strategy—a strategy which does not recommend players to execute infinitely many actions in a finite amount

of time—and to win a game a player must use a receptive strategy. Analysis using only receptive strategies is arguably more meaningful, because strategies of the controller that are not receptive are not physically implementable. Unfortunately, receptiveness is not a sufficient condition for a strategy to be implementable, e.g., consider the strategy in which controller needs to act in the following time sequence: $\langle 0, \frac{1}{2}, 1, 1\frac{1}{4}, 2, 2\frac{1}{8}, 3, 3\frac{1}{16}, \ldots \rangle$ **[CHR02]**. Moreover, reachability-time games studied by Brihaye et al. are not known to be determined and they compute the upper value of the game.

Jurdziński and I **[JT07]** studied a turn-based model of timed games where the locations of the timed automata are partitioned between two players. For these models of timed game, we proved that reachability-time games are positionally determined. Our algorithm gives a uniform solution, and by analysing our algorithm we show that reachability-time games are EXPTIME-complete for timed automata with at least two clocks. However, unlike Brihaye et al. **[BHPR07]**, we do not require players to play only receptive strategies.

1.3.3.4. Rechability-Price Games

A natural extension of reachability-time games for timed automata is reachability-price games for priced timed automata.

La Torre, Mukhopadhyay, and Murano **[LTMM02]** studied reachability-price games on a restricted class, the so-called acyclic timed automata, of linearly-priced timed automata. They gave a doubly-exponential time algorithm to solve the reachability-price games on timed automata, given that the timed automaton is acyclic, i.e., control graph does not have any cycles.

Alur, Bernadsky, and Madhusudan **[ABM04]** and Bouyer et al. **[BCFL04]** studied, independently, the reachability-price games on arbitrary linearly-priced priced timed automata, and gave semi-algorithms to compute its upper value of the reachability-price games. Their algorithms are guaranteed to terminate for linearly-priced timed automata with non-Zeno prices, i.e., every cycle of the automaton has a bounded positive price. The elegant paper of Alur et al. **[ABM04]** presents a value-iteration algorithm to uniformly compute the upper value of a reachability-price game. The authors also present a detailed understanding of the value iteration method in which, at every iteration, the regions are split into sub-regions such that the approximation of the upper value in a sub-region is a linear function. Alur et al. also showed that for certain examples regions get split into exponentially many such sub-regions. The solution of Bouyer et al. **[BCFL04]**, on the other hand, is based on a reduction to games on linear hybrid automata. They also presented **[BCFL05]** techniques to implement the solution using the HyTech tool.

Brihaye, Bruyere, and Raskin **[BBR05]** proved a somewhat unexpected result that checking the existence of optimal strategies in a reachability-price game is undecidable for a priced timed automaton with at least five clocks. They showed a reduction from the halting problem for two-counter machines to the problem of finding reachability-price optimal strategies in priced timed automata with five clocks and with *stopwatch prices* (i.e., with either 0 or 1 price rates of the locations).

Bouyer, Brihaye, and Markey **[BBM06]** later improved this undecidability result by reducing the halting problem for two-counter machine to finding optimal strategies in a

reachability-price game on a priced timed automaton with three clocks and with stopwatch prices. The problem of finding reachability-price optimal strategies for linearly-priced timed automata with one clock is known to be decidable, and Bouyer et al. [**BLMR06**] give a 3EXPTIME algorithm to solve this problem. The decidability of finding optimal strategies in a reachability-price game on timed automata with two clocks is unknown.

Finally, notice that the decidability of finding the value of a state in a reachability-price game remains an open problem.

1.4. Contributions of the Thesis

1.4.1. Boundary Strategies and Boundary Region Graphs

We view a competitive optimisation problem on timed automata as a multi-stage decision process where one or two players are confronted with the problem of choosing a sequence of timed moves—each consisting of a time delay and an action—in order to optimise their objectives given as a *cost function*. A solution of such problems consists of the "optimum" value of the cost function and an "optimal" strategy for each player. We highlight a useful class of strategies, called *boundary strategies*, that suggest to a player a symbolic timed move of the form (b, c, a)— "wait until the value of the clock c is in very close proximity of the integer b, and then execute a transition labelled with the action a". A distinctive feature of the competitive optimisation problems discussed in this thesis is the existence of optimal *boundary strategies*.

The existence of optimal boundary strategies allows us to work with a novel abstraction of the timed automata, called a *boundary region graph*, where players can use only boundary strategies. An important property of a boundary region graph is that for every state the set of reachable states is finite (exponential in the size of the timed automaton). Hence, the existence of optimal boundary strategies allows us to reduce a competitive optimisation problem on a timed automaton to the corresponding competitive optimisation problem on a finite graph.

Boundary region graphs generalise *corner-point abstractions* of timed automata introduced by Bouyer et al. Corner-point abstractions are useful in analysing various optimisation problems on timed automata, given that the initial state is a corner-state (a state with integral valuation to every clock). On the other hand, our boundary region graph permits the analysis of a number of competitive optimisation problems on timed automata with arbitrary initial states, including non-corner states.

1.4.2. Non-competitive Optimisation

To cover a large class of optimisation problems we introduce *concavely-priced timed automata*, that are generalisations of timed automata with price information given by certain concave functions. We identify a useful property, called *concave-regularity*, of the cost function: if a cost function is concave-regular then there exists an optimal boundary strategy for the corresponding optimisation problem. We show that a number of cost functions of practical interest, including reachability price, discounted price, average time-per-transition, average price-per-transition, and average price-per-time-unit, are concave-regular. We further show

that the complexity of solving optimisation problems for concave-regular cost functions is PSPACE-complete on timed automata with three or more clocks.

THEOREM. The minimisation problems for reachability price, discounted price, average price, price-per-time average, and price-per-reward average cost functions, for concavely-priced and concave price-reward timed automata, as appropriate, are PSPACE-complete.

We generalise some previously known complexity results (e.g., the results of Bouyer et al. on optimum reachability price problem [**BBBR07**] and on optimum price-per-reward average problem[**BBL04**]) in two directions: a) our results are valid for arbitrary initial states, and b) we consider more general concavely-priced timed automata.

1.4.3. Reachability-Time Games

A *reachability-time game* is played between two players Min and Max on the infinite graph of configurations of a timed automaton. The goals of player Min and player Max are to minimise and maximise, respectively, the time to reach a designated set of final configurations. We show that the exact values of reachability-time games on arbitrary timed automata are uniformly computable (here uniformity means that the output of our algorithm allows us, for every starting state, to compute in constant time the value of the game starting from this state). In particular, unlike the paper of Asarin and Maler [**AM99**], we do not require timed automata to be strongly non-Zeno. We also establish the exact complexity of reachability-time games: they are EXPTIME-complete and two clocks are sufficient for EXPTIME-hardness. For the latter result, we reduce from a recently discovered EXPTIME-complete problem of countdown games [**JLS07**].

THEOREM. The problem of solving reachability-time games is EXPTIME-complete on timed automata with at least two clocks.

We believe that an important contribution of our work on reachability-time games is the novel proof techniques used. We characterise the values of the game by *optimality equations* and then we use *strategy improvement* to solve them. This allows us to obtain an elementary and constructive proof of the fundamental determinacy result for reachability-time games, which at the same time yields an efficient algorithm matching the EXPTIME lower bound for the problem. Those techniques were known for finite state systems [**Put94, VJ00**], but we are not aware of any earlier algorithmic results based on optimality equations and strategy improvement for real-time systems such as timed automata.

1.4.4. Average-Time Games

An average time game is played on the infinite graph of configurations of a finite timed automaton. The two players, Min and Max, construct an infinite run of the automaton by taking turns to perform a timed transition. Player Min wants to minimise the average time per transition and player Max wants to maximise it. We give a solution of the average-time games using a reduction to the average-price game on the finite graph of reachable configurations of the boundary region graph. A direct consequence is an elementary

proof of determinacy for average-time games. Our solution allows computing the value of average-time games for an arbitrary starting state (i.e., including non-corner states). We also establish the exact complexity of average-time games: they are EXPTIME-complete and two clocks are sufficient for EXPTIME-hardness. For the hardness result we reduce from EXPTIME-complete problem of countdown games [**JLS07**].

THEOREM. The problem of solving average-time games is EXPTIME-complete on timed automata with at least two clocks.

1.5. Organisation of the Thesis

The rest of the thesis is organised in four parts.

1.5.1. Part 1: Background

Part 1 (Background) consists of two chapters: Chapter 2 (Competitive Optimisation on Finite Graphs) and Chapter 3 (Timed Automata).

In Chapter 2 we present algorithms to solve optimisation problems and two-player games on finite graphs. We demonstrate that dynamic programming techniques are not only instrumental in designing algorithms, but also quite helpful in proving theorems (proof of existence of positional strategy and determinacy proofs, for example). Although the results discussed in Chapter 2 are not new, the chapter is quite useful as it shows the techniques and algorithms used in the rest of the thesis in a simpler setting.

In Chapter 3 we introduce concepts, notations, and mathematical shorthands related timed automata. After introducing the syntax of timed automata, we present two extensions (priced timed automata and timed game automata) of timed automata and some abstractions of timed automata (including region graph and boundary region graph) which are useful in solving competitive optimisation problems on timed automata.

1.5.2. Part 2: Competitive Optimisation on Timed Automata

Part 2 (Competitive Optimisation on Timed Automata) of the thesis presents our main contributions, and is organised in three chapters: Chapter 4 (Noncompetitive Optimisation), Chapter 5 (Reachability-Time Games), and Chapter 6 (Average-Time Games).

Chapter 4 considers noncompetitive optimisation on timed automata, which in game-theoretical terminology is equivalent of one-player games. In this chapter we introduce *concavely-priced automata*, an extension of linearly-priced timed automata. We prove that several noncompetitive optimisation problems satisfying certain concave-regularity condition are PSPACE-complete.

Chapter 5 presents reachability-time games on the infinite graph of configurations of timed automata. Using optimality equations we show that a solution of reachability-time games on timed automata can be obtained by solving a reachability-price game on a finite graph (the *boundary region graph*).

Chapter 6 presents average-time games on timed automata. We introduce a new abstraction of timed automata, called the *closed region graph*, and we use it to show that the

value of the average-time game on a timed automaton is equal to the value of the average-price game on a finite graph (the *boundary region graph*).

1.5.3. Part 3: Conclusion

Part 3 (Conclusion) consists of the final chapter Chapter 7 (Summary and Future Work) where we summarise the thesis and discuss some possible extensions of the work presented in this thesis.

1.5.4. Part 4: Appendix

Part 4 (Appendix) is partitioned into four units: Appendix A (Notations and Acronyms), Appendix B (Results from Real Analysis), Appendix C (Some Determinacy Results), and Appendix D (Implementation).

In Appendix A we present some notations and acronyms used across the chapters in this thesis. Therefore, we suggest the reader to skim through Tables 1, 2, and 3 before reading the rest of the thesis.

In Appendix B we give a primer on *concave functions* and *Lipschitz continuity*. We also present some results from real analysis which are referenced in this thesis.

In Appendix C we present matrix games and stochastic games, and discuss determinacy result by von Neumann and Shapley.

Finally, in Appendix D we present the details of a lexical grammar and parser grammar for the specification of the timed automata.

Part 1

Background

2

Competitive Optimisation on Finite Graphs

> If you keep proving stuff that others have done, getting confidence, increasing the complexities of your solutions - for the fun of it - then one day you'll turn around and discover that nobody actually did that one! And that's the way to become a computer scientist.
>
> Richard Feynmann

In this chapter we provide an introduction to competitive optimisation on finite graphs [1]. We discuss optimisation problems and two-player games on finite graphs, and present dynamic programming based algorithms to solve them. Competitive optimisation for the following cost functions are discussed: reachability price, discounted price, average price, and price-per-reward average. This chapter lays the foundations that are essential in the development of algorithms to solve competitive optimisation problems on the infinite graph of configurations of timed automata.

2.1. Formal Definition

DEFINITION 2.1.1 (Finite Graph)**.** A (directed) *finite graph* is a pair $G = (V, E)$, where:

- V is a finite set whose elements are called *vertices*, and
- $E \subseteq V \times V$ is a set of ordered pairs of vertices whose elements are called (directed) *edges*.

[1] Disclaimer: the concepts, theorems, and their proofs presented in this chapter are standard. Similar treatment can be found in an advanced setting of Markov Decision Processes in the excellent books of Puterman [**Put94**], and Filar and Vrieze [**FV97**].

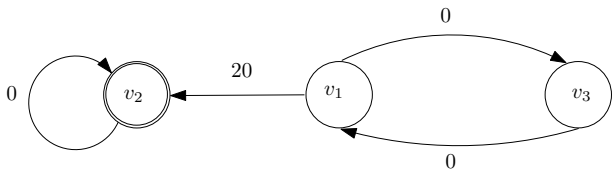


FIGURE 2.1. A priced finite graph with final vertices.

ASSUMPTION 2.1.2. For technical convenience we assume that every vertex has at least one outgoing edge, i.e., for every vertex $v \in V$ there exists a vertex $v' \in V$ such that $(v, v') \in E$.

A *run* (path) in a graph G is a sequence of vertices $\langle v_0, v_1, v_2, \ldots \rangle \in V^\omega$, such that for every positive integer i we have $(v_{i-1}, v_i) \in E$. A finite run in a graph G is a finite sequence of vertices $\langle v_0, v_1, v_2, \ldots, v_n \rangle \in V^*$, such that for every positive integer $i \leq n$ we have $(v_{i-1}, v_i) \in E$. We write Runs and Runs_{fin} for the sets of infinite and finite runs, respectively; we write $\text{Runs}(v)$ and $\text{Runs}_{\text{fin}}(v)$ for the sets of infinite and finite runs starting from the vertex $v \in V$, respectively.

DEFINITION 2.1.3 (Priced Graph and Price-Reward Graph)**.** A *priced graph* (V, E, π) consists of a finite graph (V, E) and a price function $\pi : E \to \mathbb{R}$; a *price-reward graph* (V, E, π, ϱ) consists of a graph (V, E) and price and reward functions $\pi, \varrho : E \to \mathbb{R}$, respectively.

ASSUMPTION 2.1.4 (Reward Divergence)**.** We assume that a price-reward graph (V, E, π, ϱ) is reward-diverging, that is for all runs $r \in \text{Runs}$ we have $\lim_{n\to\infty} |\varrho_n(r)| = \infty$.

For every run $r = \langle v_0, v_1, v_2, \ldots \rangle \in \text{Runs}$, price and reward functions $\pi, \varrho : E \to \mathbb{R}$, and for every positive integer n we define the following shorthands:

$$\pi_n(r) = \sum_{i=1}^{n} \pi(v_{i-1}, v_i) \text{ and } \varrho_n(r) = \sum_{i=1}^{n} \varrho(v_{i-1}, v_i).$$

A finite run $r = \langle v_0, v_1, \ldots, v_n \rangle \in \text{Runs}_{\text{fin}}$ of a graph G is called a *cycle* if we have that $v_n = v_0$. We say that a cycle $r = \langle v_0, v_1, \ldots, v_n \rangle \in \text{Runs}_{\text{fin}}$ is a zero cycle if we have that $\pi_n(r) = 0$.

A finite graph with final vertices is a tuple (V, E, F), where (V, E) is a finite graph and $F \subseteq V$ is the set of *final vertices*. Priced graphs and price-reward graphs with final vertices are defined in straightforward manner. For a run $r = \langle v_0, v_1, v_2, \ldots \rangle$ in a graph with final vertices $G = (V, E, F)$, we define $\mathsf{Stop}(r) = \inf_{i \geq 0}\{i : v_i \in F\}$.

EXAMPLE 2.1.5. A finite priced graph with final vertices (V, E, F, π) is shown in Figure 2.1, whose set of vertices is $V = \{v_1, v_2, v_3\}$, set of edges is $\{(v_1, v_2), (v_2, v_2), (v_1, v_3), (v_3, v_1)\}$, set of final vertices is $\{v_2\}$, and the price function is as follows: $\pi(v_1, v_2) = 20$ and $\pi(v_2, v_2) = \pi(v_2, v_3) = \pi(v_3, v_2) = 0$.

Let us fix a set of vertices V, a set of edges E, a set of final vertices $F \subseteq V$, a price function $\pi : E \to \mathbb{R}$ and a reward function $\rho : E \to \mathbb{R}$ for the rest of this chapter. In the rest of this chapter, when it is clear from the context, we use the term "graph" for every kind of graph—finite graph (V, E), priced graph (V, E, π), price-reward graph (V, E, π, ρ),

finite graph with final vertices (V, E, F), priced graph with final vertices (V, E, F, π) and price-reward graph with final vertices (V, E, F, π, ρ).

2.2. Noncompetitive Optimisation on Finite Graphs

Let $\mathsf{Cost} : \text{Runs} \to \mathbb{R}$ be a *cost function* that for every run $r \in \text{Runs}$ determines its cost $\mathsf{Cost}(r)$. We then define the *minimum cost function* $\mathsf{Cost}_* : V \to \mathbb{R}$, by $\mathsf{Cost}_*(v) = \inf_{r \in \text{Runs}(v)} \mathsf{Cost}(r)$. The *minimisation problem* for a cost function Cost is: "given a graph G, a vertex $v \in V$ and a number $D \in \mathbb{Q}$, determine whether $\mathsf{Cost}_*(v) \leq D$."

The following list of cost functions gives rise to a number of corresponding minimisation problems.

(1) *Reachability price.* The cost function reachability price $\mathsf{RP} : \text{Runs} \to \mathbb{R}$ is defined as follows: for every run $r \in \text{Runs}$ we have

$$\mathsf{RP}(r) = \begin{cases} \pi_N(r) & \text{if } N = \mathsf{Stop}(r) < \infty \\ \infty & \text{otherwise.} \end{cases}$$

(2) *Discounted price.* For a discount factor $\lambda \in (0, 1)$, the cost function discounted price $\mathsf{DP}(\lambda) : \text{Runs} \to \mathbb{R}$ is defined as follows: for every run $r \in \text{Runs}$ we have

$$\mathsf{DP}(\lambda)(r) = (1 - \lambda) \sum_{i=1}^{\infty} \lambda^{i-1} \pi(v_{i-1}, v_i).$$

(3) *Average price.* The cost function average price $\mathsf{AP} : \text{Runs} \to \mathbb{R}$ is defined as follows: for every run $r \in \text{Runs}$ we have

$$\mathsf{AP}(r) = \limsup_{n \to \infty} \frac{\pi_n(r)}{n}.$$

(4) *Price-per-reward average.* Finally, we define price-per-reward average cost function $\mathsf{PRAvg} : \text{Runs} \to \mathbb{R}$ as follows: for every run $r \in \text{Runs}$ we have

$$\mathsf{PRAvg}(r) = \limsup_{n \to \infty} \frac{\pi_n(r)}{\varrho_n(r)}.$$

A *positional strategy* is a function $\mu : V \to V$ such that for every vertex $v \in V$, we have $(v, \mu(v)) \in E$. We write Π for the set of positional strategies. A run from a vertex $v \in V$ according to strategy μ is the unique run $\text{Run}(v, \mu) = \langle v_0, v_1, v_2, \ldots \rangle$, such that $v_0 = v$, and for every positive integer i we have $\mu(v_{i-1}) = v_i$.

A positional strategy μ_* is *optimal* for a cost function $\mathsf{Cost} : \text{Runs} \to \mathbb{R}$, if for every vertex $v \in V$ we have $\mathsf{Cost}_*(v) = \mathsf{Cost}(\text{Run}(v, \mu_*))$. Observe that existence of an optimal positional strategy for a cost function means that, from every starting vertex, there is a run that minimises the cost, and that is a simple path leading to a simple cycle repeated infinite many times. The following is a well known result (see, for example, **[Put94]**) for finite graphs.

THEOREM 2.2.1 (Existence of optimal positional strategies)**.** For every finite graph, and for each of the reachability, discounted, average price, and price-per-reward average cost functions, there is an optimal positional strategy.

The rest of this section is devoted to the proof of this theorem. For each of the cost functions defined above, we design a set of optimality equations such that the existence of a solution of those equations implies the existence of a positional strategy. A solution of optimality equations also characterises the minimum cost function. We give a constructive proof of the existence of a solution by giving a strategy improvement algorithm, which returns a solution of the optimality equations.

2.2.1. Solving Reachability-Price Minimisation Problem

2.2.1.1. Optimality Equations

Let $G = (V, E, F, \pi)$ be a finite graph with final vertices. To keep the discussion simple, we assume that the graph G satisfy the following assumption:

ASSUMPTION 2.2.2. For every vertex $v \in V$ we have that minimum reachability-price is finite.

Before we consider the problem in full generality, we try to solve a simpler problem and assume the following:

ASSUMPTION 2.2.3. The graph G does not have any zero cycles.

For the finite price graph $G = (V, E, F, \pi)$ let us consider the following set of equations:

$$\mathcal{P}(v) \quad = \quad \begin{cases} 0, & \text{if } v \in F, \\ \min_{(v,w)\in E} \{\pi(v,w) + \mathcal{P}(w)\}, & \text{otherwise.} \end{cases}$$

We denote these equations by $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$. We say that a function $p : V \to \mathbb{R}$ is a solution of the optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$, and we write $p \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$, if all equations in $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ hold for the valuations $\mathcal{P}(v) \mapsto p(v)$.

The following proposition states the relation between optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ and a solution of minimum reachability-price problem.

PROPOSITION 2.2.4 (A solution of optimality equations gives optimal reachability-price)**.** Let G be a finite priced graph. Under Assumptions 2.2.2 and 2.2.3 we have that $p \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ implies $\mathsf{RP}_*(v) = p(v)$ for all $v \in V$.

PROOF. We prove the lemma in two parts. First we show that for every strategy $\mu \in \Sigma$, we have $p(v) \leq \mathsf{RP}(\mathrm{Run}(v, \mu))$ for every $v \in V$. In the second part, we show that there exists a positional strategy $\mu' \in \Pi$ such that for every vertex $v \in V$ we have $\mathsf{RP}(\mathrm{Run}(v, \mu')) = p(v)$.

– Let $\mathrm{Run}(v,\mu) = \langle v = v_0, v_1, \ldots\rangle$ be the run from the vertex v according to the strategy μ. Since $p \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$, for all $i \geq 0$, we have the following.

$$\begin{aligned} p(v_i) &= 0 \text{ if } v_i \in F \\ p(v_i) &\leq \pi(v_i, v_{i+1}) + p(v_{i+1}), \text{ otherwise .} \end{aligned} \tag{2.2.1}$$

Under Assumptions 2.2.2 and 2.2.3 we have that there is an index $n \in \mathbb{N}$ such that $\mathsf{Stop}(\mathrm{Run}(v,\mu)) = n$. Summing (2.2.1) side-wise for $0 \leq i < n$, we get the following.

$$p(v) \leq \sum_{i=1}^{n} \pi(v_i, v_{i-1}) + p(v_n).$$

Since $v_n \in F$, we have $p(v_n) = 0$ and we get the desired inequality.

$$p(v) \leq \mathsf{RP}(\mathrm{Run}(v,\mu)).$$

– For a set W the function $\mathsf{Choose} : 2^W \to W$ is defined such that for every set $X \subseteq W$ we have $\mathsf{Choose}(X) \in X$. In other words, Choose is a choice function which for every non-empty set selects an element of this set. We fix an arbitrary choice function for technical reasons, so that we have a way to canonically choose an arbitrary element from every non-empty set.

Let $\mu' \in \Pi$ be the following positional strategy:

$$\mu'(v) = \mathsf{Choose}(\operatorname*{argmin}_{w \in V} \{\pi(v,w) + p(w) \,:\, (v,w) \in E\}).$$

Since $p \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$, for such strategy μ' it is straightforward to notice that for all $v \in V \setminus F$ we have:

$$p(v) = \pi(v, \mu'(v)) + p(\mu'(v)) \,. \tag{2.2.2}$$

Let the run $\mathrm{Run}(v,\mu')$ be $\langle v = v_0, v_1, v_2, \ldots\rangle$. Again under Assumptions 2.2.2 and 2.2.3 we have that there is an index $n \in \mathbb{N}$ such that $\mathsf{Stop}(\mathrm{Run}(v,\mu')) = n$. Summing (2.2.2) for all $i, 0 \leq i < n$, we get the following.

$$p(v) = \sum_{i=1}^{n} \pi(v_i, v_{i-1}) + p(v_n).$$

Since $v_n \in F$, we have $p(v_n) = 0$ and we get the desired equality.

$$p(v) = \mathsf{RP}(\mathrm{Run}(v,\mu')).$$

We have thus shown that: for every strategy $\mu \in \Sigma$, we have $p(v) \leq \mathsf{RP}(\mathrm{Run}(v,\mu))$ for every $v \in V$; and there exists a positional strategy $\mu' \in \Pi$ such that $\mathsf{RP}(\mathrm{Run}(v,\mu')) = p(v)$ for every vertex $v \in V$. Hence it follows that $p(v) = \mathsf{RP}_*(v)$ for every vertex $v \in V$. ∎

From the second part of the proof of the previous proposition, the following proposition follows:

PROPOSITION 2.2.5 (A solution of optimality equations gives positional optimal strategy)**.** Let G be a priced finite graph. Under Assumptions 2.2.2 and 2.2.3 we have that if there exists a solution of optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ then there exists a positional optimal strategy for minimum reachability-price problem.

The following example shows that a solution of the optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ does not give the correct reachability-price, if the graph G contains a zero cycle.

EXAMPLE 2.2.6 (Problems with zero cycles)**.** Consider the graph shown in Figure 2.1. Notice that the graph contains zero cycles. For example the price of the cycle $\langle v_1, v_3, v_1\rangle$ is 0. Optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ are as follows:

$$\begin{aligned}\mathcal{P}(v_2) &= 0\\ \mathcal{P}(v_1) &= \min\{20+\mathcal{P}(v_2), \mathcal{P}(v_3)\}\\ \mathcal{P}(v_3) &= \min\{\mathcal{P}(v_1)\}.\end{aligned}$$

In this case infinitely many solutions of optimality equations exist; as all the functions $p : V \to \mathbb{R}$ such that $p(v_1) = k$, $p(v_2) = 0$ and $p(v_3) = k$, for any $k \leq 20$, satisfy the optimality equations. However only one (i.e., when $k = 20$) of these functions gives correct minimum reachability-price.

To solve the reachability-price problem for general graphs, we propose the following optimality equations:

$$(\mathcal{P}(v), \mathcal{D}(v)) = \begin{cases}(0,0), & \text{if } v \in F,\\ \min^{\mathrm{lex}}_{(v,w)\in E}\{(\pi(v,w)+\mathcal{P}(w), 1+\mathcal{D}(w))\}, & \text{otherwise.}\end{cases}$$

For the sake of notational convenience we reuse the name $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ for these set of equations as well. We say that the functions $p : V \to \mathbb{R}$ and $d : V \to \mathbb{N}$ satisfy the optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ and we write $(p,d) \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$, if all equations in $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ hold for the valuations $\mathcal{P}(v) \mapsto p(v)$ and $\mathcal{D}(v) \mapsto d(v)$.

The introduction of $\mathcal{D}$ variables solves the problem due to zero cycles. The proof of the following proposition is similar to the proof of Proposition 2.2.4 and hence we skip the proof here. We provide a complete proof of a lemma similar to this proposition, but in the context of timed automata (see Proposition 5.1.3).

PROPOSITION 2.2.7 (A solution of optimality equations gives optimal reachability-price)**.** Under the Assumption 2.2.2 we have that if $(p,d) \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ then we have $p(v) = \mathsf{RP}_*(v)$ for all $v \in V$.

PROPOSITION 2.2.8 (A solution of optimality equations gives positional optimal strategy)**.** Under the Assumption 2.2.2 we have that if there exists a solution (p,d) of the optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ then there exists a positional optimal strategy for reachability-price problem.

In the rest of this subsection, we give a constructive proof of the following proposition by giving a strategy improvement algorithm which yields a solution of the optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$.

PROPOSITION 2.2.9 (Existence of a solution)**.** If the Assumption 2.2.2 holds then there exists a solution of the optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$.

2.2.1.2. Solving 0-player Optimality Equations

For a graph $G = (V, E, F)$ and a positional strategy $\mu \in \Pi_{\mathrm{Min}}$, we define the graph $G\restriction\mu = (V, E', F)$ such that $E' = \{(v, \mu(v)) \,:\, v \in V\}$.

If the graph $G = (V, E, F)$ is such that every vertex has a unique successor, i.e., for every $v \in V$ we have that the set $E(v) = \{w \,:\, (v, w) \in E\}$ is a singleton, then we call such a graph a 0-player graph. Observe that for an arbitrary graph G and a positional strategy $\mu \in \Pi_{\mathrm{Min}}$ the graph $G\restriction\mu$ is a 0-player graph. A 0-player priced graph is defined in straightforward manner.

For a 0-player priced graph $G = (V, E, F, \pi)$, the optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ can be rewritten in the following manner:

$$(\mathcal{P}(v), \mathcal{D}(v)) \;=\; \begin{cases} (0,0)\,, & \text{if } v \in F, \\ (\pi(v, E(v)) + \mathcal{P}(E(v)), 1 + \mathcal{D}(E(v)))\,, & \text{otherwise.} \end{cases}$$

Remark. For brevity we have abused the notation to denote the unique vertex w such that $(v, w) \in E$ by the singleton set $E(v)$ containing w.

For a 0-player priced graph G, we write $\mathcal{OE}^{\mathsf{RP}}(G)$ for these set of equations. The optimality equations for 0-player reachability problem can be solved in time $O(|V|)$.

2.2.1.3. Solving 1-player Optimality Equations

For given functions $p : V \to \mathbb{R}$ and $d : V \to \mathbb{N}$, for every vertex $v \in V$, let us define the set $M_*(v, p, d)$ as follows:

$$M_*(v, p, d) = \operatorname*{argmin}^{\mathrm{lex}}_{w \in V} \{(\pi(v, w) + p(w),\, 1 + d(w)) \,:\, (v, w) \in E\}.$$

Consider the strategy improvement function $\mathsf{Improve}_{\mathrm{Min}} : \Pi \times [V \to \mathbb{R}] \times [V \to \mathbb{N}] \to \Pi$ defined by:

$$\mathsf{Improve}_{\mathrm{Min}}(\sigma, p, d)(v) = \begin{cases} \sigma(v) & \text{if } \sigma(v) \in M_*(v, p, d) \\ \mathsf{Choose}(M_*(v, p, d)) & \text{otherwise,} \end{cases}$$

where $p : V \to \mathbb{R}$, $d : V \to \mathbb{N}$, and $v \in V$. From the definition of the function $\mathrm{Improve}_{\mathrm{Min}}$, the following proposition is immediate.

PROPOSITION 2.2.10 (Fixed point of $\mathrm{Improve}_{\mathrm{Min}}$ are solutions of $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$)**.** Let $\mu \in \Pi_{\mathrm{Min}}$ and let $(p^\mu, d^\mu) \models \mathcal{OE}^{\mathsf{RP}}(G\restriction\mu)$. If $\mathrm{Improve}_{\mathrm{Min}}(\mu, p^\mu, d^\mu) = \mu$ then $(p^\mu, d^\mu) \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$.

Input: A graph $G = (V, E, F, \pi)$
Output: A solution of $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$

```
begin
    (Initialisation). Choose an arbitrary positional strategy μ_0 ∈ Π_Min;
    Set i = 0;
    repeat
        (Value Computation). Compute the solution (p_i, d_i) of OE^RP(G↾μ_i);
        (Strategy Improvement). Compute μ_{i+1} = Improve_Min(μ_i, p_i, d_i);
        Set i := i + 1;
    until μ_{i+1} ≡ μ_i ;
    return (p_i, d_i);
end
```

FIGURE 2.2. Strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$.

A pseudocode of the strategy improvement algorithm to solve the optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ is shown as Algorithm 2.2. Before we prove the correctness and termination properties of the algorithm, we would like to introduce a few notations and their properties.

We say that functions $p : V \to \mathbb{R}$ and $d : V \to \mathbb{N}$ satisfy the set of optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\geq}(G)$, and we write $(p, d) \models \mathcal{OE}^{\mathsf{RP}}_{\geq}(G)$, if

$$(p(v), d(v)) \quad \geq^{\mathrm{lex}} \quad \begin{cases} (0,0)\,, & \text{if } v \in F, \\ \max^{\mathrm{lex}} \left\{ \, (\pi(v,w) + p(w), 1 + d(w)) \, : \, (v,w) \in E \right\}, & \text{otherwise.} \end{cases}$$

The following proposition summarise a useful relation between a solution of $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ and $\mathcal{OE}^{\mathsf{RP}}_{\geq}(G)$.

PROPOSITION 2.2.11. Let $p, p_{\geq} : V \to \mathbb{R}$ and $d, d_{\geq} : V \to \mathbb{N}$ be such that $(p, d) \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ and $(p_{\geq}, d_{\geq}) \models \mathcal{OE}^{\mathsf{RP}}_{\geq}(G)$. Then we have $(p_{\geq}, d_{\geq}) \geq^{\mathrm{lex}} (p, d)$, and if $(p_{\geq}, d_{\geq}) \not\models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ then $(p_{\geq}, d_{\geq}) >^{\mathrm{lex}} (p, d)$.

PROOF. First we establish by induction on $d(v)$ that $(p_{\geq}, d_{\geq}) \geq^{\mathrm{lex}} (p, d)$. The base case, $d(v) = 0$, is trivial because, since $v \in F$, we have that $(p(v), d(v)) = (0, 0)$ and $(p_{\geq}, d_{\geq}) \geq (0, 0)$.

Let $v \in V \setminus F$ be such that $d(v) = n + 1$ and let $(v, w) \in E$ be such that $(p(v), d(v)) = (\pi(v, w) + p(w), 1 + d(w))$. Notice that it implies that $d(w) = n$ and hence by induction hypothesis we have $(p_{\geq}(w), d_{\geq}(w)) \geq^{\mathrm{lex}} (p(w), d(w))$. Since $(p_{\geq}, d_{\geq}) \models \mathcal{OE}^{\mathsf{RP}}_{\geq}(G)$, we have

$$\begin{aligned} (p_{\geq}(v), d_{\geq}(v)) \quad &\geq^{\mathrm{lex}} \quad (\pi(v,w) + p_{\geq}(w), 1 + d_{\geq}(w)) \\ &\geq^{\mathrm{lex}} \quad (\pi(v,w) + p(w), 1 + d(w)) \\ &= \quad (p(v), d(v)) \ . \end{aligned} \tag{2.2.3}$$

The second inequality follows from $(p_{\geq}(w), d_{\geq}(w)) \geq^{\text{lex}} (p(w), d(w))$. This concludes the proof that $(p_{\geq}, d_{\geq}) \geq^{\text{lex}} (p, d)$.

Now we prove that if $(p_{\geq}, d_{\geq}) \not\models \mathcal{OE}^{\mathsf{RP}}_{\text{Min}}(G)$ then there is a vertex $v \in V$ such that $(p_{\geq}(v), d_{\geq}(v)) >^{\text{lex}} (p(v), d(v))$. Indeed, if $(p_{\geq}, d_{\geq}) \not\models \mathcal{OE}^{\mathsf{RP}}_{\text{Min}}(G)$ then either we have that $(p_{\geq}(v), d_{\geq}(v)) <^{\text{lex}} (0, 0)$ for some vertex $v \in F$, or there is some vertex $v \in V \setminus F$ such that the inequality (2.2.3) is strict and hence we get $(p_{\geq}(v), d_{\geq}(v)) >^{\text{lex}} (p(v), d(v))$. ∎

The following lemma states that every step of strategy improvement algorithm improves the strategy.

LEMMA 2.2.12 (Strict strategy improvement)**.** Let $\mu, \mu' \in \Pi_{\text{Min}}$, let $(p, d) \models \mathcal{OE}^{\mathsf{RP}}(G{\restriction}\mu)$ and $(p', d') \models \mathcal{OE}^{\mathsf{RP}}(G{\restriction}\mu')$, and let $\mu' = \text{Improve}_{\text{Min}}(\mu, p, d)$. Then $(p, d) \geq^{\text{lex}} (p', d')$ and if $\mu \neq \mu'$ then $(p, d) >^{\text{lex}} (p', d')$.

PROOF. First we argue that $(p, d) \models \mathcal{OE}^{\mathsf{RP}}_{\geq}(G{\restriction}\mu')$, which by Proposition 2.2.11 implies that $(p, d) \leq^{\text{lex}} (p', d')$. Indeed, for every $v \in V \setminus F$ if $\mu(v) = w$ and $\mu'(v) = w'$ then we have

$$\begin{aligned} (p(v), d(v)) &= (\pi(v, w) + p(w), 1 + d(w)), \text{ as } (p, d) \models \mathcal{OE}^{\mathsf{RP}}(G{\restriction}\mu) \\ &\geq^{\text{lex}} (\pi(v, w') + p(w'), 1 + d(w')), \text{ by definition of } \text{Improve}_{\text{Min}}. \end{aligned}$$

Moreover, if $\mu \neq \mu'$ then there is a $v \in V \setminus F$ for which the above inequality is strict. Then $(p, d) \not\models \mathcal{OE}^{\mathsf{RP}}_{\text{Min}}(G{\restriction}\mu')$, because every vertex in $G{\restriction}\mu'$ has a unique successor, and hence by Proposition 2.2.11 we conclude that $(p, d) >^{\text{lex}} (p', d')$. ∎

LEMMA 2.2.13 (Correctness and termination of the strategy improvement algorithm)**.** The strategy improvement algorithm for $\mathcal{OE}^{\mathsf{RP}}_{\text{Min}}(G)$ terminates in finitely many steps and returns a solution (p, d) of $\mathcal{OE}^{\mathsf{RP}}_{\text{Min}}(G)$.

PROOF. Since the total number of positional strategies in a finite priced graph is finite (bounded from above by $|V|^{|V|}$), this lemma follows directly from Lemma 2.2.12 and Proposition 2.2.10. ∎

2.2.2. Solving Discounted-Price Minimisation Problem

For a priced graph G and a discount factor $\lambda \in (0, 1)$, a solution of minimisation problem for discounted-price function can be characterised by the following equations:

$$\mathcal{D}(v) = \min_{(v,w) \in E} \{(1 - \lambda) \cdot \pi(v, w) + \lambda \cdot \mathcal{D}(w)\}, \text{ for all } v \in V.$$

We denote these equations by $\mathcal{OE}^{\mathsf{DP}}_{\text{Min}}(G, \lambda)$. We say that a function $d : V \to \mathbb{R}$ is a solution of the set of equations $\mathcal{OE}^{\mathsf{DP}}_{\text{Min}}(G, \lambda)$, and we write $d \models \mathcal{OE}^{\mathsf{DP}}_{\text{Min}}(G, \lambda)$, if for all $v \in V$ the equations in $\mathcal{OE}^{\mathsf{DP}}_{\text{Min}}(G, \lambda)$ holds for the valuation $\mathcal{D}(v) \mapsto d(v)$.

The following proposition shows that a solution of optimality equations $\mathcal{OE}^{\mathsf{DP}}_{\text{Min}}(G, \lambda)$ gives a solution of minimum discounted-price problem.

PROPOSITION 2.2.14 (A solution of optimality equations gives optimal discounted price)**.** For a priced graph G and a discount factor $\lambda \in (0,1)$, if $d \models \mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G,\lambda)$ then for all $v \in V$ we have that $d(v) = \mathsf{DP}(\lambda)_*(v)$.

PROOF. Let $d \models \mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G,\lambda)$. There are two parts of the proof of this lemma. First we show that for every strategy $\mu \in \Sigma_{\mathrm{Min}}$ we have that $d(v) \leq \mathsf{DP}(\lambda)(\mathrm{Run}(v,\mu))$ for every $v \in V$. In the second part we show that there exists a positional strategy $\mu' \in \Pi_{\mathrm{Min}}$ such that for every vertex $v \in V$ we have that $d(v) = \mathsf{DP}(\lambda)(\mathrm{Run}(v,\mu'))$.

– Let μ be an arbitrary strategy and let $\mathrm{Run}(v,\mu)$ be the sequence $\langle v_0 = v, v_1, v_2, \ldots\rangle$. Since $d \models \mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G,\lambda)$, for all $i \geq 0$ we have that

$$d(v_i) \leq (1-\lambda)\cdot\pi(v_i, v_{i+1}) + \lambda \cdot d(v_{i+1}).$$

For all $0 \leq i < n$ multiplying both sides of this equation by λ^i and summing sidewise we get the following inequality:

$$d(v_0) \leq (1-\lambda)\sum_{i=0}^{n-1}\lambda^i\cdot\pi(v_i,v_{i+1}) + \lambda^n\cdot d(v_n).$$

Taking limit of both sides we get the desired inequality:

$$d(v) = d(v_0) \leq \mathsf{DP}(\lambda)(\mathrm{Run}(v,\mu)).$$

– Let $\mu' \in \Pi_{\mathrm{Min}}$ be the following positional strategy: for all $v \in V$ we have that

$$\mu'(v) = \mathsf{Choose}(\operatorname*{argmin}_{w\in V}\{(1-\lambda)\cdot\pi(v,w) + \lambda\cdot d(w) \;:\; (v,w)\in E\}).$$

Since $d : V \to \mathbb{R}$ is a solution of optimality equations $\mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G,\lambda)$, from previous equation it follows that for all $v \in V$ we have

$$d(v) = (1-\lambda)\cdot\pi(v,\mu'(v)) + \lambda\cdot d(\mu'(v)).$$

By doing a similar analysis to that in the previous part of this proof, it can be shown that:

$$d(v) = \mathsf{DP}(\lambda)(\mathrm{Run}(v,\mu')).$$

∎

From the second part of the proof of the previous proposition, the following proposition is immediate.

PROPOSITION 2.2.15 (A solution of optimality equations gives positional optimal strategy)**.** If there exists a solution of optimality equations $\mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G,\lambda)$ then there exists a positional optimal strategy for minimum discounted-price problem.

2.2.2.1. Existence of a solution

PROPOSITION 2.2.16 (Existence of a solution)**.** For every priced graph G and every discount factor $\lambda \in (0,1)$, there exists a solution of the optimality equations $\mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G,\lambda)$.

Again, the proof of existence of a solution of optimality equation is via strategy improvement algorithm. The rest of this subsection is devoted to the strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G,\lambda)$.

2.2.2.2. Solving 0-player Optimality Equations

For a 0-player graph $G = (V, E, F, \pi)$ the optimality equations $\mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G,\lambda)$ can be rewritten as follows:

$$\mathcal{D}(v) = (1-\lambda)\cdot\pi(v, E(v)) + \lambda\cdot\mathcal{D}(E(v)), \text{ for all } v \in V.$$

For a 0-player priced graph G we denote this set of equations by $\mathcal{OE}^{\mathsf{DP}}(G,\lambda)$.

The runs of a 0-player priced graph are simple paths leading to a simple cycle repeated infinitely many times. Let us suppose that the run starting from a vertex $v \in V$ is the sequence $\langle v = v_0, v_1, \ldots, (v_k, v_{k+1}, \ldots, v_{k+n-1}, v_{k+n} = v_k)^\omega\rangle$. The solution of the optimality equation $d \models \mathcal{OE}^{\mathsf{DP}}(G,\lambda)$ can be computed in polynomial time:

$$d(v_0) = (1-\lambda)\left(\sum_{i=0}^{k-1}\lambda^i\cdot\pi(v_i,v_{i+1}) + \sum_{i=k}^{k+n-1}\frac{\lambda^i}{(1-\lambda^n)}\pi(v_i,v_{i+1})\right).$$

2.2.2.3. Solving 1-player Optimality Equations

For given function $d : V \to \mathbb{R}$ and for every vertex $v \in V$, let us define the set $M_*(v,d)$ as follows:

$$M_*(v,d) = \operatorname*{argmin}_{w\in V}\{(1-\lambda)\cdot\pi(v,w) + \lambda\cdot p(w) \;:\; (v,w)\in E\}.$$

Consider the following strategy improvement function $\mathsf{Improve}_{\mathrm{Min}} : \Pi\times[V\to\mathbb{R}]\to\Pi$ defined as

$$\mathsf{Improve}_{\mathrm{Min}}(\sigma,d)(v) = \begin{cases}\sigma(v) & \text{if } \sigma(v)\in M_*(v,d)\\ \mathsf{Choose}(M_*(v,d)) & \text{otherwise,}\end{cases}$$

where $d : V \to \mathbb{R}$ and $v \in V$.

PROPOSITION 2.2.17 (Fixed point of $\mathrm{Improve}_{\mathrm{Min}}$ are solutions of $\mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G,\lambda)$)**.** Let $\mu \in \Pi_{\mathrm{Min}}$ and let $d^\mu \models \mathcal{OE}^{\mathsf{DP}}(G\restriction\mu,\lambda)$. If $\mathrm{Improve}_{\mathrm{Min}}(\mu, d^\mu) = \mu$ then $d^\mu \models \mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G,\lambda)$.

Strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G,\lambda)$ is shown in Algorithm 2.3. The correctness of the algorithm is immediate from the following lemmas.

Input: A graph $G = (V, E, F, \pi)$ and $\lambda \in (0, 1)$
Output: A solution of $\mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G, \lambda)$

```
begin
    (Initialisation). Choose an arbitrary positional strategy μ_0 ∈ Π_Min;
    Set i = 0;
    repeat
        (Value Computation). Compute the solution d_i of OE^DP(G↾μ_i, λ);
        (Strategy Improvement). Compute μ_{i+1} = Improve_Min(μ_i, d_i);
        Set i := i + 1;
    until μ_{i+1} ≡ μ_i ;
    return d_i;
end
```

FIGURE 2.3. Strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G, \lambda)$.

LEMMA 2.2.18 (Strict strategy improvement for Min)**.** Let strategies $\mu, \mu' \in \Pi_{\mathrm{Min}}$, let $d \models \mathcal{OE}^{\mathsf{DP}}(G{\restriction}\mu, \lambda)$ and $d' \models \mathcal{OE}^{\mathsf{DP}}(G{\restriction}\mu', \lambda)$, and let $\mu' = \mathrm{Improve}_{\mathrm{Min}}(\mu, d)$. Then $d \geq d'$ and if $\mu \neq \mu'$ then $d > d'$.

LEMMA 2.2.19 (Correctness and termination of the strategy improvement algorithm)**.** The strategy improvement algorithm for $\mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G, \lambda)$ terminates in finitely many steps and returns a solution of $\mathcal{OE}^{\mathsf{DP}}_{\mathrm{Min}}(G, \lambda)$.

2.2.3. Solving Average-price and Price-per-reward Average Minimisation Problems

It is easy to see that average-price minimisation problems are special cases of price-per-reward average minimisation problems, where the reward function $\varrho : E \to \mathbb{R}$ is such that $\varrho(e) = 1$, for every edge $e \in E$. Hence we shall not discuss the average-price minimisation problem separately.

Given a graph $G = (V, E, F, \pi, \varrho)$ the following equations capture the optimal price-per-reward average cost:

$$\begin{aligned} \mathcal{G}(v) &= \min_{(v,w)\in E} \{\mathcal{G}(w)\}, \text{ for all } v \in V, \text{ and} \\ \mathcal{B}(v) &= \min_{(v,w)\in E} \{\pi(v,w) - \varrho(v,w)\cdot\mathcal{G}(v) + \mathcal{B}(w) \,:\, \mathcal{G}(v) = \mathcal{G}(w)\} \text{ for all } v \in V. \end{aligned}$$

Sometimes we write these equations in the following compact form. It is routine to verify that the two sets of equations are equivalent.

$$(\mathcal{G}(v), \mathcal{B}(v)) = \min^{\mathrm{lex}}_{(v,w)\in E} \{\, (\mathcal{G}(w),\ \pi(v,w) - \varrho(v,w)\cdot\mathcal{G}(v) + \mathcal{B}(w)) \,\} \text{ for all } v \in V.$$

We denote these equations by $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$, and we write $(g, b) \models \mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$ to denote the fact that functions $g : V \to \mathbb{R}$ and $b : V \to \mathbb{R}$ are a solution of $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$. For a solution (g, b) of $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$ and a vertex $v \in V$, we say that $g(v)$ is the *gain* and $b(v)$ is the *bias* of the vertex v.

PROPOSITION 2.2.20 (A solution of optimality equations gives optimal price-per-reward average)**.** For a finite price-reward graph G, if $(g, b) \models \mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$ then for all $v \in V$ we have that $g(v) = \mathsf{PRAvg}_*(v)$.

PROOF. We prove the proposition in two parts. First we show that for every strategy $\mu \in \Sigma$, we have $G(v) \leq \mathsf{PRAvg}(\mathrm{Run}(v, \mu))$ for every vertex $v \in V$. In the second part, we show that there exists a positional strategy $\mu' \in \Pi$ such that $\mathsf{PRAvg}(\mathrm{Run}(v, \mu')) = G(v)$ for every vertex $v \in V$.

– Let $\mathrm{Run}(v, \mu) = \langle v = v_0, v_1, \ldots \rangle$ be the run from the vertex v according to the strategy μ. Since g and b are a solution of optimality equations $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$, for all $i \geq 0$, we have $g(v_i) \leq g(v_{i+1})$. Since there are finitely many vertices, for all $i \geq |V|$ we have that $g(v_i) = g(v_{i+1}) = g$. Note that $g(v) \leq g$.

  Now from the optimality equation it follows that for all $i \geq |V|$, we have

$$b(v_i) \quad \leq \quad \pi(v_i, v_{i+1}) - \varrho(v_i, v_{i+1}) \cdot g + b(v_{i+1}).$$

  Summing the above bias equation for n terms, we get:

$$b(v_{|V|}) \leq \sum_{i=|V|}^{|V|+n} \pi(v_i, v_{i+1}) - g \cdot \sum_{i=|V|}^{|V|+n} \varrho(v_i, v_{i+1}) + b(v_{|V|+n}),$$

  and hence:

$$b(v_{|V|}) - b(v_{|V|+n}) \leq \sum_{i=|V|}^{|V|+n} \pi(v_i, v_{i+1}) - g \cdot \sum_{i=|V|}^{|V|+n} \varrho(v_i, v_{i+1}).$$

  Let $\kappa = \sum_{i=0}^{|V|-1} (\pi(v_i, v_{i+1}) - g \cdot \varrho(v_i, v_{i+1}))$ and $N = |V| + n$. We can rewrite the previous inequality as:

$$b(v_{|V|}) - b(v_N) \leq \sum_{i=0}^{N} \pi(v_i, v_{i+1}) - g \cdot \sum_{i=0}^{N} \varrho(v_i, v_{i+1}) - \kappa.$$

  Adding κ both sides and then dividing both sides by $\sum_{i=0}^{N} \varrho(v_i, v_{i+1})$ we get:

$$\frac{b(v_{|V|}) - b(v_N) + \kappa}{\sum_{i=0}^{N} \varrho(v_i, v_{i+1})} \leq \frac{\sum_{i=0}^{N} \pi(v_i, v_{i+1})}{\sum_{i=0}^{N} \varrho(v_i, v_{i+1})} - g.$$

  Notice that the expression $b(v_{|V|}) - b(v_N) + \kappa$ is bounded and the price-reward graph is strongly reward-diverging, i.e., $\lim_{n\to\infty} \sum_{i=0}^{N} \varrho(v_i, v_{i+1}) = \infty$. It follows from these observations that:

$$g \leq \mathsf{PRAvg}(\mathrm{Run}(v, \mu)),$$

  and since $g(v) \leq g$ we get the desired inequality.

– Let us compute the positional strategy $\mu' \in \Pi$ from a solution (g, b) of optimality equations in the following manner: for all $v \in V$ we have that

$$\mu'(v) \in \operatorname*{argmin}_{w \in V}{}^{\text{lex}} \left\{ \left(g(w),\ \pi(v,w) - \varrho(v,w) \cdot g(v) + b(w)\right) \ :\ (v,w) \in E \right\}).$$

Since $(g, b) \models \mathcal{OE}^{\mathsf{PRAvg}}_{\text{Min}}(G)$, it is easy to verify that for all $v \in V$ we have:

$$\begin{aligned} g(v) &= g(\mu'(v)), \text{ and} \\ b(v) &= \pi(v, \mu'(v)) - g(v) \cdot \varrho(v, \mu'(v)) + b(\mu'(v)). \end{aligned}$$

Let $\text{Run}(v, \mu') = \langle v = v_0, v_1, v_2, \ldots \rangle$. For $0 \leq i \leq n$, summing the bias equation side-wise, we get the following:

$$b(v) = \sum_{i=0}^{n} \pi(v_i, v_{i+1}) - g(v) \cdot \sum_{i=0}^{n} \varrho(v_i, v_{i+1}) + b(v_n).$$

Using a similar analysis to the first part, we get that

$$g(v) = \mathsf{PRAvg}(\text{Run}(v, \mu')),$$

as required.

∎

From the second part of the proof of the previous proposition, the following proposition follows easily:

PROPOSITION 2.2.21 (A solution of optimality equations gives positional optimal strategy)**.** For a finite price-reward graph G, if there exists a solution of $\mathcal{OE}^{\mathsf{PRAvg}}_{\text{Min}}(G)$ then there exists a positional optimal strategy for the minimum price-per-reward average problem.

2.2.3.1. Existence of a solution

PROPOSITION 2.2.22 (Existence of a solution)**.** For every finite price-reward graph G, there exists a solution to the optimality equations $\mathcal{OE}^{\mathsf{PRAvg}}_{\text{Min}}(G)$.

In the rest of this subsection, we give a constructive proof of the Proposition 2.2.22 by giving a strategy improvement algorithm which yields a solution of the optimality equations.

2.2.3.2. Solving 0-player Optimality Equations

For a 0-player price-reward graph G, the optimality equations $\mathcal{OE}^{\mathsf{PRAvg}}_{\text{Min}}(G)$ can be rewritten as follows:

$$\left(\mathcal{G}(v), \mathcal{B}(v)\right) \quad = \quad \left(\mathcal{G}(E(v)),\ \pi(v, E(v)) - \varrho(v, E(v)) \cdot \mathcal{G}(v) + \mathcal{B}(E(v))\right) \ \text{ for all } v \in V.$$

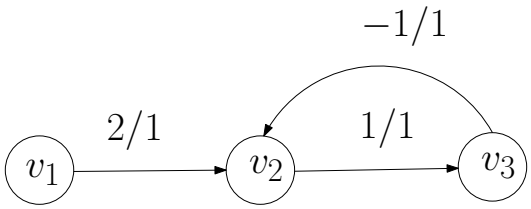


FIGURE 2.4. A finite price-reward graph.

For a 0-player price-reward graph G, we denote this set of equations by $\mathcal{OE}^{\mathsf{PRAvg}}(G)$. The following example shows that a solution of optimality equations $\mathcal{OE}^{\mathsf{PRAvg}}(G)$ is not unique.

EXAMPLE 2.2.23 (Biases are not unique)**.** Consider a 0-player graph $G = (V, E, \pi, \varrho)$ (see Figure 2.4), where $V = \{v_1, v_2, v_3\}$ and $E = \{(v_1, v_2), (v_2, v_3), (v_3, v_2)\}$. The price function $\pi : V \to \mathbb{R}$ is such that $\pi(v_1, v_2) = 2$, $\pi(v_2, v_3) = 1$ and $\pi(v_3, v_2) = -1$. The reward function $\rho : V \to \mathbb{R}$ is such that $\varrho(v, v') = 1$ for all $(v, v') \in E$. Optimality equations $\mathcal{OE}^{\mathsf{PRAvg}}(G)$ are as follows:

$$\begin{aligned}
(\mathcal{G}(v_1), \mathcal{B}(v_1)) &= (\mathcal{G}(v_2), 2 - 1 \cdot \mathcal{G}(v_1) + \mathcal{B}(v_2)) \\
(\mathcal{G}(v_2), \mathcal{B}(v_2)) &= (\mathcal{G}(v_3), 1 - 1 \cdot \mathcal{G}(v_2) + \mathcal{B}(v_3)) \\
(\mathcal{G}(v_3), \mathcal{B}(v_3)) &= (\mathcal{G}(v_2), -1 - 1 \cdot \mathcal{G}(v_3) + \mathcal{B}(v_2))
\end{aligned}$$

It is easy to verify that for any value $k \in \mathbb{R}$ the functions $g : V \to \mathbb{R}$ and $b_k : V \to \mathbb{R}$ such that $g(v_1) = g(v_2) = g(v_3) = 0$; $b_k(v_1) = k + 3$, $b_k(v_2) = k + 1$ and $b_k(v_3) = k$ satisfy the set of optimality equations.

However for a 0-player price-reward graph G, biases can be made "unique" by applying one of the following heuristics [**Put94**]:

(1) For every cycle of the 0-player price-reward graph, choose a vertex in that cycle and assign its bias to some number $k \in \mathbb{R}$.
(2) For every cycle of the 0-player price-reward graph, add an extra constraint that the sum of the biases of the vertices on the cycle is 0.

The gain of every vertex and the biases (now unique) of the rest of the vertices can be computed in polynomial time.

Remark. For an arbitrary price-reward graph the following optimality equations—with an extra set of $\mathcal{H}$ variables, one for every vertex—can also make the biases of the vertices unique:

$$\begin{aligned}
(\mathcal{G}(v), \mathcal{B}(v)) &= \min^{\text{lex}}_{(v,w)\in E} \left\{ (\mathcal{G}(w),\ \pi(v, w) - \varrho(v, w) \cdot \mathcal{G}(v) + \mathcal{B}(w)) \right\} \text{ for all } v \in V, \text{ and} \\
\mathcal{H}(v) &= \min_{(v,w)\in E} \left\{ -\mathcal{B}(v) + \mathcal{H}(w) \ :\ (\mathcal{G}(v), \mathcal{B}(v)) = (\mathcal{G}(w, \mathcal{B}(w)) \right\} \text{ for all } v \in V.
\end{aligned}$$

A careful analysis of these optimality equations shows that the biases of the vertices in a solution of these equations are the same as the biases of the vertices in a solution of the

original optimality equations $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$, with the second heuristic of assigning biases in a 0-player graph.

2.2.3.3. Solving 1-player Optimality Equations

Given functions $g : V \to \mathbb{R}$, $b : V \to \mathbb{R}$, and a vertex $v \in V$, let us define the set $M_*(v, g, b)$ as follows:

$$M_*(v, g, b) = \operatorname*{argmin}^{\mathrm{lex}}_{w \in V} \{(\mathcal{G}(w),\ \pi(v, w) - \varrho(v, w) \cdot \mathcal{G}(v) + \mathcal{B}(w)) \ :\ (v, w) \in E\}.$$

Consider the strategy improvement function $\mathrm{Improve}_{\mathrm{Min}} : \Pi \times [V \to \mathbb{R}] \times [V \to \mathbb{R}] \to \Pi$, defined as

$$\mathrm{Improve}_{\mathrm{Min}}(\mu, g, b)(v) = \begin{cases} \mu(v) & \text{if } \mu(v) \in M_*(v, g, b) \\ \mathsf{Choose}(M_*(v, g, b)) & \text{otherwise,} \end{cases}$$

where $g : V \to \mathbb{R}$, $b : V \to \mathbb{R}$, and $v \in V$.

PROPOSITION 2.2.24 (Fixed point of $\mathrm{Improve}_{\mathrm{Min}}$ are solutions of $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$)**.** Let $\mu \in \Pi_{\mathrm{Min}}$ and let $(g^\mu, b^\mu) \models \mathcal{OE}^{\mathsf{PRAvg}}(G\restriction\mu)$. If $\mathrm{Improve}_{\mathrm{Min}}(\mu, g^\mu, b^\mu) = \mu$ then $(g^\mu, b^\mu) \models \mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$.

Input: A graph $G = (V, E, F, \pi, \varrho)$
Output: A solution of $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$

```
1  begin
2      (Initialisation). Choose an arbitrary positional strategy μ_0 ∈ Π_Min;
3      Set i = 0;
4      repeat
5          (Value Computation). Compute the solution (g_i, b_i) of OE^PRAvg(G↾μ_i);
6          (Strategy Improvement). Compute μ_{i+1} = Improve_Min(μ_i, g_i, b_i);
7          Set i := i + 1;
8      until μ_{i+1} ≡ μ_i ;
9      return (g_i, b_i);
10 end
```

FIGURE 2.5. Strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$.

A pseudocode for the strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$ is shown in Algorithm 2.5. The proof of the following lemma is similar to the proof of Lemma 2.2.12 and hence we omit the proof.

LEMMA 2.2.25 (Strict strategy improvement for player Min)**.** Let strategies $\mu, \mu' \in \Pi_{\mathrm{Min}}$, let $(g, b) \models \mathcal{OE}^{\mathsf{PRAvg}}(G\restriction\mu)$ and $(g', b') \models \mathcal{OE}^{\mathsf{PRAvg}}(G\restriction\mu')$, and let $\mu' = \mathrm{Improve}_{\mathrm{Min}}(\mu, g, b)$. Then $(g, b) \geq^{\mathrm{lex}} (g', b')$ and if $\mu \neq \mu'$ then $(g, b) >^{\mathrm{lex}} (g', b')$.

LEMMA 2.2.26 (Correctness and termination of the strategy improvement algorithm)**.** The strategy improvement algorithm for $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$ terminates in finitely many steps and returns a solution (g, b) of $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G)$.

PROOF. Since the total number of positional strategies of player Min in a finite graph is finite (less than or equal to $|V|^{|V|}$), the lemma follows directly from Lemma 2.2.25 and Proposition 2.2.24. ∎

2.2.4. A Note on Value Iteration Algorithm

So far, we have presented strategy improvement algorithms to solve optimality equations for reachability price, discounted price, average price, and price-per-reward average cost functions. In this subsection we present some results related to these cost functions which allow us to write value iteration algorithm to solve noncompetitive optimisation problems.

For a graph G, let Runs_n be the set of runs of length n. For a cost function $\mathsf{Cost} : \mathrm{Runs} \to \mathbb{R}$, we say that a sequence $\langle \mathsf{Cost}_n : \mathrm{Runs}_n \to \mathbb{R} \rangle_{n \in \mathbb{N}}$ approximates the cost function Cost if for all runs $r \in \mathrm{Runs}$ we have that

$$\mathsf{Cost}(r) = \limsup_{n \to \infty} \mathsf{Cost}_n(r).$$

For a sequence $\langle \mathsf{Cost}_n : \mathrm{Runs}_n \to \mathbb{R} \rangle_{n \in \mathbb{N}}$, we define the *n-step minimum cost function* $\mathsf{Cost}_{n,*} : V \to \mathbb{R}$, by:

$$\mathsf{Cost}_{n,*}(v) = \inf_{r \in \mathrm{Runs}_n(v)} \mathsf{Cost}_n(r).$$

DEFINITION 2.2.27 (Uniform Convergence)**.** A cost function $\mathsf{Cost} : \mathrm{Runs} \to \mathbb{R}$ is *uniform convergent* if for every vertex $v \in V$ we have:

$$\mathsf{Cost}_*(v) = \limsup_{n \to \infty} \mathsf{Cost}_{n,*}(v).$$

For the reachability price, discounted price, average price, and price-per-reward average cost functions, we define the following sequences that approximate the corresponding cost functions.

(1) *Reachability price.* For every $n \in \mathbb{N}$ we define $\mathsf{RP}_n : \mathrm{Runs}_n \to \mathbb{R}$ by

$$\mathsf{RP}_n(r) = \begin{cases} \pi_N(r) & \text{if } N = \mathsf{Stop}(r) \leq n \\ n & \text{otherwise,} \end{cases}$$

for every run $r \in \mathrm{Runs}_n$.

(2) *Discounted price.* For every $n \in \mathbb{N}$ and discount factor $\lambda \in (0, 1)$, we define $\mathsf{DP}_n(\lambda) : \mathrm{Runs}_n \to \mathbb{R}$ by

$$\mathsf{DP}_n(\lambda)(r) = (1 - \lambda) \sum_{i=1}^{n} \lambda^{i-1} \pi(v_{i-1}, v_i),$$

for every run $r \in \mathrm{Runs}_n$.

(3) *Average price.* For every $n \in \mathbb{N}$ we define $\mathsf{AP}_n : \mathrm{Runs}_n \to \mathbb{R}$ by

$$\mathsf{AP}_n(r) = \frac{\pi_n(r)}{n},$$

for every run $r \in \mathrm{Runs}_n$.

(4) *Price-per-reward average.* Finally, for every $n \in \mathbb{N}$ we define $\mathsf{PRAvg}_n : \mathrm{Runs}_n \to \mathbb{R}$ by

$$\mathsf{PRAvg}_n(r) = \frac{\pi_n(r)}{\varrho_n(r)},$$

for every run $r \in \mathrm{Runs}_n$.

Notice that if the cost functions RP, DP, AP, and PRAvg are uniform convergent, then the value iteration method for these problems is immediate: at every iteration n, generate successively the corresponding n-step minimum cost $\mathsf{Cost}_n(v)$ for every vertex $v \in V$.

THEOREM 2.2.28 (Uniform convergence)**.** For a finite graph, the following cost functions are uniform convergent: reachability price (Proposition 2.1.2 in [**Ber01**]), discounted price (Proposition 1.2 in [**Ber01**]), average price (Section 7.4 in [**Ber95**]), and price-per-reward average (Section 7.4 in [**Ber95**]).

2.3. Games on Finite Graphs

DEFINITION 2.3.1 (Game Graph)**.** A finite *game graph* is a tuple $\Gamma = (V, E, V_{\mathrm{Min}}, V_{\mathrm{Max}})$, where:

- (V, E) is a finite graph,
- $V_{\mathrm{Min}} \subseteq V$ is the set of vertices controlled by player Min, and
- $V_{\mathrm{Max}} \subseteq V$ is the set of vertices controlled by player Max.

We require that the sets V_{Max} and V_{Min} form a partition of the set of vertices V.

Let us fix a finite game automaton $\Gamma = (G, V_{\mathrm{Min}}, V_{\mathrm{Max}})$ for the rest of this section.

In the context of games we sometimes refer to a run as a *play* of the game. A play starts at a vertex $v_0 \in V$. If $v_0 \in V_p$, for $p \in \{\, \mathrm{Max}, \mathrm{Min} \,\}$, then player p chooses a successor of the current vertex v_0, i.e., a vertex v_1, such that $(v_0, v_1) \in E$, and v_1 becomes the new current vertex. When this happens then we say that player p has made a move from the current vertex. Players keep making moves in this fashion indefinitely thus forming an infinite run $\langle v_0, v_1, v_2, \ldots \rangle$ in the game graph.

DEFINITION 2.3.2 (Cost Game on a Finite Game Graph)**.** A *cost game* on a finite game graph is a tuple $(\Gamma, \mathsf{Cost}_{\mathrm{Min}}, \mathsf{Cost}_{\mathrm{Max}})$, where:

- $\Gamma = (V, E, V_{\mathrm{Min}}, V_{\mathrm{Max}})$ is a game graph,
- $\mathsf{Cost}_{\mathrm{Min}} : \mathrm{Runs} \to \mathbb{R}$ is a payoff function for player Min, that gives the amount Min loses in a play, and
- $\mathsf{Cost}_{\mathrm{Max}} : \mathrm{Runs} \to \mathbb{R}$ is the payoff function for player Max, that gives the amount Max wins in a play.

A cost game is called *zero-sum* if we have that $\mathsf{Cost}_{\mathrm{Min}}(r) = \mathsf{Cost}_{\mathrm{Max}}(r)$ for every play r of the game.

A *strategy* for the player Min is a (partial) function $\mu : \mathrm{Runs}_{\mathrm{fin}} \to V$ such that for every run $r = \langle v_0, v_1, \ldots, v_n \rangle$ if $v_n \in V_{\mathrm{Min}}$ then $\mu(r)$ is defined and it is such that $(v_n, \mu(r)) \in E$. Similarly we define a strategy for the player Max. We write Σ_{Min} for the set of strategies of player Min and Σ_{Max} for the set of strategies of player Max. For strategies $\mu \in \Sigma_{\mathrm{Min}}$ and $\chi \in \Sigma_{\mathrm{Max}}$, and for an initial vertex $v \in V$, we write $\mathrm{Run}(v, \mu, \chi)$ for the unique path formed if the game starts in the vertex v and then players use strategies μ and χ, respectively.

The *upper value* $\overline{\mathsf{Val}}(v)$ of a vertex $v \in V$ is defined by:

$$\overline{\mathsf{Val}}(v) = \inf_{\mu \in \Sigma_{\mathrm{Min}}} \sup_{\chi \in \Sigma_{\mathrm{Max}}} \mathsf{Cost}_{\mathrm{Min}}(\mathrm{Run}(v, \mu, \chi)),$$

and the *lower value* $\underline{\mathsf{Val}}(v)$ of a vertex $v \in V$ is defined by:

$$\underline{\mathsf{Val}}(v) = \sup_{\chi \in \Sigma_{\mathrm{Max}}} \inf_{\mu \in \Sigma_{\mathrm{Min}}} \mathsf{Cost}_{\mathrm{Max}}(\mathrm{Run}(v, \mu, \chi)).$$

We say that the game is determined if for all $v \in V$ we have $\overline{\mathsf{Val}}(v) = \underline{\mathsf{Val}}(v)$. In this case we say that the value $\mathsf{Val}(v)$ exists and it is $\mathsf{Val}(v) = \underline{\mathsf{Val}}(v) = \overline{\mathsf{Val}}(v)$. We say that the strategies $\mu_* \in \Sigma_{\mathrm{Min}}$ and $\chi_* \in \Sigma_{\mathrm{Max}}$ are *optimal* for the respective players if for all $v \in V$ we have that

$$\inf_{\mu \in \Sigma_{\mathrm{Min}}} \mathsf{Cost}_{\mathrm{Max}}(\mathrm{Run}(v, \mu, \chi_*)) = \mathsf{Val}(v) = \sup_{\chi \in \Sigma_{\mathrm{Max}}} \mathsf{Cost}_{\mathrm{Min}}(\mathrm{Run}(v, \mu_*, \chi)).$$

Let Π_{Min} and Π_{Max} be the set of positional strategies of player Min and player Max, respectively. We say that a game is *positionally determined* if players have positional optimal strategies.

If a cost game is determined, then the competitive optimisation problem for a cost game $(\Gamma, \mathsf{Cost}_{\mathrm{Min}}, \mathsf{Cost}_{\mathrm{Max}})$ is: "given a graph graph Γ, a vertex $v \in V$ and a number $D \in \mathbb{Q}$, determine whether $\mathsf{Val}(v) \leq D$."

We consider the following cost games in this chapter:

(1) *Reachability-price game.* A reachability-price game $(\Gamma, \mathsf{RP}_{\mathrm{Min}}, \mathsf{RP}_{\mathrm{Max}})$ has the following payoff functions:

$$\mathsf{RP}_{\mathrm{Min}}(r) = \mathsf{RP}_{\mathrm{Max}}(r) \quad = \quad \begin{cases} \pi_N(r) & \text{if } N = \mathsf{Stop}(r) < \infty \\ \infty & \text{otherwise.} \end{cases}$$

(2) *Discounted-price game.* For a given discount factor $\lambda \in (0, 1)$, a discounted-price game $(\Gamma, \mathsf{DP}_{\mathrm{Min}}(\lambda), \mathsf{DP}_{\mathrm{Max}}(\lambda))$ has the following payoff functions:

$$\mathsf{DP}_{\mathrm{Min}}(\lambda)(r) = \mathsf{DP}_{\mathrm{Max}}(\lambda)(r) \quad = \quad (1 - \lambda) \sum_{i=1}^{\infty} \lambda^{i-1} \pi(v_{i-1}, v_i).$$

(3) *Average-price game.* An average-price game $(\Gamma, \mathsf{AP}_{\mathrm{Min}}, \mathsf{AP}_{\mathrm{Max}})$ has the following payoff functions:

$$\mathsf{AP}_{\mathrm{Min}}(r) = \limsup_{n \to \infty} \frac{\pi_n(r)}{n} \text{ and } \mathsf{AP}_{\mathrm{Max}}(r) = \liminf_{n \to \infty} \frac{\pi_n(r)}{n}.$$

(4) *Price-per-reward-average game.* In a similar manner, a price-per-reward average game $(\Gamma, \mathsf{PRAvg}_{\mathrm{Min}}, \mathsf{PRAvg}_{\mathrm{Max}})$ has the following payoff functions:

$$\mathsf{PRAvg}_{\mathrm{Min}}(r) = \limsup_{n\to\infty} \frac{\pi_n(r)}{\varrho_n(r)} \text{ and } \mathsf{PRAvg}_{\mathrm{Max}}(r) = \liminf_{n\to\infty} \frac{\pi_n(r)}{\varrho_n(r)}.$$

The following is a well known (see, for example, [**FV97**]) result about games on finite graphs:

THEOREM 2.3.3. Reachability-price games, discounted-price games, average-price games, and price-per-reward-average games on finite game graphs are positionally determined.

The rest of this section is devoted to the proof of this theorem. Like the noncompetitive case, we solve these games by designing a set of optimality equations such that existence of a solution of the equations implies positional determinacy of the game. Moreover, a solution also characterises the value of the game at every vertex. We give a constructive proof of the existence of a solution by giving a strategy improvement algorithm, which terminates with a solution of the optimality equations.

2.3.1. Solving Reachability-price Games

Again, to keep the discussion simple, we assume the following restriction on the game graph that we consider:

ASSUMPTION 2.3.4. For every vertex $v \in V$, we have that:

(1) player Min has a strategy to reach F, and
(2) player Min does not have a strategy to have a negative average while avoiding F.

2.3.1.1. Optimality Equations

To solve reachability-price game $(\Gamma, \mathsf{RP}_{\mathrm{Min}}, \mathsf{RP}_{\mathrm{Max}})$, let us consider the following set of optimality equations:

$$(\mathcal{P}(v), \mathcal{D}(v)) \;=\; \begin{cases} (0,0)\,, & \text{if } v \in F, \\ \min^{\mathrm{lex}}_{(v,w)\in E} \{\, (\pi(v,w) + \mathcal{P}(w), 1 + \mathcal{D}(w)) \,\}, & \text{if } v \in V_{\mathrm{Min}} \setminus F, \\ \max^{\mathrm{lex}}_{(v,w)\in E} \{\, (\pi(v,w) + \mathcal{P}(w), 1 + \mathcal{D}(w)) \,\}, & \text{if } v \in V_{\mathrm{Max}} \setminus F. \end{cases}$$

We denote these equations by $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{MinMax}}(G)$. We say that the functions $p : V \to \mathbb{R}$ and $d : V \to \mathbb{N}$ are a solution of optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{MinMax}}(G)$, and we write $(p,d) \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{MinMax}}(G)$, if all equations in $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{MinMax}}(G)$ hold for the valuations $\mathcal{P}(v) \mapsto p(v)$ and $\mathcal{D}(v) \mapsto d(v)$.

The following lemma states the relation between a solution of optimality equations and value of the reachability-price game on a priced graph.

LEMMA 2.3.5 (A solution of optimality equations gives value of the game)**.** Under Assumption 2.3.4 we have that if $(p,d) \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{MinMax}}(G)$ implies $p(v) = \mathsf{Val}(v)$, for all $v \in V$.

PROOF. Let $(p, d) \models \mathcal{OE}^{\mathsf{RP}}_{\text{MinMax}}(G)$. Using this solution of the optimality equation, let us define the positional strategy μ_* of player Min as follows: for every vertex $v \in V_{\text{Min}}$, we have:

$$\mu_*(v) \in \operatorname*{argmin}^{\text{lex}}_{w \in V} \big\{ (\pi(v,w) + p(w),\, 1 + d(w)) \,:\, (v,w) \in E \big\}.$$

Similarly we define the positional strategy χ_* as follows: for every vertex $v \in V_{\text{Max}}$, we have:

$$\chi_*(v) \in \operatorname*{argmax}^{\text{lex}}_{w \in V} \big\{ (\pi(v,w) + p(w),\, 1 + d(w)) \,:\, (v,w) \in E \big\}.$$

Note that the strategies μ_* and χ_* are defined in such a way that for all $v \in V_{\text{Min}}$, we have $p(v) = \pi(v, \mu_*(v)) + p(\mu_*(v))$, and for all $v \in V_{\text{Max}}$, we have that $p(v) = \pi(v, \chi_*(v)) + p(\chi_*(v))$. It is easy to verify that $p(v) = \mathsf{RP}(\text{Run}(v, \mu_*, \chi_*)$. Now to prove the lemma, what remains to be shown is that $\mu_* \in \Sigma_{\text{Min}}$ and $\chi_* \in \Sigma_{\text{Max}}$ are optimal strategies.

The rest of the proof is in two parts. In first part we show that $p(v) \geq \mathsf{Val}(v)$ and in the second part we show that $p(v) \leq \mathsf{Val}(v)$.

– Let $\mu \in \Sigma_{\text{Min}}$ be a strategy (not necessarily positional) of player Min and let us consider the initial vertex $v \in V$. Let the run $\text{Run}(v, \mu, \chi_*)$ be $\langle v_0 = v, v_1, v_2, \ldots \rangle$. For all $i \geq 0$, if $v_i \in V_{\text{Min}}$ then we have the following inequality:

$$(p(v_i), d(v_i)) \leq^{\text{lex}} (\pi(v_i, v_{i+1}) + p(v_{i+1}), 1 + d(v_{i+1})\,,$$

as (p, d) is a solution of optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\text{MinMax}}(G)$. Moreover for all $i \geq 0$, if $v_i \in V_{\text{Max}}$ then from the definition of χ_* we get the equality:

$$(p(v_i), d(v_i)) = (\pi(v_i, v_{i+1}) + p(v_{i+1}), 1 + d(v_{i+1})\,.$$

Hence it implies that for all $i \geq 0$ we have

$$p(v_i) \leq \pi(v_i, v_{i+1}) + p(v_{i+1}). \tag{2.3.1}$$

If $\mathsf{Stop}(\text{Run}(v, \mu, \chi_*)) = \infty$ then by definition we have that $\mathsf{RP}(\text{Run}(v, \mu, \chi_*)) = \infty$, and then is inequality $p(v_i) \leq \mathsf{RP}(\text{Run}(v, \mu, \chi_*))$ is trivial. Let us assume that $\mathsf{Stop}(\text{Run}(v, \mu, \chi_*)) = n < \infty$. Summing the inequalities (2.3.1) for $0 \leq i < n$, side-wise we get:

$$p(v_0) \leq \sum_{i=0}^{\mathsf{Stop}(\text{Run}(v,\mu,\chi_*))-1} \pi(v_i, v_{i+1}) + p(v_n).$$

Since $v_n \in F$, we have that $p(v_n) = 0$; also note that $v_0 = v$. It follows that for every strategy $\mu \in \Sigma_{\text{Min}}$ we have that $p(v) \leq \mathsf{RP}(\text{Run}(v, \mu, \chi_*)$. Since $\mu_* \in \Sigma_{\text{Min}}$ we have that

$$p(v) \leq \mathsf{RP}(\text{Run}(v, \mu_*, \chi_*)) = \mathsf{Val}(v).$$

– Let $\chi \in \Sigma_{\text{Max}}$ be an arbitrary strategy of player Max and let us consider the initial vertex $v \in V$. Let the run $\text{Run}(v, \mu_*, \chi)$ be $\langle v_0 = v, v_1, v_2, \ldots \rangle$. For all $i \geq 0$, if $v_i \in V_{\text{Max}}$ then we have the following inequality:

$$(p(v_i), d(v_i)) \geq^{\text{lex}} (\pi(v_i, v_{i+1}) + p(v_{i+1}), 1 + d(v_{i+1})\,,$$

as (p, d) is a solution of optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\text{MinMax}}(G)$. Moreover for all $i \geq 0$, if $v_i \in V_{\text{Min}}$ then from the definition of μ_* we get the equality:

$$(p(v_i), d(v_i)) = (\pi(v_i, v_{i+1}) + p(v_{i+1}), 1 + d(v_{i+1})\,.$$

Hence it implies that for all $i \geq 0$ we have

$$p(v_i) \geq \pi(v_i, v_{i+1}) + p(v_{i+1}).$$

If $\mathsf{Stop}(\text{Run}(v, \mu_*, \chi)) = \infty$ then it is easy to verify that either the player Min does not have a strategy to reach F, or the strategy μ_* have negative average while avoiding F. Since both cases are not possible due to Assumption 2.3.4, we know that $\mathsf{Stop}(\text{Run}(v, \mu_*, \chi)) = n < \infty$ Now using a similar analysis to the first part, it is straightforward to verify that

$$p(v) \geq \mathsf{Val}(v).$$

∎

From the second part of the proof of the previous lemma, the following proposition is immediate.

PROPOSITION 2.3.6 (A solution of optimality equations gives positional optimal strategies for both players)**.** For a priced graph G, if there exists a solution of $\mathcal{OE}^{\mathsf{RP}}_{\text{MinMax}}(G)$ then both players have positional optimal strategies.

We prove the following lemma using a strategy improvement algorithm, which upon termination gives a solution of the optimality equations.

LEMMA 2.3.7 (Existence of a solution)**.** For every priced graph G, there exists a solution of the optimality equations $\mathcal{OE}^{\mathsf{RP}}_{\text{MinMax}}(G)$.

2.3.1.2. Solving 2-player Optimality Equations

For given functions $p : V \to \mathbb{R}$ and $d : V \to \mathbb{N}$, let us define the set $M^*(v, p, d)$ for every vertex $v \in V$ as follows:

$$M^*(v, p, d) = \operatorname*{argmax}^{\text{lex}}_{w \in V} \{(\pi(v, w) + p(w),\ 1 + d(w)) \ :\ (v, w) \in E\}.$$

Consider the strategy improvement function $\text{Improve}_{\text{Max}} : \Pi_{\text{Max}} \times [V \to \mathbb{R}] \times [V \to \mathbb{N}] \to \Pi_{\text{Max}}$ defined by:

$$\text{Improve}_{\text{Max}}(\chi, p, d)(v) = \begin{cases} \chi(v) & \text{if } \chi(v) \in M^*(v, p, d) \\ \mathsf{Choose}(M^*(v, p, d)) & \text{otherwise,} \end{cases}$$

where $p : V \to \mathbb{R}$, $d : V \to \mathbb{N}$, and $v \in V$.

PROPOSITION 2.3.8 (Fixed points of $\text{Improve}_{\text{Max}}$ are solutions of $\mathcal{OE}^{\mathsf{RP}}_{\text{MinMax}}$)**.** Let $\chi \in \Pi_{\text{Max}}$ and let $(p^\chi, d^\chi) \models \mathcal{OE}^{\mathsf{RP}}_{\text{Min}}(G{\restriction}\chi)$. If $\text{Improve}_{\text{Max}}(\chi, p^\chi, d^\chi) = \chi$ then $(p^\chi, d^\chi) \models \mathcal{OE}^{\mathsf{RP}}_{\text{MinMax}}(G)$.

Input: Priced Graph $G = ((V, E, F), \pi)$
Output: A solution of $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{MinMax}}(G)$

```
begin
    (Initialisation). Choose an arbitrary positional strategy χ_0 ∈ Π_Max;
    Set i := 0;
    repeat
        (Value Computation). Compute the solution (p_i, d_i) of OE^RP_Min(G↾χ_i).;
        (Strategy Improvement). Compute χ_{i+1} = Improve_Max(χ_i, p_i, d_i);
        Set i := i + 1;
    until χ_{i+1} ≡ χ_i ;
    return (p_i, d_i);
end
```

FIGURE 2.6. Strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{MinMax}}(G)$.

Notice that for every positional strategy $\chi \in \Pi_{\mathrm{Max}}$ of player Max, in the graph $G{\restriction}\chi_i$ player Max does not have any choice and hence for the sake of notational simplicity we can assume that all the vertices in the graph $G{\restriction}\chi$ belong to player Min and hence Algorithm 2.2 can be used to solve $\mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G{\restriction}\chi)$.

We say that $(p, d) \models \mathcal{OE}^{\mathsf{RP}}_{\leq}(G)$, if

$$(p(v), d(v)) \leq^{\mathrm{lex}} \begin{cases} (0,0), & \text{if } v \in F, \\ \min^{\mathrm{lex}}_{w \in V} \left\{ \left(\pi(v,w) + p(w), 1 + d(w)\right) \,:\, (v,w) \in E \right\}, & \text{otherwise.} \end{cases}$$

PROPOSITION 2.3.9. Let $p, p_{\leq} : V \to \mathbb{R}$ and $d, d_{\leq} : V \to \mathbb{N}$ be such that $(p, d) \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ and $(p_{\leq}, d_{\leq}) \models \mathcal{OE}^{\mathsf{RP}}_{\leq}(G)$. Then we have $(p_{\leq}, d_{\leq}) \leq^{\mathrm{lex}} (p, d)$, and if $(p_{\leq}, d_{\leq}) \not\models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G)$ then $(p_{\leq}, d_{\leq}) <^{\mathrm{lex}} (p, d)$.

PROOF. The proof is similar to that of Proposition 2.2.11. ∎

LEMMA 2.3.10 (Strict strategy improvement for player Max)**.** Let strategies $\chi, \chi' \in \Pi_{\mathrm{Max}}$, let $(p, d) \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G{\restriction}\chi)$ and $(p', d') \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G{\restriction}\chi')$, and let $\chi' = \mathrm{Improve}_{\mathrm{Max}}(\chi, p, d)$. Then we have that $(p, d) \leq^{\mathrm{lex}} (p', d')$ and if $\chi' \neq \chi$ then $(p, d) <^{\mathrm{lex}} (p', d')$.

PROOF. First we argue that $(p, d) \models \mathcal{OE}^{\mathsf{RP}}_{\leq}(G{\restriction}\chi')$, which by Proposition 2.3.9 implies that $(p, d) \geq^{\mathrm{lex}} (p', d')$. Indeed, for every $v \in V \setminus F$ if $\chi(v) = w$ and $\chi'(v) = w'$ then we have

$$\begin{aligned} (p(v), d(v)) &= \left(\pi(v,w) + p(w), 1 + d(w)\right), \text{ as } (p, d) \models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G{\restriction}\chi) \\ &\leq^{\mathrm{lex}} \left(\pi(v,w') + p(w'), 1 + d(w')\right), \text{ by the definition of } \mathrm{Improve}_{\mathrm{Max}}. \end{aligned}$$

Moreover, if $\chi \neq \chi'$ then there is a $v \in V \setminus F$ for which the above inequality is strict. Then $(p, d) \not\models \mathcal{OE}^{\mathsf{RP}}_{\mathrm{Min}}(G{\restriction}\chi')$, because every vertex in $G{\restriction}\chi'$ has a unique successor, and hence by Proposition 2.3.9 we conclude that $(p, d) <^{\mathrm{lex}} (p', d')$. ∎

LEMMA 2.3.11 (Correctness and termination of strategy improvement algorithm)**.** The strategy improvement algorithm for $\mathcal{OE}^{\mathsf{RP}}_{\text{MinMax}}$ terminates in finitely many steps and returns a solution (p, d) of $\mathcal{OE}^{\mathsf{RP}}_{\text{MinMax}}$.

2.3.2. Solving Discounted-price Games

2.3.2.1. Optimality Equations

To solve a discounted-price game $(\Gamma, \mathsf{DP}_{\text{Min}}(\lambda), \mathsf{DP}_{\text{Max}}(\lambda))$ with a discount factor $\lambda \in (0, 1)$, let us consider the following set of optimality equations.

$$\mathcal{D}(v) = \begin{cases} \min_{(v,w)\in E} \{(1-\lambda)\cdot \pi(v,w) + \lambda \cdot \mathcal{D}(w)\}, & \text{if } v \in V_{\text{Min}}, \\ \max_{(v,w)\in E} \{(1-\lambda)\cdot \pi(v,w) + \lambda \cdot \mathcal{D}(w)\}, & \text{if } v \in V_{\text{Max}}. \end{cases}$$

We denote these equations by $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$. We say that a functions $d : V \to \mathbb{R}$ is a solution of optimality equations $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$, and we write $d \models \mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$, if all equations in $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$ hold for the valuations $\mathcal{D}(v) \mapsto d(v)$.

The following lemma states the relation between a solution of optimality equations and the value of the discounted-price game on a priced graph.

LEMMA 2.3.12 (A solution of optimality equations gives value of the game)**.** If $d : V \to \mathbb{R}$ is a solution of $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$ then for all $v \in V$ we have that $d(v) = \mathsf{Val}(v)$.

PROPOSITION 2.3.13 (A solution of optimality equations gives positional optimal strategies for both players)**.** For a priced graph G and discount factor $\lambda \in (0, 1)$, if there exists a solution of $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$ then both players have positional optimal strategies.

2.3.2.2. Existence of a solution

LEMMA 2.3.14 (Existence of a solution)**.** For every priced graph G and discount factor $\lambda \in (0, 1)$, there exists a solution of the optimality equations $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$.

We prove this lemma using the following strategy improvement algorithm, which terminates with a solution of the optimality equations.

2.3.2.3. Solving 2-player Optimality Equations

For given function $d : V \to \mathbb{R}$, let us define the set $M^*(v, d)$ for every vertex $v \in V$ as follows:

$$M^*(v, d) = \operatorname*{argmax}_{w\in V} \{(1-\lambda)\cdot \pi(v,w) + \lambda \cdot d(w) \,:\, (v,w) \in E\}.$$

Consider the strategy improvement function $\text{Improve}_{\text{Max}} : \Pi_{\text{Max}} \times [V \to \mathbb{R}] \to \Pi_{\text{Max}}$ defined by:

$$\text{Improve}_{\text{Max}}(\chi, d)(v) = \begin{cases} \chi(v) & \text{if } \chi(v) \in M^*(v, d) \\ \mathsf{Choose}(M^*(v, d)) & \text{otherwise.} \end{cases}$$

where $d : V \to \mathbb{R}$ and $v \in V$.

PROPOSITION 2.3.15 (Fixed point of $\text{Improve}_{\text{Max}}$ are solutions of $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$)**.** Let $\chi \in \Pi_{\text{Max}}$ and let $(p^\chi, d^\chi) \models \mathcal{OE}^{\mathsf{DP}}_{\text{Min}}(G\restriction\chi, \lambda)$. If $\text{Improve}_{\text{Max}}(\chi, d^\chi) = \chi$ then we have $(p^\chi, d^\chi) \models \mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$.

A strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$ is given as Algorithm 2.7.

Input: Priced graph $G = ((V, E, F), \pi)$ and $\lambda \in (0, 1)$
Output: A solution of $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$

```
begin
    (Initialisation). Choose an arbitrary positional strategy χ_0 ∈ Π_Max;
    Set i := 0;
    repeat
        (Value Computation). Compute the solution d_i of OE^DP_Min(G↾χ_i, λ).;
        (Strategy Improvement). Compute χ_{i+1} = Improve_Max(χ_i, d_i);
        Set i := i + 1;
    until χ_{i+1} ≡ χ_i ;
    return d_i;
end
```

FIGURE 2.7. Strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$.

Proofs of the following lemmas concerning the correctness of the algorithm are routine, and omitted.

LEMMA 2.3.16 (Strict strategy improvement for player Max)**.** Let $\chi, \chi' \in \Pi_{\text{Max}}$, let $d \models \mathcal{OE}^{\mathsf{DP}}_{\text{Min}}(G\restriction\chi, \lambda)$ and $d' \models \mathcal{OE}^{\mathsf{DP}}_{\text{Min}}(G\restriction\chi', \lambda)$, and let $\chi' = \text{Improve}_{\text{Max}}(\chi, d)$. Then $d \leq d'$ and if $\chi' \neq \chi$ then $d < d'$.

LEMMA 2.3.17 (Correctness and termination of strategy improvement algorithm)**.** The strategy improvement algorithm for $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$ terminates in finitely many steps and returns a solution $d : V \to \mathbb{R}$ of $\mathcal{OE}^{\mathsf{DP}}_{\text{MinMax}}(G, \lambda)$.

2.3.3. Solving Average-price and Price-per-reward Average Games

2.3.3.1. Optimality Equations

To solve price-per-reward average game $(\Gamma, \mathsf{PRAvg}_{\text{Min}}, \mathsf{PRAvg}_{\text{Max}})$ on a price-reward game graph, let us consider the following set of optimality equations.

$$(\mathcal{G}(v), \mathcal{B}(v)) = \begin{cases} \min^{\text{lex}}_{(v,w)\in E} \{ (\mathcal{G}(w),\ \pi(v,w) - \varrho(v,w)\cdot\mathcal{G}(v) + \mathcal{B}(w)) \}, & \text{if } v \in V_{\text{Min}}, \\ \max^{\text{lex}}_{(v,w)\in E} \{ (\mathcal{G}(w),\ \pi(v,w) - \varrho(v,w)\cdot\mathcal{G}(v) + \mathcal{B}(w)) \}, & \text{if } v \in V_{\text{Max}}. \end{cases}$$

We denote these equations by $\mathcal{OE}^{\mathsf{PRAvg}}_{\text{MinMax}}(G)$. We say that the functions $g : V \to \mathbb{R}$ and $b : V \to \mathbb{R}$ are a solution of optimality equations $\mathcal{OE}^{\mathsf{PRAvg}}_{\text{MinMax}}(G)$, and we write $(g, b) \models \mathcal{OE}^{\mathsf{PRAvg}}_{\text{MinMax}}(G)$, if all equations in $\mathcal{OE}^{\mathsf{PRAvg}}_{\text{MinMax}}(G)$ hold for the valuations $\mathcal{G}(v) \mapsto g(v)$ and $\mathcal{B}(v) \mapsto b(v)$.

The following lemma states the relation between a solution of optimality equations and value of the price-per-reward average game on a price-reward graph.

LEMMA 2.3.18 (A solution of optimality equations gives value of the game)**.** If $(g, b) \models \mathcal{OE}^{\mathsf{PRAvg}}_{\text{MinMax}}(G)$ then for all $v \in V$ we have that $g(v) = \mathsf{Val}(v)$.

PROPOSITION 2.3.19 (A solution of optimality equations gives positional optimal strategies for both players)**.** For a price-reward graph G, if there exists a solution of $\mathcal{OE}^{\mathsf{PRAvg}}_{\text{MinMax}}(G)$ then both players have positional optimal strategies.

2.3.3.2. Existence of a solution

LEMMA 2.3.20 (Existence of a solution)**.** For every price-reward graph G, there exists a solution of the optimality equations $\mathcal{OE}^{\mathsf{PRAvg}}_{\text{MinMax}}(G)$.

We prove this lemma using the following strategy improvement algorithm, which terminates with a solution of the optimality equations.

2.3.3.3. Solving 2-player Optimality Equations

For given functions $g : V \to \mathbb{R}$ and $b : V \to \mathbb{R}$, let us define the set $M^*(v, g, b)$ for every vertex $v \in V$ as follows:

$$M^*(v, g, b) = \operatorname*{argmax}^{\text{lex}}_{w\in V} \{ (g(w),\ \pi(v,w) - \varrho(v,w)\cdot g(v) + b(w)) \ :\ (v,w) \in E \}.$$

Consider the strategy improvement function $\text{Improve}_{\text{Max}} : \Pi_{\text{Max}} \times [V \to \mathbb{R}] \times [V \to \mathbb{R}] \to \Pi_{\text{Max}}$ defines by:

$$\text{Improve}_{\text{Max}}(\chi, g, b)(v) = \begin{cases} \chi(v) & \text{if } \chi(v) \in M^*(v, g, b) \\ \mathsf{Choose}(M^*(v, g, b)) & \text{otherwise.} \end{cases}$$

where $g : V \to \mathbb{R}$, $b : V \to \mathbb{R}$, and $v \in V$.

PROPOSITION 2.3.21 (Fixed point of $\text{Improve}_{\text{Max}}$ are solutions of $\mathcal{OE}^{\mathsf{PRAvg}}_{\text{MinMax}}$)**.** Let $\chi \in \Pi_{\text{Max}}$ and let $(g^\chi, b^\chi) \models \mathcal{OE}^{\mathsf{PRAvg}}_{\text{Min}}(G{\restriction}\chi)$. If $\text{Improve}_{\text{Max}}(\chi, g^\chi, b^\chi) = \chi$ then $(g^\chi, b^\chi) \models \mathcal{OE}^{\mathsf{PRAvg}}_{\text{MinMax}}(G)$.

The strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{MinMax}}(G)$ is shown as Algorithm 2.8 and its correctness follows from the following lemmas.

Input: Price-reward Graph $G = ((V, E, F), \pi, \varrho)$
Output: A solution of $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{MinMax}}(G)$

```
begin
    (Initialisation). Choose an arbitrary positional strategy χ_0 ∈ Π_Max;
    Set i := 0;
    repeat
        (Value Computation). Compute the solution (g_i, b_i) of OE^PRAvg_Min(G↾χ_i).;
        (Strategy Improvement). Compute χ_{i+1} = Improve_Max(χ_i, g_i, b_i);
        Set i := i + 1;
    until χ_{i+1} ≡ χ_i ;
    return (g_i, b_i);
end
```

FIGURE 2.8. Strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{MinMax}}(G)$.

LEMMA 2.3.22 (Strict strategy improvement for Max)**.** Let the strategies $\chi, \chi' \in \Pi_{\mathrm{Max}}$, let $(g, b) \models \mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G\restriction\chi)$ and $(g', b') \models \mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{Min}}(G\restriction\chi')$, and let $\chi' = \mathrm{Improve}_{\mathrm{Max}}(\chi, g, b)$. Then $(g, b) \leq^{\mathrm{lex}} (g', b')$ and if $\chi' \neq \chi$ then $(g, b) <^{\mathrm{lex}} (g', b')$.

LEMMA 2.3.23 (Correctness and termination of strategy improvement algorithm)**.** The strategy improvement algorithm for $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{MinMax}}$ terminates in finitely many steps and returns a solution (g, b) of $\mathcal{OE}^{\mathsf{PRAvg}}_{\mathrm{MinMax}}$.

2.4. Discussion

In this chapter we presented algorithms to solve optimisation problems and two-player games on finite graphs. We demonstrated that dynamic programming techniques are not only instrumental in designing algorithms, but also quite helpful in proving theorems (for example, proofs of positional determinacy).

The central theme of this thesis is competitive optimisation problems on timed automata. There are two natural ways in which we can attempt to extend the solution techniques introduced in this chapter—for competitive problems on finite graphs—to solve competitive optimisation problems on infinite graph of configurations of timed automata.

The First approach is to design optimality equations for the infinite graph of configurations of timed automata, and then use iterative algorithms to solve those equations. As we shall see later in Chapter 5, we took this approach to solve reachability-time games on timed automata.

The second approach is to reduce the competitive optimisation problem on timed automata to the competitive optimisation problem on certain finite graph, and then use the algorithms presented in this chapter to solve those problems. In the next chapter we

introduce boundary region graph, an abstraction of timed automata, which is quite useful for this purpose. In Chapter 4 we show that a number of optimisation problems on timed automata can be reduced to corresponding optimisation problems on a finitary sub-graph of their boundary region graph. In Chapter 6 we show that the problem of solving average-time games on timed automata can be reduced to the problem of solving average price games on a finitary sub-graph of their boundary region graph.

3

Timed Automata

> For compositional reasoning, we should be able to define the semantics of a component and prove its properties without knowing the details of the other parts of the system. Shifting to dense-time semantics offers a natural and mathematically clean way to this effect.
>
> Rajeev Alur

Mathematical formalisms to model real-time systems can broadly be divided in two categories: discrete-time models and dense-timed models. In discrete-time models the variables measuring time are (non-negative) integer-valued, while in dense-time models the variables measuring time are (non-negative) real-valued. To model a real-time system with discrete-time models we are forced to fix a time-unit in advance and to ignore the relative order of two events within a unit of time. If we choose a small enough time-unit then meaningful results can be obtained using discrete-time models. However, probably, the strongest case against discrete-time models is that if a system is to be composed of several components, the semantics of each component can not be given independently. Dense-time models handles this issue elegantly as we are not required to fix a unit of time in advance. Timed automata **[AD90]** are dense-time computational models for real-time systems.

This chapter introduces timed automata and some of its extensions and sets notations and mathematical shorthands that are used consistently in the rest of the thesis. This chapter also introduces some abstractions—including a novel boundary region graph abstraction—of timed automata, which are useful in solving competitive optimisation problems on the infinite graphs of configurations of timed automata. Before we present formal definitions of timed automata, let us give two short examples of modelling real-time systems using timed automata.

3.1. Examples

Let us consider a light bulb [**UPP**] with a press-button switch which has two modes of operation: dim and bright. If the bulb is in the *off* mode and we press the switch, the light bulb lights with a *dim* light. If we press the switch again within a pre-specified unit of time, say one second, the light bulb lights with *bright* light. However, if the bulb is dimly lit and we press the switch after one second the light-bulb turns off. Once the light-bulb is brightly lit and switch is pressed the light-bulb turns off.

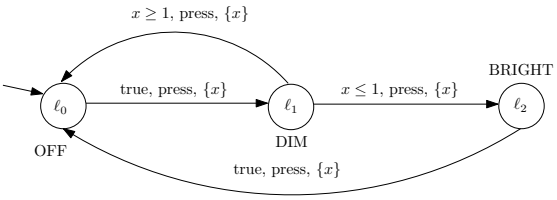


FIGURE 3.1. A light-bulb modelled using a timed automaton.

A straightforward model of such a light bulb is shown in Figure 3.1 as a timed automaton: a finite state transition system with a finite set of continuous variables called clocks. Clocks can appear as guards on the transitions where they can be compared against integers and after taking a transition it is possible to reset them to 0. When drawing a timed automaton we annotate a transition with the triple : (clock guard, action name, set of clocks to be reset). All the clock variables evolve continuously at uniform rate. Since it is possible to reset clock variables after taking a transition, timed automata can express complex timing behaviours of the real-time systems (e.g., a clock can remember the time since a particular action was fired).

Let us explain the timed automaton shown in Figure 3.1. According to this model, the light bulb system can be in one of the three locations: ℓ_0 where the light bulb is *off*, ℓ_1 where the light bulb is dimly lit, and ℓ_2 where the light bulb is brightly lit. The variable x appearing in the figure is the only clock used in this example and it measures the time spent in the location ℓ_1. The explanations for the rest of the example is straightforward.

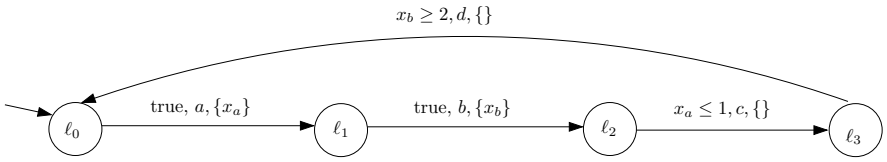


FIGURE 3.2. A timed automaton with more than one clock.

The light bulb example used only one clock, but there are systems whose modelling requires multiple clocks. As an example consider the timed automaton in Figure 3.2 [**AD94**]. This automaton models a system that repeatedly executes a sequence of actions a, b, c, and d, and in that order. The timing constraint on the events is such that the action c must follow

the action a within one time unit and the action d must be executed after more than 2 time-units from the previous action b. Notice that such system can not be modelled using a single clock.

3.2. Formal Definition

In this section we formally introduce timed automata and related concepts. This section is organised as follows. In the first subsection we introduce clocks, clock valuations, regions and zones. In Subsection 3.2.2 we give the definition of timed automata and introduce some key notations. Our definition of a timed automaton may appear to differ from the usual ones [**AD94, Bou06**]. The differences are, however, superficial and mostly syntactic. Finally, in Subsection 3.2.3, we introduce the concept of runs and strategies in the context of a timed automaton.

3.2.1. Clocks, Valuations, Regions, and Zones

Fix a constant $\mathcal{K} \in \mathbb{N}$ for the rest of the thesis. Let C be a finite set of *clocks*. Clocks in timed automata are usually allowed to take arbitrary non-negative real values. For the sake of simplicity, we restrict clocks to be bounded by some constant $\mathcal{K} \in \mathbb{N}$, that is we consider only *bounded* timed automata models.

A ($\mathcal{K}$-bounded) *clock valuation* is a function $\nu : C \to [\![\mathcal{K}]\!]_{\mathbb{R}}$; we write $\mathcal{V}_C$ for the set $[C \to [\![\mathcal{K}]\!]_{\mathbb{R}}]$ of clock valuations over C. If $\nu \in \mathcal{V}_C$ and $t \in \mathbb{R}_{\oplus}$ then we write $\nu + t : C \to \mathbb{R}_{\oplus}$ for the function defined by $(\nu + t)(c) = \nu(c) + t$, for all $c \in C$. Notice that $(\nu + t)$ may not be in the set $\mathcal{V}_C$ of bounded clock valuation if $\nu(c) + t > \mathcal{K}$ for some $c \in C$. For a set $C' \subseteq C$ of clocks and a clock valuation $\nu \in \mathcal{V}_C$, we define $\mathsf{Reset}(\nu, C')(c) = 0$ if $c \in C'$, and $\mathsf{Reset}(\nu, C')(c) = \nu(c)$ if $c \notin C'$. A *corner* is an integer clock valuation, i.e., α is a corner if $\alpha(c) \in [\![\mathcal{K}]\!]_{\mathbb{N}}$ for every clock $c \in C$. If $\nu \in \mathcal{V}_C$ is a clock valuation then we write $\lfloor \nu \rfloor$ for the corner defined by $\lfloor \nu \rfloor(c) = \lfloor \nu(c) \rfloor$.

EXAMPLE 3.2.1 (Clock valuations and reset functions)**.** Let us consider a timed automaton with three clocks x, y, z, i.e., $C = \{x, y, z\}$. Let us assume that for some valuation ν the value of clock x, y, and z are 1.2, 2.3, and 1.333, respectively, then we display this clock valuation as $\nu = \{x = 1.2,\ y = 2.3,\ z = 1.333\}$. The corner valuation $\lfloor \nu \rfloor$ of the valuation ν is the clock valuation $\{x = 1,\ y = 2,\ z = 1\}$. For the clock valuation ν and a set of clocks $C' = \{y\}$ the function $\mathsf{Reset}(\nu, C')$ returns the clock valuation $\nu' = \{x = 1.2,\ y = 0,\ z = 1.333\}$.

The set of *clock constraints* over the set of clocks C is the set of conjunctions of *simple clock constraints* , which are constraints of the form $c \bowtie i$ or $c - c' \bowtie i$, where $c, c' \in C$, $i \in [\![\mathcal{K}]\!]_{\mathbb{N}}$, and $\bowtie \in \{ <, >, =, \leq, \geq \}$. There are finitely many simple clock constraints. For every clock valuation $\nu \in \mathcal{V}_C$, let $\mathsf{SCC}(\nu)$ be the set of simple clock constraints which hold in $\nu \in \mathcal{V}_C$.

A *clock region* is a maximal set $P \subseteq \mathcal{V}_C$, such that for all $\nu, \nu' \in P$, $\mathsf{SCC}(\nu) = \mathsf{SCC}(\nu')$. In other words, every clock region is an equivalence class of the indistinguishability-by-clock-constraints relation, and vice versa. Note that ν and ν' are in the same clock region iff all clocks have the same integer parts in ν and ν', and if the partial orders of the clocks,

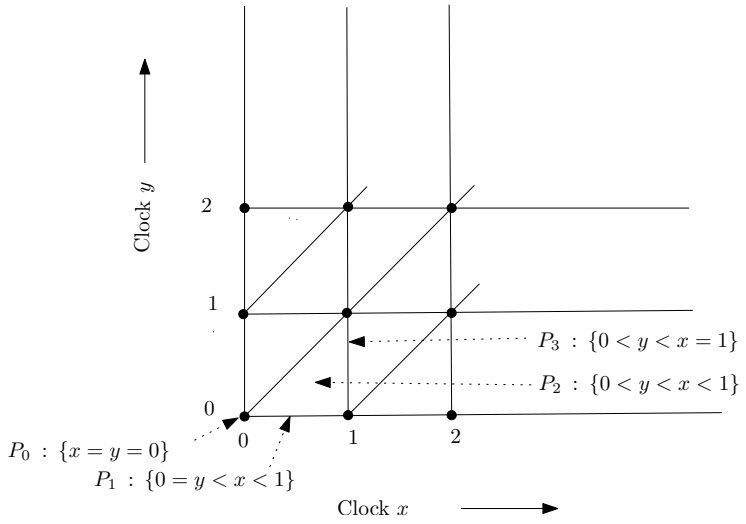


FIGURE 3.3. Clock regions of a timed automaton.

determined by their fractional parts in ν and ν', are the same. For all $\nu \in \mathcal{V}_C$, we write $[\nu]$ for the clock region of ν.

EXAMPLE 3.2.2 (Clock regions)**.** The set of the clock valuation with two clocks, x and y, and bound $\mathcal{K} = 2$ is illustrated in Figure 3.3. Notice that all the intersection points (e.g., clock region P_0), open line segments (e.g., clock region P_1 and P_3), and open areas (e.g., clock region P_2) are different clock regions. Total number of regions in this example is 33 (9 intersection points, 16 open line segments and 8 triangular open areas).

A *clock zone* is a convex set of clock valuations, which is a union of a set of clock regions. We write $\mathcal{Z}$ for the set of clock zones. Note that a set of clock valuations is a zone iff it is definable by a clock constraint. For $W \subseteq \mathcal{V}_C$, we write $\mathsf{clos}(W)$ for the smallest closed set in $\mathcal{V}_C$ which contains W. Observe that for every clock zone W, the set $\mathsf{clos}(W)$ is also a clock zone.

Let L be a finite set of *locations* . A *configuration* is a pair (ℓ, ν), where $\ell \in L$ is a location and $\nu \in \mathcal{V}_C$ is a clock valuation; we write Q for the set of configurations. If $s = (\ell, \nu) \in Q$ and $c \in C$, then we write $s(c)$ for $\nu(c)$.

A *region* is a pair (ℓ, P), where ℓ is a location and P is a clock region. If $s = (\ell, \nu)$ is a configuration then we write $[s]$ for the region $(\ell, [\nu])$. A region $R = (\ell, P)$ corresponds to a set of configurations $\{(\ell, \nu) : [\nu] = P\}$, hence sometimes we write $s \in R$ as a shorthand for $[s] = R$. We write $\mathcal{R}$ for the set of regions. A set $Z \subseteq Q$ is a *zone* if for every $\ell \in L$, there is a clock zone W_ℓ (possibly empty), such that $Z = \{(\ell, \nu) : \ell \in L \text{ and } \nu \in W_\ell\}$. For a region $R = (\ell, P) \in \mathcal{R}$, we write $\mathsf{clos}(R)$ for the zone $\{(\ell, \nu) : \nu \in \mathsf{clos}(P)\}$.

3.2.2. Timed Automata

We are now in a position to introduce a timed automaton.

DEFINITION 3.2.3 (Timed Automata)**.** A *timed automaton* is a tuple $\mathcal{T} = (L, C, S, A, E, \delta, \xi, F)$, where:

- L is a finite set of *locations*,
- C is a finite set of *clocks*,
- $S \subseteq L \times \mathcal{V}_C$ is a set of *states* [1],
- A is a finite set of *actions*,
- $E : A \to 2^S$ is an *action enabledness function*,
- $\delta : L \times A \to L$ is a *transition function*,
- $\xi : A \to 2^C$ is a *clock reset function*, and
- $F \subseteq S$ is a set of *final states*.

It is required that S, F, and $E(a)$ for all $a \in A$, are zones.

Clock zones, from which zones S, F, and $E(a)$, for all $a \in A$, are built, are typically specified by clock constraints. Therefore, when we consider a timed automaton as an input of an algorithm, its size should be understood as the sum of sizes of encodings of L, C, A, δ, and ξ, and the sizes of encodings of clock constraints defining zones S, F, and $E(a)$, for all $a \in A$.

Remark. In standard definitions [**Alu91, Bou09**] of timed automata, transitions are allowed to be non-deterministic, i.e., the transition function has the form $\delta : L \times A \to 2^L$. However, in our work timed automata are not used as formal language acceptors, but rather as generators of (classes of) transition systems. It is the game structure (i.e., the partition of states for the two players) that is in fact introducing two forms of non-deterministic choices, sometimes referred to, e.g., as "angelic" and "demonic" non-determinism. In this context working with "deterministic" automata is more meaningful because it does not introduce an unnecessary "third" form of non-determinism. Since we never interpret automata as formal language acceptors, restriction to "deterministic" automata is hence without loss of generality and avoids unnecessary confusion.

For a configuration $s = (\ell, \nu) \in Q$ and $t \in \mathbb{R}_\oplus$, we define $s + t$ to be the configuration $s' = (\ell, \nu + t)$ if $\nu + t \in \mathcal{V}_C$, and we then write $s \rightharpoonup_t s'$. We write $s \to_t s'$ if $s \rightharpoonup_t s'$ and for all $t' \in [0, t]$, we have $(\ell, \nu + t') \in S$. For an action $a \in A$, we define $\mathsf{Succ}(s, a)$ to be the configuration $s' = (\ell', \nu')$, where $\ell' = \delta(\ell, a)$ and $\nu' = \mathsf{Reset}(\nu, \xi(a))$, and we then write $s \overset{a}{\rightharpoonup} s'$. We write $s \overset{a}{\to} s'$ if $s \overset{a}{\rightharpoonup} s'$; $s, s' \in S$; and $s \in E(a)$.

ASSUMPTION 3.2.4. For technical convenience, and without loss of generality, we will assume throughout that for every $s \in S$, there exists $a \in A$, such that $s \overset{a}{\to} s'$.

For $s, s' \in S$, we say that s' is in the future of s, or equivalently, that s is in the past of s', if there is $t \in \mathbb{R}_\oplus$, such that $s \to_t s'$; we then write $s \to_* s'$.

[1] The set of states is usually specified as an *invariant function* [**HNSY92**] $\mathcal{I} : L \to \mathcal{Z}$ on the locations such that $S = \{(\ell, \nu) : \nu \in \mathcal{I}(\ell)\}$.

3.2.3. Timed Actions, Runs, and Strategies

A *timed action* is a pair $\tau = (t, a) \in \mathbb{R}_{\oplus} \times A$. For $s \in Q$, we define $\mathsf{Succ}(s, \tau) = \mathsf{Succ}(s, (t, a))$ to be the configuration $s' = \mathsf{Succ}(s + t, a)$, i.e., such that $s \rightharpoonup_t s'' \xrightarrow{a} s'$, and we then write $s \xrightarrow{a}_t s'$. We write $s \xrightarrow{a}_t s'$ if $s \rightarrow_t s'' \xrightarrow{a} s'$, and we then say that $(s, (t, a), s')$ is a *transition* of the timed automaton. If $\tau = (t, a)$ then we write $s \xrightarrow{\tau} s'$ instead of $s \xrightarrow{a}_t s'$, and $s \xrightarrow{\tau} s'$ instead of $s \xrightarrow{a}_t s'$.

A finite run of a timed automaton $\mathcal{T}$ is a sequence

$$\langle s_0, \tau_1, s_1, \tau_2, \ldots, \tau_n, s_n \rangle \in S \times ((\mathbb{R}_{\oplus} \times A) \times S)^*,$$

such that for every positive integer $i \leq n$ we have that (s_{i-1}, τ_i, s_i) is a transition, i.e., $s_{i-1} \xrightarrow{\tau_i} s_i$. For a finite run $r = \langle s_0, \tau_1, s_1, \tau_2, \ldots, \tau_n, s_n \rangle$ we define $\mathsf{Length}(r) = n$ and we define $\mathsf{Last}(r) = s_n$ to be the state in which the run ends. We write $\mathrm{Runs}_{\mathrm{fin}}$ for the set of finite runs, and $\mathrm{Runs}_{\mathrm{fin}}(s)$ for the set of finite runs starting from state $s \in S$.

An infinite run of a timed automaton $\mathcal{T}$ is a sequence

$$\langle s_0, \tau_1, s_1, \tau_2, \ldots \rangle \in S \times ((\mathbb{R}_{\oplus} \times A) \times S)^{\omega},$$

such that for all $i \geq 1$, we have $s_{i-1} \xrightarrow{\tau_i} s_i$. For an infinite run r, we define $\mathsf{Length}(r) = \infty$. For a run $r = \langle s_0, \tau_1, s_1, \tau_2, \ldots \rangle$, we define $\mathsf{Stop}(r) = \inf\{i : s_i \in F\}$. We write Runs for the set of infinite runs, and $\mathrm{Runs}(s)$ for the set of infinite runs starting from state $s \in S$. For a run $r = \langle s_0, \tau_1, s_1, \tau_2, \ldots \rangle$, we define time of the run as $\mathsf{Time}(r) = \sum_{i=1}^{\mathsf{Length}(r)} t_i$.

A *strategy* is a function $\sigma : \mathrm{Runs}_{\mathrm{fin}} \to \mathbb{R}_{\oplus} \times A$, such that if $\mathsf{Last}(r) = s \in S$ and $\sigma(r) = \tau$ then $s \xrightarrow{\tau} s'$. We write Σ for the set of strategies. A run according to a strategy σ from a state $s \in S$ is the unique run $\mathrm{Run}(s, \sigma) = \langle s_0, \tau_1, s_1, \tau_2, \ldots \rangle$, such that $s_0 = s$, and for every $i \geq 1$, we have $\sigma(\mathrm{Run}_i(s, \sigma)) = \tau_{i+1}$, where $\mathrm{Run}_i(s, \sigma) = \langle s_0, \tau_1, s_1, \ldots, s_{i-1}, \tau_i, s_i \rangle$.

We say that a strategy σ is *positional* if for all finite runs $r, r' \in \mathrm{Runs}_{\mathrm{fin}}$, we have that $\mathsf{Last}(r) = \mathsf{Last}(r')$ implies $\sigma(r) = \sigma(r')$. A positional strategy can be then represented as a function $\sigma : S \to \mathbb{R}_{\oplus} \times A$, which uniquely determines the strategy $\sigma^{\infty} \in \Sigma$ as follows: $\sigma^{\infty}(r) = \sigma(\mathsf{Last}(r))$, for all finite runs $r \in \mathrm{Runs}_{\mathrm{fin}}$. We write Π for the sets of positional strategies.

3.3. Some Properties of Regions

Let us recall that a *clock region* is an equivalence class of the indistinguishability-by-clock-constraints relation, while a *region* is a pair of a location and a clock region. We write $\mathcal{R}$ for the set of regions of a timed automaton $\mathcal{T}$. In this section we discuss some properties of regions and we begin by defining *fractional signature* and *cellular signature* of a clock valuation.

3.3.1. Fractional Signature and Cellular Signature

For a clock valuation ν we define its *fractional signature* $\wr \nu \wr$ to be the sequence $(f_0, f_1, \ldots, f_m)$, such that $f_0 = 0$, $f_i < f_j$ if $i < j$, for all $i, j \leq m$, and $f_1, f_2, \ldots, f_m$ are all the non-zero

fractional parts of clock values in the clock valuation ν. In other words, for every $i \geq 1$, there is a clock c, such that $\wr\nu(c)\int = f_i$, and for every clock $c \in C$, there is $i \leq m$, such that $\wr\nu(c)\int = f_i$. Let $(f_0, f_1, \ldots, f_m)$ be the fractional signature $\wr\nu\int$. Then the *cellular signature* $\Vdash\nu\Vdash$ of a clock valuation ν is defined to be the partition $(D^0, D^1, \ldots, D^m)$ of the set C of clocks, such that $D^i = \{c \in C : \wr\nu(c)\int = f_i\}$, for all $i \leq m$. We then define $\mathsf{Cell}(\nu, c)$ to be the number i such that $c \in D^i$.

EXAMPLE 3.3.1 (fractional and cellular signatures)**.** Let us consider clock valuations $\nu = \{x = 1.2,\ y = 2.3,\ z = 1.333\}$ and $\nu' = \{x = 1.2,\ y = 0,\ z = 1.333\}$ from Example 3.2.1. The fractional signature of ν is $\wr\nu\int = (0,\ 0.2,\ 0.3,\ 0.333)$, and the fractional signature of ν' is $\wr\nu\int = (0,\ 0.2,\ 0.333)$. The cellular signature of ν is $\Vdash\nu\Vdash = (\emptyset, \{x\}, \{y\}, \{z\})$, while the cellular signature of ν' is $\Vdash\nu'\Vdash = (\{y\}, \{x\}, \{z\})$.

An equivalent definition of a clock region is given in terms of cellular signature: a *clock region* is a pair (α, D), where α is a corner and D is a cellular signature. For a clock valuation $\nu \in \mathcal{V}_C$, we write $[\nu]$ for the clock region $(\lfloor\nu\rfloor, \Vdash\nu\Vdash)$.

EXAMPLE 3.3.2. We can write the clock region of the clock valuation $\nu = \{x = 1.2,\ y = 2.3,\ z = 1.333\}$ as $(\{x = 1,\ y = 2,\ z = 1\}, (\emptyset, \{x\}, \{y\}, \{z\}))$. Clock region P_2 from Figure 3.3 can be written as $(\{x = 0, y = 0\}, (\emptyset, \{y\}, \{x\}))$, while the clock region P_3 can be written as $(\{x = 1, y = 0\}, (\{x\}, \{y\}))$.

3.3.2. Thick and Thin Regions

For $R, R' \in \mathcal{R}$, we say that R' is in the future of R, or that R is in the past of R', if for all $s \in R$, there is $s' \in R'$, such that s' is in the future of s; we then write $R \rightarrow_* R'$. We say that R' is the *time successor* of R if $R \rightarrow_* R'$, $R \neq R'$, and for every $R'' \in \mathcal{R}$, we have that $R \rightarrow_* R'' \rightarrow_* R'$ implies $R'' = R$ or $R'' = R'$; we then write $R \rightarrow_{+1} R'$ or $R' \leftarrow_{+1} R$. Similarly, for $R, R' \in \mathcal{R}$, we write $R \xrightarrow{a} R'$ if there is $s \in R$, and there is $s' \in R'$, such that $s \xrightarrow{a} s'$.

We say that a region $R \in \mathcal{R}$ is *thin* if for every $s \in R$ and every $\varepsilon > 0$, we have that $[s] \neq [s + \varepsilon]$; other regions are called *thick*. We write $\mathcal{R}_{\text{Thin}}$ and $\mathcal{R}_{\text{Thick}}$ for the sets of thin and thick regions, respectively. Note that if $R \in \mathcal{R}_{\text{Thick}}$ then for every $s \in R$, there is an $\varepsilon > 0$, such that $[s] = [s + \varepsilon]$. Observe also, that the time successor of a thin region is thick, and vice versa.

Another noteworthy property of a thin region is that for every thin region $R \in \mathcal{R}_{\text{Thin}}$ there is a clock $c \in C$ and a nonnegative integer $b \in \mathbb{N}_+$ such that for every state $s \in R$ we have that $s(c) = b$. In other words if $(D^0, D^1, \ldots, D^m)$ is the cellular signature of a thin region then D^0 is nonempty.

EXAMPLE 3.3.3. Let ℓ be a location of a timed automaton with two clocks x and y and bound $\mathcal{K} = 2$. The clock regions of this timed automaton are shown in Figure 3.3. The region (ℓ, P_2) and the region (ℓ, P_3) are in the future of the region (ℓ, P_1). The region (ℓ, P_2) is the time successor of the region (ℓ, P_1), and the region (ℓ, P_3) is the time successor of the region (ℓ, P_2). The regions $(\ell, P_0), (\ell, P_1)$ and (ℓ, P_3) are thin regions, while the region (ℓ, P_2) is a thick region. Let us consider the regions $(\ell, P_1), (\ell, P_2)$, and (ℓ, P_3) in their cellular signature

form:

$$(\ell, P_1) = (\ell, (\{x = 0, y = 0\}, (\{y\}, \{x\}))),$$

$$(\ell, P_2) = (\ell, (\{x = 0, y = 0\}, (\emptyset, \{y\}, \{x\}))), \text{ and}$$

$$(\ell, P_3) = (\ell, (\{x = 1, y = 0\}, (\{x\}, \{y\}))).$$

Notice how the corner and the cellular signature of a region changes as it evolves in time.

3.3.3. Time-Abstract Bisimulation

DEFINITION 3.3.4 (Time-Abstract Bisimulation **[Bou09]**)**.** A relation $B \subseteq S \times S$ defined over the set of states of a timed automata is a *time-abstract bisimulation* if for every pair of states $s_1, s_2 \in S$ such that $(s_1, s_2) \in B$, for every nonnegative real number $t_1 \in \mathbb{R}_{\oplus}$, and every action $a \in A$ such that $s_1 \xrightarrow{a}_{t_1} s_1'$, there exists a nonnegative real number $t_2 \in \mathbb{R}_{\oplus}$ such that $s_2 \xrightarrow{a}_{t_2} s_2'$ and $(s_1', s_2') \in B$.

Alur and Dill proved the following fact about the region equivalence relation.

LEMMA 3.3.5. **[AD90]** Region equivalence relation is a time-abstract bisimulation.

3.4. Extensions of Timed Automata

The extensions of timed automata which are relevant to this thesis are priced timed automata and timed game automata.

3.4.1. Priced Timed Automata

Priced timed automata are extensions of timed automata and are useful in modelling optimal budgeting and scheduling problems **[BFH+01, BBL08, AM01]** for real-time systems.

DEFINITION 3.4.1 (Priced Timed Automata)**.** A *priced timed automaton* $(\mathcal{T}, \pi)$ consists of a timed automaton $\mathcal{T}$ and a price function $\pi : S \times \mathbb{R}_{\oplus} \times A \to \mathbb{R}$. For every state $s \in S$ and every timed move $(t, a) \in \mathbb{R}_{\oplus} \times A$, the price function π determines the price $\pi(s, t, a)$ of taking the timed move (t, a) from state s, i.e., the price of the transition $(s, (t, a), \mathsf{Succ}(s, (t, a)))$.

Linearly-priced timed automata **[BFH+01]**, also known in the literature as weighted timed automata **[ALTP01, BBBR07, ABM04]**, augment timed automata with price information, such that the price of waiting in a location is proportional to the waiting time, hence the name linearly-priced timed automata.

DEFINITION 3.4.2 (Linearly-Priced TA)**.** A *linearly-priced timed automaton* $(\mathcal{T}, p)$ consists of timed automaton $\mathcal{T}$, and a *price labelling function* $p : L \cup A \to \mathbb{R}$ that assigns a *price rate* $p(\ell)$ to every location $\ell \in L$, and a price $p(a)$ to every action a. A linearly-priced timed automaton $(\mathcal{T}, p)$ is a priced timed automaton $(\mathcal{T}, \pi)$ such that for every state $s = (\ell, \nu)$ and every timed move (t, a) we have $\pi(s, (t, a)) = p(\ell) \cdot t + p(a)$.

Linearly-priced timed automata are useful in modelling scheduling problems of real-time systems where the price-per-time-unit curve of every resource in the system is linear, i.e., the price-rate of every resource is constant. In practice, however, price-per-time-unit curve of resources can be non-linear—in particular convex or concave. Convex price-per-time-unit functions arise in the situations where price-rate of a resource is non-decreasing with time (e.g. overtime labour), while concave price-per-time-unit functions arise if the price-rate is non-increasing (e.g. price of hiring an equipment for a short duration). For a detailed discussion on the importance of convex and concave price functions in scheduling problems, we refer the reader to Falk and Horowitz [**FH72**].

Since linearly-priced timed automata are inadequate to model such systems, we propose generalisations of linearly-priced timed automata to concavely-priced and convexly-priced timed automata. Unlike for linearly-priced timed automata, we do not specify explicitly how the price function $\pi : S \times \mathbb{R}_{\oplus} \times A$ is represented; for conceptual simplicity it is convenient to think of it as an oracle or a computational black box. For the definitions and properties of *concave functions* and K-continuity (or Lipschitz-continuity with constant K), we refer the reader to Appendix B.

DEFINITION 3.4.3 (Concavely-Priced Timed Automata)**.** A *concavely-priced timed automaton* $(\mathcal{T}, \pi, K)$ is a priced timed automaton $(\mathcal{T}, \pi)$ with a constant $K > 0$, such that for all actions $a \in A$ and for all regions $R, R' \in \mathcal{R}$, the function $\pi^a_{R,R'} : (s,t) \mapsto \pi(s,t,a)$ is concave and K-continuous[2] on $D_{R,R'} = \{(s,t) \in S \times \mathbb{R}_{\oplus} : s \in R \text{ and } (s+t) \in R'\}$.

A *convexly-priced timed automaton* is defined in an analogous manner. Notice that every linearly-priced timed automaton is both a concavely-priced timed automaton and a convexly-priced timed automaton. To study price-per-reward optimisation, we need a dual-priced timed automata, and for this purpose we define concave price-reward timed automata.

DEFINITION 3.4.4 (Concave Price-Reward Timed Automata)**.** A *concave price-reward timed automaton* $(\mathcal{T}, \pi, \varrho, K)$ is a timed automaton $\mathcal{T}$ with *price* and *reward* functions $\pi, \varrho : S \times \mathbb{R}_{\oplus} \times A \to \mathbb{R}$, and a constant $K > 0$ such that for all actions $a \in A$ and for all regions $R, R' \in \mathcal{R}$, the functions $(s,t) \mapsto \pi(s,t,a)$ and $(s,t) \mapsto \varrho(s,t,a)$ are K-continuous, and concave and convex, respectively, on $\{(s,t) \in S \times \mathbb{R}_{\oplus} : s \in R \text{ and } (s+t) \in R'\}$.

We require the following assumption regarding the divergence of reward in a price-reward timed automaton.

ASSUMPTION 3.4.5 (Regional Non-Zenoness Assumption on Reward)**.** A timed automaton is *regionally non-Zeno* [3] with respect to ϱ, i.e., there exists a positive real number k such that for every run $r = \langle s_0, \tau_1, s_1, \tau_2, \ldots, \tau_n, s_n \rangle \in \text{Runs}_{\text{fin}}$, with $[s_0] = [s_n]$, (i.e., such that the run r forms a cycle in the finite region graph of the the timed automaton), we have that $\sum_{i=1}^{n} \varrho(s_{i-1}, \tau_i) \geq k$.

[2] If the timed automaton is bounded and $\pi^a_{R,R'}$ is concave, then such a K exists without loss of generality.

[3] This assumption is similar to the structural non-Zenoness assumption [**Tri99**]. While structural non-Zeno assumption concerns cycles of control graph of locations and transitions, regional non-Zeno assumption is less restrictive and requires non-Zeno rewards in cycles of region graph.

Such assumptions are standard for price-per-reward optimisation problems on finite graphs (see Assumption 2.1.4) and timed automata (Bouyer et al. [**BBL04**]).

A *convex price-reward timed automaton* is defined in an analogous manner.

In Chapter 4 we discuss minimisation problems on concavely-priced timed automata and concave price-reward timed automata. The treatment for maximisation problems on convexly-priced timed automaton and convex price-reward timed automata is similar and hence omitted.

3.4.2. Games on Timed Automata

Optimal controller synthesis of real-time open system (assuming adversarial environment) can be specified as two-player zero-sum games on timed automata. Broadly speaking, there are two different ways to specify a game arena on a timed automaton: turn based games—specified by a partition of the set of locations between the two players, and concurrent games—specified by a partition of the set of actions between the two players.

3.4.2.1. Turn Based Games

DEFINITION 3.4.6 (Timed Game Automata—Turn Based)**.** A *timed game automaton* is a tuple $\Gamma = (\mathcal{T}, L_{\text{Min}}, L_{\text{Max}})$, where:

- $\mathcal{T} = (L, C, S, A, E, \delta, \xi, F)$ is a timed automaton,
- $L_{\text{Min}} \subseteq L$ is the set of locations controlled by player Min, and
- $L_{\text{Max}} \subseteq L$ is the set of locations controlled by player Max.

We require that the sets L_{Max} and L_{Min} form a partition of the set of locations L.

A play starts at a state $s_0 \in S$. If $s_0 \in L_p$, for $p \in \{\text{Max}, \text{Min}\}$, then player p chooses a valid timed move (t_1, a_1) and $s_1 = \mathsf{Succ}(s_0, (t_1, a_1))$ becomes the new current state. When this happens then we say that player p has made a timed move from the current state. Players keep making timed moves in this way indefinitely, thus forming an infinite run $r = \langle s_0, \tau_1 s_1, \tau_2, \ldots \rangle$ of the timed automaton.

A *strategy* for Min is a function $\mu : \text{Runs}_{\text{fin}} \to A \times \mathbb{R}_{\oplus}$, such that if $\mathsf{Last}(r) = s \in S_{\text{Min}}$ and $\mu(r) = \tau$ then $s \xrightarrow{\tau} s'$, where $s' = \mathsf{Succ}(s, \tau)$. Similarly, a strategy for player Max is a function $\chi : \text{Runs}_{\text{fin}} \to A \times \mathbb{R}_{\oplus}$, such that if $\mathsf{Last}(r) = s \in S_{\text{Max}}$ and $\chi(r) = \tau$ then $s \xrightarrow{\tau} s'$, where $s' = \mathsf{Succ}(s, \tau)$. We write Σ_{Min} for the set of strategies for player Min, and we write Σ_{Max} for the set of strategies for player Max.

If players Min and Max use strategies μ and χ, resp., then the (μ, χ)-run from a state s is the unique run $\text{Run}(s, \mu, \chi) = \langle s_0, \tau_1, s_1, \tau_2, \ldots \rangle$, such that $s_0 = s$, and for every $i \geq 1$, if $s_i \in S_{\text{Min}}$, or $s_i \in S_{\text{Max}}$, then $\mu(\text{Run}_i(s, \mu, \chi)) = \tau_{i+1}$, or $\chi(\text{Run}_i(s, \mu, \chi)) = \tau_{i+1}$, resp., where $\text{Run}_i(s, \mu, \chi) = \langle s_0, \tau_1, s_1, \ldots, s_{i-1}, \tau_i, s_i \rangle$.

We say that a strategy μ for Min is *positional* if for all finite runs $r, r' \in \text{Runs}_{\text{fin}}$, we have that $\mathsf{Last}(r) = \mathsf{Last}(r')$ implies $\mu(r) = \mu(r')$. A positional strategy for player Min can be then represented as a function $\mu : S_{\text{Min}} \to A \times \mathbb{R}_{\oplus}$, which uniquely determines the strategy $\mu^{\infty} \in \Sigma_{\text{Min}}$ as follows: $\mu^{\infty}(r) = \mu(\mathsf{Last}(r))$, for all finite runs $r \in \text{Runs}_{\text{fin}}$. Positional

strategies for player Max are defined and represented in the analogous way. We write Π_{Min} and Π_{Max} for the sets of positional strategies for player Min and for player Max, respectively.

3.4.2.2. Concurrent Games

Minor variants of the following timed game automata are considered in literature [**AM99, BHPR07, ABM04**].

DEFINITION 3.4.7 (Timed Game Automata [**AM99**]—Concurrent Games)**.** A timed game automaton is a tuple $(L, C, S, A_{\text{Min}}, A_{\text{Max}}, \epsilon, E, \delta, \xi, F)$, where

- L is a finite set of *locations*,
- C is a finite set of *clocks*,
- $S \subseteq L \times \mathcal{V}_C$ is a set of *states*,
- A_{Min} is a finite set *actions* controlled by player Min,
- A_{Max} is a finite set *actions* controlled by player Max,
- ϵ is a special empty action,
- $E : A_{\text{Min}} \cup A_{\text{Max}} \to 2^S$ is an *action enabledness function*,
- $\delta : L \times A^{\epsilon}_{\text{Min}} \times A^{\epsilon}_{\text{Max}} \to L$ is a *transition function*,
- $\xi : A^{\epsilon}_{\text{Min}} \times A^{\epsilon}_{\text{Max}} \to 2^C$ is a *clock reset function*, and
- $F \subseteq S$ is a set of *final states*.

Here $A^{\epsilon}_{\text{Min}}$ and $A^{\epsilon}_{\text{Max}}$ stand for $A_{\text{Min}} \cup \{\epsilon\}$ and $A_{\text{Max}} \cup \{\epsilon\}$, respectively. It is required that S, F, and $E(a)$ for all $a \in A$, are zones.

Starting from some state $s_0 \in S$ both players choose timed moves $\tau^n \in \mathbb{R}_{\oplus} \times A_{\text{Min}}$ and $\tau^x \in \mathbb{R}_{\oplus} \times A_{\text{Max}}$ simultaneously; the next state of the system is determined by their joint action and is given by the successor function $\mathsf{Succ} : S \times \mathbb{R}_{\oplus} \times A_{\text{Min}} \times \mathbb{R}_{\oplus} \times A_{\text{Max}} \to S$ in the following manner:

$$\mathsf{Succ}((\ell, \nu), (t_n, a_n), (t_x, a_x)) = \begin{cases} (\delta(\ell, a_n, \epsilon), \mathsf{Reset}(\nu + t_n, \xi(a_n, \epsilon))) & \text{if } t_n < t_x \\ (\delta(\ell, \epsilon, a_x), \mathsf{Reset}(\nu + t_x, \xi(\epsilon, a_x))) & \text{if } t_x < t_n \\ (\delta(\ell, a_n, a_x), \mathsf{Reset}(\nu + t, \xi(a_n, a_x))) & \text{if } t = t_x = t_n. \end{cases}$$

The play then evolves to the resulting state and the players continue to play indefinitely to form an infinite run of the timed automaton. The strategies for the players can be defined in a straightforward manner.

3.4.2.3. A Case for Turn-based Games

Arguably, concurrent games can be used to model a larger class of open real-time systems than turn based games. However, in this thesis, we focus our attention to turn-based games. Turn based games are simple to analyse using dynamic programming techniques, and are a good first step in establishing effective techniques for solving competitive optimisation on timed automata. Another reason for choosing the turn-based model is that concurrent games on timed automata often lack determinacy (see, e.g., de Alfaro et al. [**dAFH$^+$03**]).

3.5. Competitive Optimisation on Timed Automata

In this section we define competitive optimisation problems on timed automata which are of interest in this thesis. Since some of these problems are known by different names in the literature, in this section we set a uniform terminology for these problems.

3.5.1. The Noncompetitive Case

In this thesis we consider minimisation problems for various cost functions on concavely-priced timed automata. Henceforth we reserve the term optimisation, optimum, and optimal to refer to minimisation, minimum, and minimal, respectively. Fix a timed automaton $\mathcal{T}$, price and reward functions $\pi, \varrho : S \times \mathbb{R}_{\oplus} \times A \to \mathbb{R}$, and an initial state $s \in S$.

A cost function is a function which takes an infinite run and returns its cost. Given a cost function $\mathsf{Cost} : \text{Runs} \to \mathbb{R}$ we define the optimal cost function $\mathsf{Cost}_* : S \to \mathbb{R}$ in the following way:

$$\mathsf{Cost}_*(s) \quad = \quad \inf_{r \in \text{Runs}(s)} \{\mathsf{Cost}(r)\} = \inf_{\sigma \in \Sigma} \{\mathsf{Cost}(\text{Run}(s, \sigma))\}.$$

The *optimisation problem* for the cost function Cost is to compute the optimum cost $\mathsf{Cost}_*(s)$ of a given state $s \in S$. The decision version of the optimisation problem is as follows: "given a timed automaton $\mathcal{T}$, a state $s \in S$ and a number $D \in \mathbb{Q}$, determine whether $\mathsf{Cost}_*(s) \leq D$."

We say that a strategy $\sigma_* \in \Sigma$ is *optimal* for the cost function Cost if we have that $\mathsf{Cost}(\text{Run}(s, \sigma_*)) = \mathsf{Cost}_*(s)$. For a given $\varepsilon > 0$, we say that a strategy $\sigma \in \Sigma$ is ε-optimal if we have that $\mathsf{Cost}(\text{Run}(s, \sigma)) \leq \mathsf{Cost}_*(s) + \varepsilon$.

Let $r = \langle s_0, \tau_1, s_1, \tau_2, \ldots \rangle \in \text{Runs}$ be a run of the timed automaton $\mathcal{T}$, where $\tau_i = (t_i, a_i)$ for every positive integer i. Moreover, for π and ϱ, the price and reward functions, respectively, of a priced (or price-reward) timed automaton, and for every positive integer n, we define: $T_n(r) = \sum_{i=1}^{n} t_i$, $\pi_n(r) = \sum_{i=1}^{n} \pi(s_{i-1}, \tau_i)$, and $\varrho_n(r) = \sum_{i=1}^{n} \varrho(s_{i-1}, \tau_i)$.

The following list of cost functions gives rise to a number of corresponding optimisation problems.

(1) *Reachability time*. The reachability-time cost function is defined in the following way: for every run $r \in \text{Runs}$ we have

$$\mathsf{RT}(r) = \begin{cases} T_N(r), & \text{if } N = \mathsf{Stop}(r) < \infty, \\ \infty, & \text{otherwise.} \end{cases}$$

(2) *Reachability price*. The reachability-price cost function is defined in the following way: for every run $r \in \text{Runs}$ we have

$$\mathsf{RP}(r) = \begin{cases} \pi_N(r), & \text{if } N = \mathsf{Stop}(r) < \infty, \\ \infty, & \text{otherwise.} \end{cases}$$

(3) *Discounted time*. The discounted-time cost function is defined in the following way: for every run $r \in \text{Runs}$ and every *discount factor* $\lambda \in (0, 1)$ we have

$$\mathsf{DT}(\lambda)(r) = (1 - \lambda) \sum_{i=1}^{\infty} \lambda^{i-1} t_i.$$

(4) *Discounted price*. The discounted-price cost function is defined in the following way: for every run $r \in \text{Runs}$ and every *discount factor* $\lambda \in (0,1)$ we have

$$\mathsf{DP}(\lambda)(r) = (1-\lambda)\sum_{i=1}^{\infty} \lambda^{i-1}\pi(s_{i-1},\tau_i).$$

(5) *Average time*. The average-time cost function is defined in the following way: for every run $r \in \text{Runs}$ we have

$$\mathsf{AT}(r) = \limsup_{n\to\infty} \frac{T_n}{n}.$$

(6) *Average price*. The average-price cost function is defined in the following way: for every run $r \in \text{Runs}$ we have

$$\mathsf{AP}(r) = \limsup_{n\to\infty} \frac{\pi_n(r)}{n}.$$

(7) *Price-per-time average*. The price-per-time average cost function is defined in the following way: for every run $r \in \text{Runs}$ we have

$$\mathsf{PTAvg}(r) = \limsup_{n\to\infty} \frac{\pi_n(r)}{T_n(r)}.$$

(8) *Price-per-reward average*. The price-per-reward average cost function is defined in the following way: for every run $r \in \text{Runs}$ we have

$$\mathsf{PRAvg}(r) = \limsup_{n\to\infty} \frac{\pi_n(r)}{\varrho_n(r)}.$$

In Chapter 4 we show (Theorem 4.2.1) that for arbitrary initial state $s \in S$, the optimisation problem for all these cost functions on a concavely-price timed automata or concave price-reward timed automata, as appropriate, is PSPACE-complete.

3.5.2. Competitive Optimisation

Fix a timed game automaton $\Gamma = (\mathcal{T}, L_{\text{Min}}, L_{\text{Max}})$, price and reward functions $\pi, \varrho : S \times \mathbb{R}_{\oplus} \times A \to \mathbb{R}$, and an initial state $s \in S$. Let Σ_{Min} and Σ_{Max} be the set of strategies for player Min and player Max, respectively. In the context of games we sometimes refer to a run as a *play* of the game.

DEFINITION 3.5.1 (Cost Game on a Timed Game Automaton)**.** A *cost game* on a timed game automaton is a tuple $(\Gamma, \mathsf{Cost}_{\text{Min}}, \mathsf{Cost}_{\text{Max}})$, where:

- $\Gamma = (\mathcal{T}, L_{\text{Min}}, L_{\text{Max}})$ is a timed game automaton,
- $\mathsf{Cost}_{\text{Min}} : \text{Runs} \to \mathbb{R}$ is a payoff function for player Min which gives the amount Min loses in a play, and
- $\mathsf{Cost}_{\text{Max}} : \text{Runs} \to \mathbb{R}$ is the payoff function for player Max, which gives the amount Max wins in a play.

A cost game is called *zero-sum* if we have that $\mathsf{Cost}_{\mathrm{Min}}(r) = \mathsf{Cost}_{\mathrm{Max}}(r)$ for every play r of the game.

Payoff functions $\mathsf{Cost}_{\mathrm{Max}} : \mathrm{Runs} \to \mathbb{R}$ and $\mathsf{Cost}_{\mathrm{Min}} : \mathrm{Runs} \to \mathbb{R}$ naturally gives rise to the payoff functions $\mathsf{Cost}_{\mathrm{Max}} : S \times \Sigma_{\mathrm{Min}} \times \Sigma_{\mathrm{Max}} \to \mathbb{R}$ and $\mathsf{Cost}_{\mathrm{Min}} : S \times \Sigma_{\mathrm{Min}} \times \Sigma_{\mathrm{Max}} \to \mathbb{R}$ in the following way: for strategies $\mu \in \Sigma_{\mathrm{Min}}$ and $\chi \in \Sigma_{\mathrm{Max}}$ of respective players and a state $s \in S$ we have $\mathsf{Cost}_{\mathrm{Min}}(s, \mu, \chi) = \mathsf{Cost}_{\mathrm{Min}}(\mathrm{Run}(s, \mu, \chi))$ and $\mathsf{Cost}_{\mathrm{Max}}(s, \mu, \chi) = \mathsf{Cost}_{\mathrm{Max}}(\mathrm{Run}(s, \mu, \chi))$.

The *upper value* of a cost game at the state s is defined as the maximum loss that player Min can secure, in a play starting at the state s, irrespective of the strategies used by player Max. Similarly the *lower value* of a cost game at the state s is defined as the minimum win that player Max can secure, in a play starting at the state s, irrespective of the strategies used by player Min. Formally we define the *upper value* $\overline{\mathsf{Val}}^{\Gamma}(s)$ and the lower-value $\underline{\mathsf{Val}}^{\Gamma}(s)$ of the game at the state s by as follows:

$$\overline{\mathsf{Val}}^{\Gamma}(s) = \inf_{\mu \in \Sigma_{\mathrm{Min}}} \sup_{\chi \in \Sigma_{\mathrm{Max}}} \mathsf{Cost}_{\mathrm{Min}}(s, \mu, \chi), \text{ and } \underline{\mathsf{Val}}^{\Gamma}(s) = \sup_{\chi \in \Sigma_{\mathrm{Max}}} \inf_{\mu \in \Sigma_{\mathrm{Min}}} \mathsf{Cost}_{\mathrm{Max}}(s, \mu, \chi).$$

From Proposition 1.2.4 we know that the lower value of a game at a state is always less than the upper value of the game at that state, i.e., for every state $s \in S$ the inequality $\underline{\mathsf{Val}}^{\Gamma}(s) \leq \overline{\mathsf{Val}}^{\Gamma}(s)$ always holds.

A cost game is determined if for every state $s \in S$, we have $\underline{\mathsf{Val}}^{\Gamma}(s) = \overline{\mathsf{Val}}^{\Gamma}(s)$. We then say that the value of the game exists. We write $\mathsf{Val}^{\Gamma}(s)$ for this number and we call it the *value* of the game at the state s.

The strategies $\mu^* \in \Sigma_{\mathrm{Min}}$ and $\chi^* \in \Sigma_{\mathrm{Max}}$ are *optimal* for the respective players, if for every state $s \in S$ we have that

$$\sup_{\chi \in \Sigma_{\mathrm{Max}}} \mathsf{Cost}_{\mathrm{Min}}(s, \mu^*, \chi) = \overline{\mathsf{Val}}^{\Gamma}(s), \text{ and } \inf_{\mu \in \Sigma_{\mathrm{Min}}} \mathsf{Cost}_{\mathrm{Min}}(s, \mu^*, \chi) = \underline{\mathsf{Val}}^{\Gamma}(s).$$

We say that a game is *positionally determined* if players have positional optimal strategies.

If the game is determined then the competitive optimisation problem is to compute value of the game $\mathsf{Val}(s)$ for a give state $s \in S$. The corresponding decision problem is given a cost game $(\Gamma, \mathsf{Cost}_{\mathrm{Min}}, \mathsf{Cost}_{\mathrm{Max}})$, a state $s \in S$ and a number $D \in \mathbb{Q}$, determine whether $\mathsf{Val}^{\Gamma}(s) \leq D$.

The following cost games on timed automata are of interest.

(1) *Reachability game.* A reachability game $(\Gamma, \mathsf{Reach}_{\mathrm{Min}}, \mathsf{Reach}_{\mathrm{Max}})$ has the following payoff functions:

$$\mathsf{Reach}_{\mathrm{Min}}(r) = \mathsf{Reach}_{\mathrm{Max}}(r) = \begin{cases} N & \text{if } N = \mathsf{Stop}(r) < \infty \\ \infty & \text{otherwise.} \end{cases}$$

(2) *Reachability-time game.* A reachability-time game $(\Gamma, \mathsf{RT}_{\mathrm{Min}}, \mathsf{RT}_{\mathrm{Max}})$ has the following payoff functions:

$$\mathsf{RT}_{\mathrm{Min}}(r) = \mathsf{RT}_{\mathrm{Max}}(r) = \begin{cases} T_N(r) & \text{if } N = \mathsf{Stop}(r) < \infty \\ \infty & \text{otherwise.} \end{cases}$$

(3) *Reachability-price game.* A reachability-price game $(\Gamma, \mathsf{RP}_{\mathrm{Min}}, \mathsf{RP}_{\mathrm{Max}})$ has the following payoff functions:

$$\mathsf{RP}_{\mathrm{Min}}(r) = \mathsf{RP}_{\mathrm{Max}}(r) = \begin{cases} \pi_N(r) & \text{if } N = \mathsf{Stop}(r) < \infty \\ \infty & \text{otherwise.} \end{cases}$$

(4) *Discounted-time game.* For a given discount factor $\lambda \in (0,1)$, a discounted-time game $(\Gamma, \mathsf{DT}_{\mathrm{Min}}(\lambda), \mathsf{DT}_{\mathrm{Max}}(\lambda))$ has the following payoff functions:

$$\mathsf{DT}_{\mathrm{Min}}(\lambda)(r) = \mathsf{DT}_{\mathrm{Max}}(\lambda)(r) = (1-\lambda)\sum_{i=1}^{\infty} \lambda^{i-1} t_i.$$

(5) *Discounted-price game.* For a given discount factor $\lambda \in (0,1)$, a discounted-price game $(\Gamma, \mathsf{DP}_{\mathrm{Min}}(\lambda), \mathsf{DP}_{\mathrm{Max}}(\lambda))$ has the following payoff functions:

$$\mathsf{DP}_{\mathrm{Min}}(\lambda)(r) = \mathsf{DP}_{\mathrm{Max}}(\lambda)(r) = (1-\lambda)\sum_{i=1}^{\infty} \lambda^{i-1} \pi(s_{i-1}, \tau_i).$$

(6) *Average-time game.* An average-time game $(\Gamma, \mathsf{AP}_{\mathrm{Min}}, \mathsf{AP}_{\mathrm{Max}})$ has the following payoff functions:

$$\mathsf{AT}_{\mathrm{Min}}(r) = \limsup_{n\to\infty} \frac{T_n(r)}{n} \text{ and } \mathsf{AT}_{\mathrm{Max}}(r) = \liminf_{n\to\infty} \frac{T_n(r)}{n}.$$

(7) *Average-price game.* An average-price game $(\Gamma, \mathsf{AP}_{\mathrm{Min}}, \mathsf{AP}_{\mathrm{Max}})$ has the following payoff functions:

$$\mathsf{AP}_{\mathrm{Min}}(r) = \limsup_{n\to\infty} \frac{\pi_n(r)}{n} \text{ and } \mathsf{AP}_{\mathrm{Max}}(r) = \liminf_{n\to\infty} \frac{\pi_n(r)}{n}.$$

(8) *Price-per-time average game.* In a similar manner, a price-per-time average game $(\Gamma, \mathsf{PTAvg}_{\mathrm{Min}}, \mathsf{PTAvg}_{\mathrm{Max}})$ has the following payoff functions:

$$\mathsf{PTAvg}_{\mathrm{Min}}(r) = \limsup_{n\to\infty} \frac{\pi_n(r)}{T_n(r)} \text{ and } \mathsf{PTAvg}_{\mathrm{Max}}(r) = \liminf_{n\to\infty} \frac{\pi_n(r)}{T_n(r)}.$$

(9) *Price-per-reward average game.* In a similar manner, a price-per-reward average game $(\Gamma, \mathsf{PRAvg}_{\mathrm{Min}}, \mathsf{PRAvg}_{\mathrm{Max}})$ has the following payoff functions:

$$\mathsf{PRAvg}_{\mathrm{Min}}(r) = \limsup_{n\to\infty} \frac{\pi_n(r)}{\varrho_n(r)} \text{ and } \mathsf{PRAvg}_{\mathrm{Max}}(r) = \liminf_{n\to\infty} \frac{\pi_n(r)}{\varrho_n(r)}.$$

In this thesis we study reachability-time games and average-time games in Chapter 5 and Chapter 6, respectively, and show that these game are determined (Theorem 5.1.2 and Theorem 6.4.2). We also prove that decision problems corresponding to these games are EXPTIME-complete for timed automata with at least two clocks (Theorem 5.5.4 and Theorem 6.5.1).

Reachability games were shown to be EXPTIME-complete for timed automata by Henzinger and Kopke [**HK99**]. In Chapter 5 we strengthen this results by showing that reachability games are EXPTIME-hard for timed automata with at least two clocks (Theorem 5.5.3).

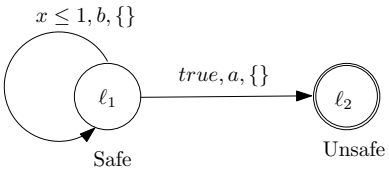


FIGURE 3.4. A Zeno Timed Automaton.

3.6. A Note on Zeno Runs

The syntax of timed automata permits the specification of physically impossible behaviours of a real-time system. For instance, it is possible that a run of a timed automaton performs infinitely many actions in a finite amount of time. Such runs are called Zeno runs. A run $r = \langle s_0, (t_1, a_1), s_1, (t_2, a_2), s_2, \ldots \rangle \in$ Runs of a timed automaton $\mathcal{T}$ is called a *Zeno run* if $\mathsf{Time}(r) = \sum_{i=1}^{\infty} t_i$ is finite. We say that a timed automaton is *Zeno* if some of its runs are Zeno. For a given initial state $s \in S$ we say that a strategy $\sigma \in \Sigma$ is *Zeno* is $\mathrm{Run}(s, \sigma)$ is a Zeno run.

The practical applications of competitive optimisation on timed automata are in optimal scheduling and optimal controller synthesis. If, for some competitive optimisation problem, the optimal strategy for player Min (controller) is a Zeno strategy then it is impossible to implement such a controller in practice.

For example, let us consider the timed automaton in Figure 3.4. The timed automaton has two location ℓ_1, where the system is safe and ℓ_2, where system is unsafe. Notice that this automaton is Zeno, as from location ℓ_1 the action a may be executed infinitely many times within a unit of time. In particular, starting from the state $s_0 = (\ell_1, 0)$ the timed automaton allows the following Zeno run:

$$r = \langle (\ell_1, 0), (1/2, b), (\ell_1, 1/2), (1/4, b), (\ell_1, 3/4), \ldots, (\ell_1, 1 - (1/2^i)), (1/2^{i+1}, b), \ldots \rangle.$$

This run is Zeno since $\mathsf{Time}(r) = \sum_{i=1}^{\infty} 1/2^i = 1$. Let us consider the following verification problem: decide whether this automaton is "safe" with respect to the start state s_0, i.e., decide whether there exists a strategy such that we avoid ever reaching the unsafe state ℓ_2? The answer to this question is both yes and no! Yes, because we can avoid reaching the unsafe state ℓ_2 by following the run r; and no, because executing such a run is tantamount to executing infinitely many actions in a finite amount of time (one time unit), which is generally considered to be physically impossible.

3.6.1. Avoiding Zeno Strategies

One way of avoiding problems with Zeno strategies is to put syntactical constraints on timed automata. For example, if we restrict the timed automata so that for every run $r = \langle s_0, \tau_1, s_1, \tau_2, \ldots, \tau_n, s_n \rangle \in \mathrm{Runs}_{\mathrm{fin}}$, such that $s_0 = (\ell_0, \nu_0)$, $s_n = (\ell_n, \nu_n)$, and $\ell_0 = \ell_n$ (i.e., such that the run r forms a cycle in the finite graph of the locations and transitions of the timed automaton), we have that $\sum_{i=1}^{n} t_i \geq 1$. Compliance to such a restriction can be insured by assuming that the timed automaton under consideration satisfies the following, easily verifiable, property:

DEFINITION 3.6.1 (Structural Non-Zenoness Assumption [**BGS00**])**.** A timed automaton is structurally non-Zeno if in every cycle in the finite graph of locations and transitions at least one clock is reset, and it is tested against some positive lower bound.

Asarin and Maler [**AM99**] used this assumption to solve reachability-time games on timed automata. Another related assumption that appears in some publications [**Tri99**] is *strong non-Zenoness assumption*, which states that every cycle in the control graph of timed automata "spends" time. Note that structural non-Zenoness implies strong non-Zenoness.

Another way to avoid implementation issues with Zeno strategies is to forbid [**MPS95**] the controller (or the player Min) to use Zeno strategies, while the environment (or player Max) can use Zeno strategies to its advantage. Such restrictions, however, may result in sub-optimal strategies.

Henzinger et al. [**dAFH$^+$03, BHPR07**] proposed an elegant way to handle the problems of Zeno strategies by introducing timed games where players are penalised if they use Zeno strategies. In their work a player can win a game only by using the so-called receptive strategies—strategies where the player is not responsible for a Zeno run. Unfortunately receptive strategies can also be physically unimplementable. For instance consider a strategy which suggest to a player to take actions in the following time sequence (example taken from [**CHR02**]):

$$\langle 1, 1\frac{1}{2}, 2, 2\frac{1}{4}, 3, 3\frac{1}{8}, 4, 4\frac{1}{16}, ...\rangle.$$

Notice that although this strategy is non-Zeno, it can be implemented only if the software/hardware with infinite precision is available. However, one can argue that if such hardware/software is indeed available, even Zeno strategies are implementable.

In this thesis we do not require the timed automata to be non-Zeno. As a consequence, given a Zeno timed automaton as input, our algorithms may return unrealistic Zeno optimal strategies for players. One related restriction (Assumption 3.4.5) is on the divergence of reward function in price-reward timed automata. However, note that such an assumption (Assumption 2.1.4) on reward is required even for the weighted finite graphs.

3.7. Abstractions of Timed Automata

In this section we review some abstractions of timed automata, and comment on their suitability to solve competitive optimisation problems.

3.7.1. Region Automata and Region Graphs

DEFINITION 3.7.1 (Region Automata [**AD90**])**.** The *region automaton* $\mathcal{T}_{\mathrm{RA}} = (\mathcal{R}, \mathcal{M}, \mathcal{R}_F)$ of a timed automaton $\mathcal{T}$ is an edge-labelled finite graph with final vertices, where:

- the set $\mathcal{R}$ of regions of $\mathcal{T}$ is the set of *vertices*;
- the *labelled edge relation* $\mathcal{M} \subseteq \mathcal{R} \times \mathcal{R} \times A \times \mathcal{R}$ is defined by

$$\mathcal{M} = \{(R, R'', a, R') \,:\, R \to_* R'' \xrightarrow{a} R'\};\ \text{and}$$

- the set of final vertices $\mathcal{R}_F \subseteq \mathcal{R}$ is such that for every $R \in \mathcal{R}_F$ we have that $R \subseteq F$ and for every $R \in \mathcal{R} \setminus \mathcal{R}_F$ we have $R \cap F = \emptyset$.

By counting the total number of cellular signatures we get an upper bound on the number of regions in a ($\mathcal{K}$-bounded) timed automaton as $|L| \cdot d! \cdot (2\mathcal{K}+2)^d$, where d is the number of clocks.

PROPOSITION 3.7.2 (Size of the region automata **[AD94]**)**.** The size—number of vertices and edges—of the region automaton $\mathcal{T}_{\mathrm{RA}}$ of a timed automaton $\mathcal{T}$ is exponential in the size of the timed automaton.

Region automata are useful abstractions to solve, among others, the (qualitative) reachability problems and (qualitative) reachability games on timed automata. The following proposition is immediate thanks to the time-abstract bisimulation property (Lemma 3.3.5) of the region equivalence relation.

PROPOSITION 3.7.3. In a timed automaton $\mathcal{T}$ a final state is reachable from a state s if and only if a final vertex is reachable from the vertex $[s]$ in the corresponding region automaton $\mathcal{T}_{\mathrm{RA}}$.

The semantics of a region automaton $\mathcal{T}_{\mathrm{RA}}$ is given by the region graph $\widetilde{\mathcal{T}}$.

DEFINITION 3.7.4 (Region Graph)**.** The *region graph* of a timed automaton $\mathcal{T}$ is a labelled transition system $\widetilde{\mathcal{T}} = (\widetilde{S}, \widetilde{E}, \widetilde{S}_F)$, where:

- $\widetilde{S}$ is the set of *states* defined as $\widetilde{S} = \{(s,R) \in S \times \mathcal{R} \,:\, s \in R\}$;
- $\widetilde{E}$ is the *labelled transition relation* defined as

$$\widetilde{E} = \{((s,R),(t,R'',a),(s',R')) \in \widetilde{S} \times (\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \times \widetilde{S} \\ : R \to_* R'' \xrightarrow{a} R' \text{ and } s' = \mathsf{Succ}(s,t,a) \text{ and } s+t \in R''\};$$ and

- $\widetilde{S}_F \subseteq \widetilde{S}$ is the set of *final states* defined as $\widetilde{S}_F = \{(s,R) \in \widetilde{S} \,:\, R \subseteq F\}$.

The timed automaton $\mathcal{T}$ and the corresponding region graph $\widetilde{\mathcal{T}}$ are equivalent in the following sense.

PROPOSITION 3.7.5. Let $\mathcal{T}$ be a timed automaton and $\widetilde{\mathcal{T}} = (\widetilde{S}, \widetilde{E}, \widetilde{S}_F)$ be its region graph. For every $s, s' \in S$ and timed action $(t,a) \in \mathbb{R}_{\oplus} \times A$, we have $s \xrightarrow{a}_t s'$ if and only if $((s,[s]),(t,[s+t],a),(s',[s'])) \in \widetilde{E}$.

Region automata abstract away the timing information and hence they are not suitable for solving quantitative optimisation problems. While investigating the average price-per-reward problem on timed automata, Bouyer et. al **[BBL04]** introduced a refinement of the region automaton abstraction, which they called the *corner-point abstraction*.

3.7.2. Corner-Point Abstraction

The main idea behind the corner-point abstraction is as follows: if the initial state is a corner state then nearly optimal runs for average price-per-reward problem consist of time delays very close to integers. A corner-point abstraction is a finite edge-labelled graph whose vertices are corner states lying in the topological closure of the regions. Since corner-point

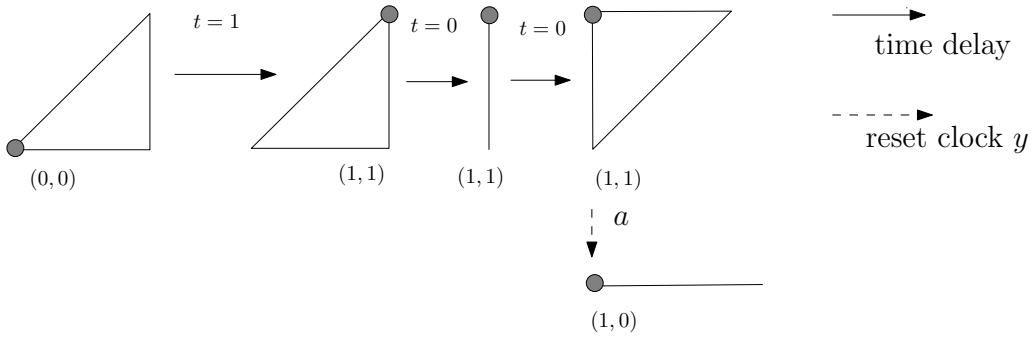


FIGURE 3.5. Evolution of regions in a corner-point abstraction of a timed automaton (the idea of this figure is from Bouyer [**Bou06**]).

abstraction ignores all time delays except the integer ones, it coincides with digital clock semantics considered by Henzinger et al. [**HMP92**].

Before we formally define corner-point abstraction we need to introduce the concept of boundary of a region. We say that $s \in Q$ is in the *boundary of the region* [4] R, and we write $s \in \mathsf{bd}(R)$, if one of the following conditions hold:

- $R \in \mathcal{R}_{\text{Thin}}$ and $s \in R$; or
- $R \in \mathcal{R}_{\text{Thick}}$ and there exists a region R' such that $R' \to_{+1} R$ and $s \in R'$; or
- $R \in \mathcal{R}_{\text{Thick}}$ and there exists a region R' such that $R' \leftarrow_{+1} R$ and $s \in R'$;

We are now in position to define corner-point abstraction.

DEFINITION 3.7.6 (Corner-point Abstraction [**Bou09**])**.** A corner-point abstraction $\mathcal{T}_{\text{CP}} = (V, E, V_F)$ of a timed automaton $\mathcal{T}$ is a finite edge-labelled graph, where:

- V is a finite set of *vertices* defined as

$$V = \{(s, R) \in Q \times \mathcal{R} \,:\, s = (\ell, \nu) \in \mathsf{clos}(R) \text{ and } \lfloor \nu \rfloor = \nu\}$$

  (by $\lfloor \nu \rfloor = \nu$ we mean that ν is a corner);
- E is the *labelled edge relation* defined as

$$E = \{((s, R), (t, R'', a), (s', R')) \in V \times \mathbb{R}_{\oplus} \times \mathcal{R} \times A \times V \\ : R \to_* R'' \xrightarrow{a} R' \text{ and } s' = \mathsf{Succ}(s, t, a) \text{ and } s + t \in \mathsf{bd}(R'')\};$$

- V_F is the set of *final vertices* defined as $V_F = \{(s, R) \in V \,:\, R \subseteq F\}$;

Observe that $((s, R), (t, R'', a), (s', R')) \in E$ implies that t is a nonnegative integer and $(s + t, R'')$ is a vertex of $\mathcal{T}_{\text{CP}}$. An example run of a corner-point abstraction is shown in Figure 3.5.

Using the corner-point abstraction, Bouyer et al. [**BBL04**] showed that the price-per-reward average optimisation problem for a linearly-priced timed automaton is in PSPACE if the initial state is a corner state. Recently, Bouyer et al. [**BBBR07**] also used corner-point abstraction to solve the reachability-price optimisation problem for linearly-priced timed automata, again with the restriction that the initial state is a corner state.

[4] Observe that it is not the boundary of R in topological sense.

3.7.3. Boundary Region Automaton and Boundary Region Graph

Our original motivation for introducing the *boundary region automaton* was to solve the reachability price and price-per-reward average optimisation problems on linearly-priced timed automata with arbitrary initial states, i.e., including non-corner states. The assumption that nearly optimal runs for these two optimisation problems consist of time delays very close to nonnegative integers is incorrect if the initial state is a non-corner state. We observed that if the initial state is an arbitrary state, a more general invariant holds— nearly optimal runs for these two optimisation problems consist of waiting until very close to the boundary of some region before taking an action. We call such timed actions *boundary timed actions* and we denote them by a triple $(b, c, a) \in [\![\mathcal{K}]\!]_{\mathbb{N}} \times C \times A$ which represents the following symbolic timed action: take the action a when the value of clock c is "very close" to b.

DEFINITION 3.7.7 (Boundary Timed Actions)**.** Define the finite set of *boundary timed actions* $\mathcal{A} = [\![\mathcal{K}]\!]_{\mathbb{N}} \times C \times A$. For $s \in Q$ and $\alpha = (b, c, a) \in \mathcal{A}$, we define $t(s, \alpha) = b - s(c)$ if $s(c) \leq b$, and $t(s, \alpha) = 0$ if $s(c) > b$; and we define $\mathsf{Succ}(s, \alpha)$ to be the state $s' = \mathsf{Succ}(s, \tau(\alpha))$, where $\tau(\alpha) = (t(s, \alpha), a)$; we then write $s \xrightarrow{\alpha} s'$. We also write $s \xrightarrow{\alpha} s'$ if $s \xrightarrow{\tau(\alpha)} s'$.

Note that if $\alpha \in \mathcal{A}$ and $s \xrightarrow{\alpha} s'$ then $[s'] \in \mathcal{R}_{\text{Thin}}$. Observe that for every thin region $R' \in \mathcal{R}_{\text{Thin}}$, there is a number $b \in [\![\mathcal{K}]\!]_{\mathbb{N}}$ and a clock $c \in C$, such that for every $R \in \mathcal{R}$ in the past of R', we have that $s \in R$ implies $(s + (b - s(c)) \in R'$; we then write $R \rightarrow_{b,c} R'$. For $\alpha = (a, b, c) \in \mathcal{A}$ and $R, R' \in \mathcal{R}$, we write $R \xrightarrow{\alpha} R'$ or $R \xrightarrow{a}_{b,c} R'$, if $R \rightarrow_{b,c} R'' \xrightarrow{a} R'$, for some $R'' \in \mathcal{R}_{\text{Thin}}$.

Boundary Region Automata. The motivation for the boundary region automata is the following. Let $a \in A$, $s = (\ell, \nu) \in R$ and $R \rightarrow_* R' \xrightarrow{a} R''$.

- If $R' \in \mathcal{R}_{\text{Thick}}$, then there are infinitely many $t \in \mathbb{R}_{\oplus}$ such that $s{+}t \in R'$. For all competitive optimisation problems solved in this thesis, one of the main results that we establish is that in the state s, amongst all such t's, for one of the boundaries of R', the closer $\nu{+}t$ is to this boundary, the 'better' the timed action (t, a) becomes for that problem. However, since R' is a thick region, the set $\{t \in \mathbb{R}_{\oplus} \mid s{+}t \in R'\}$ is an open interval, and hence does not contain its boundary values. Observe that the infimum equals $b_{\text{inf}}{-}\nu(c_{\text{inf}})$ where $R \rightarrow_{b_{\text{inf}}, c_{\text{inf}}} R_{\text{inf}} \rightarrow_{+1} R'$ and the supremum equals $b_{\text{sup}}{-}\nu(c_{\text{sup}})$ where $R \rightarrow_{b_{\text{sup}}, c_{\text{sup}}} R_{\text{sup}} \leftarrow_{+1} R'$. In the boundary region automata, we include these 'best' timed action through the actions $((b_{\text{inf}}, c_{\text{inf}}, a), R')$ and $((b_{\text{sup}}, c_{\text{sup}}, a), R')$.
- If $R' \in \mathcal{R}_{\text{Thin}}$, then there exists a unique $t \in \mathbb{R}_{\oplus}$ such that $(\ell, \nu{+}t) \in R'$. Moreover since R' is a thin region there exists a clock $c \in C$ and a number $b \in \mathbb{N}$ such that $R \rightarrow_{b,c} R'$ and $t = b{-}\nu(c)$. In the boundary region automata we summarise this 'best' timed action from region R via region R' through the action $((b, c, a), R')$.

With this intuition in mind, let us present the definition of a boundary region automata.

DEFINITION 3.7.8 (Boundary Region Automaton)**.** The *boundary region automaton* $\mathcal{T}_{\mathrm{BR}} = (\mathcal{R}, \widehat{\mathcal{M}}, \mathcal{R}_F)$ of a timed automaton $\mathcal{T}$ is a finite edge-labelled graph with final vertices, where:

- the set $\mathcal{R}$ of regions of $\mathcal{T}$ is the set of *vertices*;
- the *labelled edge relation* $\widehat{\mathcal{M}} \subseteq \mathcal{R} \times \mathcal{R} \times \mathcal{A} \times \mathcal{R}$ is such that for every boundary timed action $\alpha = (b, c, a) \in \mathcal{A}$ and for all $R, R', R'' \in \mathcal{R}$ we have that $(R, R'', \alpha, R') \in \widehat{\mathcal{M}}$, if and only if one of the following conditions holds—
  - $R \to_{b,c} R'' \xrightarrow{a} R'$, or
  - there is region $R''' \in \mathcal{R}_{\mathrm{Thin}}$ such that $R \to_{b,c} R''' \to_{+1} R'' \xrightarrow{a} R'$, or
  - there is a region $R''' \in \mathcal{R}_{\mathrm{Thin}}$ such that $R \to_{b,c} R''' \leftarrow_{+1} R'' \xrightarrow{a} R'$;
- the set of *final vertices* $\mathcal{R}_F \subseteq \mathcal{R}$ is such that for every $R \in \mathcal{R}_F$ we have that $R \subseteq F$ and for every $R \in \mathcal{R} \setminus \mathcal{R}_F$ we have that $R \cap F = \emptyset$.

The semantics of a boundary region automaton is given by the *boundary region graph*.

DEFINITION 3.7.9 (Boundary Region Graph)**.** The *boundary region graph* $\widehat{\mathcal{T}} = (\widehat{S}, \widehat{E}, \widehat{S}_F)$ of a timed automaton $\mathcal{T}$ is a labelled transition system, where:

- $\widehat{S}$ is the set of *states* defined as

$$\widehat{S} = \{(s, R) \in Q \times \mathcal{R} \,:\, s \in \mathsf{clos}(R)\};$$

- $\widehat{E}$ is the *labelled transition relation* defined as

$$\widehat{E} = \{((s, R), (t, R'', a), (s', R')) \in \widehat{S} \times (\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \times \widehat{S} \\ : R \to_* R'' \xrightarrow{a} R' \text{ and } s' = \mathsf{Succ}(s, t, a) \text{ and } s + t \in \mathsf{bd}(R'')\}; \text{ and}$$

- $\widehat{S}_F$ is the set of *final vertices* defined as $\widehat{S}_F = \{(s, R) \in \widehat{S} \,:\, R \subseteq F\}$.

Boundary region graphs have the following property.

PROPOSITION 3.7.10. For every state in a boundary region graph $\widehat{\mathcal{T}} = (\widehat{S}, \widehat{E}, \widehat{S}_F)$, the set of reachable states is finite.

We prove Proposition 3.7.10 by constructing such reachable subgraph of a boundary region graph (Definition 3.7.12). Before we formally define the reachable subgraph of a boundary region graph from a certain state (s, R), we need to introduce some concepts.

The fractional signature of a state $(s, R) \in \widehat{S}$ of a boundary region graph is defined as $\wr(s, R)\int = \wr\nu\int$ where $s = (\ell, \nu)$. A state $(s = (\ell, \nu), R)$ of a boundary region graph is called a corner if its clock valuation ν is a corner. For a nonnegative integer $k \leq m$ we define the k-shift of a fractional signature $(f_0, f_1, \ldots, f_m)$ as the fractional signature $(f'_k, f'_{k+1}, \ldots, f'_m, f'_0, \ldots, f'_{k-1})$ such that for all nonnegative integers $i \leq m$ we have $f'_i = \wr f_i + 1 - f_k\int$. We say that a fractional signature $(f'_0, f'_1, \ldots, f'_n)$ is a subsequence of another fractional signature $(f_0, f_1, \ldots, f_m)$ if $n \leq m$ and for all nonnegative integers $i < n$ we have $f'_i \leq f'_{i+1}$; and for every nonnegative integer $i \leq n$ there exists a nonnegative integer $j \leq m$ such that $f'_i = f_j$.

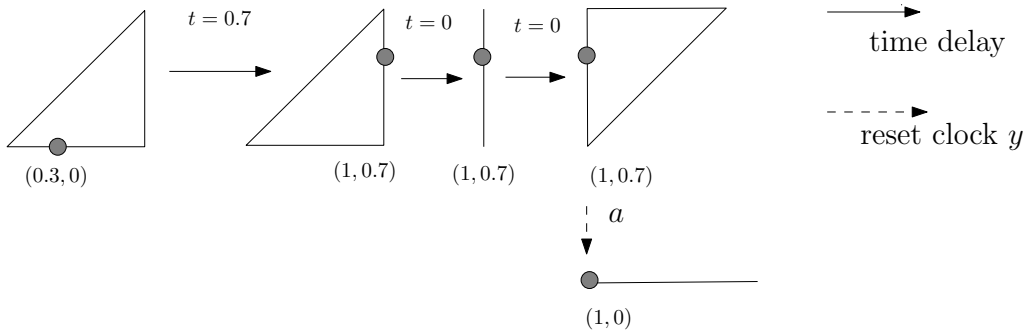


FIGURE 3.6. Evolution of regions in a boundary region graph of a timed automaton.

The following proposition is straightforward (see Example 3.7.14) from the definition of fractional signatures and boundary region graphs.

PROPOSITION 3.7.11. In a boundary region graph $\widehat{\mathcal{T}} = (\widehat{S}, \widehat{E}, \widehat{S}_F)$ if a state $q' \in \widehat{S}$ is reachable from a state $q \in \widehat{S}$ then fractional signature of q' is k-shift of a subsequence of the fractional signature of q.

DEFINITION 3.7.12 (Reachable Sub-Graph of Boundary Region Graph)**.** The *reachable subgraph* of a boundary region graph $\widehat{\mathcal{T}} = (\widehat{S}, \widehat{E}, \widehat{S}_F)$ from a state $q \in \widehat{S}$ is the finite graph $\widehat{\mathcal{T}}^q = (\widehat{S}^q, \widehat{E}, \widehat{S}_F^q)$, where $\widehat{S}^q$ is the finite set of states such that $q' \in \widehat{S}^q$ if $\wr q' \int$ is a k-shift of a subsequence of $\wr q \int$, and $\widehat{S}_F^q = \widehat{S}_F \cap \widehat{S}^q$.

From the definitions of the corner-point abstraction and the boundary region graph the following proposition is easy to verify.

PROPOSITION 3.7.13. Let $\mathcal{T}$ be a timed automaton. For every corner state $q \in \widehat{S}$ of the boundary region graph $\widehat{\mathcal{T}} = (\widehat{S}, \widehat{E}, \widehat{S}_F)$ the reachable subgraph $\widehat{\mathcal{T}}^q$ is same as the corner-point abstraction $\mathcal{T}_{\text{CP}}$.

EXAMPLE 3.7.14. A run of a boundary region graph with the initial state $(\ell, \nu_0 = (0.3, 0))$ is shown in Figure 3.6. Some clock valuations reachable from this state are $\nu_1 = (1, 0.7)$ and $\nu_4 = (1, 0)$. Fractional signatures of these valuations are $\wr \nu_0 \int = (0, 0.3)$, $\wr \nu_1 \int = (0, 0.7)$ and $\wr \nu_4 \int = (0)$. Notice that $\wr \nu_1 \int$ is 1-shift of $\wr \nu_0 \int$, while $\wr \nu_4 \int$ is a subsequence of $\wr \nu_0 \int$. Compare this figure with Figure 3.5 and you may notice that the run in Figure 3.5 is a run of boundary region graph whose initial state is a corner state.

Boundary region automata and boundary region graphs have turned out to be useful in solving a number of competitive optimisation problems on timed automata.

Using the boundary region graph abstraction, in Chapter 4, we show that a number of optimisation problems, including reachability-price and price-per-reward average optimisation problems, for more general concavely-priced timed automata and arbitrary initial states are decidable and PSPACE-complete.

In Chapter 5 we use boundary region automata to prove positional determinacy of reachability-time games and we give an algorithm to solve reachability-time games by solving reachability-time games on the corresponding boundary region automata.

In Chapter 6 we introduce another abstraction, called the closed region graph, whose set of states is same as the set of states of corresponding boundary region graph, while the transition relation is the union of the transition relations of the corresponding region graph and boundary region graph. We use closed region graphs to argue that, at every state of a timed automaton, the value of the average-time game is equal to the value of the average-time game, at a corresponding state, in its boundary region graph.

The definition of the boundary region automata and the boundary region graphs in these chapters are slightly modified to suit the problem under consideration. The differences, however, are superficial.

Part 2

Competitive Optimisation on Timed Automata

4
Noncompetitive Optimisation

There is no opponent.

Bruce Lee

This chapter is dedicated to the study of noncompetitive optimisation problems on timed automata. To cover a larger class of optimisation problems we work with *concavely-priced* timed automata, which are a generalisation of timed automata with price information. We consider minimisation problems for a variety of cost functions on concavely-priced timed automata in a uniform manner. For this purpose, we define the concept of *concave-regularity* of cost functions and show that minimisation problems for concave-regular cost functions are PSPACE-complete for timed automata with at least three clocks. Finally, we show that reachability time, reachability price, discounted time, discounted price, average time, average price, price-per-time average, and price-per-reward average cost functions are concave-regular.

This chapter assumes familiarity with Lipschitz-continuous functions and their properties, convex sets, concave and convex functions, quasiconcave and quasiconvex functions, and their properties. In Appendix B we give a short introduction to these concepts.

4.1. Concavely-Priced Timed Automata

Linearly-priced timed automata are extensions of timed automata with price information, and optimisation problems on linearly-priced timed automata are shown to be useful in modelling optimal scheduling and budgeting problems [**AM01, Bou06, BBL08**] of real-time systems. For instance, consider a situation where we have to model an optimal scheduling problem using a timed automaton whose different locations correspond to different resources being consumed. If the price-per-time-unit of every resource is constant then price function at every location is a linear function, and such a system can easily be modelled using a linearly-priced timed automata. However, if the price-per-time-unit is increasing or decreasing then linear prices are not adequate. To model such systems we propose concavely-priced timed automata and convexly-priced timed automata; both are generalisations of linearly-priced timed automata.

A *concavely-priced timed automaton* $(\mathcal{T}, \pi, K)$ is a priced timed automaton $(\mathcal{T}, \pi)$ with a constant $K > 0$, such that for all actions $a \in A$ and for all regions $R, R' \in \mathcal{R}$, the

function $\pi^a_{R,R'} : (s,t) \mapsto \pi(s,t,a)$ is concave and K-continuous on $D_{R,R'} = \{(s,t) \in S \times \mathbb{R}_\oplus : s \in R \text{ and } (s+t) \in R'\}$. A *convexly-priced timed automaton* is defined in an analogous manner.

A *concave price-reward timed automaton* $(\mathcal{T}, \pi, \varrho, K)$ is a timed automaton $\mathcal{T}$ with *price* and *reward* functions $\pi, \varrho : S \times \mathbb{R}_\oplus \times A \to \mathbb{R}$, and a constant $K > 0$, such that for all actions $a \in A$ and for all regions $R, R' \in \mathcal{R}$, the functions $(s,t) \mapsto \pi(s,t,a)$ and $(s,t) \mapsto \varrho(s,t,a)$ are K-continuous, and concave and convex, respectively, on $\{(s,t) \in S \times \mathbb{R}_\oplus : s \in R \text{ and } (s+t) \in R'\}$. A *convex price-reward timed automaton* is defined in an analogous manner. For technical convenience, we assume that the reward function is regionally non-Zeno (see Assumption 3.4.5).

In this chapter we only discuss minimisation problems on concavely-priced timed automata and concave price-reward timed automata. The discussion on maximisation problems on convexly-priced timed automata and convex price-reward timed automata is analogous and has been omitted. Fix a timed automaton $\mathcal{T}(L, C, S, A, E, \delta, \xi, F)$, price and reward functions $\pi, \varrho : S \times \mathbb{R}_\oplus \times A \to \mathbb{R}$, and a constant $K > 0$.

4.2. Optimisation Problems on Priced Timed Automata

Let $\mathsf{Cost} : \text{Runs} \to \mathbb{R}$ be a cost function that for every run $r \in \text{Runs}$ determines its cost $\mathsf{Cost}(r)$. We then define the *minimum cost* function $\mathsf{Cost}_* : S \to \mathbb{R}$, by:

$$\mathsf{Cost}_*(s) \quad = \quad \inf_{r \in \text{Runs}(s)} \{\mathsf{Cost}(r)\} = \inf_{\sigma \in \Sigma} \{\mathsf{Cost}(\text{Run}(s,\sigma))\}.$$

The *minimisation problem* for the cost function Cost is to compute the minimum cost $\mathsf{Cost}_*(s)$ for a given state $s \in S$. The decision version of the optimisation problem is as follows: "given a timed automaton $\mathcal{T}$, a state $s \in S$ and a number $D \in \mathbb{Q}$, determine whether $\mathsf{Cost}_*(s) \leq D$."

We say that a strategy $\sigma_* \in \Sigma$ is *optimal* for the cost function Cost if we have that $\mathsf{Cost}(\text{Run}(s,\sigma_*)) = \mathsf{Cost}_*(s)$. For a given $\varepsilon > 0$, we say that a strategy $\sigma \in \Sigma$ is ε-optimal if we have that $\mathsf{Cost}(\text{Run}(s,\sigma)) \leq \mathsf{Cost}_*(s) + \varepsilon$.

Let $r = \langle s_0, \tau_1, s_1, \tau_2, \ldots \rangle \in \text{Runs}$ be a run of the timed automaton $\mathcal{T}$, where $\tau_i = (t_i, a_i)$ for every positive integer i. Moreover, for π and ϱ, the price and reward functions, respectively, of a priced (or price-reward) timed automaton, and for every positive integer n, we define: $T_n(r) = \sum_{i=1}^n t_i$, $\pi_n(r) = \sum_{i=1}^n \pi(s_{i-1}, \tau_i)$, and $\varrho_n(r) = \sum_{i=1}^n \varrho(s_{i-1}, \tau_i)$.

The following list of cost functions gives rise to a number of corresponding optimisation problems.

(1) *Reachability price*. The reachability-price cost function is defined in the following way: for every run $r \in \text{Runs}$ we have

$$\mathsf{RP}(r) = \begin{cases} \pi_N(r), & \text{if } N = \mathsf{Stop}(r) < \infty, \\ \infty, & \text{otherwise.} \end{cases}$$

(2) *Discounted price*. The discounted-price cost function is defined in the following way: for every run $r \in \text{Runs}$ and every *discount factor* $\lambda \in (0,1)$ we have

$$\mathsf{DP}(\lambda)(r) = (1-\lambda)\sum_{i=1}^{\infty} \lambda^{i-1}\pi(s_{i-1},\tau_i).$$

(3) *Average price*. The average-price cost function is defined in the following way: for every run $r \in \text{Runs}$ we have

$$\mathsf{AP}(r) = \limsup_{n\to\infty} \frac{\pi_n(r)}{n}.$$

(4) *Price-per-time average*. The price-per-time average cost function is defined in the following way: for every run $r \in \text{Runs}$ we have

$$\mathsf{PTAvg}(r) = \limsup_{n\to\infty} \frac{\pi_n(r)}{T_n(r)}.$$

(5) *Price-per-reward average*. The price-per-reward average cost function is defined in the following way: for every run $r \in \text{Runs}$ we have

$$\mathsf{PRAvg}(r) = \limsup_{n\to\infty} \frac{\pi_n(r)}{\varrho_n(r)}.$$

The following is the main result of the chapter:

THEOREM 4.2.1. The minimisation problems for reachability price, discounted price, average price, price-per-time average, and price-per-reward average cost functions, for concavely-priced and concave price-reward timed automata, as appropriate, are PSPACE-complete.

The reachability problem for timed automata can be easily reduced, in logarithmic space, to the minimisation problems discussed above so, by Theorem 1.3.1, they are all PSPACE-hard. In the following sections we prove that they are all in PSPACE, and hence we establish the main Theorem 4.2.1.

The following sections are organised as follows. In Section 4.3 we revisit region graph $\widetilde{\mathcal{T}}$ and boundary region graph $\widehat{\mathcal{T}}$ of a timed automaton $\mathcal{T}$, and introduce a few notations. In Section 4.4 we introduce the concept of *concave-regular cost functions*, and prove that minimisation problems on $\mathcal{T}$ for such cost functions can be reduced to corresponding minimisation problems on $\widehat{\mathcal{T}}$. In Section 4.5 we show that reachability price, discounted price, average price, price-per-time average, and price-per-reward average cost functions are concave-regular.

4.3. Region Graphs

For the purpose of this chapter, we collectively refer to region graph and boundary region graph as region graphs. In Subsections 4.3.1 and 4.3.2 we introduce region graph and boundary region graph, respectively, of a concavely-priced timed automaton. In Subsection 4.3.3 we define reachability price, discounted price, average price, price-per-time average, and price-per-reward average minimisation problems on these graph.

4.3.1. Region Graph

Let us recall the definition of the region graph of a timed automaton introduced in Chapter 3. The *region graph* of a timed automaton $\mathcal{T} = (L, C, S, A, E, \delta, \xi, F)$ is a labelled transition system $\widetilde{\mathcal{T}} = (\widetilde{S}, \widetilde{E}, \widetilde{S}_F)$, where:

- $\widetilde{S}$ is the set of *states* defined as $\widetilde{S} = \{(s, R) \in S \times \mathcal{R} \,:\, s \in R\}$;
- $\widetilde{E}$ is the *labelled transition relation* defined as

$$\widetilde{E} = \{((s,R),(t,R'',a),(s',R')) \in \widetilde{S} \times (\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \times \widetilde{S} \\ : R \to_* R'' \xrightarrow{a} R' \text{ and } s' = \mathsf{Succ}(s,t,a) \text{ and } s + t \in R''\}; \text{ and}$$

- $\widetilde{S}_F \subseteq \widetilde{S}$ is the set of *final states* defined as $\widetilde{S}_F = \{(s, R) \in \widetilde{S} \,:\, R \subseteq F\}$.

We say that $\big((s,R),(t,R'',a),(s',R')\big)$ is a transition in $\widetilde{\mathcal{T}} = (\widetilde{S}, \widetilde{E}, \widetilde{S}_F)$ if we have that $\big((s,R),(t,R'',a),(s',R')\big) \in \widetilde{E}$. We then also say that there is an (t, R'', a)-transition from state (s, R) in $\widetilde{\mathcal{T}}$.

A run of $\widetilde{\mathcal{T}}$ is a sequence $\langle (s_0,R_0),(t_1,R_1',a_1),(s_1,R_1),(t_2,R_2',a_2),(s_2,R_2),\ldots\rangle$, such that for every $i \in \mathbb{N}$ we have that $\big((s_i,R_i),(t_{i+1},R_{i+1}'',a_{i+1}),(s_{i+1}',R_{i+1}')\big)$ is a transition in $\widetilde{\mathcal{T}}$. We write $\mathrm{Runs}^{\widetilde{\mathcal{T}}}$ for the set of runs of $\widetilde{\mathcal{T}}$, and for $(s, R) \in \widetilde{S}$, we write $\mathrm{Runs}^{\widetilde{\mathcal{T}}}(s, R)$ for the set of runs of $\widetilde{\mathcal{T}}$ whose initial state is (s, R).

The timed automaton $\mathcal{T}$ and the region graph $\widetilde{\mathcal{T}}$ are equivalent in the following sense.

PROPOSITION 4.3.1. For every $s \in S$ and $(t, a) \in \mathbb{R}_{\oplus} \times A$, there is a (t, a)-transition from s in $\mathcal{T}$ if and only if there is a $(t, [s + t], a)$-transition from $(s, [s])$ in $\widetilde{\mathcal{T}}$.

Let $(\mathcal{T}, \pi, K)$ be a concavely-priced timed automaton. We define the price function $\widetilde{\pi} : \widetilde{S} \times (\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \to \mathbb{R}$ in the following way. For $(s, R) \in \widetilde{S}$ and $(t, R'', a) \in \mathbb{R}_{\oplus} \times \mathcal{R} \times A$, such that there is a (t, R'', a)-transition from (s, R) in $\widetilde{\mathcal{T}}$, we define $\widetilde{\pi}\big((s,R),(t,R'',a)\big) = \pi(s,t,a)$. For a concave price-reward automaton $(\mathcal{T}, \pi, \varrho, K)$, we define functions $\widetilde{\pi}$ and $\widetilde{\varrho}$ in an analogous way.

4.3.2. Boundary Region Graph

Let us recall the definition of a *boundary region graph* from Chapter 3. The *boundary region graph* $\widehat{\mathcal{T}} = (\widehat{S}, \widehat{E}, \widehat{S}_F)$ of a timed automaton $\mathcal{T} = (L, C, S, A, E, \delta, \xi, F)$ is a labelled transition system, where:

- $\widehat{S}$ is the set of *states* defined as

$$\widehat{S} = \{(s, R) \in Q \times \mathcal{R} \,:\, s \in \mathsf{clos}(R)\};$$

- $\widehat{E}$ is the *labelled transition relation* defined as

$$\widehat{E} = \{((s,R),(t,R'',a),(s',R')) \in \widehat{S} \times (\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \times \widehat{S} \\ : R \to_* R'' \xrightarrow{a} R' \text{ and } s' = \mathsf{Succ}(s,t,a) \text{ and } s + t \in \mathsf{bd}(R'')\}; \text{ and}$$

– $\widehat{S}_F$ is the set of *final vertices* defined as $\widehat{S}_F = \{(s,R) \in \widehat{S} \,:\, R \subseteq F\}$.

For $(s,R),(s',R') \in \widehat{S}$ and $(t,R'',a) \in \mathbb{R}_\oplus \times \mathcal{R} \times A$, we say that $\big((s,R),(t,R'',a),(s',R')\big)$ is a transition in $\widehat{\mathcal{T}} = (\widehat{S},\widehat{E},\widehat{S}_F)$ if $\big((s,R),(t,R'',a),(s',R')\big) \in \widehat{E}$.

A run of $\widehat{\mathcal{T}}$ is a sequence $\langle (s_0,R_0),(t_1,R'_1,a_1),(s_1,R_1),(t_2,R'_2,a_2),\ldots\rangle$, such that for every $i \in \mathbb{N}$, we have that $\big((s_i,R_i),(t_{i+1},R''_{i+1},a_{i+1}),(s'_{i+1},R'_{i+1})\big)$ is a transition in $\widehat{\mathcal{T}}$. We write $\mathrm{Runs}^{\widehat{\mathcal{T}}}$ for the set of runs of $\widehat{\mathcal{T}}$, and for $(s,R) \in \widehat{S}$, we write $\mathrm{Runs}^{\widehat{\mathcal{T}}}(s,R)$ for the set of runs of $\widehat{\mathcal{T}}$ whose initial state is (s,R).

Let $(\mathcal{T},\pi,K)$ be a concavely-priced timed automaton. We define the price function $\widehat{\pi} : \widehat{S} \times (\mathbb{R}_\oplus \times \mathcal{R} \times A) \to \mathbb{R}$ in the following way. Recall that for $a \in A$ and $R,R'' \in \mathcal{R}$, the function $\pi^a_{R,R''} : (s,t) \mapsto \pi(s,a,t)$ defined on the set $D_{R,R''} = \{(s,t) : s \in R \text{ and } (s+t) \in R''\}$ is continuous. We write $\overline{\pi^a_{R,R''}}$ for the unique continuous extension of $\pi^a_{R,R''}$ to the closure $\overline{D_{R,R''}}$ of the set $D_{R,R''}$. For $(s,R) \in \widehat{S}$ and $(t,R'',a) \in \mathbb{R}_\oplus \times \mathcal{R} \times A$, such that there is an (t,R'',a)-transition from (s,R) in $\widehat{\mathcal{T}}$, we define

$$\widehat{\pi}\big((s,R),(t,R'',a)\big) = \overline{\pi^a_{R,R''}}(s,t).$$

Thanks to the following proposition we can, and sometimes will, abuse notation by writing $\pi\big((s,R),(t,R,a)\big)$ instead of $\widetilde{\pi}\big((s,R),(t,R,a)\big)$ or $\widehat{\pi}\big((s,R),(t,R,a)\big)$.

PROPOSITION 4.3.2. If (t,R,a) is a transition from the state (s,R) in both the region graph and the boundary region graph then $\widetilde{\pi}\big((s,R),(t,R,a)\big) = \widehat{\pi}\big((s,R),(t,R,a)\big)$.

For a concave price-reward automaton $(\mathcal{T},\pi,\varrho,K)$, we define functions $\widehat{\pi}$ and $\widehat{\varrho}$ in an analogous way.

4.3.3. Optimisation Problems on the Region Graphs

Before we discuss optimisation problems on region graph and boundary region graph, we define pre-runs of a timed automaton. The concept of pre-runs is a generalisation of the runs of $\widetilde{\mathcal{T}}$ and $\widehat{\mathcal{T}}$, and it allows us to define cost functions on region graphs in a general setting.

DEFINITION 4.3.3 (Pre-Run)**.** A *pre-run* of $\mathcal{T}$ is a sequence

$$\langle (s_0,R_0),(t_1,R'_1,a_1),(s_1,R_1),(t_1,R'_2,a_2),\ldots\rangle \in (Q\times\mathcal{R}) \times ((\mathbb{R}_\oplus \times \mathcal{R} \times A) \times (Q \times \mathcal{R}))^\omega,$$

such that $s_{i+1} = \mathsf{Succ}(s_i,(t_{i+1},a_{i+1}))$ and $R_i \to_* R'_{i+1} \xrightarrow{a_{i+1}} R_{i+1}$ for every $i \in \mathbb{N}$.

Observe that a pre-run follows the wait, the action, and the clock reset discipline, but not necessarily the guards. To be more precise, in a pre-run $\langle (s_0,R_0),(t_1,R'_1,a_1),(s_1,R_1),\ldots\rangle$, for every index i, we do not require that the action a_{i+1} is enabled in the state $s_i + t_{i+1}$, or that the state s_i is in the closure of region R_i.

We write $\mathsf{PreRuns}$ for the set of pre-runs and $\mathsf{PreRuns}(s,R)$ for the set of pre-runs starting from (s,R). We have the following relations among the sets of runs of region graphs and the set of pre-runs:

$$\mathrm{Runs}^{\widehat{\mathcal{T}}} \subseteq \mathsf{PreRuns}, \text{ and } \mathrm{Runs}^{\widehat{\mathcal{T}}}(s,R) \subseteq \mathsf{PreRuns}(s,R) \text{ for all} (s,R) \in \widehat{S};$$
$$\mathrm{Runs}^{\widetilde{\mathcal{T}}} \subseteq \mathsf{PreRuns}, \text{ and } \mathrm{Runs}^{\widetilde{\mathcal{T}}}(s,R) \subseteq \mathsf{PreRuns}(s,R) \text{ for all} (s,R) \in \widetilde{S}.$$

For a cost function $\mathsf{Cost} : \mathsf{PreRuns} \to \mathbb{R}$, the *minimum cost* functions $\mathsf{Cost}^{\widetilde{\mathcal{T}}}_* : \widetilde{S} \to \mathbb{R}$ and $\mathsf{Cost}^{\widehat{\mathcal{T}}}_* : \widehat{S} \to \mathbb{R}$ for region graph $\widetilde{\mathcal{T}}$ and boundary region graph $\widehat{\mathcal{T}}$, respectively, are defined by:

$$\mathsf{Cost}^{\widetilde{\mathcal{T}}}_*(s,R) = \inf_{r \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}(s,R)} \mathsf{Cost}(r), \quad \text{and} \quad \mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,R) = \inf_{r \in \mathrm{Runs}^{\widehat{\mathcal{T}}}(s,R)} \mathsf{Cost}(r).$$

The corresponding *minimisation problems* for $\widetilde{\mathcal{T}}$ and $\widehat{\mathcal{T}}$ are: given a state $s \in S$ and a number $D \in \mathbb{Q}$, determine whether $\mathsf{Cost}^{\widetilde{\mathcal{T}}}_*(s,[s]) \leq D$ and $\mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s]) \leq D$, respectively.

Let $r = \langle (s_0,R_0),(t_1,R'_1,a_1),(s_1,R_1),(t_2,R'_2,a_2),\ldots\rangle$ be a run of $\widetilde{\mathcal{T}}$ or $\widehat{\mathcal{T}}$. We define $\mathsf{Stop}(r) = \inf\{i : R_i \subseteq F\}$. Also, for all $n \in \mathbb{N}$ we define $T_n(r) = \sum_{i=1}^{n} t_i$, $\pi_n(r) = \sum_{i=1}^{n} \pi((s_{i-1},R_{i-1}),(R'_i,t_i,a_i))$, and $\varrho_n(r) = \sum_{i=1}^{n} \varrho((s_{i-1},R_{i-1}),(R'_i,t_i,a_i))$. With those notations, we define the reachability price, discounted price, average price, price-per-time average, and price-per-reward average cost functions, on the sets of runs of $\widetilde{\mathcal{T}}$ and $\widehat{\mathcal{T}}$, in exactly the same way as for runs of the timed automaton $\mathcal{T}$; see Section 4.2.

The following is an easy corollary of Proposition 4.3.1.

PROPOSITION 4.3.4. If Cost is any of the reachability price, discounted price, average price, price-per-time average, or price-per-reward average cost functions, then for all $s \in S$, we have $\mathsf{Cost}^{\mathcal{T}}_*(s) = \mathsf{Cost}^{\widetilde{\mathcal{T}}}_*(s,[s])$.

The following theorem is one of the main technical results of the chapter.

THEOREM 4.3.5. If Cost is any of the reachability price, discounted price, average time, average price, price-per-time average, or price-per-reward average cost functions, then for all $s \in S$, we have $\mathsf{Cost}^{\widetilde{\mathcal{T}}}_*(s,[s]) = \mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s])$.

From Proposition 3.7.10 we know that for every state $s \in S$, the number of states reachable from s in the boundary region graph $\widehat{\mathcal{T}}$ is finite. By Theorem 2.2.1, it follows that optimal positional strategies exist in $\widehat{\mathcal{T}}$ for all the above-mentioned cost functions. Therefore, and since a run from a state according to a positional strategy in $\widehat{\mathcal{T}}$ can be guessed, and its cost computed, in PSPACE (with respect to the size of the input, i.e., a timed automaton $\mathcal{T}$), it suffices to prove Theorem 4.3.5 in order to obtain the main Theorem 4.2.1. We dedicate the remaining two sections to the proof of Theorem 4.3.5.

4.4. Correctness of the Boundary Region Graph Abstraction

In this section we define *concave-regular* cost functions and show that optimisation problems for concave-regular cost functions on concavely-priced timed automaton can be reduced to similar optimisation problem on boundary region graph.

4.4.1. Approximations of Cost Functions

For a run $r \in \mathsf{PreRuns}$ and $n \in \mathbb{N}$, we write $\mathrm{Prefix}(r, n)$ for the finite pre-run consisting of the first n transitions of r. We write $\mathsf{PreRuns}_n$ for the set of pre-runs of length $n \in \mathbb{N}$, and we write $\mathrm{Runs}_n^{\widetilde{\mathcal{T}}}$ and $\mathrm{Runs}_n^{\widehat{\mathcal{T}}}$ for the sets of runs of $\widetilde{\mathcal{T}}$ and $\widehat{\mathcal{T}}$, respectively, of length $n \in \mathbb{N}$.

DEFINITION 4.4.1 (Approximation of Cost Function)**.** We say that a sequence of functions $\langle \mathsf{Cost}_n : \mathsf{PreRuns}_n \to \mathbb{R} \rangle_{n \in \mathbb{N}}$ approximates a cost function $\mathsf{Cost} : \mathrm{Runs}^{\widetilde{\mathcal{T}}} \to \mathbb{R}$ or $\mathsf{Cost} : \mathrm{Runs}^{\widehat{\mathcal{T}}} \to \mathbb{R}$, respectively, if for all $r \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}$, or for all $r \in \mathrm{Runs}^{\widehat{\mathcal{T}}}$, respectively, we have that $\mathsf{Cost}(r) = \limsup_{n \to \infty} \mathsf{Cost}_n(\mathrm{Prefix}(r, n))$.

For a run $r \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}$ we sometimes abuse notation—for the sake of brevity—by writing $\mathsf{Cost}_n(r)$ instead of $\mathsf{Cost}_n(\mathrm{Prefix}(r, n))$; the same applies to runs in $\mathrm{Runs}^{\widehat{\mathcal{T}}}$.

For a sequence of functions $\langle \mathsf{Cost}_n : \mathsf{PreRuns}_n \to \mathbb{R} \rangle_{n \in \mathbb{N}}$ and for every $n \in \mathbb{N}$, we define the *n-step-minimum-cost functions* $\mathsf{Cost}_{n,*}^{\widetilde{\mathcal{T}}} : \widetilde{S} \to \mathbb{R}$ and $\mathsf{Cost}_{n,*}^{\widehat{\mathcal{T}}} : \widehat{S} \to \mathbb{R}$, by:

$$\mathsf{Cost}_{n,*}^{\widetilde{\mathcal{T}}}(s, R) = \inf_{r \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}(s,R)} \mathsf{Cost}_n(r), \text{ and } \mathsf{Cost}_{n,*}^{\widehat{\mathcal{T}}}(s, R) = \inf_{r \in \mathrm{Runs}^{\widehat{\mathcal{T}}}(s,R)} \mathsf{Cost}_n(r).$$

4.4.2. Run Types and Related Concepts

In the next subsection we show that if approximation of a cost function is *quasiconcave* (for definition, see Chapter B) in certain sense, then there exists $N \in \mathbb{N}$ such that for all $n \geq N$, we have that for every run r in $\widetilde{\mathcal{T}}$, there exists a run r' in $\widehat{\mathcal{T}}$ with $\mathsf{Cost}_n(r') \leq \mathsf{Cost}_n(r)$. To prove that result we need to introduce the concept of run types.

DEFINITION 4.4.2 (Run Types)**.** A *run type* is a sequence

$$\langle R_0, (R_1', a_1), R_1, (R_2', a_2), \ldots \rangle \in \mathcal{R} \times ((\mathcal{R} \times A) \times \mathcal{R})^\omega$$

such that for every $i \in \mathbb{N}$ we have that $R_i \to_* R_{i+1}' \xrightarrow{a} R_{i+1}$.

We say that a pre-run $r = \langle (s_0, R_0), (t_1, R_1', a_1), (s_1, R_1), (t_1, R_2', a_2), (s_2, R_2), \ldots \rangle$ is of type $\langle R_0, (R_1', a_1), R_1, (R_2', a_2), \ldots \rangle$. Also, we define the type of a run $r = \langle s_0, (t_1, a_1), s_1, (t_2, a_2), \ldots \rangle$ of a timed automaton $\mathcal{T}$ as the run type $\langle R_0, (R_1', a_1), R_1, (R_2', a_2), \ldots \rangle$, where $R_i = [s_i]$ and $R_{i+1}' = [s_i + t_{i+1}]$ for all $i \in \mathbb{N}$.

We write Types for the set of run types, and we write $\mathrm{Types}(R)$ for the set of run types starting from region $R \in \mathcal{R}$. We say that a run type $\Lambda = \langle R_0, (R_1', a_1), R_1, (R_2', a_2), \ldots \rangle$ is *positional* if for every index $i, j \in \mathbb{N}$ we have that $R_i = R_j$ implies $(R_{i+1}', a_{i+1}) = (R_{j+1}', a_{j+1})$.

Given a run type $\Lambda \in \mathrm{Types}(R)$, a state $s \in R$, and a sequence of timing delays $\bar{t} = \langle t_1, t_2, \ldots \rangle$, we write $\mathsf{PreRun}_s^\Lambda(\bar{t})$ for the pre-run of type Λ which starts in state s and whose successive time delays follow the sequence $\bar{t} = \langle t_1, t_2, \ldots \rangle$. Formally, given a run type $\Lambda = \langle R_0, (R_1', a_1), R_1, (R_2', a_2), \ldots \rangle \in \mathrm{Types}$, a starting state $s \in R_0$, and a tuple $\bar{t} = \langle t_1, t_2, \ldots \rangle \in \mathbb{R}_\oplus^\omega$, we define

$$\mathsf{PreRun}_s^\Lambda(\bar{t}) = \langle (s_0, R_0), (t_1, R_1', a_1), (s_1, R_1), (t_1, R_2', a_2), \ldots \rangle,$$

where $s_0 = s$ and $s_{i+1} = \mathsf{Succ}(s_i, (t_{i+1}, a_{i+1}))$ for every $i \in \mathbb{N}$.

Given a run type $\Lambda \in \mathrm{Types}(R)$, a state $s \in R$, and a finite sequence of timing delays $\bar{t} = \langle t_1, t_2, \ldots, t_n \rangle$, we write $\mathsf{PreRun}^{\Lambda}_{n,s}(\bar{t})$ for the finite pre-run of type Λ which starts in state s and whose successive time delays follow the sequence $\langle t_1, t_2, \ldots, t_n \rangle$. Formally, given a run type $\Lambda = \langle R_0, (R'_1, a_1), R_1, (R'_2, a_2), R_2, \ldots \rangle$, a state $s \in R_0$, and a tuple $(t_1, t_2, \ldots, t_n) \in \mathbb{R}^n_{\oplus}$, we define

$$\mathsf{PreRun}^{\Lambda}_{n,s}(t_1, t_2, \ldots, t_n) = \mathrm{Prefix}(\mathsf{PreRun}^{\Lambda}_{s}(\bar{t}), n),$$

where the first n elements of $\bar{t} \in \mathbb{R}^{\omega}_{\oplus}$ are $t_1, t_2, \ldots, t_n$.

4.4.3. Runs in $\widetilde{\mathcal{T}}$ and $\widehat{\mathcal{T}}$ of same Type

We compare the runs in region graph $\widetilde{\mathcal{T}}$ and in boundary region graph $\widehat{\mathcal{T}}$ and prove two key results: Lemma 4.4.6 and Proposition 4.4.7. Lemma 4.4.6 states that if Cost_n is quasiconcave in certain sense then for every run $r \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}(s, [s])$, there exists a run $r' \in \mathrm{Runs}^{\widehat{\mathcal{T}}}(s, [s])$ of same type such that $\mathsf{Cost}_n(r') \leq \mathsf{Cost}_n(r)$. On the other hand, Proposition 4.4.7 states that for every $r = \langle (s_0, R_0), (t_1, R'_1 a_1), \ldots \rangle \in \mathrm{Runs}^{\widehat{\mathcal{T}}}(s, [s])$ and number $\varepsilon > 0$, there exists $r' = \langle (s'_0, R_0), (t'_1, R'_1, a_1), \ldots \rangle \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}(s, [s])$ of same type such that for all $i \geq 1$, we have $|t_i - t'_i| < \varepsilon$ and $\|s_i - s'_i\|_\infty < \varepsilon$.

To prove Lemma 4.4.6 we need to show that all the finite runs (or length n) in region graph $\widetilde{\mathcal{T}}$ of type Λ form a convex polytope $\Delta^{\Lambda}_{n,s}$, whose vertices correspond to finite runs in boundary region graph $\widehat{\mathcal{T}}$ of same type. We then invoke a well-known result from non-linear programming to prove the desired result.

We define $\Delta^{\Lambda}_{n,s} \subseteq \mathbb{R}^n_{\oplus}$ to consist of the tuples $(t_1, t_2, \ldots, t_n) \in \mathbb{R}^n_{\oplus}$, such that the pre-run $\mathsf{PreRun}^{\Lambda}_{n,s}(t_1, t_2, \ldots, t_n)$ is a finite run in the region graph $\widetilde{\mathcal{T}}$, i.e., $\mathsf{PreRun}^{\Lambda}_{n,s}(t_1, t_2, \ldots, t_n) \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}_n$.

Claim 4.4.3. For every state $s \in S$, a run type $\Lambda \in \mathrm{Types}([s])$, and $n \in \mathbb{N}$, the set $\Delta^{\Lambda}_{n,s}$ is a convex polytope.

Proof. Let $\Lambda = \langle R_0, (R'_1, a_1), R_1, (R'_2, a_2), R_2, \ldots \rangle$ be a run type. By definition, we have that the tuple $(t_1, t_2, \ldots, t_n) \in \Delta^{\Lambda}_{n,s}$ if the pre-run

$$\mathsf{PreRun}^{\Lambda}_{n,s}(t_1, t_2, \ldots, t_n) = \langle (s_0, R_0), (t_1, R'_1, a_1), (s_1, R_1), \ldots, (t_n, R'_n, a_n), (s_n, R_n) \rangle,$$

is a finite run in the region graph $\widetilde{\mathcal{T}}$. It implies that if $(t_1, t_2, \ldots, t_n) \in \Delta^{\Lambda}_{n,s}$ then the tuple $(t_1, t_2, \ldots, t_n)$ satisfy the following system of linear inequalities.

- For every positive integer $i \leq n$, if $R'_i \in \mathcal{R}_{\mathrm{Thin}}$ then we have one linear equality of the form $t_i = b_i - s_{i-1}(c_i)$, where $b_i \in \mathbb{N}$ and $c_i \in C$ are such that for every $s' \in R'_i$ we have $s'(c_i) = b_i$.
- For every positive integer $i \leq n$, if $R'_i \in \mathcal{R}_{\mathrm{Thick}}$ then we have two inequalities of the form $t_i < b_i - s_{i-1}(c_i)$ and $t_i > b'_i - s_{i-1}(c'_i)$, where $b_i, b'_i \in \mathbb{N}$ and $c_i, c'_i \in C$ are such that for every $s' \in R \leftarrow_{+1} R'_i$ we have $s'(c_i) = b_i$, and every $s' \in R \rightarrow_{+1} R'_i$ we have $s'(c'_i) = b'_i$.
- For every nonnegative integer $i \leq n$ we have $t_i \geq 0$.

– For every nonnegative integer $i \leq n$ and for every $c \in C$, the variable $s_i(c)$ is the following linear function of the initial state s and variables $t_1, t_2, \ldots, t_n$:

$$s_i(c) = \begin{cases} s(c), & \text{if } i = 0, \\ 0, & \text{if } i \neq 0 \text{ and } c \in \xi(a_i), \\ s_{i-1}(c) + t_i, & \text{if } i \neq 0 \text{ and } c \notin \xi(a_i). \end{cases}$$

∎

PROPOSITION 4.4.4. Let $R \in \mathcal{R}$, $\Lambda \in \text{Types}(R)$, $s \in R$, and $n \in \mathbb{N}$. There is a 1-to-1 correspondence between runs—starting from s, of type Λ, and of length n—in $\widehat{\mathcal{T}}$, and vertices of the convex polytope $\overline{\Delta^{\Lambda}_{n,s}}$.
More precisely, $r = \langle (s_0, R_0), (t_1, R'_1, a_1), (s_1, R_1), \ldots, (t_n, R'_n, a_n), (s_n, R_n) \rangle$ is a run (of type Λ) in $\widehat{\mathcal{T}}$ if and only if there is a vertex $(t_1, t_2, \ldots, t_n)$ of $\overline{\Delta^{\Lambda}_{n,s}}$, such that $r = \mathsf{PreRun}^{\Lambda}_{n,s}(t_1, t_2, \ldots, t_n)$.

The following is a well-known result (e.g., Martos [**Mar65**], Bertsekas et al. [**BNO03**]).

PROPOSITION 4.4.5. Let $f : \Delta \to \mathbb{R}$ be a continuous quasiconcave function, where $\Delta \subseteq \mathbb{R}^n$ is a polytope. Let $\overline{f}$ be the unique continuous extension of f to the closure $\overline{\Delta}$ of Δ.

– There exists a vertex v of $\overline{\Delta}$, such that $\overline{f}(v) = \inf_{x \in \Delta} f(x)$.
– For every $\varepsilon > 0$, there exists $x \in \Delta$, such that $f(x) \leq \overline{f}(v) + \varepsilon$.

Let a sequence $\langle \mathsf{Cost}_n \rangle_{n \in \mathbb{N}}$ approximate a cost function Cost. We define the function $\mathsf{Cost}^{\Lambda}_{n,s} : \Delta^{\Lambda}_{n,s} \to \mathbb{R}$ by $\mathsf{Cost}^{\Lambda}_{n,s}(t_1, t_2, \ldots, t_n) = \mathsf{Cost}_n(\mathsf{PreRun}^{\Lambda}_{n,s}(t_1, t_2, \ldots, t_n))$. The following lemma can be derived from Propositions 4.4.4 and 4.4.5.

LEMMA 4.4.6. Let $\mathsf{Cost}^{\Lambda}_{n,s}$ be quasiconcave on $\Delta^{\Lambda}_{n,s}$.

(1) For every run $\widetilde{r} \in \text{Runs}^{\widetilde{\mathcal{T}}}(s, [s])$ of type Λ, and for every $n \in \mathbb{N}$, there is a run $\widehat{r} \in \text{Runs}^{\widehat{\mathcal{T}}}(s, [s])$ of type Λ, such that $\mathsf{Cost}_n(\widehat{r}) \leq \mathsf{Cost}_n(\widetilde{r})$.
(2) For every run $\widehat{r} \in \text{Runs}^{\widehat{\mathcal{T}}}(s, [s])$, and for every $\varepsilon > 0$, there is a run $\widetilde{r} \in \text{Runs}^{\widetilde{\mathcal{T}}}(s, [s])$ of type Λ, such that $\mathsf{Cost}_n(\widetilde{r}) \leq \mathsf{Cost}_n(\widehat{r}) + \varepsilon$.

Let us consider two pre-runs $r = \langle (s_0, R_0), (t_1, R'_1, a_1), (s_1, R_1), (t_2, R'_2, a_2), \ldots \rangle$ and $r' = \langle (s'_0, R_0), (t'_1, R'_1, a_1), (s'_1, R_1), (t'_2, R'_2, a_2), \ldots \rangle$ of the same type. We define $r - r' = (s_0 - s'_0, t_1 - t'_1, s_1 - s'_1, t_2 - t'_2, \ldots)$, where for all $i \in \mathbb{N}$, the expression $(s_i - s'_i)$ stands for the finite sequence $\langle s_i(c) - s'_i(c) \rangle_{c \in C}$. For a sequence $\overline{x} = \langle x_i \rangle_{i \in \mathbb{N}} \in \mathbb{R}^{\omega}$ of reals, we define $\|\overline{x}\|_{\infty} = \sup_{i \in \mathbb{N}} |x_i|$.

PROPOSITION 4.4.7. For every run $\widehat{r} \in \text{Runs}^{\widehat{\mathcal{T}}}(s, [s])$, and for every $\varepsilon > 0$, and there is a run $\widetilde{r} \in \text{Runs}^{\widetilde{\mathcal{T}}}(s, [s])$, of the same type as $\widehat{r}$, such that $\|\widehat{r} - \widetilde{r}\|_{\infty} \leq \varepsilon$.

PROOF. Let $\widehat{r} = \langle (s_0, R_0), (t_1, R'_1, a_1), (s_1, R_1), (t_2, R'_2, a_2), \ldots \rangle \in \text{Runs}^{\widehat{\mathcal{T}}}$. Note that since $\widehat{r} \in \text{Runs}^{\widehat{\mathcal{T}}}$, for every $i \in \mathbb{N}$, there are $b_i \in [\![\mathcal{K}]\!]_{\mathbb{N}}$ and $c_i \in C$, such that $t_i = b_i - s_{i-1}(c_i)$. Let $\varepsilon' > 0$ and $\widetilde{r} = \langle (s'_0 = s_0, R_0), (t'_1, R'_1, a_1), (s'_1, R_1), (t'_2, R'_2, a_2), \ldots \rangle \in \text{Runs}^{\widetilde{\mathcal{T}}}$ (of the same type as $\widehat{r}$) be such that for all $i \in \mathbb{N}$, we choose $t'_i \in \mathbb{R}_{\oplus}$ so that $|t'_i - (b_i - s'_{i-1}(c_i))| < \varepsilon'$. We claim

that for every $\varepsilon > 0$ there exists a choice of $\varepsilon' > 0$ such that $\|\widehat{r} - \widetilde{r}\|_\infty \leq \varepsilon$. The claim is based on the following facts:

- For every transition $((s,R),(t,R'',a),(s',R')) \in \widehat{E}$, state $(s_r,R) \in \widetilde{S}$, and positive real $\delta > 0$, there exists $t_r \in \mathbb{R}_\oplus$ such that $((s_r,R),(t_r,R'',a),(s'_r,R')) \in \widetilde{E}$ and $|(b - s_r(c)) - t_r| < \delta$, where $b \in [\![\mathcal{K}]\!]_{\mathbb{N}}$ and $c \in C$ are such that $b - s(c) = t$. Moreover, for all $c' \in C$, if $c' \in \mathsf{Reset}(a)$ then we have $s'(c') = s'_r(c') = 0$, else we have $|s'(c') - s'_r(c')| < 2 \cdot \|s - s_r\|_\infty + \delta$.
- In a boundary region graph every clock is reset at least once in every $|\mathcal{R}|$ transitions.

Hence if we chose $\varepsilon' < \varepsilon/(2^{|\mathcal{R}|} - 1)$ then it follows that for all $i \in \mathbb{N}$, we have $\|s_i - s'_i\|_\infty < \varepsilon$, and hence $\|\widehat{r} - \widetilde{r}\|_\infty \leq \varepsilon$. ∎

4.4.4. Concave-Regular Cost Functions

We are now in a position to define concave-regular cost functions.

DEFINITION 4.4.8 (Concave-Regular Cost Functions)**.** A cost function $\mathsf{Cost} : \mathsf{PreRuns} \to \mathbb{R}$ is *concave-regular* if it satisfies the following properties.

(1) *(Quasiconcavity).* For every region $R \in \mathcal{R}$ and for every run type $\Lambda \in \mathrm{Types}(R)$, there is $N \in \mathbb{N}$, such that for every state $s \in R$ and for every $n \geq N$, the function $\mathsf{Cost}^\Lambda_{n,s}$ is quasiconcave on $\Delta^\Lambda_{n,s}$.
(2) *(Regular Lipschitz-continuity).* There is a constant $\kappa \geq 0$, such that for every region $R \in \mathcal{R}$ and for every *positional run type* $\Lambda \in \mathrm{Types}(R)$, there is $N \in \mathbb{N}$, such that for every state $s \in R$ and for every $n \geq N$, the function $\mathsf{Cost}^\Lambda_{n,s}$ is κ-continuous on $\Delta^\Lambda_{n,s}$.
(3) *(Positional optimality).* There is a positional optimal strategy for Cost in $\widehat{\mathcal{T}}$.
(4) *(Uniform convergence).* For every $s \in S$ we have that

$$\mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s]) = \limsup_{n\to\infty} \mathsf{Cost}^{\widehat{\mathcal{T}}}_{n,*}(s,[s]).$$

Notice that properties 3 and 4 refer to the properties of Cost function in the boundary region graph.

THEOREM 4.4.9. If $\mathsf{Cost} : \mathsf{PreRuns} \to \mathbb{R}$ is concave-regular then for all states $s \in S$, we have $\mathsf{Cost}^{\widetilde{\mathcal{T}}}_*(s,[s]) = \mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s])$.

PROOF. Let $\mathsf{Cost} : \mathsf{PreRuns} \to \mathbb{R}$ is concave-regular. This proof is in two parts.

- First we prove that for all $s \in S$, we have $\mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s]) \leq \mathsf{Cost}^{\widetilde{\mathcal{T}}}_*(s,[s])$. It suffices to show that for every run $\widetilde{r} \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}(s,[s])$, we have $\mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s]) \leq \mathsf{Cost}(\widetilde{r})$.

  Let $\widetilde{r} \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}(s,[s])$ be a run in $\widetilde{\mathcal{T}}$ of type Λ. By the *quasiconcavity property* of Cost, there is $N \in \mathbb{N}$, such that for all $n \geq N$, the function $\mathsf{Cost}^\Lambda_{n,s}$ is quasiconcave on $\Delta^\Lambda_{n,s}$. Hence—by the first part of Lemma 4.4.6—for every $n \geq N$, there is a run $\widehat{r_n} \in \mathrm{Runs}^{\widehat{\mathcal{T}}}(s,[s])$ of type Λ, such that $\mathsf{Cost}_n(\widehat{r_n}) \leq \mathsf{Cost}_n(\widetilde{r})$. Since $\widehat{r_n} \in \mathrm{Runs}^{\widehat{\mathcal{T}}}(s,[s])$ and by definition $\mathsf{Cost}^{\widehat{\mathcal{T}}}_{n,*}(s,[s]) = \inf_{r \in \mathrm{Runs}^{\widehat{\mathcal{T}}}(s,[s])} \mathsf{Cost}_n(r)$, we

get

$$\mathsf{Cost}^{\widehat{\mathcal{T}}}_{n,*}(s,[s]) \leq \mathsf{Cost}_n(\widetilde{r}).$$

Taking the limit supremum of both sides of the last inequality yields:

$$\limsup_{n\to\infty} \mathsf{Cost}^{\widehat{\mathcal{T}}}_{n,*}(s,[s]) \leq \limsup_{n\to\infty} \mathsf{Cost}_n(\widetilde{r}) = \mathsf{Cost}(\widetilde{r}).$$

By the *uniform convergence property* of Cost and the previous inequality, it follows that $\mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s]) \leq \mathsf{Cost}(\widetilde{r})$.

– Next we prove that for all $s \in S$, we have $\mathsf{Cost}^{\widetilde{\mathcal{T}}}_*(s,[s]) \leq \mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s])$. It suffices to argue that for every $s \in S$ and $\varepsilon > 0$, there is a run $\widetilde{r} \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}(s,[s])$, such that $|\mathsf{Cost}(\widetilde{r}) - \mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s])| \leq \varepsilon$.

Let $\widehat{r} \in \mathrm{Runs}^{\widehat{\mathcal{T}}}(s,[s])$ be such that $\mathsf{Cost}(\widehat{r}) = \mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s])$ and the type Λ of r is positional. The existence of such a run $r \in \mathrm{Runs}^{\widehat{\mathcal{T}}}(s,[s])$ follows from *positional optimality property* of Cost.

Let $\varepsilon > 0$ and let $\widetilde{r} \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}(s,[s])$ be a run—of the same type, Λ, as the run $\widehat{r} \in \mathrm{Runs}^{\widehat{\mathcal{T}}}(s,[s])$—such that $\|\widetilde{r} - \widehat{r}\|_\infty \leq \varepsilon'$, for some $\varepsilon' > 0$ to be chosen later; existence of such $\widetilde{r} \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}(s,[s])$ follows from Proposition 4.4.7.

Since the run type Λ of the runs $\widehat{r}$ and $\widetilde{r}$ is positional, by the *regular Lipschitz-continuity property* of Cost, there is $\kappa \geq 0$ and $N \in \mathbb{N}$, such that for all $n \geq N$, we have:

$$|\mathsf{Cost}_n(\widetilde{r}) - \mathsf{Cost}_n(\widehat{r})| \leq \kappa \|\widetilde{r} - \widehat{r}\|_\infty \leq \kappa\varepsilon'.$$

Hence—by choosing $\varepsilon' > 0$ so that $\varepsilon' \leq \varepsilon/\kappa$—we obtain that:

$$|\mathsf{Cost}_n(\widetilde{r}) - \mathsf{Cost}_n(\widehat{r})| \leq \varepsilon,$$

for every $n \geq N$. Recall, however, that we have chosen $\widehat{r} \in \mathrm{Runs}^{\widehat{\mathcal{T}}}(s,[s])$ such that $\mathsf{Cost}(\widehat{r}) = \mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s])$. From the last two inequalities it follows that $|\mathsf{Cost}(\widetilde{r}) - \mathsf{Cost}^{\widehat{\mathcal{T}}}_*(s,[s])| \leq \varepsilon$.

The proof is now complete. ∎

4.5. Concave-Regularity of Cost Functions

So far we have shown that, for concave-regular cost functions, the optimal value of the cost function in $\mathcal{T}$ is equal to the optimal value in $\widehat{\mathcal{T}}$. To complete the proof of Theorem 4.3.5, we prove the following result.

THEOREM 4.5.1. Reachability price, discounted price, average price, price-per-time average, and price-per-reward average cost functions are concave-regular for the concavely-priced (or the concave price-reward, as appropriate) timed automata.

This section is dedicated to the proof of Theorem 4.5.1. We begin by showing quasiconcavity and Lipschitz-continuity properties of the above-mentioned cost functions,

and then in Subsection 4.5.2 we comment on their positional optimality and uniform convergence.

4.5.1. Quasiconcavity and Regular Lipschitz-Continuity

We begin this subsection by showing the quasiconcavity and Lipschitz-continuity of total accumulated price in n steps, and using that we prove quasiconcavity and Lipschitz-continuity of reachability-price, discounted-price, and price-per-reward average cost functions.

4.5.1.1. Total Accumulated Price in Finite Steps

Let $\Lambda = \langle R_0, (R'_1, a_1), R_1, \ldots \rangle$ be a run type. For a state $s \in R_0$ and $\bar{t} = (t_1, t_2, \ldots, t_n) \in \mathbb{R}^n_\oplus$, we define $\pi^\Lambda_n : S \times \mathbb{R}^n_\oplus \to \mathbb{R}$ and $\pi^\Lambda_{n,s} : \mathbb{R}^n_\oplus \to \mathbb{R}$ in the following manner:

$$\pi^\Lambda_n(s, \bar{t}) = \pi_n(\mathsf{PreRun}^\Lambda_s(\bar{t})), \text{ and } \pi^\Lambda_{n,s}(\bar{t}) = \pi_n(\mathsf{PreRun}^\Lambda_s(\bar{t})).$$

The shorthands ϱ^Λ_n and $\varrho^\Lambda_{n,s}$ are defined analogously. We define $\Delta^\Lambda_n \subseteq Q \times \mathbb{R}^n_\oplus$ to consist of the tuples $(s, t_1, t_2, \ldots, t_n) \in Q \times \mathbb{R}^n_\oplus$, such that the pre-run $\mathsf{PreRun}^\Lambda_{n,s}(t_1, t_2, \ldots, t_n)$ is a finite run in the region graph $\widetilde{\mathcal{T}}$, i.e., $\mathsf{PreRun}^\Lambda_{n,s}(t_1, t_2, \ldots, t_n) \in \mathrm{Runs}^{\widetilde{\mathcal{T}}}_n$.

PROPOSITION 4.5.2. Let $(\mathcal{T}, \pi, K)$ be a concavely-priced timed automaton and $(\mathcal{T}, \pi, \varrho, K)$ be a concave price-reward timed automaton. For region $R \in \mathcal{R}$, run type $\Lambda \in \mathrm{Types}(R)$, state $s \in S$, and $n \geq 1$ we have that π^Λ_n and ϱ^Λ_n are concave and convex, respectively, on Δ^Λ_n.

PROOF. We show, by induction, that π^Λ_n is concave on Δ^Λ_n for every $n \in \mathbb{N}$. The proof of convexity of ϱ^Λ_n on Δ^Λ_n is similar and hence omitted.

Let $\Lambda = \langle R_0, (R'_1, a_1), R_1, \ldots \rangle$ be a run type. For the base case we need to show that $\pi^\Lambda_1 : (s,t) \to \pi\big((s, R_0)(t, R'_1, a_1)\big)$ is concave on $\Delta^\Lambda_1 = \{(s,t) \in S \times \mathbb{R}_\oplus : s \in R_0 \text{ and } (s + t) \in R'_1\}$. Recall the function $\pi^a_{R,R'}$ and its domain $D_{R,R'}$ that we introduced while defining concavely-priced timed automata. The concavity of π^Λ_1 on Δ^Λ_1 follows from the concavity of $\pi^{a_1}_{R_0,R'_1}$ on D_{R_0,R'_1}.

Now we show that π^Λ_{n+1} is concave on Δ^Λ_{n+1} if π^Λ_n is concave on Δ^Λ_n. Let Λ' be the run type $\langle R_1, (R'_2, a_2), R_2, \ldots \rangle$. We have that

$$\pi^\Lambda_{n+1}(s, t_1, t_2, \ldots, t_{n+1}) = \pi\big((s, R_0), (t_1, R'_1, a_1)\big) + \pi^{\Lambda'}_n(\mathsf{Succ}(s, (t_1, a_1)), t_2, \ldots, t_{n+1}).$$

By definition $\mathsf{Succ}(s, (t_1, a_1))$ is affine on Δ^Λ_1 (and hence on Δ^Λ_{n+1}) and from inductive hypothesis $\pi^{\Lambda'}_n$ is concave on $\Delta^{\Lambda'}_n$; from the property of concave functions being closed under affine composition (Lemma B.2.3) we get that $\pi^{\Lambda'}_n(\mathsf{Succ}(s, (t_1, a_1)), t_2, \ldots, t_{n+1})$ is concave on Δ^Λ_{n+1}. Since $\pi(s, (t_1, a_1))$ is concave on Δ^Λ_1 (and hence on Δ^Λ_{n+1}) and the sum of two concave functions is concave (Lemma B.2.2), we get that π^Λ_{n+1} is concave on Δ^Λ_{n+1}. ∎

COROLLARY 4.5.3. Let $(\mathcal{T}, \pi, K)$ be a concavely-priced timed automaton and $(\mathcal{T}, \pi, \varrho, K)$ be a concave price-reward timed automaton. For every region $R \in \mathcal{R}$, for every run type

$\Lambda \in \text{Types}(R)$, for every $s \in S$, and for every $n \geq 0$ we have that $\pi^{\Lambda}_{n,s}$ and $\varrho^{\Lambda}_{n,s}$ are concave and convex, respectively, on $\Delta^{\Lambda}_{n,s}$.

PROPOSITION 4.5.4. Let $(\mathcal{T}, \pi, K)$ be a concavely-priced timed automaton and $(\mathcal{T}, \pi, \varrho, K)$ be a concave price-reward timed automaton. For every region $R \in \mathcal{R}$, for every run type $\Lambda \in \text{Types}(R)$, for every $s \in S$, and for every $n \geq 0$ we have that π_n and ϱ_n are κ-continuous on Δ^{Λ}_n, where $\kappa = n \cdot K$.

PROOF. We prove by induction that π^{Λ}_n is Lipschitz-continuous on Δ^{Λ}_n with constant $(n \cdot K)$ for every $n \in \mathbb{N}$. The proof of Lipschitz-continuity of ϱ^{Λ}_n on Δ^{Λ}_n is similar and hence omitted.

Let $\Lambda = \langle R_0, (R'_1, a_1), R_1, \ldots \rangle$ be a run type. The base case, that the function $\pi^{\Lambda}_1 : (s,t) \to \pi\big((s, R_0)(t, R'_1, a_1)\big)$ is K-continuous on the domain $\Delta^{\Lambda}_1 = \{(s,t) \in S \times \mathbb{R}_{\oplus} : s \in R_0 \text{ and } (s+t) \in R'_1\}$, follows from the Lipschitz-continuity assumption of the price functions in the definition of concavely-priced timed automata.

Now we show that π^{Λ}_{n+1} is Lipschitz-continuous on Δ^{Λ}_{n+1} with constant $(n+1) \cdot K$ given π^{Λ}_n is Lipschitz-continuous on Δ^{Λ}_n with constant $(n \cdot K)$. Let Λ' be the run type $\langle R_1, (R'_2, a_2), R_2, \ldots \rangle$. We have that

$$\pi^{\Lambda}_{n+1}(s, t_1, t_2, \ldots, t_{n+1}) = \pi\big((s, R_0), (t_1, R'_1, a_1)\big) + \pi^{\Lambda'}_n(\mathsf{Succ}(s, (t_1, a_1)), t_2, \ldots, t_{n+1}).$$

The function $\mathsf{Succ}(s, (t_1, a_1))$ is Lipschitz-continuous on Δ^{Λ}_1 (and hence on Δ^{Λ}_{n+1}) with constant 1 and from inductive hypothesis $\pi^{\Lambda'}_n$ is Lipschitz-continuous on $\Delta^{\Lambda'}_n$ with constant $(n \cdot K)$. From the Lemma B.1.2 (composition of Lipschitz-continuous functions), we get that $\pi^{\Lambda'}_n(\mathsf{Succ}(s, (t_1, a_1)), t_2, \ldots, t_{n+1})$ is Lipschitz-continuous on Δ^{Λ}_{n+1} with constant $(n \cdot K)$. Since $\pi(s, (t_1, a_1))$ is K-continuous on Δ^{Λ}_1 (and hence on Δ^{Λ}_{n+1}), from Lemma B.1.1 (weighted sum of Lipschitz-continuous functions) we get that π^{Λ}_{n+1} is Lipschitz-continuous on Δ^{Λ}_{n+1} with constant $((n+1) \cdot K)$. ∎

COROLLARY 4.5.5. Let $(\mathcal{T}, \pi, K)$ be a concavely-priced timed automaton and $(\mathcal{T}, \pi, \varrho, K)$ be a concave price-reward timed automaton. For every region $R \in \mathcal{R}$, for every run type $\Lambda \in \text{Types}(R)$, for every $s \in S$, and for every $n \geq 0$ we have that $\pi_{n,s}$ and $\varrho_{n,s}$ are κ-continuous on $\Delta^{\Lambda}_{n,s}$, where $\kappa = n \cdot K$.

4.5.1.2. Reachability Price Cost Function

We define the sequence of functions $\langle \mathsf{RP}_n : \mathsf{PreRuns}_n \to \mathbb{R} \rangle_{n \in \mathbb{N}}$ in the following way: for a run $r = \langle (s_0, R_0), (t_1, R'_1, a_1), (s_1, R_1), (t_2, R'_2, a_2), \ldots \rangle$ we have that

$$\mathsf{RP}_n(r) = \begin{cases} \pi_N(r) & \text{if } N = \mathsf{Stop}(r) < n \\ n & \text{otherwise.} \end{cases}$$

It is easy to see that $\langle \mathsf{RP}_n \rangle_{n \in \mathbb{N}}$ approximates RP.

The following proposition follows from the definition of RP_n and the Corollary 4.5.3.

PROPOSITION 4.5.6 (Quasiconcavity of reachability price)**.** Let $(\mathcal{T}, \pi, K)$ be a concavely-priced timed automaton. For every region $R \in \mathcal{R}$, for every run type $\Lambda \in \text{Types}(R)$, for every $s \in S$, and for every $n \geq 0$ we have that $\mathsf{RP}^{\Lambda}_{n,s}$ is concave on $\Delta^{\Lambda}_{n,s}$.

For a positional run type Λ notice either $\mathsf{Stop}(r) = \infty$ or $\mathsf{Stop}(r) \leq |\mathcal{R}|$, for every run r of type Λ. In both cases it is straightforward to verify that for every $n \geq |\mathcal{R}| + 1$ the function $\mathsf{RP}^{\Lambda}_{n,s}$ is Lipschitz-continuous on $\Delta^{\Lambda}_{n,s}$ with constant $((|\mathcal{R}| + 1) \cdot K)$.

PROPOSITION 4.5.7. [L-continuity of reachability price] Let $(\mathcal{T}, \pi, K)$ be a concavely-priced timed automaton. For every region $R \in \mathcal{R}$, for every run type $\Lambda \in \text{Types}(R)$, for every $s \in S$, and for every $n \geq |\mathcal{R}| + 1$ we have that $\mathsf{RP}^{\Lambda}_{n,s}$ is Lipschitz-continuous on $\Delta^{\Lambda}_{n,s}$ with constant $(|\mathcal{R}| + 1) \cdot K$.

4.5.1.3. Discounted Price Cost Function

We define the sequence of functions $\langle \mathsf{DP}_n(\lambda) : \mathsf{PreRuns}_n \to \mathbb{R} \rangle_{n \in \mathbb{N}}$ in the following way: for a run $r = \langle (s_0, R_0), (t_1, R'_1, a_1), (s_1, R_1), (t_2, R'_2, a_2), \ldots \rangle$ we have that

$$\mathsf{DP}_n(\lambda)(r) = (1 - \lambda) \sum_{i=1}^{n} \lambda^{i-1} \pi((s_{i-1}, R_{i-1}), (R'_i, t_i, a_i)).$$

It is straightforward to verify that $\langle \mathsf{DP}_n(\lambda) \rangle_{n \in \mathbb{N}}$ approximates $\mathsf{DP}(\lambda)$.

The proof of the following proposition is along the same lines as the proof of Proposition 4.5.2. The only difference, however, is that the proof uses the property of concave functions being closed under nonnegative weighted sum (Lemma B.2.2). Notice that λ^i is nonnegative for all $i \in \mathbb{N}$.

PROPOSITION 4.5.8 (Quasiconcavity of discounted price)**.** Let $(\mathcal{T}, \pi, K)$ be a concavely-priced timed automaton. For every region $R \in \mathcal{R}$, for every run type $\Lambda \in \text{Types}(R)$, for every $s \in S$, and for every $n \geq 0$ we have that $\mathsf{DP}^{\Lambda}_{n,s}(\lambda)$ is concave on $\Delta^{\Lambda}_{n,s}$.

The proof of the following proposition is similar to the proof of Proposition 4.5.4.

PROPOSITION 4.5.9. Let $(\mathcal{T}, \pi, K)$ be a concavely-priced timed automaton. For every region $R \in \mathcal{R}$, for every run type $\Lambda \in \text{Types}(R)$, for every $s \in S$, for every $\lambda \in (0, 1)$, and for every $n \geq 0$ we have that $\mathsf{DP}^{\Lambda}_{n,s}(\lambda)$ is Lipschitz-continuous on $\Delta^{\Lambda}_{n,s}$ with constant $(1 - \lambda^n)K$.

4.5.1.4. Price-per-Reward Average Cost Function

We define the sequence of functions $\langle \mathsf{PRAvg}_n : \mathsf{PreRuns}_n \to \mathbb{R} \rangle_{n \in \mathbb{N}}$ in the following way: for a run $r = \langle (s_0, R_0), (t_1, R'_1, a_1), (s_1, R_1), (t_2, R'_2, a_2), \ldots \rangle$ we have that

$$\mathsf{PRAvg}_n(r) = \frac{\pi_n(r)}{\varrho_n(r)}.$$

It is straightforward to verify that $\langle \mathsf{PRAvg}_n \rangle_{n \in \mathbb{N}}$ approximates PRAvg.

PROPOSITION 4.5.10 (Quasiconcavity of price-per-reward average)**.** Let $(\mathcal{T}, \pi, \varrho, K)$ be a concave price-reward timed automaton. For every region $R \in \mathcal{R}$, for every run type $\Lambda \in \text{Types}(R)$, for every $s \in S$, and for every $n \geq (|\mathcal{R}| + 1)$ we have that $\mathsf{PRAvg}^{\Lambda}_{n,s}(r)$ is quasiconcave on $\Delta^{\Lambda}_{n,s}$ if the functions $\pi^{\Lambda}_{n,s}$ and $\varrho^{\Lambda}_{n,s}$ on the domain $\Delta^{\Lambda}_{n,s}$ satisfy one of the following properties:

(1) $\pi^{\Lambda}_{n,s}$ is nonnegative and $\varrho^{\Lambda}_{n,s}$ is positive; or
(2) $\varrho^{\Lambda}_{n,s}$ is positive and affine.

PROOF. The function $\mathsf{PRAvg}^{\Lambda}_{n,s}$ is the ratio of the functions $\pi^{\Lambda}_{n,s}$ and $\varrho^{\Lambda}_{n,s}$. Since the reward function is regionally non-Zeno (Assumption 3.4.5), it follows that for all $n \geq (|\mathcal{R}| + 1)$, we have that $\varrho^{\Lambda}_{n,s}$ is nonzero and hence $\mathsf{PRAvg}^{\Lambda}_{n,s}$ is well defined for all $n \geq (|\mathcal{R}| + 1)$. From Lemma B.2.7 we know that $\mathsf{PRAvg}^{\Lambda}_{n,s}$ is quasiconcave if the functions $\pi^{\Lambda}_{n,s}$ and $\varrho^{\Lambda}_{n,s}$ on the domain $\Delta^{\Lambda}_{n,s}$ satisfy one of the following properties:

(1) $\pi^{\Lambda}_{n,s}$ is nonnegative and concave, and $\varrho^{\Lambda}_{n,s}$ is positive and convex; or
(2) $\pi^{\Lambda}_{n,s}$ is concave, and $\varrho^{\Lambda}_{n,s}$ is positive and affine.

The current proposition, now, follows from Proposition 4.5.2 ($\pi^{\Lambda}_{n,s}$ and $\varrho^{\Lambda}_{n,s}$ are concave and convex, respectively). ∎

Observe that the average time, average price, price-per-time average cost functions satisfy the second requirement on reward function (i.e., $\varrho^{\Lambda}_{n,s}$ is positive and affine) stated in Proposition 4.5.10.

PROPOSITION 4.5.11. Let $(\mathcal{T}, \pi, \varrho, K)$ be a concave price-reward timed automaton. For every region $R \in \mathcal{R}$, for every run type $\Lambda \in \text{Types}(R)$, for every $s \in S$, and for every $n \geq 0$ we have that $\mathsf{PRAvg}^{\Lambda}_{n,s}$ is Lipschitz-continuous on $\Delta^{\Lambda}_{n,s}$.

PROOF. The function $\mathsf{PRAvg}^{\Lambda}_{n,s}$ is the ratio of the functions $\pi^{\Lambda}_{n,s}$ and $\varrho^{\Lambda}_{n,s}$. From Proposition 4.5.4 we have that $\pi^{\Lambda}_{n,s}$ and $\varrho^{\Lambda}_{n,s}$ are Lipschitz-continuous on $\Delta^{\Lambda}_{n,s}$ with constant $(n \cdot K)$. From Lemma B.1.3 (ratio of two Lipschitz-continuous functions) we have that if $\pi^{\Lambda}_{n,s}$ and $\varrho^{\Lambda}_{n,s}$ are bounded from above by a constant M and $\varrho^{\Lambda}_{n,s}$ is bounded from below by constant N then their ratio function is Lipschitz-continuous with constant $(N + M) \cdot (n \cdot K)/N^2$.

Because we consider $\mathcal{K}$-bounded timed automata, we have that each t_i (for $1 \leq i \leq n$) in the tuple $(t_1, t_2, \ldots, t_n) \in \Delta^{\Lambda}_{n,s}$ is bounded from above by $\mathcal{K}$. As $\pi^{\Lambda}_{n,s}$ and $\varrho^{\Lambda}_{n,s}$ are Lipschitz-continuous on $\Delta^{\Lambda}_{n,s}$ with constant $(n \cdot K)$, it follows that $\pi^{\Lambda}_{n,s}$ and $\varrho^{\Lambda}_{n,s}$ are bounded from above by $M = n\gamma$, where $\gamma = K \cdot \mathcal{K}$.

Set $\beta = |\mathcal{R}| + 1$. Since the reward function is regionally non-Zeno (Assumption 3.4.5), we have that for every $n \geq \beta$ the reward function $\varrho^{\Lambda}_{n,s}$ is bounded from below by $N = \frac{n}{\beta}$.

Hence it follows that for every $n \geq \beta$ the function $\mathsf{PRAvg}^{\Lambda}_{n,s}$ is Lipschitz-continuous with constant $\frac{(N+M)\cdot(n\cdot K)}{N^2} = (1 + \beta\gamma) \cdot K \cdot \beta$. ∎

4.5.2. Positional Optimality and Uniform Convergence

The positional optimality and the uniform convergence properties of a cost function need only be verified with boundary region graph.

From Proposition 3.7.10 we know that for every boundary region graph $\widehat{\mathcal{T}}$ and for every state $(s, R) \in \widehat{S}$, the set of reachable states in $\widehat{\mathcal{T}}$ from (s, R) is finite. Hence to prove positional optimality and uniform convergence properties for reachability price, discounted price, and price-per-reward average cost function in a boundary region graph, it is sufficient to show that these cost functions satisfy positional optimality and uniform convergence properties in finite graphs.

From Theorems 2.2.1 and 2.2.28, we have that for every finite graph, and hence for every boundary region graph, the cost functions reachability price, discounted price, and price-per-reward average satisfy positional optimality and uniform convergence properties.

5

Reachability-Time Games

> If you must play, decide on three things at the start: the rules of the game, the stakes, and the quitting time.
>
> Chinese Proverb

In a reachability-time game, players Min and Max choose moves so that the time to reach a final state in a timed automaton is minimised or maximised, respectively. Asarin and Maler [**AM99**] showed decidability of reachability-time games on strongly non-Zeno timed automata using a value iteration algorithm. This chapter complements their work by providing a strategy improvement algorithm for the problem. It also generalises their decidability result because the proposed strategy improvement algorithm solves reachability-time games on all timed automata. The exact computational complexity of solving reachability-time games is also established: the problem is EXPTIME-complete for timed automata with at least two clocks.

5.1. Introduction

5.1.1. Definition

DEFINITION 5.1.1 (Reachability-Time Games)**.** A *reachability-time game* on a timed automaton is a tuple $(\Gamma, \mathsf{RT}_{\mathrm{Min}}, \mathsf{RT}_{\mathrm{Max}})$, where:

- $\Gamma = (\mathcal{T}, L_{\mathrm{Min}}, L_{\mathrm{Max}})$ is a timed game automaton such that $\mathcal{T} = (L, C, S, A, E, \delta, \xi, F)$ is a timed automaton, L_{Min} is the set of locations controlled by player Min, and L_{Max} is the set of locations controlled by player Max;
- $\mathsf{RT}_{\mathrm{Min}} : \mathrm{Runs} \to \mathbb{R}$ and $\mathsf{RT}_{\mathrm{Max}} : \mathrm{Runs} \to \mathbb{R}$ are payoff functions, which for every run of the timed automaton return the amount the player Min loses and the player Max wins, respectively. The functions $\mathsf{RT}_{\mathrm{Min}}$ and $\mathsf{RT}_{\mathrm{Max}}$ are defined in the following way: for a run $r = \langle s_0, (t_1, a_1), s_1, (t_2, a_2), \ldots \rangle \in \mathrm{Runs}$ we have

$$\mathsf{RT}_{\mathrm{Min}}(r) = \mathsf{RT}_{\mathrm{Max}}(r) = \begin{cases} \sum_{i=1}^{\mathsf{Stop}(r)} t_i & \text{if } \mathsf{Stop}(r) < \infty \\ \infty & \text{otherwise.} \end{cases}$$

Since the functions $\mathsf{RT}_{\mathrm{Min}}$ and $\mathsf{RT}_{\mathrm{Max}}$ are equal, we write $\mathsf{RT} : \mathrm{Runs} \to \mathbb{R}$ for this function.

We define $Q_{\mathrm{Min}} = \{(\ell, \nu) \in Q : \ell \in L_{\mathrm{Min}}\}$, $Q_{\mathrm{Max}} = Q \setminus Q_{\mathrm{Min}}$, $S_{\mathrm{Min}} = S \cap Q_{\mathrm{Min}}$, $S_{\mathrm{Max}} = S \setminus S_{\mathrm{Min}}$, $\mathcal{R}_{\mathrm{Min}} = \{[s] : s \in Q_{\mathrm{Min}}\}$, and $\mathcal{R}_{\mathrm{Max}} = \mathcal{R} \setminus \mathcal{R}_{\mathrm{Min}}$.

The strategies of player Min and player Max are defined as usual (see Section 3.4.2). We write Σ_{Min} for the set of strategies for player Min, and we write Σ_{Max} for the set of strategies for player Max. We write Π_{Min} and Π_{Max} for the sets of positional strategies for player Min and for player Max, respectively. Reachability-time payoff function $\mathsf{RT} : \mathrm{Runs} \to \mathbb{R}$ naturally gives rise to the function $\mathsf{RT} : S \times \Sigma_{\mathrm{Min}} \times \Sigma_{\mathrm{Max}} \to \mathbb{R}$ in the following way. For strategies $\mu \in \Sigma_{\mathrm{Min}}$ and $\chi \in \Sigma_{\mathrm{Max}}$ of respective players and a state $s \in S$ we have $\mathsf{RT}(s, \mu, \chi) = \mathsf{RT}(\mathrm{Run}(s, \mu, \chi))$.

5.1.2. Value of Reachability-Time Game

If player Min uses the strategy $\mu \in \Sigma_{\mathrm{Min}}$ and player Max uses the strategy $\chi \in \Sigma_{\mathrm{Max}}$ then player Min loses the value $\mathsf{RT}(s, \mu, \chi)$ and player Max wins the value $\mathsf{RT}(s, \mu, \chi)$. In a reachability-time game player Min is interested in minimising the value she loses and player Max is interested in maximising the value he wins. We define the *upper value* $\overline{\mathsf{Val}}(s)$ and the *lower value* $\underline{\mathsf{Val}}(s)$ of the reachability-time game at the state $s \in S$ by

$$\overline{\mathsf{Val}}(s) = \inf_{\mu \in \Sigma_{\mathrm{Min}}} \sup_{\chi \in \Sigma_{\mathrm{Max}}} \mathsf{RT}(\mathrm{Run}(s, \mu, \chi)), \text{ and } \underline{\mathsf{Val}}(s) = \sup_{\chi \in \Sigma_{\mathrm{Max}}} \inf_{\mu \in \Sigma_{\mathrm{Min}}} \mathsf{RT}(\mathrm{Run}(s, \mu, \chi)).$$

From Proposition 1.2.4 the inequality $\underline{\mathsf{Val}}(s) \leq \overline{\mathsf{Val}}(s)$ always holds. A reachability-time game is *determined* if for every $s \in S$, the lower and upper values at s are equal to each other; then we say that the *value* $\mathsf{Val}(s)$ exists and $\mathsf{Val}(s) = \underline{\mathsf{Val}}(s) = \overline{\mathsf{Val}}(s)$.

For strategies $\mu \in \Sigma_{\mathrm{Min}}$ and $\chi \in \Sigma_{\mathrm{Max}}$, we define

$$\mathsf{Val}^{\mu}(s) = \sup_{\chi \in \Sigma_{\mathrm{Min}}} \mathsf{RT}(\mathrm{Run}(s, \mu, \chi)), \text{ and } \mathsf{Val}_{\chi}(s) = \inf_{\mu \in \Sigma_{\mathrm{Min}}} \mathsf{RT}(\mathrm{Run}(s, \mu, \chi)).$$

We say that a strategy $\mu \in \Sigma_{\mathrm{Min}}$ or $\chi \in \Sigma_{\mathrm{Max}}$, respectively, is *optimal* if for every $s \in S$, we have $\mathsf{Val}^{\mu}(s) = \mathsf{Val}(s)$ or $\mathsf{Val}_{\chi}(s) = \mathsf{Val}(s)$, respectively. For an $\varepsilon > 0$, we say that a strategy $\mu \in \Sigma_{\mathrm{Min}}$ or $\chi \in \Sigma_{\mathrm{Max}}$ is *ε-optimal* if for every $s \in S$, we have $\mathsf{Val}^{\mu}(s) \leq \mathsf{Val}(s) + \varepsilon$ or $\mathsf{Val}_{\chi}(s) \geq \mathsf{Val}(s) - \varepsilon$, respectively. Note that if a game is determined then for every $\varepsilon > 0$, both players have ε-optimal strategies.

For an $\varepsilon > 0$, we say that a strategy $\mu \in \Sigma_{\mathrm{Min}}$ for Min is *ε-optimal* if for every $s \in S$, we have $\mathsf{Val}^{\mu}(s) \leq \mathsf{Val}(s) + \varepsilon$. For an $\varepsilon > 0$, we say that a strategy $\chi \in \Sigma_{\mathrm{Max}}$ for Max is *ε-optimal* if for every $s \in S$, we have $\mathsf{Val}_{\chi}(s) \geq \mathsf{Val}(s) - \varepsilon$. Optimal and ε-optimal strategies for player Max are defined analogously.

We say that a reachability-time game is *positionally determined* if for every $s \in S$, we have

$$\sup_{\mu \in \Pi_{\mathrm{Min}}} \inf_{\chi \in \Sigma_{\mathrm{Max}}} \mathsf{RT}(\mathrm{Run}(s, \mu, \chi)) = \mathsf{Val}(s) = \inf_{\chi \in \Pi_{\mathrm{Max}}} \sup_{\mu \in \Sigma_{\mathrm{Min}}} \mathsf{RT}(\mathrm{Run}(s, \mu, \chi)).$$

Note that if the reachability-time game is positionally determined then for every $\varepsilon > 0$, both players have *positional* ε-optimal strategies. Our results (Lemma 5.1.3, Theorem 5.3.3,

and Theorem 5.4.12) yield a constructive proof of the following fundamental result for reachability-time games.

THEOREM 5.1.2 (Positional determinacy)**.** Reachability-time games are positionally determined.

5.1.3. Optimality Equations

Let Γ be a timed game automaton, and let $T : S \to \mathbb{R}$ and $D : S \to \mathbb{N}$.

We write $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma)$, and we say that (T, D) is a solution of optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma)$, if for all $s \in S$, we have:

- if $D(s) = \infty$ then $T(s) = \infty$; and
- if $s \in F$ then $(T(s), D(s)) = (0, 0)$; and
- if $s \in S_{\mathrm{Min}} \setminus F$ then

$$\begin{aligned} T(s) &= \inf_{a,t}\{t + T(s') \,:\, s \xrightarrow{a}_t s'\}, \text{ and} \\ D(s) &= \min\left\{1 + d' \,:\, T(s) = \inf_{a,t}\{t + T(s') \,:\, s \xrightarrow{a}_t s' \text{ and } D(s') = d'\}\right\}; \end{aligned}$$

- if $s \in S_{\mathrm{Max}} \setminus F$ then

$$\begin{aligned} T(s) &= \sup_{a,t}\{t + T(s') \,:\, s \xrightarrow{a}_t s'\}, \text{ and} \\ D(s) &= \max\left\{1 + d' \,:\, T(s) = \sup_{a,t}\{t + T(s') \,:\, s \xrightarrow{a}_t s' \text{ and } D(s') = d'\}\right\}; \end{aligned}$$

LEMMA 5.1.3 (ε-Optimal strategies from optimality equations)**.** If $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma)$, then for all $s \in S$, we have $\mathsf{Val}(s) = T(s)$ and for every $\varepsilon > 0$, both players have *positional* ε-optimal strategies.

PROOF. We show that for every $\varepsilon > 0$, there exists a positional strategy $\mu_\varepsilon : S_{\mathrm{Min}} \to A \times \mathbb{R}_\oplus$ for player Min, such that for every strategy χ for player Max, if $s \in S$ is such that $D(s) < \infty$, then we have $\mathsf{RT}(\mathrm{Run}(s, \mu_\varepsilon, \chi)) \leq T(s) + \varepsilon$. The proof, that for every $\varepsilon > 0$, there exists a positional strategy $\chi_\varepsilon : S_{\mathrm{Max}} \to A \times \mathbb{R}_\oplus$ for player Max, such that for every strategy μ for player Min, if $s \in S$ is such that $D(s) < \infty$ then we have $\mathsf{RT}(\mathrm{Run}(s, \mu, \chi_\varepsilon)) \geq T(s) - \varepsilon$, is similar and omitted. The proof, that if $D(s) = \infty$ then player Max has a strategy to prevent ever reaching a final state, is routine and omitted as well. Together, these facts imply that T is equal to the value function of the reachability-time game, and the positional strategies μ_ε and χ_ε, defined in the proof below for all $\varepsilon > 0$, are ε-optimal.

For $\varepsilon' > 0$, $T : S \to \mathbb{R}$, and $s \in S_{\mathrm{Min}} \setminus F$, we say that a timed action $(a, t) \in A \times \mathbb{R}_\oplus$ is ε'-optimal for (T, D) in s if $s \xrightarrow{a}_t s'$, and

$$D(s') \leq D(s) - 1, \text{ and} \tag{5.1.1}$$

$$t + T(s') \leq T(s) + \varepsilon'. \tag{5.1.2}$$

Observe that for every state $s \in S_{\mathrm{Min}}$ and for every $\varepsilon' > 0$, there is a ε'-optimal timed action for (T, D) in s because $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma)$. Moreover, again by $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma)$

we have that for every $s \in S_{\text{Max}} \setminus F$ and timed action (a,t), such that $s \xrightarrow{a}_t s'$, we have

$$D(s') \leq D(s) - 1, \text{ and} \tag{5.1.3}$$
$$t + T(s') \leq T(s). \tag{5.1.4}$$

Let $\varepsilon > 0$; we define $\mu_\varepsilon : S_{\text{Min}} \to A \times \mathbb{R}_\oplus$ by setting $\mu_\varepsilon(s)$, for every $s \in S_{\text{Min}}$, to be a timed action which is $\varepsilon'(s)$-optimal for (T, D) in s, where $\varepsilon'(s) > 0$ is sufficiently small (to be determined later). Let χ be an arbitrary strategy for player Max and let $r = \text{Run}(s, \mu_\varepsilon, \chi) = \langle s_0, (a_1, t_1), s_1, (a_2, t_2), \ldots \rangle$. Let $N = \mathsf{Stop}(r)$. Our goal is to prove that $\mathsf{RT}(r) \leq T(s) + \varepsilon$, i.e., that $T(s) \geq \sum_{k=1}^{N} t_k - \varepsilon$.

For every state $s \in S$, such that $D(s) < \infty$, define $\varepsilon'(s) = \varepsilon \cdot 2^{-D(s)}$. Note that if we add left- and right-hand sides of the inequalities (5.1.2) or (5.1.4), respectively, for all states s_i, and $\varepsilon'(s_i)$-optimal timed actions $\mu_\varepsilon(s_i)$ if $s_i \in S_{\text{Min}}$, where $i = 0, 1, \ldots, N-1$, then we get

$$T(s) \;=\; T(s_0) \;\geq\; \sum_{k=1}^{N} t_k - \sum_{k=0}^{N-1} \varepsilon'(s_k) \;\geq\; \sum_{k=0}^{N-1} t_k - \varepsilon.$$

The first inequality holds by $T(s_N) = T(s_{\mathsf{Stop}(r)}) = 0$, and the second inequality holds because

$$\sum_{k=0}^{N-1} \varepsilon'(s_k) \;=\; \sum_{k=0}^{N-1} (\varepsilon \cdot 2^{-D(s_k)}) \;\leq\; \varepsilon \cdot \sum_{d=1}^{\infty} 2^{-d} \;\leq\; \varepsilon,$$

where the first inequality follows by (5.1.1) and (5.1.3).

It may be worth noting that if the finite values of the function D are bounded, i.e., if $B < \infty$, where $B = \sup_{s \in S}\{D(s) : D(s) < \infty\}$, then in the above proof it is sufficient to define $\varepsilon'(s) = \varepsilon / B$, for all $s \in S$, which gives arguably more realistically "physically implementable" ε-optimal strategies. ∎

5.2. Simple Functions

Asarin and Maler [**AM99**] presented a value-iteration algorithm to compute upper-value in a reachability-time game. They observed that all the functions used in their algorithm belong to a special class that can be finitely represented, which they called *simple functions*. Simple functions play an important role in our work as we show the value of a reachability-time game restricted to a region is a simple function.

DEFINITION 5.2.1 (Simple Functions [**AM99**])**.** Let $X \subseteq Q$. A function $F : X \to \mathbb{R}$ is a *simple function* if either: there is $e \in \mathbb{Z}$, such that for every $s \in X$, we have $F(s) = e$; or there are $e \in \mathbb{Z}$ and $c \in C$, such that for every $s \in X$, we have $F(s) = e - s(c)$.

Let $X \subseteq Q$ be convex and let $F : X \to \mathbb{R}$ be a continuous function. We write $\overline{F}$ for the unique continuous function $F' : \overline{X} \to \mathbb{R}$, such that for all $s \in X$, we have $F'(s) = F(s)$. Observe that if F is simple, then $\overline{F}$ is simple. For functions $F, F' : X \to \mathbb{R}$ we define functions $\max(F, F'), \min(F, F') : X \to \mathbb{R}$ by $\max(F, F')(s) = \max\{ F(s), F'(s) \}$ and $\min(F, F')(s) = \min\{ F(s), F'(s) \}$, for every $s \in X$.

LEMMA 5.2.2. Let $F, F' : R \to \mathbb{R}$ be simple functions defined on a region $R \in \mathcal{R}$. Then either $\min(\overline{F}, \overline{F'}) = \overline{F}$ and $\max(\overline{F}, \overline{F'}) = \overline{F'}$, or $\min(\overline{F}, \overline{F'}) = \overline{F'}$ and $\max(\overline{F}, \overline{F'}) = \overline{F}$. In particular, both $\min(\overline{F}, \overline{F'})$ and $\max(\overline{F}, \overline{F'})$ are simple functions.

PROOF. We prove the lemma for functions $\min(F, F')$ and $\max(F, F')$ instead of $\min(\overline{F}, \overline{F'})$ and $\max(\overline{F}, \overline{F'})$, respectively. Extending the result to the unique continuous extensions to $\overline{X}$ is routine. The case when both F and F' are constant functions is straightforward. Hence it suffices to consider the following two cases.

Case 1. Let $F(s) = e - s(c)$ and let $F'(s) = e'$, for some $e, e' \in \mathbb{Z}$ and a clock $c \in C$. Note that for every state $s \in R$, we have $\lfloor F'(s) - F(s) \rfloor = (e' - e) + \lfloor s(c) \rfloor$ and hence $\lfloor F' - F \rfloor$ is a constant function in region R. Therefore either $F'(s) - F(s) \geq 0$ for all $s \in R$, or $F'(s) - F(s) \leq 0$ for all $s \in R$, i.e., either $\min(F, F') = F$ and $\max(F, F') = F'$, or $\min(F, F') = F'$ and $\max(F, F') = F$.

Case 2. Let $F(s) = e - s(c)$ and $F'(s) = e' - s(c')$, for some $e, e' \in \mathbb{Z}$ and clocks $c, c' \in C$. Note that for every state $s \in R$, we have $\lfloor F'(s) - F(s) \rfloor = (e' - e) + \lfloor s(c') - s(c) \rfloor$ and

$$\lfloor s(c') - s(c) \rfloor = \begin{cases} \lfloor s(c') \rfloor - \lfloor s(c) \rfloor & \text{if } \langle s(c') \rangle \geq \langle s(c) \rangle, \\ \lfloor s(c') \rfloor - \lfloor s(c) \rfloor - 1 & \text{if } \langle s(c') \rangle < \langle s(c) \rangle. \end{cases}$$

In particular, as in the previous case we have that $\lfloor F' - F \rfloor$ is a constant function in region R and hence one of the functions F or F' is equal to $\max(F, F')$ and the other is equal to $\min(F, F')$. ∎

5.3. Reachability-Time Games on Boundary Region Automata

In this section we introduce boundary region (game) automaton Γ_{BR} of a timed game automaton Γ, and define the set of optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$ for a boundary region automaton Γ_{BR}. We further show that a solution of optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$ of Γ_{BR} gives a solution of optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma)$ for the timed game automaton Γ.

5.3.1. Boundary Region (Game) Automata

Let $\Gamma = (\mathcal{T}, L_{\mathrm{Min}}, L_{\mathrm{Max}})$ be timed game automaton. Recall from Definition 3.7.7 that $\mathcal{A} = A \times [\![\mathcal{K}]\!]_{\mathbb{N}} \times C$ is the finite set of boundary timed actions. The *boundary region (game) automaton* Γ_{BR} to be the finite edge-labelled graph $(\mathcal{R}, \mathcal{M})$, where the set $\mathcal{R}$ of regions of timed automaton $\mathcal{T}$ is the set of vertices, and the labelled edge relation $\mathcal{M} \subseteq \mathcal{R} \times \mathcal{A} \times \mathcal{R}$ is defined in the following way. For $\alpha = (a, b, c) \in \mathcal{A}$ and $R, R' \in \mathcal{R}$ we have $(R, \alpha, R') \in \mathcal{M}$, sometimes denoted by $R \overset{\alpha}{\rightsquigarrow} R'$, if and only if one of the following conditions holds:

- there is an $R'' \in \mathcal{R}$, such that $R \to_{b,c} R'' \xrightarrow{a} R'$; or
- $R \in \mathcal{R}_{\mathrm{Min}}$, and there are $R'', R''' \in \mathcal{R}$, such that $R \to_{b,c} R'' \to_{+1} R''' \xrightarrow{a} R'$; or
- $R \in \mathcal{R}_{\mathrm{Max}}$, and there are $R'', R''' \in \mathcal{R}$, such that $R \to_{b,c} R'' \leftarrow_{+1} R''' \xrightarrow{a} R'$.

Remark. Note that this definition of the boundary region automata has been slightly modified (compare with Definition 3.7.8) to take advantage of the fact that, in a reachability-time game, timed moves which wait till the farthest (closest) boundary of a region are

sub-optimal for player Min (Max). Also observe that the set of edges, of the boundary region automata introduced here, are of the form $\mathcal{R} \times \mathcal{A} \times \mathcal{R}$ instead of $\mathcal{R} \times \mathcal{R} \times \mathcal{A} \times \mathcal{R}$ as introduced in Chapter 3. In doing so there is no loss of information in the case of reachability-time games, as the time of two different moves with the same boundary actions and same initial region, e.g., (R, R_1, α, R_1') and (R, R_2, α, R_2') is the same, while in a priced timed automaton (e.g., concavely-priced timed automaton) prices of these moves may differ.

5.3.2. Optimality Equations

Recall, from Section 5.1.3, that a solution of equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma)$ for a reachability-time game Γ is a pair of functions (T, D), such that $T : S \to \mathbb{R}$ and $D : S \to \mathbb{N}$. Our goal is to define analogous optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$ for the boundary region automaton Γ_{BR}.

If $R \overset{\alpha}{\rightsquigarrow} R'$, where $R, R' \in \mathcal{R}$ and $\alpha \in \mathcal{A}$, then $s \in R$ does not in general imply that $\mathsf{Succ}(s, \alpha) \in R'$; it is however the case that $s \in R$ implies $\mathsf{Succ}(s, \alpha) \in \overline{R'}$. In order to correctly capture the constraints for successor states which fall out of the "target" region R' of a move of the form $R \overset{\alpha}{\rightsquigarrow} R'$, we consider, as solutions of optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$, *regional functions* of types $T : \mathcal{R} \to [S \rightharpoonup \mathbb{R}]$ and $D : \mathcal{R} \to [S \rightharpoonup \mathbb{N}]$, where for every $R \in \mathcal{R}$, the domain of partial functions $T(R)$ and $D(R)$ is $\overline{R}$. Sometimes, when defining a regional function $F : \mathcal{R} \to [S \rightharpoonup \mathbb{R}]$, it will only be natural to define $F(R)$ for all $s \in R$, instead of all $s \in \overline{R}$. This is not a problem, however, because as discussed in Section 5.2 defining $F(R)$ on the region R uniquely determines the continuous extension of $F(R)$ to $\overline{R}$. For a function $F : \mathcal{R} \to [S \rightharpoonup \mathbb{R}]$, we define the function $\widetilde{F} : S \to \mathbb{R}$ by $\widetilde{F}(s) = F([s])(s)$.

Optimality equations. Let $T : \mathcal{R} \to [S \to \mathbb{R}]$ and let $D : \mathcal{R} \to [S \to \mathbb{N}]$. We write $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$ if for all $s \in S$, we have the following:

- if $s \in F$ then $\big(\widetilde{T}(s), \widetilde{D}(s)\big) = (0, 0)$;
- if $s \in S_{\mathrm{Min}} \setminus F$ then

$$\big(\widetilde{T}(s), \widetilde{D}(s)\big) = \min^{\mathrm{lex}}_{m \in \mathcal{M}} \big\{ \big(T(R')^{\oplus}_{\alpha}(s), D(R')^{\boxplus}_{\alpha}(s)\big) \,:\, m = ([s], \alpha, R') \big\};$$

- if $s \in S_{\mathrm{Max}} \setminus F$ then

$$\big(\widetilde{T}(s), \widetilde{D}(s)\big) = \max^{\mathrm{lex}}_{m \in \mathcal{M}} \big\{ \big(T(R')^{\oplus}_{\alpha}(s), D(R')^{\boxplus}_{\alpha}(s)\big) \,:\, m = ([s], \alpha, R') \big\}.$$

5.3.3. Optimality Equations of Timed Automata vs Boundary Region Automata

In this subsection we show that the function $(T, D) \mapsto (\widetilde{T}, \widetilde{D})$ translates solutions of the reachability-time optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$ for the boundary region automaton Γ_{BR} to solutions of optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma)$ for the reachability-time game Γ. In other words, we establish that the function $\Gamma \mapsto \Gamma_{\mathrm{BR}}$ is a reduction from the problem of computing values in reachability-time games to the problem of solving optimality equations

for boundary region automata. Then in Section 5.4 we give an algorithm to solve optimality equations for $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$.

Before we state and prove the main result (Theorem 5.3.3) of this subsection we need to introduce some mathematical shorthands and their properties. For $\alpha \in \mathcal{A}$ and $R, R' \in \mathcal{R}$, if $R \xrightarrow{\alpha} R'$ and $F : R' \to \mathbb{R}$ then we define the functions $F^{\oplus}_{\alpha} : R \to \mathbb{R}$ and $F^{\boxplus}_{\alpha} : R \to \mathbb{R}$ by $F^{\oplus}_{\alpha}(s) = t(s,\alpha) + F(\mathsf{Succ}(s,\alpha))$, and $F^{\boxplus}_{\alpha}(s) = 1 + F(\mathsf{Succ}(s,\alpha))$, for all $s \in R$.

PROPOSITION 5.3.1. Let $\alpha \in \mathcal{A}$ and $R, R' \in \mathcal{R}$. If $R \xrightarrow{\alpha} R'$ and $F : R' \to \mathbb{R}$ is simple, then $F^{\oplus}_{\alpha}$ is simple.

PROOF. Let $\alpha = (a, b, c)$. If F is a constant function, i.e., if there is some $e \in \mathbb{Z}$, such that for all $s' \in R'$, we have $F(s') = e$, then $F^{\oplus}_{\alpha}(s) = t(s,\alpha) + e$. If $s(c) > b$ for all $s \in R$, then $t(s,\alpha) = 0$ for all $s \in R$, and hence $F^{\oplus}_{\alpha}(s) = e$ and $F^{\oplus}_{\alpha}$ is simple. If instead $s(c) \leq b$ for all $s \in R$, then $F^{\oplus}_{\alpha}(s) = (b - s(c)) + e = (b + e) - s(c)$ and hence it is a simple function.

The other case is when F is not a constant function, i.e., if there are a constant $e \in \mathbb{Z}$ and a clock $c' \in C$, such that for all $s' \in R'$, we have $F(s') = e - s'(c')$. We consider two sub cases.

- If $c' \in \xi(a)$ then $F^{\oplus}_{\alpha}(s) = t(s,a) + (e - s'(c')) = t(s,\alpha) + e$, because by the assumption that $c' \in \xi(a)$ we have that $s'(c') = 0$. If $s(c) > b$ for all $s \in R$, then $t(s,\alpha) = 0$ for all $s \in R$, and hence $F^{\oplus}_{\alpha}(s) = e$ which is a simple function. If instead $s(c) \leq b$ for all $s \in R$, then $F^{\oplus}_{\alpha}(s) = (b + e) - s(c)$ which is also a simple function.
- If instead $c' \notin \xi(a)$ then $F^{\oplus}_{\alpha}(s) = t(s,\alpha) + (e - (s(c') + t(s,\alpha))) = e - s(c')$, because by the assumption that $c' \notin \xi(a)$ we have that $s'(c') = s(c') + t(s,\alpha)$, and hence $F^{\oplus}_{\alpha}$ is a simple function.

The proof is now complete. ∎

For $a \in A$ and $R, R', R'' \in \mathcal{R}$, if $R \to_* R'' \xrightarrow{a} R'$, $s \in R$, and $F : R' \to \mathbb{R}$, then we define the partial function $F^{\oplus}_{s,a} : \mathbb{R}_{\oplus} \rightharpoonup \mathbb{R}$ by $F^{\oplus}_{s,a}(t) = t + F(\mathsf{Succ}(s,(a,t)))$, for all $t \in \mathbb{R}_{\oplus}$, such that $(s + t) \in R''$. Note that the domain $\{t \in \mathbb{R}_{\oplus} : (s + t) \in R''\}$ of $F^{\oplus}_{s,a}$ is an interval.

PROPOSITION 5.3.2. Let $a \in A$ and $R, R', R'' \in \mathcal{R}$. If $R \to_* R'' \xrightarrow{a} R'$, $s \in R$, and $F : R' \to \mathbb{R}$ is simple, then $F^{\oplus}_{s,a} : I \to \mathbb{R}$, where $I = \{t \in \mathbb{R}_{\oplus} : (s + t) \in R''\}$, is continuous and nondecreasing.

PROOF. We consider two cases. If F is a constant function, i.e., if there is $e \in \mathbb{Z}$, such that for all $s' \in R'$ we have $F(s') = e$, then $F^{\oplus}_{s,a}(t) = t + F(\mathsf{Succ}(s,(a,t))) = t + e$, which is a continuous and nondecreasing function of t.

The other case is when F is not a constant function, i.e., if there are a constant $e \in \mathbb{Z}$ and a clock $c' \in C$, such that for all $s' \in R'$, we have $F(s') = e - s'(c')$. We consider two sub cases. If $c' \in \xi(a)$ then $F^{\oplus}_{s,a}(t) = t + e$ which is continuous and nondecreasing. If instead $c' \notin \xi(a)$ then $F^{\oplus}_{s,a}(t) = t + (e - (s + t)(c')) = t + e - (s(c') + t) = e - s(c')$, i.e., $F^{\oplus}_{s,a}$ is a constant function and hence continuous and nondecreasing. ∎

7 We say that a function $F : \mathcal{R} \to [S \to \mathbb{R}]$ is *regionally simple* or *regionally constant*, respectively, if for every region $R \in \mathcal{R}$, the function $F(R) : \overline{R} \to \mathbb{R}$ is simple or constant, respectively.

THEOREM 5.3.3 (Correctness of the reduction)**.** If $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$, T is regionally simple, and D is regionally constant, then $(\widetilde{T}, \widetilde{D}) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma)$.

PROOF. We need to show that for every state $s \in S_{\mathrm{Min}} \setminus F$, we have

$$\widetilde{T}(s) = \inf_{a,t}\{t + \widetilde{T}(s') : s \xrightarrow{a}_t s'\}, \text{ and} \tag{5.3.1}$$

$$\widetilde{D}(s) = \min_{d' \in \mathbb{N}}\big\{1 + d' : \widetilde{T}(s) = \inf_{a,t}\{t + \widetilde{T}(s') : s \xrightarrow{a}_t s' \text{ and } \widetilde{D}(s') = d'\}\big\}. \tag{5.3.2}$$

The proof of the corresponding equalities for states $s \in S_{\mathrm{Max}} \setminus F$ is similar and omitted.

First we prove the equality (5.3.1).

$$\begin{aligned}
\widetilde{T}(s) &= \min_{m \in \mathcal{M}}\big\{T(R')^{\oplus}_{\alpha}(s) : m = ([s], \alpha, R')\big\} \\
&= \min\Big\{ \min_{R'',a,R'}\big\{T(R')^{\oplus}_{s,a}(b - s(c)) : [s] \to_{b,c} R'' \xrightarrow{a} R'\big\}, \\
&\qquad \min_{R'',a,R'}\big\{T(R')^{\oplus}_{s,a}(b - s(c)) : [s] \to_{b,c} R''' \to_{+1} R'' \xrightarrow{a} R'\big\}\Big\} \\
&= \min_{R'',a,R'}\big\{ \inf_{t}\{T(R')^{\oplus}_{s,a}(t) : [s + t] = R''\} : [s] \to_{*} R'' \xrightarrow{a} R'\big\} \\
&= \min_{R'',a,R'}\big\{ \inf_{t}\{t + \widetilde{T}(\mathsf{Succ}(s, (a, t))) : [s + t] = R''\} : [s] \to_{*} R'' \xrightarrow{a} R'\big\} \\
&= \inf_{a,t}\{t + \widetilde{T}(s') : s \xrightarrow{a}_t s'\}
\end{aligned}$$

The first equality holds by the assumption that $T \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$. The second equality holds by the definition of the move relation $\mathcal{M}$ of the boundary region automaton Γ_{BR}, and because if $\alpha = (a, b, c)$ then

$$\begin{aligned}
T(R')^{\oplus}_{\alpha}(s) &= b - s(c) + T(R')(\mathsf{Succ}(s, (a, b - s(c)))) \\
&= T(R')^{\oplus}_{s,a}(b - s(c)).
\end{aligned}$$

For the third equality we invoke regional simplicity of T which by Proposition 5.3.2 implies that the function $T(R')^{\oplus}_{s,a}$ is continuous and nondecreasing. If either $[s] \to_{b,c} R'' \xrightarrow{a} R'$, or $[s] \to_{b,c} R''' \to_{+1} R'' \xrightarrow{a} R'$, then we have that $\inf\{t : [s + t] = R''\} = b - s(c)$, and hence

$$\inf_{t}\{T(R')^{\oplus}_{s,a}(t) : [s + t] = R''\} = T(R')^{\oplus}_{s,a}(b - s(c)),$$

because $T(R')^{\oplus}_{s,a}$ is continuous and nondecreasing. The fourth equality holds as $[s + t] = R''$ and $R'' \xrightarrow{a} R'$ imply that $[\mathsf{Succ}(s, (a, t))] = R'$, and hence we have $T(R')(\mathsf{Succ}(s, (a, t))) = \widetilde{T}(\mathsf{Succ}(s, (a, t)))$.

Now we prove the equality (b).

$$\begin{aligned}\widetilde{D}(s) &= \min_{m\in\mathcal{M}}\left\{D(R')^{\boxplus}_{\alpha}(s) \,:\, \widetilde{T}(s) = T(R')^{\oplus}_{\alpha}(s) \text{ and } m = ([s],\alpha,R')\right\}\\ &= \min_{d'\in\mathbb{N}}\left\{1+d' \,:\, \widetilde{T}(s) = T(R')^{\oplus}_{\alpha}(s) \text{ and } ([s],\alpha,R')\in\mathcal{M} \text{ and } D(R')\equiv d'\right\}\\ &= \min_{d'\in\mathbb{N}}\left\{1+d' \,:\, \widetilde{T}(s) = \inf_{a,t}\{t+\widetilde{T}(s') \,:\, s\xrightarrow{a}_t s' \text{ and } \widetilde{D}(s')=d'\}\right\}\end{aligned}$$

The first equality holds by the assumption that $(T,D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$. The second equality holds because of the assumption that D is regionally constant, and we write $D(R') \equiv d'$, where $d' \in \mathbb{N}$, to express that for all $s \in R'$, we have $D(R')(s) = d'$. Finally, to establish the third equality it is sufficient to perform a calculation analogous to the above proof of (a), in order to show that $\widetilde{T}(s) = T(R')^{\oplus}_{\alpha}(s)$ and $([s],\alpha,R') \in \mathcal{M}$ and $D(R') \equiv d'$ if and only if $\widetilde{T}(s) = \inf_{a,t}\{t+\widetilde{T}(s') \,:\, s\xrightarrow{a}_t s' \text{ and } \widetilde{D}(s')=d'\}$. ∎

5.4. Solving Optimality Equations by Strategy Improvement

So far, we showed that if there exists a solution (T,D) of $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$ such that T is regionally simple and D is regionally constant, then $(\widetilde{T},\widetilde{D})$ is a solution of $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma)$. In this section, using a strategy improvement algorithm, we give a constructive proof of the fact that for every boundary region automaton $\mathcal{T}_{\mathrm{BR}}$ there exists a solution $(T,D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$ such that T is regionally simple and D is regionally constant.

We begin by defining a special class of strategies in boundary region automata which we call *regionally constant positional strategies*. These strategies are of interest as all strategies appearing in our strategy improvement algorithm are regionally constant positional strategies. In Subsection 5.4.2 we define optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$ and $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Min}}(\Gamma_{\mathrm{BR}})$ characterising maximum and minimum, respectively, reachability-price in a boundary region automata. In Subsection 5.4.3 we present strategy improvement algorithm to solve $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$, which is called as a subroutine in the strategy improvement algorithm presented in Subsection 5.4.4 to solve $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$.

5.4.1. Positional Strategies in Boundary Region Automata

A positional strategy for player Max in a boundary region automaton Γ_{BR} is a function $\chi : S_{\mathrm{Max}} \to \mathcal{M}$, such that for every $s \in S_{\mathrm{Max}}$, we have $\chi(s) = ([s],\alpha,R)$, for some $\alpha \in \mathcal{A}$ and $R \in \mathcal{R}$. A strategy $\chi : S_{\mathrm{Max}} \to \mathcal{M}$ is *regionally constant* if for all $s, s' \in S_{\mathrm{Max}}$, we have that $[s] = [s']$ implies $\chi(s) = \chi(s')$; we can then write $\chi([s])$ for $\chi(s)$. Positional strategies for player Min are defined analogously. We write Δ_{Max} and Δ_{Min} for the sets of positional strategies for players Max and Min, respectively.

If $\chi \in \Delta_{\mathrm{Max}}$ is regionally constant then we define the strategy subgraph $\Gamma_{\mathrm{BR}}\restriction\chi$ to be the subgraph $(\mathcal{R},\mathcal{M}_\chi)$ where $\mathcal{M}_\chi \subseteq \mathcal{M}$ consists of: all moves $(R,\alpha,R') \in \mathcal{M}$, such that $R \in \mathcal{R}_{\mathrm{Min}}$; and of all moves $m = (R,\alpha,R')$, such that $R \in \mathcal{R}_{\mathrm{Max}}$ and $\chi(R) = m$. The strategy subgraph $\Gamma_{\mathrm{BR}}\restriction\mu$ for a regionally constant positional strategy $\mu \in \Delta_{\mathrm{Min}}$ for player Min is defined analogously.

We say that $R \in \mathcal{R}$ is *choiceless* in a boundary region automaton Γ_{BR} if R has a unique successor in Γ_{BR}. We say that Γ_{BR} is 0-player if all $R \in \mathcal{R}$ are choiceless in Γ_{BR}; we say that Γ_{BR} is 1-player if either all $R \in \mathcal{R}_{\mathrm{Min}}$ or all $R \in \mathcal{R}_{\mathrm{Max}}$ are choiceless in Γ_{BR}; every boundary region automaton Γ_{BR} is 2-player. Note that if χ and μ are positional strategies in Γ_{BR} for players Max and Min, respectively, then $\Gamma_{\mathrm{BR}}\restriction\chi$ and $\Gamma_{\mathrm{BR}}\restriction\mu$ are 1-player and $(\Gamma_{\mathrm{BR}}\restriction\chi)\restriction\mu$ is 0-player.

For functions $T : \mathcal{R} \to [S \to \mathbb{R}]$ and $D : \mathcal{R} \to [S \to \mathbb{R}]$, and $s \in S_{\mathrm{Max}}$, we define sets $M^*(s,(T,D))$ and $M_*(s,(T,D))$, respectively, of moves enabled in s which are (lexicographically) (T,D)-optimal for player Max and Min, respectively:

$$\begin{aligned} M^*(s,(T,D)) &= \operatorname*{argmax}^{\mathrm{lex}}_{m\in\mathcal{M}} \left\{ \left(T(R')^{\oplus}_{\alpha}(s), D(R')^{\boxplus}_{\alpha}(s)\right) \,:\, m = ([s],\alpha,R') \right\}, \text{ and} \\ M_*(s,(T,D)) &= \operatorname*{argmin}^{\mathrm{lex}}_{m\in\mathcal{M}} \left\{ \left(T(R')^{\oplus}_{\alpha}(s), D(R')^{\boxplus}_{\alpha}(s)\right) \,:\, m = ([s],\alpha,R') \right\}. \end{aligned}$$

Let $\mathsf{Choose} : 2^{\mathcal{M}} \to \mathcal{M}$ be a function such that for every non-empty set of moves $M \subseteq \mathcal{M}$, we have $\mathsf{Choose}(M) \in M$. For regional functions $T : \mathcal{R} \to [S \rightharpoonup \mathbb{R}]$ and $D : \mathcal{R} \to [S \rightharpoonup \mathbb{N}]$, the canonical (T,D)-optimal strategies $\chi_{(T,D)}$ and $\mu_{(T,D)}$ for player Max and Min, respectively, are defined by: $\chi_{(T,D)}(s) = \mathsf{Choose}(M^*(s,(T,D)))$, for every $s \in S_{\mathrm{Max}}$; and $\mu_{(T,D)}(s) = \mathsf{Choose}(M_*(s,(T,D)))$, for every $s \in S_{\mathrm{Min}}$.

5.4.2. Optimality Equations $\mathcal{OE}^{\mathsf{RT}}_{\mathbf{Max}}(\Gamma_{\mathbf{BR}})$, $\mathcal{OE}^{\mathsf{RT}}_{\mathbf{Max}}(\Gamma_{\mathbf{BR}})$, and $\mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathbf{BR}})$

Optimality equations sets $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$, $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$, and $\mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathrm{BR}})$ are introduced. Intuitively, $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$ ($\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$) characterises optimal reachability-price when player Min (Max) is choiceless, while $\mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathrm{BR}})$ characterises optimal reachability-price when both players are choiceless. We also define the sets $\mathcal{OE}^{\mathsf{RT}}_{\geq}(\Gamma_{\mathrm{BR}})$ and $\mathcal{OE}^{\mathsf{RT}}_{\leq}(\Gamma_{\mathrm{BR}})$ that are useful technical tools in showing strict improvement in every iteration of strategy improvement algorithm.

Let $T : \mathcal{R} \to [S \to \mathbb{R}]$ and $D : \mathcal{R} \to [S \to \mathbb{N}]$. We write $(T,D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$ if for all $s \in S$ we have the following:

$$\left(\widetilde{T}(s), \widetilde{D}(s)\right) = \begin{cases} (0,0) & \text{if } s \in F \\ \max^{\mathrm{lex}}_{m\in\mathcal{M}} \left\{ \left(T(R')^{\oplus}_{\alpha}(s), D(R')^{\boxplus}_{\alpha}(s)\right) \,:\, m = ([s],\alpha,R') \right\}, & \text{otherwise.} \end{cases}$$

Similarly we write $(T,D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Min}}(\Gamma_{\mathrm{BR}})$, if for all $s \in F$, we have the following:

$$\left(\widetilde{T}(s), \widetilde{D}(s)\right) = \begin{cases} (0,0) & \text{if } s \in F \\ \min^{\mathrm{lex}}_{m\in\mathcal{M}} \left\{ \left(T(R')^{\oplus}_{\alpha}(s), D(R')^{\boxplus}_{\alpha}(s)\right) \,:\, m = ([s],\alpha,R') \right\}, & \text{otherwise.} \end{cases}$$

If Γ_{BR} is 0-player then $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$ and $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Min}}(\Gamma_{\mathrm{BR}})$ are equivalent to each other and denoted by $\mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathrm{BR}})$.

We write $(T,D) \models \mathcal{OE}^{\mathsf{RT}}_{\geq}(\Gamma_{\mathrm{BR}})$ if for all $s \in S$ we have:

$$\left(\widetilde{T}(s), \widetilde{D}(s)\right) \geq^{\text{lex}} \begin{cases} (0,0) & \text{if } s \in F \\ \max^{\text{lex}}{}_{m \in \mathcal{M}} \left\{ \left(T(R')^{\oplus}_{\alpha}(s), D(R')^{\boxplus}_{\alpha}(s)\right) \,:\, m = ([s], \alpha, R') \right\}, & \text{otherwise.} \end{cases}$$

Similarly we write $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\leq}(\Gamma_{\text{BR}})$ if for all $s \in S$, we have:

$$\left(\widetilde{T}(s), \widetilde{D}(s)\right) \leq^{\text{lex}} \begin{cases} (0,0) & \text{if } s \in F \\ \min^{\text{lex}}{}_{m \in \mathcal{M}} \left\{ \left(T(R')^{\oplus}_{\alpha}(s), D(R')^{\boxplus}_{\alpha}(s)\right) \,:\, m = ([s], \alpha, R') \right\}, & \text{otherwise.} \end{cases}$$

PROPOSITION 5.4.1 (Relaxations of optimality equations)**.** If $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\text{Max}}(\Gamma_{\text{BR}})$ then $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\geq}(\Gamma_{\text{BR}})$, and if $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\text{Min}}(\Gamma_{\text{BR}})$ then $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\leq}(\Gamma_{\text{BR}})$.

LEMMA 5.4.2 (Solution of $\mathcal{OE}^{\mathsf{RT}}(\Gamma_{\text{BR}})$ is regionally simple)**.** Let Γ_{BR} be a 0-player boundary region automaton. If $(T, D) \models \mathcal{OE}^{\mathsf{RT}}(\Gamma_{\text{BR}})$ then T is regionally simple and D is regionally constant.

PROOF. In a 0-player boundary region automaton Γ_{BR}, for every region R, there is at most one outgoing labelled edge $(R, \alpha, R') \in \mathcal{M}$, and hence for every region R, there is a unique $\mathcal{M}$-path from R in Γ_{BR}. For every region $R \in \mathcal{R}$, we define the distance $d(R) \in \mathbb{N}$ to be the smallest number of edges in the unique $\mathcal{M}$-path from R, that one needs to reach a final region. It is easy to show that for every state $s \in S$, we have that $D([s])(s) = d([s])$, and hence D is regionally constant.

We prove that for every region $R \in \mathcal{R}$, the function $T(R) : \overline{R} \to \mathbb{R}$ is simple, by induction on $d(R)$. If $d(R) = 0$ then $T(R)(s) = 0$ for all $s \in \overline{R}$, and hence $T(R)$ is simple on $\overline{R}$.

Let $d(R) = n + 1$ and let $(R, \alpha, R') \in \mathcal{M}$ be the unique edge going out of R in Γ_{BR}. Observe that $T(R) = T(R')^{\oplus}_{\alpha}$ because for every $s \in R$, we have $T(R)(s) = T([s])(s) = T(R')^{\oplus}_{\alpha}(s)$, where the second equality follows from $(T, D) \models \mathcal{OE}^{\mathsf{RT}}(\Gamma_{\text{BR}})$. Moreover, by the induction hypothesis the function $T(R') : \overline{R'} \to \mathbb{R}$ is simple, and hence by Proposition 5.3.1 we get that $T(R')^{\oplus}_{\alpha} = T(R)$ is simple.

If $d(R) = \infty$, i.e., if the unique $\mathcal{M}$-path from R in Γ_{BR} never reaches a final region, then we set $T(R')(s) = \infty$, for all $s \in \overline{R}$. Therefore $T(R') : \overline{R} \to \mathbb{R}$ is a constant function and hence it is simple. ∎

5.4.3. Solving 1-Player Reachability-Time Optimality Equations $\mathcal{OE}^{\mathsf{RT}}_{\text{Max}}(\Gamma_{\text{BR}})$

In this section we give a strategy improvement algorithm for solving maximum reachability-time optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\text{Max}}(\Gamma_{\text{BR}})$ for a 1-player boundary region automaton Γ_{BR}.

We define the following strategy improvement operator $\text{Improve}_{\text{Max}}$:

$$\text{Improve}_{\text{Max}}(\chi, (T, D))(s) = \begin{cases} \chi(s) & \text{if } \chi(s) \in M^*(s, (T, D)), \\ \mathsf{Choose}(M^*(s, T)) & \text{if } \chi(s) \notin M^*(s, (T, D)). \end{cases}$$

Note that $\text{Improve}_{\text{Max}}(\chi,(T,D))(s)$ may differ from the canonical (T,D)-optimal choice $\chi_{(T,D)}(s)$ only if $\chi(s)$ is itself (T,D)-optimal in state s, i.e., if $\chi(s) \in M^*(s,(T,D))$.

LEMMA 5.4.3 (Improvement preserves regional constancy of strategies)**.** If $\chi \in \Delta_{\text{Max}}$ is regionally constant, $T : \mathcal{R} \to [S \to \mathbb{R}]$ is regionally simple, and $D : \mathcal{R} \to [S \to \mathbb{N}]$ is regionally constant, then $\text{Improve}_{\text{Max}}(\chi,(T,D))$ is regionally constant.

PROOF. We need to prove that for $s,s' \in S$, if $[s] = [s']$ then $\chi'(s) = \chi'(s')$, where $\chi' = \text{Improve}_{\text{Max}}(\chi,(T,D))$. By regionality of χ it is sufficient to prove that $M^*(s,(T,D)) = M^*(s',(T,D))$. By regional simplicity of T, and by Proposition 5.3.1, we have that functions $T(R)^{\oplus}_{\alpha} : [s] \to \mathbb{R}$, for all $m = ([s],\alpha,R) \in \mathcal{M}$, are simple. Then we have

$$\begin{aligned} M^*(s,(T,D)) &= \operatorname*{argmax}^{\text{lex}}_{m\in\mathcal{M}} \left\{ \left(T(R)^{\oplus}_{\alpha}(s), D(R)^{\boxplus}_{\alpha}(s)\right) \,:\, m = ([s],\alpha,R) \right\} \\ &= \operatorname*{argmax}^{\text{lex}}_{m\in\mathcal{M}} \left\{ \left(T(R)^{\oplus}_{\alpha}(s'), D(R)^{\boxplus}_{\alpha}(s')\right) \,:\, m = ([s'],\alpha,R) \right\} \\ &= M^*(s',(T,D)), \end{aligned}$$

where the second equality follows from $[s] = [s']$, regional constancy of D, and by the application of Lemma 5.2.2 to the (finite) set of functions $\{T(R)^{\oplus}_{\alpha} \,:\, ([s],\alpha,R) \in \mathcal{M}\}$. ∎

Input: Boundary Region Automaton Γ_{BR}
Output: A solution of $\mathcal{OE}^{\mathsf{RT}}_{\text{Max}}(\Gamma_{\text{BR}})$

```
1  begin
2      (Initialisation). Choose a regionally constant positional strategy χ_0 ∈ Δ_Max for
       player Max in Γ_BR;
3      Set i := 0;
4      repeat
5          (Value Computation). Compute the solution (T_i, D_i) of OE^RT(Γ_BR ⌈χ_i);
6          (Strategy Improvement). Compute χ_{i+1} = Improve_Max(χ_i, (T_i, D_i));
7          Set i := i + 1;
8      until χ_{i+1} ≡ χ_i ;
9      return (T_i, D_i);
10 end
```

FIGURE 5.1. Strategy improvement algorithm for $\mathcal{OE}^{\mathsf{RT}}_{\text{Max}}(\Gamma_{\text{BR}})$.

PROPOSITION 5.4.4 (Fixpoints of $\text{Improve}_{\text{Max}}$ are solutions of $\mathcal{OE}^{\mathsf{RT}}_{\text{Max}}(\Gamma_{\text{BR}})$)**.** Let $\chi \in \Delta_{\text{Max}}$ and let $(T^{\chi},D^{\chi}) \models \mathcal{OE}^{\mathsf{RT}}(\Gamma_{\text{BR}}\restriction\chi)$. If $\text{Improve}_{\text{Max}}(\chi,(T^{\chi},D^{\chi})) = \chi$ then we have that $(T^{\chi},D^{\chi}) \models \mathcal{OE}^{\mathsf{RT}}_{\text{Max}}(\Gamma_{\text{BR}})$.

If $F,F' : \mathcal{R} \to [S \to \mathbb{R}]$ then we write $F \leq F'$ if for all $R \in \mathcal{R}$, and for all $s \in \overline{R}$, we have $F(R)(s) \leq F'(R)(s)$. Moreover, $F < F'$ if $F \leq F'$ and there is $R \in \mathcal{R}$ and $s \in R$, such that $F(R)(s) < F'(R)(s)$. If $F,G,F',G' : \mathcal{R} \to [S \to \mathbb{R}]$ then $(F,G) \leq^{\text{lex}} (F',G')$ if $F < F'$, or if $F = F'$ and $G \leq G'$.

The following proposition characterises the solution of optimality equations $\mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathrm{BR}})$ for a 0-player boundary region automaton Γ_{BR}, as the lexicographically *maximum* solution of the system of inequalities $\mathcal{OE}^{\mathsf{RT}}_{\leq}(\Gamma_{\mathrm{BR}})$.

PROPOSITION 5.4.5 (Solution of $\mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathrm{BR}})$ is the maximum solution of $\mathcal{OE}^{\mathsf{RT}}_{\leq}(\Gamma_{\mathrm{BR}})$)**.** Let $T, T_{\leq} : \mathcal{R} \to [S \to \mathbb{R}]$ and $D, D_{\leq} : \mathcal{R} \to [S \to \mathbb{N}]$ be such that $(T, D) \models \mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathrm{BR}})$ and $(T_{\leq}, D_{\leq}) \models \mathcal{OE}^{\mathsf{RT}}_{\leq}(\Gamma_{\mathrm{BR}})$. Then we have $(T_{\leq}, D_{\leq}) \leq^{\mathrm{lex}} (T, D)$, and if $(T_{\leq}, D_{\leq}) \not\models \mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathrm{BR}})$ then the inequality is strict, i.e., we have $(T_{\leq}, D_{\leq}) <^{\mathrm{lex}} (T, D)$.

PROOF. Our first goal is to establish that for every $s \in S$, we have $(\widetilde{T_{\leq}}(s), \widetilde{D_{\leq}}(s)) \leq_{\mathrm{lex}} (\widetilde{T}(s), \widetilde{D}(s))$. We proceed by induction on $\widetilde{D}(s)$, i.e., on the length of the $\chi_{(T,D)}$-path in Γ_{BR} from $[s]$ to a final region. The trivial base case is when $[s]$ is a final region, because then $(\widetilde{T}(s), \widetilde{D}(s)) = (0, 0)$ and $(\widetilde{T_{\leq}}(s), \widetilde{D_{\leq}}(s)) \leq_{\mathrm{lex}} (0, 0)$. Let $s \in S \setminus F$ be such that $\widetilde{D}(s) = n + 1$. Then $\widetilde{D}(\mathsf{Succ}(s, \chi_{(T,D)}(s))) = n$ and if $\chi_{(T,D)}(s) = ([s], \alpha, R')$ then we have the following:

$$\begin{array}{rcl} \big(\widetilde{T_{\leq}}(s), \widetilde{D_{\leq}}(s)\big) & \leq_{\mathrm{lex}} & \big(T_{\leq}(R')^{\oplus}_{\alpha}(s), D_{\leq}(R')^{\boxplus}_{\alpha}(s)\big) \\ & \leq_{\mathrm{lex}} & \big(T(R')^{\oplus}_{\alpha}(s), D(R')^{\boxplus}_{\alpha}(s)\big) \\ & = & \big(\widetilde{T}(s), \widetilde{D}(s)\big)\,, \end{array} \tag{5.4.1}$$

where the first inequality follows from $(T_{\leq}, D_{\leq}) \models \mathcal{OE}^{\mathsf{RT}}_{\leq}(\Gamma_{\mathrm{BR}})$, the second inequality follows from the induction hypothesis, and the last equality follows from $(T, D) \models \mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathrm{BR}})$ and $\chi_{(T,D)}(s) = ([s], \alpha, R')$. This concludes the proof that $(T_{\leq}, D_{\leq}) \leq_{\mathrm{lex}} (T, D)$.

We prove that if $(T_{\leq}, D_{\leq}) \not\models \mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathrm{BR}})$ then there is $s \in S$, such that $(\widetilde{T_{\leq}}(s), \widetilde{D_{\leq}}(s)) <_{\mathrm{lex}} (\widetilde{T}(s), \widetilde{D}(s))$. Indeed, if $(T_{\leq}, D_{\leq}) \not\models \mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathrm{BR}})$ then either $(\widetilde{T_{\leq}}(s), \widetilde{D_{\leq}}(s)) <_{\mathrm{lex}} (0, 0)$ for some $s \in F$, or there is $s \in S \setminus F$, for which the inequality in (5.4.1) is strict and hence we get $(\widetilde{T_{\leq}}(s), \widetilde{D_{\leq}}(s)) <_{\mathrm{lex}} (\widetilde{T}(s), \widetilde{D}(s))$. ∎

LEMMA 5.4.6 (Strict strategy improvement for player Max)**.** Let $\chi, \chi' \in \Delta_{\mathrm{Max}}$, let $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Min}}(\Gamma_{\mathrm{BR}}{\restriction}\chi)$ and $(T', D') \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Min}}(\Gamma_{\mathrm{BR}}{\restriction}\chi')$, and let $\chi' = \mathrm{Improve}_{\mathrm{Max}}(\chi, (T, D))$. Then we have that $(T, D) \leq^{\mathrm{lex}} (T', D')$ and if $\chi \neq \chi'$ then $(T, D) <^{\mathrm{lex}} (T', D')$.

PROOF. First we argue that $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\leq}(\Gamma_{\mathrm{BR}}{\restriction}\chi')$ which by Proposition 5.4.5 implies that $(T, D) \leq (T', D')$. Indeed for every $s \in S \setminus F$, if $\chi(s) = ([s], \alpha, R)$ and $\chi'(s) = ([s], \alpha', R')$ then we have

$$\begin{array}{rcl} \big(\widetilde{T}(s), \widetilde{D}(s)\big) & = & \big(T(R)^{\oplus}_{\alpha}(s), D(R)^{\boxplus}_{\alpha}(s)\big) \\ & \leq^{\mathrm{lex}} & \big(T(R')^{\oplus}_{\alpha'}(s), D(R')^{\boxplus}_{\alpha'}(s)\big), \end{array}$$

where the equality follows from $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Min}}(\Gamma_{\mathrm{BR}}{\restriction}\chi)$, and the inequality follows from the definition of $\mathrm{Improve}_{\mathrm{Max}}$. Moreover, if $\chi \neq \chi'$ then there is $s \in S_{\mathrm{Max}} \setminus F$ for which the above inequality is strict. Then $(T, D) \not\models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Min}}(\Gamma_{\mathrm{BR}}{\restriction}\chi')$ because every vertex in $\Gamma_{\mathrm{BR}}{\restriction}\chi'$ has a unique successor, and hence again by Proposition 5.4.5 we conclude that $(T, D) <^{\mathrm{lex}} (T', D')$. ∎

The following theorem is an immediate corollary of Lemmas 5.4.2 and 5.4.3 (the algorithm considers only regionally constant strategies), of Lemma 5.4.6 and finiteness of

the number of regionally constant positional strategies for Max (the algorithm terminates), and of Proposition 5.4.4 (the algorithm returns a solution of optimality equations).

THEOREM 5.4.7 (Correctness and termination of strategy improvement for $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$)**.** The strategy improvement algorithm for $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$ terminates in finitely many steps and returns a solution (T, D) of $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$, such that T is regionally simple and D is regionally constant.

5.4.4. Solving 2-Player Reachability-Time Optimality Equations $\mathcal{OE}^{\mathsf{RT}}_{\mathbf{MinMax}}(\Gamma_{\mathbf{BR}})$

In this section we give a strategy improvement algorithm for solving optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$ for a 2-player boundary region automaton Γ_{BR}. The structure of the algorithm is very similar to that of Algorithm 5.1. The only difference is that in step 2. of every iteration we solve 1-player optimality equations $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}}\restriction\mu)$ instead of 0-player optimality equations $\mathcal{OE}^{\mathsf{RT}}(\Gamma_{\mathrm{BR}}\restriction\chi)$. Note that we can perform step 2. of Algorithm 5.2 below by using Algorithm 5.1.

We define the following strategy improvement operator $\mathrm{Improve}_{\mathrm{Min}}$:

$$\mathrm{Improve}_{\mathrm{Min}}(\mu,(T,D))(s) = \begin{cases} \mu(s) & \text{if } \mu(s) \in M_*(s,(T,D)), \\ \mathsf{Choose}(M_*(s,(T,D))) & \text{if } \mu(s) \notin M_*(s,(T,D)). \end{cases}$$

Note that $\mathrm{Improve}_{\mathrm{Min}}(\mu,(T,D))(s)$ may differ from the canonical (T,D)-optimal choice $\mu_{(T,D)}(s) = \mathsf{Choose}(M_*(s,(T,D)))$ only if $\mu(s)$ is itself (T,D)-optimal in state s, i.e., if $\mu(s) \in M_*(s,(T,D))$.

The proof of the following lemma is the same as for Lemma 5.4.3.

LEMMA 5.4.8 (Improvement preserves regional constancy of strategies)**.** If $\mu \in \Delta_{\mathrm{Min}}$ is regionally constant, $T : \mathcal{R} \to [S \to \mathbb{R}]$ is regionally simple, and $D : \mathcal{R} \to [S \to \mathbb{R}]$ is regionally constant, then $\mathrm{Improve}_{\mathrm{Min}}(\mu,(T,D))$ is regionally constant.

PROPOSITION 5.4.9 (Fixpoints of $\mathrm{Improve}_{\mathrm{Min}}$ are solutions of $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$)**.** Let $\mu \in \Delta_{\mathrm{Min}}$ and $(T^\mu, D^\mu) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}}\restriction\mu)$. If $\mathrm{Improve}_{\mathrm{Min}}(\mu,(T^\mu,D^\mu)) = \mu$ then we have that $(T^\mu, D^\mu) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$.

The following proposition is proved in the same way as Proposition 5.4.5.

PROPOSITION 5.4.10 (Solution of $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$ is the minimum solution of $\mathcal{OE}^{\mathsf{RT}}_{\geq}(\Gamma_{\mathrm{BR}})$)**.** Let $T, T_\geq : \mathcal{R} \to [S \to \mathbb{R}]$ and $D, D_\geq : \mathcal{R} \to [S \to \mathbb{R}]$ be such that $(T,D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$ and $(T_\geq, D_\geq) \models \mathcal{OE}^{\mathsf{RT}}_{\geq}(\Gamma_{\mathrm{BR}})$. Then $(T_\geq, D_\geq) \geq^{\mathrm{lex}} (T,D)$, and if $(T_\geq, D_\geq) \not\models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}})$ then $(T_\geq, D_\geq) >^{\mathrm{lex}} (T,D)$.

LEMMA 5.4.11 (Strict strategy improvement for player Min)**.** Let strategies $\mu, \mu' \in \Delta_{\mathrm{Min}}$, let $(T,D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}}\restriction\mu)$ and $(T',D') \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}}\restriction\mu')$, and let $\mu' = \mathrm{Improve}_{\mathrm{Min}}(\mu,(T,D))$. Then we have $(T,D) \geq^{\mathrm{lex}} (T',D')$, and if $\mu \neq \mu'$ then $(T,D) >^{\mathrm{lex}} (T',D')$.

PROOF. First we argue that $(T,D) \models \mathcal{OE}^{\mathsf{RT}}_{\geq}(\Gamma_{\mathrm{BR}}\restriction\mu')$ which by Proposition 5.4.10 implies that $(T,D) \geq^{\mathrm{lex}} (T',D')$. Indeed for every $s \in S \setminus F$, if $\mu(s) = ([s],\alpha,R)$ and $\mu'(s) =$

Input: Boundary Region Automaton Γ_{BR}
Output: A solution of $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$

begin
  (Initialisation). Choose a regionally constant positional strategy $\mu_0 \in \Delta_{\mathrm{Min}}$ for player Min in Γ_{BR};
  Set $i := 0$;
  repeat
    (Value Computation). Compute the solution (T_i, D_i) of $\mathcal{OE}^{\mathrm{Max}}(\Gamma_{\mathrm{BR}}\restriction\mu_i)$;
    (Strategy Improvement). Compute $\mu_{i+1} = \mathrm{Improve}_{\mathrm{Min}}(\mu_i, (T_i, D_i))$;
    Set $i := i + 1$;
  until $\mu_{i+1} \equiv \mu_i$;
  return (T_i, D_i);
end

FIGURE 5.2. Strategy improvement algorithm for solving $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$.

$([s], \alpha', R')$ then we have

$$\begin{array}{rcl} \big(\widetilde{T}(s), \widetilde{D}(s)\big) & = & \big(T(R)^{\oplus}_{\alpha}(s), D(R)^{\boxplus}_{\alpha}(s)\big) \\ & \geq^{\mathrm{lex}} & \big(T(R')^{\oplus}_{\alpha'}(s), D(R')^{\boxplus}_{\alpha'}(s)\big), \end{array}$$

where the equality follows from $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}}\restriction\mu)$, and the inequality follows from the definition of $\mathrm{Improve}_{\mathrm{Min}}$. Moreover, if $\mu \neq \mu'$ then there is $s \in S_{\mathrm{Min}} \setminus F$ for which the above inequality is strict. Then $(T, D) \not\models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma_{\mathrm{BR}}\restriction\mu')$ because every vertex $R \in \mathcal{R}_{\mathrm{Min}}$ in $\Gamma_{\mathrm{BR}}\restriction\mu'$ has a unique successor, and hence again by Proposition 5.4.10 we conclude that $(T, D) >^{\mathrm{lex}} (T', D')$. ∎

The following theorem is an immediate corollary of Theorem 5.4.7 and Lemma 5.4.8, of Lemma 5.4.11 and finiteness of the number of regionally constant positional strategies for Min, and of Proposition 5.4.9.

THEOREM 5.4.12 (Correctness and termination of strategy improvement for $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$)**.** The strategy improvement algorithm for $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$ terminates in finitely many steps and returns a solution (T, D) of $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma_{\mathrm{BR}})$, such that T is regionally simple and D is regionally constant.

5.5. Complexity

LEMMA 5.5.1 (Complexity of strategy improvement)**.** Let Γ^0_{BR}, Γ^1_{BR}, and Γ^2_{BR} be 0-player, 1-player, and 2-player boundary region automata, respectively. A solution of $\mathcal{OE}^{\mathsf{RT}}(\Gamma^0_{\mathrm{BR}})$ can be computed in time $O(|\mathcal{R}|)$. The strategy improvement algorithms for $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma^1_{\mathrm{BR}})$ and $\mathcal{OE}^{\mathsf{RT}}_{\mathrm{MinMax}}(\Gamma^2_{\mathrm{BR}})$ terminate in $O(|\mathcal{R}|)$ iterations and hence run in $O(|\mathcal{R}|^2)$ and $O(|\mathcal{R}|^3)$ time, respectively.

PROOF. An $O(|\mathcal{R}|)$ algorithm to solve $\mathcal{OE}^{\mathsf{RT}}(\Gamma^0_{\mathrm{BR}})$ is implicit in the proof of Lemma 5.4.2.

Let $(T, D) \models \mathcal{OE}^{\mathsf{RT}}_{\mathrm{Max}}(\Gamma^1_{\mathrm{BR}})$; and for all $i \geq 0$, let $\chi_i \in \Delta_{\mathrm{Max}}$ be the strategy in the i-th iteration of Algorithm 5.1, and let $(T_i, D_i) \models \mathcal{OE}^{\mathsf{RT}}(\Gamma^1_{\mathrm{BR}} \restriction \chi_i)$. We claim that for every $i \geq 0$, if $D(R) \equiv i$ then for all $j \geq i$, we have $(T_j(R), D_j(R)) = (T(R), D(R))$. This can be established by a routine induction on the values of the regionally constant function D. Observe that the finite values of the function D are bounded by $|\mathcal{R}|$, because in the proof of Lemma 5.4.2 they are set to be the length of a simple path in a boundary region automaton. Algorithm 5.1 must therefore terminate no later than after $|\mathcal{R}| + 1$ iterations, because for every $i \geq 0$, in the i-th iteration there must be $R \in \mathcal{R}$ whose value $D(R)$ is set to i.

An analogous routine proof by induction on the value of D can be used to prove that Algorithm 5.2 terminates in $O(|\mathcal{R}|)$ iterations. ∎

Since the number $|\mathcal{R}|$ of regions is at most exponential in the size of a timed automaton **[AD94]**, we conclude that the strategy improvement algorithm solves reachability-time games in exponential time.

COROLLARY 5.5.2. The problem of solving reachability-time games is in EXPTIME.

In order to prove EXPTIME-hardness of solving reachability-time games on timed automata with two clocks, we reduce the EXPTIME-complete countdown games to reachability games on timed automata.

THEOREM 5.5.3. The problem of solving reachability games is EXPTIME-complete on timed automata with at least two clocks.

PROOF. In order to solve a reachability game on a timed automaton it is sufficient to solve the reachability game on the finite region graph of the automaton. Observe that every region, and hence also every configuration of the game, can be written down in polynomial space, and that every move of the game can be simulated in polynomial time. Therefore, the winner in the game can be determined by a straightforward alternating PSPACE algorithm, and hence the problem is in EXPTIME because APSPACE = EXPTIME. In order to prove

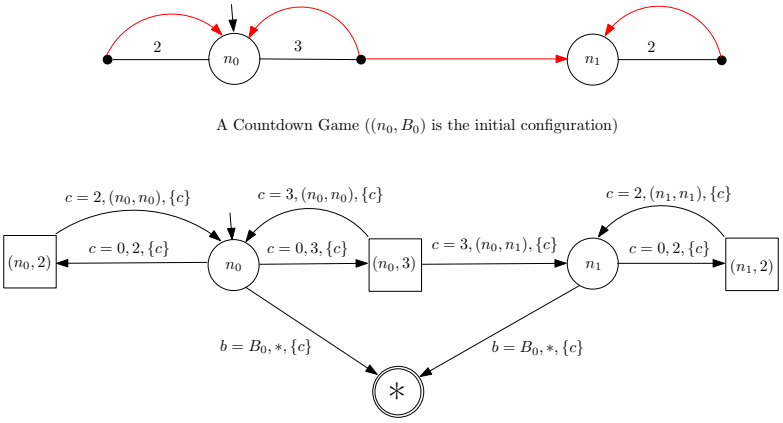


FIGURE 5.3. A Reduction from a Countdown Game to a Reachability Game.

EXPTIME-hardness of solving reachability games on timed automata with two clocks, we reduce the EXPTIME-complete problem of solving countdown games [**JLS07**] to it.

Let $G = (N, M, \pi, n_0, B_0)$ be a countdown game, where N is a finite set of nodes, $M \subseteq N \times N$ is a set of moves, $\pi : M \to \mathbb{N}_+$ assigns a positive integer number to every move, and $(n_0, B_0) \in N \times \mathbb{N}_+$ is the initial configuration. In every move of the game from a configuration $(n, B) \in N \times \mathbb{N}_+$, first player 1 chooses a number $p \in \mathbb{N}_+$, such that $p \leq B$ and $\pi(n, n') = p$ for some move $(n, n') \in M$, and then player 2 chooses a move $(n, n'') \in M$, such that $\pi(n, n'') = p$; the new configuration is then $(n'', B - p)$. Player 1 wins a play of the game when a configuration $(n, 0)$ is reached, and he loses (i.e., player 2 wins) when a configuration (n, B) is reached in which player 1 is stuck, i.e., for all moves $(n, n') \in M$, we have $\pi(n, n') > B$.

We define the timed automaton $\mathcal{T}_G = (L, C, S, A, E, \delta, \xi, F)$ by setting $C = \{ b, c \}$; $S = L \times (\llbracket B_0 \rrbracket_{\mathbb{R}})^2$; $A = \{ * \} \cup P \cup M$, where $P = \pi(M)$, the image of the function $\pi : M \to \mathbb{N}_+$;

$$
\begin{aligned}
L &= \{ * \} \cup N \cup \big\{ (n, p) \,:\, \text{there is } (n, n') \in M, \text{s.t. } \pi(n, n') = p \big\}; \\
E(a) &= \begin{cases} \big\{ (n, \nu) \,:\, n \in N \text{ and } \nu(b) = B_0 \big\} & \text{if } a = *, \\ \big\{ (n, \nu) \,:\, \text{there is } (n, n') \in M, \text{s.t. } \pi(n, n') = p \text{ and } \nu(c) = 0 \big\} & \text{if } a = p \in P, \\ \big\{ \big((n, p), \nu \big) \,:\, \pi(n, n') = p \text{ and } \nu(c) = p \big\} & \text{if } a = (n, n') \in M, \end{cases} \\
\delta(\ell, a) &= \begin{cases} * & \text{if } \ell = n \in N \text{ and } a = *, \\ (n, p) & \text{if } \ell = n \in N \text{ and } a = p \in P, \\ n' & \text{if } \ell = (n, p) \in N \times P \text{ and } a = (n, n') \in M; \end{cases}
\end{aligned}
$$

$\xi(a) = \{ c \}$, for every $a \in A$; and $F = \{ * \} \times \mathcal{V}_C$. Note that the timed automaton $\mathcal{T}_G$ has only two clocks and that the clock b is never reset.

Finally, we define the reachability game on timed game automaton $\Gamma_G = (\mathcal{T}_G, L_1, L_2)$ by setting $L_1 = N$ and $L_2 = L \setminus L_1$. It is routine to verify that player 1 has a winning strategy from state $(n_0, (0, 0)) \in S$ in the reachability game Γ_G if and only if player 1 has a winning strategy (from the initial configuration (n_0, B_0)) in the countdown game G. An example of such reduction is shown in Figure 5.3. ∎

The reachability games for timed automata can be easily reduced, in logarithmic space, to the reachability-time games so, by Theorem 5.5.3, reachability-time games are EXPTIME-hard. The following theorem follows from this observation and Corollary 5.5.2.

THEOREM 5.5.4 (Complexity of reachability-time games on timed automata)**.** The problem of solving reachability-time games is EXPTIME-complete on timed automata with at least two clocks.

6

Average-Time Games

> The truth is, you can't go forever.
>
> Tony Blair

An average-time game is played on the infinite graph of configurations of a finite timed automaton. The two players, Min and Max, construct an infinite run of the automaton by taking turns to perform a timed transition. Player Min wants to minimise the average time per transition and player Max wants to maximise it. A solution of average-time games is presented using a reduction to average-price games on finite graphs. A direct consequence is an elementary proof of determinacy for average-time games. This complements our results for reachability-time games and partially solves a problem posed by Bouyer et al. **[BBL04]**, to design an algorithm for solving average-price games on priced timed automata. The exact computational complexity of solving average-time games is also established: the problem is EXPTIME-complete for timed automata with at least two clocks.

6.1. Introduction

6.1.1. Definition

DEFINITION 6.1.1 (Average-Time Games)**.** An *average-time game* on a timed automaton is a tuple $(\Gamma, \mathsf{AT}_{\mathrm{Min}}, \mathsf{AT}_{\mathrm{Max}})$, where:

- $\Gamma = (\mathcal{T}, L_{\mathrm{Min}}, L_{\mathrm{Max}})$ is a timed game automaton such that $\mathcal{T} = (L, C, S, A, E, \delta, \xi, F)$ is a timed automaton, L_{Min} is the set of locations controlled by player Min, and L_{Max} is the set of locations controlled by player Max;
- $\mathsf{AT}_{\mathrm{Min}} : \mathrm{Runs} \to \mathbb{R}$ and $\mathsf{AT}_{\mathrm{Max}} : \mathrm{Runs} \to \mathbb{R}$ are payoff functions, which for every run of the timed automaton return the amount the player Min loses and the player Max wins, respectively. The functions $\mathsf{AT}_{\mathrm{Min}}$ and $\mathsf{AT}_{\mathrm{Max}}$ are defined as follows: for a run $r = \langle s_0, (t_1, a_1), s_1, (t_2, a_2), \ldots \rangle \in \mathrm{Runs}$ we have

$$\mathsf{AT}_{\mathrm{Min}}(r) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} t_i \text{ and } \mathsf{AT}_{\mathrm{Max}}(r) = \liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} t_i.$$

We define $Q_{\text{Min}} = \{(\ell, \nu) \in Q : \ell \in L_{\text{Min}}\}$, $Q_{\text{Max}} = Q \setminus Q_{\text{Min}}$, $S_{\text{Min}} = S \cap Q_{\text{Min}}$, $S_{\text{Max}} = S \setminus S_{\text{Min}}$, $\mathcal{R}_{\text{Min}} = \{[s] : s \in Q_{\text{Min}}\}$, and $\mathcal{R}_{\text{Max}} = \mathcal{R} \setminus \mathcal{R}_{\text{Min}}$.

The strategies of player Min and player Max are defined as usual (see Section 3.4.2). We write Σ_{Min} for the set of strategies for player Min, and we write Σ_{Max} for the set of strategies for player Max. We write Π_{Min} and Π_{Max} for the sets of positional strategies for player Min and for player Max, respectively. Average-time payoff functions naturally give rise to the functions $\mathsf{AT}_{\text{Min}} : S \times \Sigma_{\text{Min}} \times \Sigma_{\text{Max}} \to \mathbb{R}$ and $\mathsf{AT}_{\text{Max}} : S \times \Sigma_{\text{Min}} \times \Sigma_{\text{Max}} \to \mathbb{R}$ in the following way. For strategies $\mu \in \Sigma_{\text{Min}}$ and $\chi \in \Sigma_{\text{Max}}$ of respective players and a state $s \in S$ we have $\mathsf{AT}_{\text{Min}}(s, \mu, \chi) = \mathsf{AT}_{\text{Min}}(\text{Run}(s, \mu, \chi))$ and $\mathsf{AT}_{\text{Max}}(s, \mu, \chi) = \mathsf{AT}_{\text{Max}}(\text{Run}(s, \mu, \chi))$.

6.1.2. Value of Average-Time Game

If player Min uses the strategy $\mu \in \Sigma_{\text{Min}}$ and player Max uses the strategy $\chi \in \Sigma_{\text{Max}}$ then player Min loses the value $\mathsf{AT}_{\text{Min}}(s, \mu, \chi)$ and player Max wins the value $\mathsf{AT}_{\text{Max}}(s, \mu, \chi)$. In an average-time game player Min is interested in minimising the value she loses and player Max is interested in maximising the value he wins. We define the *upper value* $\overline{\mathsf{Val}}(s)$ and the *lower value* $\underline{\mathsf{Val}}(s)$ of the average-time game at the state $s \in S$ by

$$\overline{\mathsf{Val}}(s) = \inf_{\mu \in \Sigma_{\text{Min}}} \sup_{\chi \in \Sigma_{\text{Max}}} \mathsf{AT}_{\text{Min}}(s, \mu, \chi), \text{ and } \underline{\mathsf{Val}}(s) = \sup_{\chi \in \Sigma_{\text{Max}}} \inf_{\mu \in \Sigma_{\text{Min}}} \mathsf{AT}_{\text{Max}}(s, \mu, \chi).$$

By Proposition 1.2.4 the inequality $\underline{\mathsf{Val}}(s) \leq \overline{\mathsf{Val}}(s)$ always holds. An average-time game is *determined* if for every $s \in S$, the lower and upper values at s are equal to each other; then we say that the *value* $\mathsf{Val}(s)$ exists and $\mathsf{Val}(s) = \underline{\mathsf{Val}}(s) = \overline{\mathsf{Val}}(s)$.

We give an elementary proof for the determinacy of the average-time games without recourse to general results like Martin's determinacy theorem [**Mar75, Mar98**].

THEOREM 6.1.2 (Determinacy)**.** Average-time games are determined.

For strategies $\mu \in \Sigma_{\text{Min}}$ and $\chi \in \Sigma_{\text{Max}}$, we define

$$\mathsf{Val}^{\mu}(s) = \sup_{\chi \in \Sigma_{\text{Max}}} \mathsf{AT}_{\text{Min}}(s, \mu, \chi), \text{ and } \mathsf{Val}^{\chi}(s) = \inf_{\mu \in \Sigma_{\text{Min}}} \mathsf{AT}_{\text{Max}}(s, \mu, \chi).$$

For an $\varepsilon > 0$, we say that a strategy $\mu \in \Sigma_{\text{Min}}$ or $\chi \in \Sigma_{\text{Max}}$ is *ε-optimal* if for every $s \in S$ we have that $\mathsf{Val}^{\mu}(s) \leq \mathsf{Val}(s) + \varepsilon$ or $\mathsf{Val}^{\chi}(s) \geq \mathsf{Val}(s) - \varepsilon$, respectively. Note that if a game is determined then for every $\varepsilon > 0$, both players have ε-optimal strategies.

We say that a strategy $\chi \in \Sigma_{\text{Max}}$ of player Max is a *best response* to a strategy $\mu \in \Sigma_{\text{Min}}$ of player Min if for all $s \in S$ we have that $\mathsf{AT}_{\text{Min}}(s, \mu, \chi) = \sup_{\chi' \in \Sigma_{\text{Max}}} \mathsf{AT}_{\text{Min}}(s, \mu, \chi')$. Similarly we say that a strategy $\mu \in \Sigma_{\text{Min}}$ of player Min is a best response to a strategy $\chi \in \Sigma_{\text{Max}}$ of player Max if for all $s \in S$ we have that $\mathsf{AT}_{\text{Max}}(s, \mu, \chi) = \inf_{\mu' \in \Sigma_{\text{Min}}} \mathsf{AT}_{\text{Max}}(s, \mu', \chi)$.

In the next section we introduce some region-based abstractions of timed automata, including the closed region graph, and its subgraphs: the boundary region graph, and the region graph. While the region graph is semantically equivalent to the corresponding timed automaton, the boundary region graph has the property that for every starting state, the reachable state space is finite. In Section 6.4 we introduce average-time games on these graphs and show that if we have the solution of the average-time game for any of these

graphs, then we get the solution of the average-time game for the corresponding timed automaton. The key Theorem 6.1.2 follows immediately from Theorem 6.4.2.

6.2. Abstractions of Timed Automata

The corner-point abstraction, introduced by Bouyer et al. **[BBL04]**, is a refinement of a region automaton which preserves some timing information. The corner-point abstraction $\mathcal{T}_{\text{CP}} = (V, E)$ of a timed automaton $\mathcal{T}$ is a finite edge-labelled graph, where: V is a finite set of *vertices* defined as

$$V = \{(s, R) \in Q \times \mathcal{R} \,:\, s = (\ell, \nu) \in \mathsf{clos}(R) \text{ and } \lfloor \nu \rfloor = \nu\},$$

and E is the *labelled edge relation* defined as

$$E = \{((s, R), (t, R'', a), (s', R')) \in V \times \mathbb{R}_{\oplus} \times \mathcal{R} \times A \times V \\ : R \to_* R'' \xrightarrow{a} R' \text{ and } s' = \mathsf{Succ}(s, t, a) \text{ and } s + t \in \mathsf{bd}(R'')\}.$$

From Proposition 3.7.13 we know that *boundary region graphs* are a generalisation of the corner-point abstraction. We prove that the value of the average-time game on a timed automaton is equal to the value of the average-time game on the corresponding boundary region graph for all starting states. In the process, we introduce two other related abstractions, which we call the *closed region graph* and the *region graph*. We collectively refer to these three graphs as *region graphs*.

The analysis of average-time games on region graphs allows us to establish equivalence of average-time games on the original timed automaton and the boundary region graph. We also show (Lemma 6.4.3) that the value of an average-time game is constant over a region. A side-effect of this result is that the corner-point abstractions can be used to solve average-time games on timed automata for arbitrary starting states.

6.2.1. Region Graphs

A *configuration* in region graphs is a is a pair (s, R), where $s \in Q$ is a configuration of the timed automaton and $R \in \mathcal{R}$ is a region; We write Ω for the set of configurations of the region graphs. For a set $X \subseteq \Omega$ and a region $R_0 \in \mathcal{R}$, we define the set X restricted to the region R_0 as the set $\{(s, R) \in X \,:\, R = R_0\}$, and we denote this set by $X(R_0)$. For a configuration $q = (s, R) \in \Omega$ we write write $[q]$ for its region R.

DEFINITION 6.2.1 (Closed Region Graph)**.** The *closed region graph* $\overline{\mathcal{T}} = (\overline{S}, \overline{E})$ of a timed automaton $\mathcal{T}$ is a labelled transition system, where:

- $\overline{S}$ is the set of *states* defined as

$$\overline{S} = \{(s, R) \in \Omega \,:\, s \in \mathsf{clos}(R)\} \text{ and}$$

- $\overline{E}$ is the *labelled transition relation* defined as

$$\overline{E} = \{((s, R), (t, R'', a), (s', R')) \in \overline{S} \times (\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \times \overline{S} \\ : R \to_* R'' \xrightarrow{a} R' \text{ and } s' = \mathsf{Succ}(s, t, a) \text{ and } s + t \in \mathsf{clos}(R'')\}.$$

DEFINITION 6.2.2 (Boundary Region Graph)**.** The *boundary region graph* $\widehat{\mathcal{T}} = (\widehat{S}, \widehat{E})$ of a timed automaton $\mathcal{T}$ is a labelled transition system, where:

– $\widehat{S}$ is the set of *states* defined as

$$\widehat{S} = \{(s, R) \in \Omega \,:\, s \in \mathsf{clos}(R)\} \text{ and}$$

– $\widehat{E}$ is the *labelled transition relation* defined as

$$\widehat{E} = \{((s,R),(t,R'',a),(s',R')) \in \widehat{S} \times (\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \times \widehat{S} \\ : R \to_* R'' \xrightarrow{a} R' \text{ and } s' = \mathsf{Succ}(s,t,a) \text{ and } s + t \in \mathsf{bd}(R'')\}.$$

DEFINITION 6.2.3 (Region Graph)**.** A *region graph* of a timed automaton $\mathcal{T}$ is a labelled transition system $\widetilde{\mathcal{T}} = (\widetilde{S}, \widetilde{E})$, where:

– $\widetilde{S}$ is the set of *states* defined as

$$\widetilde{S} = \{(s, R) \in \Omega \,:\, s \in R\} \text{ and}$$

– $\widetilde{E}$ is the *labelled transition relation* defined as

$$\widetilde{E} = \{((s,R),(t,R'',a),(s',R')) \in \widetilde{S} \times (\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \times \widetilde{S} \\ : R \to_* R'' \xrightarrow{a} R' \text{ and } s' = \mathsf{Succ}(s,t,a) \text{ and } s + t \in R''\}.$$

Notice that there are no significant differences between the definitions of the boundary region graph and the region graph presented above and those from Chapter 3. We reproduce these definition to help the reader compare the three region graphs.

For configuration $q = (s, R) \in \Omega$, real number $t \in \mathbb{R}_{\oplus}$, region $R'' \in \mathcal{R}$, and action $a \in A$, we write $\mathsf{Succ}(q, (t, R'', a))$ for the configuration $\big(\mathsf{Succ}(s, t, a), R'\big)$ where $R'' \xrightarrow{a} R'$. Recall from Definition 3.7.7 that $\mathcal{A} = A \times [\![\mathcal{K}]\!]_{\mathbb{N}} \times C$ is the finite set of boundary timed actions. For configuration $q = (s, R) \in \Omega$, boundary timed action $\alpha = (b, c, a) \in \mathcal{A}$, and region $R'' \in \mathcal{R}$ we write $\mathsf{Succ}(q, (\alpha, R''))$ for the configuration $\mathsf{Succ}(q, (t(s, \alpha), R'', a))$.

6.2.2. Region Game Graphs

For $\Gamma = (\mathcal{T}, L_{\text{Min}}, L_{\text{Max}})$ we define the sets $\Omega_{\text{Min}} = \{(s, R) \in \Omega \,:\, R \in \mathcal{R}_{\text{Min}}\}$ and $\Omega_{\text{Max}} = \Omega \setminus \Omega_{\text{Min}}$. Similarly we define sets $\overline{S}_{\text{Min}}$, $\overline{S}_{\text{Max}}$, $\widehat{S}_{\text{Min}}$, $\widehat{S}_{\text{Max}}$, $\widetilde{S}_{\text{Min}}$, and $\widetilde{S}_{\text{Max}}$. The timed game automaton Γ naturally gives rise to the closed region game graph $\overline{\Gamma} = (\overline{\mathcal{T}}, \overline{S}_{\text{Min}}, \overline{S}_{\text{Max}})$, the boundary region game graph $\widehat{\Gamma} = (\widehat{\mathcal{T}}, \widehat{S}_{\text{Min}}, \widehat{S}_{\text{Max}})$, and the region game graph $\widetilde{\Gamma} = (\widetilde{\mathcal{T}}, \widetilde{S}_{\text{Min}}, \widetilde{S}_{\text{Max}})$. When it is clear from context, we use the terms region graphs and region game graphs interchangeably. Also, sometimes, we write $\mathcal{T}$, $\overline{\mathcal{T}}$, $\widehat{\mathcal{T}}$, and $\widetilde{\mathcal{T}}$ for Γ, $\overline{\Gamma}$, $\widehat{\Gamma}$, and $\widetilde{\Gamma}$, respectively.

6.2.3. Runs of Region Graphs

An infinite run of the closed region graph $\overline{\mathcal{T}} = (\overline{S}, \overline{E})$ is an infinite sequence

$$\langle q_0, \tau_1, q_1, \tau_1, \ldots \rangle \in \overline{S} \times \big((\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \times \overline{S}\big)^{\omega},$$

such that for every positive integer i we have $(q_{i-1}, \tau_i, q_i) \in \overline{E}$. A finite run of the closed region graph $\overline{\mathcal{T}}$ is a finite sequence

$$\langle q_0, \tau_1, q_1, \tau_1, \ldots, q_n \rangle \in \overline{S} \times \big((\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \times \overline{S}\big)^*,$$

such that for every positive integer $i \leq n$ we have $(q_{i-1}, \tau_i, q_i) \in \overline{E}$. Runs of the boundary region graph and the region graph are defined analogously.

For a graph $\mathcal{G} \in \{\overline{\mathcal{T}}, \widehat{\mathcal{T}}, \widetilde{\mathcal{T}}\}$ we write $\mathrm{Runs}^{\mathcal{G}}$ for the set of its runs and $\mathrm{Runs}^{\mathcal{G}}(q)$ for the set of its runs from a state $q \in \overline{Q}$. We write $\mathrm{Runs}^{\mathcal{G}}_{\mathrm{fin}}$ for the set of finite runs and $\mathrm{Runs}^{\mathcal{G}}_{\mathrm{fin}}(q)$ for the set of finite runs starting from a state $q \in \overline{S}$.

6.2.4. Pre-Runs and Run Types Revisited

Pre-runs, first introduce in Subsection 4.3.3, generalise runs of $\overline{\mathcal{T}}, \widetilde{\mathcal{T}}$, and $\widehat{\mathcal{T}}$, and allow us to compare the runs in $\overline{\mathcal{T}}, \widetilde{\mathcal{T}}$, and $\widehat{\mathcal{T}}$ in a uniform manner. On the other hand, the concept of the type of a run allows us to compare pre-runs passing through the same sequence of regions.

A *pre-run* is a sequence $\langle (s_0, R_0), (t_1, R'_1, a_1), (s_1, R_1), \ldots \rangle \in \Omega \times ((\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \times \Omega)^{\omega}$, such that $s_{i+1} = \mathsf{Succ}(s_i, (t_{i+1}, a_{i+1}))$ and $R_i \to_* R'_{i+1} \xrightarrow{a_{i+1}} R_{i+1}$ for every $i \in \mathbb{N}$. We write $\mathsf{PreRuns}$ for the set of pre-runs and $\mathsf{PreRuns}(s, R)$ for the set of pre-runs starting from $(s, R) \in \Omega$. The relation between various sets of runs is as follows: for all $q \in \overline{Q}$ we have

$$\mathrm{Runs}^{\widehat{\mathcal{T}}}(q) \subseteq \mathrm{Runs}^{\overline{\mathcal{T}}}(q) \subseteq \mathsf{PreRuns}(q) \text{ and}$$
$$\mathrm{Runs}^{\widetilde{\mathcal{T}}}(q) \subseteq \mathrm{Runs}^{\overline{\mathcal{T}}}(q) \subseteq \mathsf{PreRuns}(q).$$

Similarly, a finite pre-run is a finite sequence $\langle (s_0, R_0), (t_1, R'_1, a_1), (s_1, R_1), \ldots, (s_n, R_n) \rangle \in (Q \times \mathcal{R}) \times ((\mathbb{R}_{\oplus} \times \mathcal{R} \times A) \times (Q \times \mathcal{R}))^*$ such that for every nonnegative integer $i < n$ we have that $s_{i+1} = \mathsf{Succ}(s_i, (t_{i+1}, a_{i+1}))$ and $R_i \to_* R'_i \xrightarrow{a_{i+1}} R_i$. We write $\mathsf{PreRuns}_{\mathrm{fin}}$ for the set of finite pre-runs and $\mathsf{PreRuns}_{\mathrm{fin}}(s, R)$ for the set of finite pre-runs starting from $(s, R) \in \Omega$. For a finite run $r = \langle q_0, (t_1, R_1, a_1), q_1, (t_2, R_2, a_2), \ldots, q_n \rangle \in \mathsf{PreRuns}_{\mathrm{fin}}$ we define its total time as $\mathsf{Time}(r) = \sum_{i=1}^{n} t_i$, and we denote the last state of the run by $\mathsf{Last}(r) = q_n$.

A *run type* is a sequence $\langle R_0, (R'_1, a_1), R_1, (R'_2, a_2), \ldots \rangle \in \mathcal{R} \times ((\mathcal{R} \times A) \times \mathcal{R})^{\omega}$ such that for every $i \in \mathbb{N}$ we have that $R_i \to_* R'_{i+1} \xrightarrow{a} R_{i+1}$. We say that a pre-run $r = \langle (s_0, R_0), (t_1, R'_1, a_1), (s_1, R_1), (t_1, R'_2, a_2), \ldots \rangle$ is of the type $\langle R_0, (R'_1, a_1), R_1, (R'_2, a_2), \ldots \rangle$. Also, we say that a run $r = \langle s_0, (t_1, a_1), s_1, (t_2, a_2), \ldots \rangle$ of a timed automaton $\mathcal{T}$ is of the type $\langle R_0, (R'_1, a_1), R_1, (R'_2, a_2), \ldots \rangle$, where $R_i = [s_i]$ and $R'_{i+1} = [s_i + t_{i+1}]$ for all $i \in \mathbb{N}$. We also define the type of a finite runs analogously.

For a (finite or infinite) run or pre-run r, we write $[\![r]\!]_{\mathcal{R}}$ for its type. We write Types for the set of run types, and we write $\mathrm{Types}(R)$ for the set of run types starting from region $R \in \mathcal{R}$. Similarly we write $\mathrm{Types}_{\mathrm{fin}}$ for the set of finite run types, and we write $\mathrm{Types}_{\mathrm{fin}}(R)$ for the set of finite run types starting from region $R \in \mathcal{R}$.

6.3. Strategies in Region Graphs

In this section we define strategies of players in region graphs $\overline{\mathcal{T}}$, $\widetilde{\mathcal{T}}$, and $\widehat{\mathcal{T}}$, and study some of their properties. Strategies in $\widetilde{\mathcal{T}}$ are called *admissible strategies*, while strategies in $\widehat{\mathcal{T}}$ are called *boundary strategies*. We also introduce so-called *type-preserving boundary strategies* which are a key tool in proving the correctness of game reduction from timed automata to boundary region graph. In Section 6.4 we show that there are optimal type-preserving boundary strategies in $\overline{\mathcal{T}}$ and $\widehat{\mathcal{T}}$.

6.3.1. Pre-strategies and Strategies in $\overline{\mathcal{T}}, \widehat{\mathcal{T}}, \widetilde{\mathcal{T}}$

Pre-strategies generalise the concept of strategies in region graphs, and allows us to discuss the strategies in $\overline{\mathcal{T}}$, $\widehat{\mathcal{T}}$, and $\widetilde{\mathcal{T}}$ in a uniform manner. We first define pre-strategies for players in $\mathcal{T}$, and then using that we define strategies for players in closed region graph, boundary region graph, and region graph.

DEFINITION 6.3.1 (Pre-strategies)**.** A *pre-strategy* of player Min μ is a (partial) function $\mu : \mathsf{PreRuns}_{\text{fin}} \to \mathbb{R}_{\oplus} \times \mathcal{R} \times A$, such that for a run $r \in \mathsf{PreRuns}_{\text{fin}}$, if $\mathsf{Last}(r) = (s, R) \in \Omega_{\text{Min}}$ then $\mu(r) = (t, R'', a)$ is defined, and it is such that $R \to_* R'' \xrightarrow{a} R'$ for some $R' \in \mathcal{R}$. Pre-strategies of player Max are defined analogously. We write $\Sigma^{\text{pre}}_{\text{Min}}$ and $\Sigma^{\text{pre}}_{\text{Max}}$ for the set of pre-strategies of player Min and player Max, respectively.

We say that a strategy of player Min $\mu \in \Sigma^{\text{pre}}_{\text{Min}}$ is positional if for all runs $r_1, r_2 \in \mathsf{PreRuns}_{\text{fin}}$ we have that $\mathsf{Last}(r_1) = \mathsf{Last}(r_2)$ implies $\mu(r_1) = \mu(r_2)$. Similarly we define positional strategy of player Max.

We define the run starting from configuration $q \in \Omega$ where player Min and player Max use the strategies $\mu \in \Sigma^{\text{pre}}_{\text{Min}}$ and $\chi \in \Sigma^{\text{pre}}_{\text{Max}}$, respectively, in a straightforward manner and we write $\text{Run}(q, \mu, \chi)$ for this run. For every positive integer n we write $\text{Run}_n(q, \mu, \chi)$ for the prefix of the run $\text{Run}(q, \mu, \chi)$ of length n.

Now we are in a position to introduce strategies in closed region graph, region graph, and boundary region graph.

DEFINITION 6.3.2 (Strategies in Closed Region Graph)**.** A pre-strategy of player Min $\mu \in \Sigma^{\text{pre}}_{\text{Min}}$ is a strategy in a closed region graph $\overline{\mathcal{T}} = (\overline{S}, \overline{E})$ if for every run $r \in \mathsf{PreRuns}_{\text{fin}}$ such that $\mu(r) = (t, R', a)$, we have that $(s + t) \in \mathsf{clos}(R')$ where $(s, R) = \mathsf{Last}(r)$. Strategies of player Max in a closed region graph are defined analogously. We write $\overline{\Sigma}_{\text{Min}}$ and $\overline{\Sigma}_{\text{Max}}$ for the set of strategies of player Min and player Max, respectively.

DEFINITION 6.3.3 (Strategies in Region Graphs)**.** A pre-strategy of player Min $\mu \in \Sigma^{\text{pre}}_{\text{Min}}$ is a strategy in a region graph $\widetilde{\mathcal{T}} = (\widetilde{S}, \widetilde{E})$ if for every run $r \in \text{Runs}^{\widetilde{\mathcal{T}}}_{\text{fin}}$ such that $\mu(r) = (t, R'', a)$, we have that $(s + t) \in R''$ where $(s, R) = \mathsf{Last}(r)$. Strategies of player Max in a region graph are defined analogously. We call such strategies *admissible strategies*. We write $\widetilde{\Sigma}_{\text{Min}}$ and $\widetilde{\Sigma}_{\text{Max}}$ for the set of admissible strategies of player Min and player Max, respectively.

DEFINITION 6.3.4 (Strategies in Boundary Region Graph)**.** A pre-strategy of player Min $\mu \in \Sigma^{\text{pre}}_{\text{Min}}$ is a strategy in a boundary region graph $\widehat{\mathcal{T}} = (\widehat{S}, \widehat{E})$ if for every run $r \in \mathsf{PreRuns}_{\text{fin}}$

such that $\mu(r) = (t, R', a)$, we have that

$$t = \inf\{t \,:\, s + t \in \mathsf{clos}(R')\}, \tag{6.3.1}$$

where $(s, R) = \mathsf{Last}(r)$.
A pre-strategy of player Max $\chi \in \Sigma^{\text{pre}}_{\text{Max}}$ is a strategy in a boundary region graph $\widehat{\mathcal{T}}$ if for every run $r \in \mathsf{PreRuns}_{\text{fin}}$ such that $\mu(r) = (t, R', a)$, we have that

$$t = \sup\{t \,:\, s + t \in \mathsf{clos}(R')\}, \tag{6.3.2}$$

where $(s, R) = \mathsf{Last}(r)$. We call such strategies *boundary strategies*. We write $\widehat{\Sigma}_{\text{Min}}$ and $\widehat{\Sigma}_{\text{Max}}$ for the set of boundary strategies of player Min and player Max, respectively.

For notational convenience and w.l.o.g., in the definition of boundary strategies, we do not consider those timed moves of player Min (Max) which suggest waiting till the farther (nearer) boundary of a thick region.

FACT 6.3.5. For every state $s \in S$ of timed automata $\mathcal{T}$ and every strategy $\mu \in \Sigma^{\text{pre}}_{\text{Min}}$ and $\chi \in \Sigma^{\text{pre}}_{\text{Max}}$ of respective players, we have that :

- $\text{Run}((s,[s]), \mu, \chi) \in \text{Runs}^{\overline{\mathcal{T}}}(s,[s])$ if $\mu \in \overline{\Sigma}_{\text{Min}}$ and $\chi \in \overline{\Sigma}_{\text{Max}}$;
- $\text{Run}((s,[s]), \mu, \chi) \in \text{Runs}^{\widehat{\mathcal{T}}}(s,[s])$ if $\mu \in \widehat{\Sigma}_{\text{Min}}$ and $\chi \in \widehat{\Sigma}_{\text{Max}}$;
- $\text{Run}((s,[s]), \mu, \chi) \in \text{Runs}^{\widetilde{\mathcal{T}}}(s,[s])$ if $\mu \in \widetilde{\Sigma}_{\text{Min}}$ and $\chi \in \widetilde{\Sigma}_{\text{Max}}$.

Boundary Strategies and Boundary Timed Actions. Timed actions suggested by a boundary strategies are precisely boundary timed actions as defined in Definition 3.7.7. The following proposition formalises this notion.

PROPOSITION 6.3.6. For every boundary strategy $\sigma \in \widehat{\Sigma}_{\text{Min}}(\widehat{\Sigma}_{\text{Max}})$ of player Min (Max) and for every run $r \in \mathsf{PreRuns}_{\text{fin}}$, if $\sigma(r) = (t, R', a)$ then there exists a boundary timed action $\alpha = (b, c, a) \in \mathcal{A}$ such that $t(s, \alpha) = t$, where $(s, R) = \mathsf{Last}(r)$.

PROOF. Let run $r \in \mathsf{PreRuns}_{\text{fin}}$ be such that $\mathsf{Last}(r) = (s, R)$. Let $\sigma \in \widehat{\Sigma}_{\text{Min}}$ be a boundary strategy of player Min such that $\sigma(r) = (t, R', a)$. From the definition of the boundary strategies, we have that $t = \inf\{t \,:\, s + t \in \mathsf{clos}(R')\}$. To prove the proposition, all we need to show is that there exists an integer $b \in \mathbb{Z}$ and a clock $c \in C$, such that $b - s(c) = t$.

If $R' \in \mathcal{R}_{\text{Thin}}$ then there exists a clock $c' \in C$ such that for all states $s' \in \mathsf{clos}(R')$ we have that $\lbrace s'(c') \rbrace = 0$. In this case the clock $c = c'$ and the integer $b = (s + t)(c)$.

If $R \in \mathcal{R}_{\text{Thick}}$ and let $R' \leftarrow_{+1} R$ be the thin region immediately before R. Let clock $c' \in C$ be such that for all states $s' \in \mathsf{clos}(R')$ we have that $\lbrace s'(c') \rbrace = 0$. Again, in this case the desired clock $c = c'$ and the integer $b = (s + t)(c)$.

The case, where σ is a strategy of Max is similar, and hence omitted. ∎

Sometimes, in our proofs we need to use boundary timed action suggested by a boundary strategy. For this purpose we define the notation $\sigma_{\mathsf{bta}}(r)$ that gives the boundary timed action and region pair that corresponds to $\sigma(r)$. The definition of this function is formalised in the following definition.

DEFINITION 6.3.7. For a boundary strategy $\sigma \in \widehat{\Sigma}_{\mathrm{Max}}(\widehat{\Sigma}_{\mathrm{Max}})$ of player Min (Max), we define the function $\sigma_{\mathsf{bta}} : \mathsf{PreRuns}_{\mathrm{fin}} \to (\mathcal{A} \times \mathcal{R})$ as follows: if for a run $r \in \mathsf{PreRuns}_{\mathrm{fin}}$ we have $\sigma(r) = (t, R', a)$, then $\sigma_{\mathsf{bta}}(r) = ((b, c, a), R')$ such that $b - s(c) = t$, where $(s, R) = \mathsf{Last}(r)$.

6.3.2. Type-Preserving Boundary Strategies

We now introduce an important class of boundary strategies called *type-preserving boundary strategies*. Broadly speaking, these strategies suggest to players a unique boundary timed action and region pair for all the finite runs of the same type.

DEFINITION 6.3.8 (Type-Preserving Boundary Strategies)**.** A boundary strategy $\sigma \in \widehat{\Sigma}_{\mathrm{Min}}$ of player Min is *type-preserving* if $[\![r_1]\!]_{\mathcal{R}} = [\![r_2]\!]_{\mathcal{R}}$ implies $\sigma_{\mathsf{bta}}(r_1) = \sigma_{\mathsf{bta}}(r_2)$ for all $r_1, r_2 \in \mathsf{PreRuns}_{\mathrm{fin}}$. Type-preserving boundary strategies of player Max are defined analogously. We write Ξ_{Min} and Ξ_{Max} for the sets of type-preserving boundary strategies of players Min and Max, respectively.

The rationale behind the name *type-preserving* is that if $\mu \in \Xi_{\mathrm{Min}}$ and $\chi \in \Xi_{\mathrm{Max}}$, then for every $R \in \mathcal{R}$ and for $q, q' \in \Omega(R)$, the run types of the resulting runs from q and q' are the same, i.e., $[\![\mathrm{Run}(q, \mu, \chi)]\!]_{\mathcal{R}} = [\![\mathrm{Run}(q', \mu, \chi)]\!]_{\mathcal{R}}$.

6.3.2.1. Simple Functions Revisited

Let us redefine simple functions in the context of region graphs. Let $X \subseteq \Omega$. A function $F : X \to \mathbb{R}$ is a *simple function* if either: there is $e \in \mathbb{Z}$, such that for every $q = (s, R) \in X$, we have $F(q) = e$; or there are $e \in \mathbb{Z}$ and $c \in C$, such that for every $q = (s, R) \in X$, we have $F(q) = e - s(c)$. We say that a function $F : X \to \mathbb{R}$ is *regionally simple* or *regionally constant*, respectively, if for every region $R \in \mathcal{R}$, the function F, over domain $X(R)$, is simple or constant, respectively.

For regions $R, R', R'' \in \mathcal{R}$ and boundary timed action $\alpha = (b, c, a) \in \mathcal{A}$, we write $R \xrightarrow{R''}_{\alpha} R'$ if one of the following holds:

- $R \to_{b,c} R'' \xrightarrow{a} R'$, or
- there is region $R''' \in \mathcal{R}_{\mathrm{Thin}}$ such that $R \to_{b,c} R''' \to_{+1} R'' \xrightarrow{a} R'$, or
- there is a region $R''' \in \mathcal{R}_{\mathrm{Thin}}$ such that $R \to_{b,c} R''' \leftarrow_{+1} R'' \xrightarrow{a} R'$.

The proof of the following proposition is along the lines of the proof of Proposition 5.3.1.

PROPOSITION 6.3.9. Let $\alpha \in \mathcal{A}$ and regions $R, R', R'' \in \mathcal{R}$ be such that $R \xrightarrow{R''}_{\alpha} R'$. If a function $F : \Omega(R') \to \mathbb{R}$ is simple then the function $F^{\oplus}_{(\alpha, R'')} : \Omega(R) \to \mathbb{R}$, defined as $(s, R) \mapsto t(s, \alpha) + F(\mathsf{Succ}(q, (\alpha, R'')))$, is simple.

The proof of the following proposition is analogous to the proof of Proposition 5.3.2.

PROPOSITION 6.3.10. Let $a \in A$ and regions $R, R', R'' \in \mathcal{R}$ be such that $R \to_* R'' \xrightarrow{a} R'$. If a function $F : \Omega(R') \to \mathbb{R}$ is simple then for every $q = (s, R) \in \Omega(R)$, the function $F^{\oplus}_{(q, R'', a)} : I \to \mathbb{R}$, defined as $t \mapsto t + F(\mathsf{Succ}(q, (t, R'', a)))$, is continuous and nondecreasing, where $I = \{t \in \mathbb{R}_{\oplus} : (s + t) \in \mathsf{clos}(R'')\}$.

6.3.2.2. Properties of Type-preserving Boundary Strategies

The next two proposition state that if both players play with type-preserving boundary strategies then for every $n \in \mathbb{N}$ the total time spent in n transitions is regionally simple (Proposition 6.3.11), and the average time of the infinite run is regionally constant (Proposition 6.3.12).

PROPOSITION 6.3.11 (Type-preserving strategy pairs yield regionally simple time for finite runs)**.** If $\mu \in \Xi_{\text{Min}}$, $\chi \in \Xi_{\text{Max}}$, and $n \in \mathbb{N}$, then $\mathsf{Time}(\text{Run}_n(\cdot, \mu, \chi)) : \overline{Q} \to \mathbb{R}_{\oplus}$ is regionally simple.

PROOF. Let $\mu \in \Xi_{\text{Min}}$ and $\chi \in \Xi_{\text{Max}}$. We prove this lemma by induction on the value of n. The base case for $n = 0$ is trivial. Assume that for every $\mu \in \Xi_{\text{Min}}$ and $\chi \in \Xi_{\text{Max}}$ the function $\mathsf{Time}(\text{Run}_k(\cdot, \mu, \chi)) : \overline{Q} \to \mathbb{R}_{\oplus}$ is regionally simple. To prove this proposition we now need to show that for $\mu \in \Xi_{\text{Min}}$ and $\chi \in \Xi_{\text{Max}}$ the function $\mathsf{Time}(\text{Run}_{k+1}(\cdot, \mu, \chi))$ is regionally simple.

Let the strategies $\mu' \in \Xi_{\text{Min}}$ and $\chi' \in \Xi_{\text{Max}}$ be such that for every $q \in \overline{Q}$ the run $\text{Run}_k(\mathsf{Succ}(q, \mu, \chi), \mu', \chi')$ be the length k suffix of the run $\text{Run}_{k+1}(q, \mu, \chi)$. From inductive hypothesis we have that $\text{Run}_k(\cdot, \mu', \chi')$ is regionally simple. Assume that $R \in \mathcal{R}_{\text{Min}}$ and let $\mu_{\mathsf{bta}}(\langle q \rangle) = (\alpha, R'')$ for every $q \in \overline{Q}(R)$. The treatment for the case where $R \in \mathcal{R}_{\text{Max}}$ is similar. Now for every $q = (s, R) \in \overline{Q}(R)$ we have that $\mathsf{Time}(\text{Run}_{k+1}(q, \mu, \chi)) = t(s, \alpha) + \text{Run}_k(\mathsf{Succ}(q, (\alpha, R'')), \mu', \chi')$, which from Proposition 6.3.9 is a simple function. ∎

PROPOSITION 6.3.12 (Type-preserving strategy pairs yield regionally constant average time)**.** If $\mu \in \Xi_{\text{Min}}$ and $\chi \in \Xi_{\text{Max}}$ then $\mathsf{AT}_{\text{Min}}(\cdot, \mu, \chi) : \overline{Q} \to \mathbb{R}_{\oplus}$ and $\mathsf{AT}_{\text{Max}}(\cdot, \mu, \chi) : \overline{Q} \to \mathbb{R}_{\oplus}$ are regionally constant.

PROOF. Let $\mu \in \Xi_{\text{Min}}$, $\chi \in \Xi_{\text{Max}}$ and $q = (s, R), q' = (s', R) \in \overline{Q}(R)$. We have

$$\begin{aligned}
&\mathsf{AT}_{\text{Min}}(q, \mu, \chi) - \mathsf{AT}_{\text{Min}}(q', \mu, \chi) \\
&\quad = \liminf_{n \to \infty} (1/n) \cdot \mathsf{Time}(\text{Run}_n(q, \mu, \chi)) - \liminf_{n \to \infty} (1/n) \cdot \mathsf{Time}(\text{Run}_n(q', \mu, \chi)) \\
&\qquad = \liminf_{n \to \infty} (1/n) \cdot \big(b - s(c) - b + s'(c)\big) = \liminf_{n \to \infty} (1/n) \cdot \big(s'(c) - s(c)\big) = 0.
\end{aligned}$$

The first equality is by definition, the second follows from Proposition 6.3.11, and the last two equalities are trivial. Similarly we show that $\mathsf{AT}_{\text{Max}}(q, \mu, \chi) = \mathsf{AT}_{\text{Max}}(q', \mu, \chi)$. ∎

6.3.2.3. Type-preserving Boundary Strategy that Agrees with a Boundary Strategy

Given an arbitrary boundary strategy σ and a configuration $q \in \overline{Q}$, sometimes we are interested in a type-preserving boundary strategy that agrees with σ for all the runs starting from q. We denote such a strategy by $\sigma^{\downarrow q}$. The following definition formalises such strategy.

DEFINITION 6.3.13. For a boundary strategy $\mu \in \widehat{\Sigma}_{\text{Min}}$ of player Min and $q \in \overline{Q}$ we define $\mu^{\downarrow q} \in \Xi_{\text{Min}}$ to be a type-preserving boundary strategy which satisfy the following conditions:

(1) $\mu_{\mathsf{bta}}^{\downarrow q}(r) = \mu_{\mathsf{bta}}(r)$ for every $r \in \mathsf{PreRuns}_{\mathrm{fin}}(q)$, and
(2) $[\![r]\!]_{\mathcal{R}} = [\![r']\!]_{\mathcal{R}}$ implies $\mu_{\mathsf{bta}}^{\downarrow q}(r) = \mu_{\mathsf{bta}}^{\downarrow q}(r')$ for all runs $r, r' \in \mathsf{PreRuns}_{\mathrm{fin}}$.

For $\chi \in \widehat{\Sigma}_{\mathrm{Max}}$ and $q \in \overline{Q}$ we define $\chi^{\downarrow q} \in \Xi_{\mathrm{Max}}$ analogously.

Given an arbitrary strategy $\mu \in \overline{\Sigma}_{\mathrm{Min}}$ of player Min, a type-preserving boundary strategy $\chi \in \Xi_{\mathrm{Max}}$ of player Max, and a configuration $q \in \overline{Q}$ sometimes we require to specify a type-preserving strategy $\mu^{(q,\chi)} \in \Xi_{\mathrm{Min}}$ which has the property that types of runs $\mathrm{Run}(q, \mu, \chi)$ and $\mathrm{Run}(q, \mu^{(q,\chi)}, \chi)$ are the same. We then argue that from configuration $q \in \overline{Q}$ if player Max plays according to $\chi \in \Xi_{\mathrm{Max}}$ then player Min can achieve better average-time if she plays according to $\mu^{(q,\chi)}$ (see Proposition 6.3.15 and Corollary 6.3.16). The motivation for the definition of $\chi^{(q,\mu)}$ is similar.

DEFINITION 6.3.14. For an arbitrary strategy $\mu \in \overline{\Sigma}_{\mathrm{Min}}$ of player Min, a type-preserving boundary strategy $\chi \in \Xi_{\mathrm{Max}}$ of player Max, and a configuration $q = (s, R) \in \overline{Q}$, we define $\mu^{(q,\chi)} \in \Xi_{\mathrm{Min}}$ to be a type-preserving boundary strategy which satisfy the following conditions:

(1) $[\![\mathrm{Run}(q, \mu^{(q,\chi)}, \chi)]\!]_{\mathcal{R}} = [\![\mathrm{Run}(q, \mu, \chi)]\!]_{\mathcal{R}}$, and
(2) $[\![r]\!]_{\mathcal{R}} = [\![r']\!]_{\mathcal{R}}$ implies $\mu_{\mathsf{bta}}^{(q,\chi)}(r) = \mu_{\mathsf{bta}}^{(q,\chi)}(r')$ for all runs $r, r' \in \mathsf{PreRuns}_{\mathrm{fin}}$.

For $\chi \in \overline{\Sigma}_{\mathrm{Max}}$, $\mu \in \Xi_{\mathrm{Min}}$, and $q \in \overline{Q}$ the strategy $\chi^{(q,\mu)} \in \Xi_{\mathrm{Max}}$ is defined analogously.

The following proposition and its corollary shows that starting from a configuration q player Min (Max) prefers $\mu^{(q,\chi)}$ ($\chi^{(q,\mu)}$) to μ (χ) against a type-preserving strategy $\chi \in \Xi_{\mathrm{Max}}$ ($\mu \in \Xi_{\mathrm{Min}}$) of its opponent.

PROPOSITION 6.3.15. For every $\chi \in \Xi_{\mathrm{Max}}$, $\mu \in \overline{\Sigma}_{\mathrm{Min}}$ and $q \in \overline{Q}$ we have that

$$\mathsf{Time}(\mathrm{Run}_n(q, \mu, \chi)) \geq \mathsf{Time}(\mathrm{Run}_n(q, \mu^{(q,\chi)}, \chi)),$$

for every $n \in \mathbb{N}$. Similarly, for every $\mu \in \Xi_{\mathrm{Min}}$, $\chi \in \overline{\Sigma}_{\mathrm{Max}}$ and $q \in \overline{Q}$ we have that

$$\mathsf{Time}(\mathrm{Run}_n(q, \mu, \chi)) \leq \mathsf{Time}(\mathrm{Run}_n(q, \mu, \chi^{(q,\mu)})),$$

for every $n \in \mathbb{N}$.

PROOF. The proof is by induction on n. The base case, when $n = 0$, is trivial. In the rest of the proof we show that for $\chi \in \Xi_{\mathrm{Max}}$, $\mu \in \overline{\Sigma}_{\mathrm{Min}}$, and a configuration $q = (s, R) \in \overline{Q}$, we have that $\mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu, \chi)) \geq \mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu^{(q,\chi)}, \chi))$ assuming that the proposition holds for $n = k$. The proof for the case where $q \in \overline{Q}_{\mathrm{Max}}$ is trivial. In the rest of the proof we assume that $q \in \overline{Q}_{\mathrm{Min}}$.

Let us fix $\chi \in \Xi_{\mathrm{Max}}$ and $\mu \in \overline{\Sigma}_{\mathrm{Min}}$. Let the runs $\mathrm{Run}_{k+1}(q, \mu, \chi)$ and $\mathrm{Run}_{k+1}(q, \mu^{(q,\chi)}, \chi)$ be $\langle q_0, \tau_1, q_1, \ldots, q_{k+1} \rangle$ and $\langle q'_0, \tau'_1, q'_1, \ldots, q'_{k+1} \rangle$, respectively, where $q_0 = q'_0 = q$. Notice that by definition the run types of both runs are the same. Hence for every index $i \leq k+1$ we have $q_i = (s_i, R_i)$ and $q'_i = (s'_i, R_i)$, and for every index $i \leq k+1$ we have $\tau_i = (t_i, R'_i, a_i)$ and $\tau'_i = (t'_i, R'_i, a_i)$.

Let $X \in \Xi_{\mathrm{Max}}$ and $M \in \overline{\Sigma}_{\mathrm{Min}}$ be such that the run $\mathrm{Run}_k(q_1, M, X)$ be length k suffix of the run $\mathrm{Run}_{k+1}(q, \mu, \chi)$. Notice that we assume that X is type-preserving. It is easy to see

that

$$\mathsf{Time}(\mathrm{Run}_{k+1}(q,\mu,\chi)) = t_1 + \mathsf{Time}(\mathrm{Run}_k(q_1, M, X)).$$

From inductive hypothesis, we get that

$$\mathsf{Time}(\mathrm{Run}_{k+1}(q,\mu,\chi)) \geq t_1 + \mathsf{Time}(\mathrm{Run}_k(q_1, M^{(q_1,X)}, X)). \tag{6.3.3}$$

Since the strategies $M^{(q_1,X)} \in \Xi_{\mathrm{Min}}$ and $X \in \Xi_{\mathrm{Max}}$ are type-preserving, from Proposition 6.3.11 we get that $\mathsf{Time}(\mathrm{Run}_k(\cdot, M^{(q_1,X)}, X))$ is regionally simple. Let us denote the restriction of this function on domain $\overline{Q}(R_1)$ by $\mathcal{F} : \overline{Q}(R_1) \to \mathbb{R}$. Let us define the partial function $\mathcal{F}^{\oplus}_{(q,R_1',a)} : \mathbb{R}_{\oplus} \rightharpoonup \mathbb{R}$ as $t \mapsto t + \mathcal{F}(\mathsf{Succ}(q,(t,R'',a)))$, for all $t \in \mathbb{R}_{\oplus}$, such that $(s+t) \in \mathsf{clos}(R_1')$. The following inequality follows from (6.3.3):

$$\mathsf{Time}(\mathrm{Run}_{k+1}(q,\mu,\chi)) \geq t_1 + \mathcal{F}(q_1) \geq \inf_t \left\{ \mathcal{F}^{\oplus}_{(q,R_1',a)}(t) \ : \ s+t \in \mathsf{clos}(R_1') \right\}.$$

Since $\mu^{(q,\chi)}$ is a type-preserving boundary strategy of player Min, from equation (6.3.1), we know that $t_1' = \inf\{t \ : \ s+t \in \mathsf{clos}(R_1')\}$. Moreover from Proposition 6.3.10 we have that $\mathcal{F}^{\oplus}_{(q,R_1',a)}$ is continuous and nondecreasing on the domain $\{t \in \mathbb{R}_{\oplus} \ : \ (s+t) \in \mathsf{clos}(R'')\}$. Hence $\mathcal{F}^{\oplus}_{(q,R_1',a)}(t_1') = \inf_t \left\{ \mathcal{F}^{\oplus}_{(q,R_1',a)}(t) \ : \ s+t \in \mathsf{clos}(R_1') \right\}$. Combining these facts, we get the following inequalities:

$$\mathsf{Time}(\mathrm{Run}_{k+1}(q,\mu,\chi)) \geq \mathcal{F}^{\oplus}_{(q,R_1',a)}(t_1') = t_1' + \mathsf{Time}(\mathrm{Run}_k(q_1', M^{(q_1,X)}, X))$$

Since the run $\mathrm{Run}_k(q_1', M^{(q_1,X)}, X)$ is length k suffix of the run $\mathrm{Run}_{k+1}(q, \mu^{(q,\chi)}, \chi)$, we get the desired inequality. ∎

An easy corollary of this proposition is as follows:

COROLLARY 6.3.16. For every $\chi \in \Xi_{\mathrm{Max}}$, $\mu \in \overline{\Sigma}_{\mathrm{Min}}$ and for all configurations $q \in \overline{Q}$ we have that

$$\mathsf{AT}_{\mathrm{Min}}(q,\mu,\chi)) \geq \mathsf{AT}_{\mathrm{Min}}(q,\mu^{(q,\chi)},\chi)).$$

Similarly for every $\mu \in \Xi_{\mathrm{Min}}$, $\chi \in \overline{\Sigma}_{\mathrm{Max}}$ and for all configurations $q \in \overline{Q}$ we have that

$$\mathsf{AT}_{\mathrm{Max}}(q,\mu,\chi)) \leq \mathsf{AT}_{\mathrm{Max}}(q,\mu,\chi^{(q,\mu)})).$$

6.3.2.4. Admissible Strategies ε-Close to a Type-Preserving Boundary Strategy

DEFINITION 6.3.17 (Set of Admissible Strategies ε-close to a Type-Preserving Boundary Strategy)**.** For $\mu \in \Xi_{\mathrm{Min}}$ and a real number $\varepsilon > 0$, we define the set of admissible strategy $\widetilde{\Sigma}^{(\mu,\varepsilon)}_{\mathrm{Min}} \subseteq \widetilde{\Sigma}_{\mathrm{Min}}$ as follows. For every $\mu_\varepsilon \in \widetilde{\Sigma}^{(\mu,\varepsilon)}_{\mathrm{Min}}$ we have that for all runs $r \in \mathsf{PreRuns}_{\mathrm{fin}}$ if $\mu_{\mathsf{bta}}(r) = ((b,c,a),R')$ then $\mu_\varepsilon(r) = (t,R',a)$ is such that

$$s + t \in R' \text{ and } t \leq b - s(c) + \varepsilon,$$

where $(s,R) = \mathsf{Last}(r)$. Notice that (see Equation 6.3.1) such a value of t always exists. Similarly for $\chi \in \Xi_{\mathrm{Max}}$ and a real number $\varepsilon > 0$ we define the set $\widetilde{\Sigma}^{(\chi,\varepsilon)}_{\mathrm{Max}} \subseteq \widetilde{\Sigma}_{\mathrm{Max}}$ as follows.

For every $\chi_\varepsilon \in \widetilde{\Sigma}^{(\chi,\varepsilon)}_{\text{Max}}$ we have that for all runs $r \in \mathsf{PreRuns}_{\text{fin}}$ if $\chi_{\mathsf{bta}}(r) = ((b,c,a),R')$ then $\chi_\varepsilon(r) = (t,R',a)$ is such that

$$s + t \in R' \text{ and } t \geq b - s(c) - \varepsilon,$$

where $(s,R) = \mathsf{Last}(r)$.

Given an arbitrary strategy $\mu \in \overline{\Sigma}_{\text{Min}}$ of player Min, a positive real $\varepsilon > 0$, a type-preserving boundary strategy $\chi \in \Xi_{\text{Max}}$ of player Max, an ε-close strategy $\chi_\varepsilon \in \widetilde{\Sigma}^{(\chi,\varepsilon)}_{\text{Max}}$, and a configuration $q \in \overline{Q}$ sometimes we require to specify a type-preserving strategy $\mu^{(q,\chi_\varepsilon)} \in \Xi_{\text{Min}}$ which has the property that types of runs $\text{Run}(q,\mu,\chi_\varepsilon)$ and $\text{Run}(q,\mu^{(q,\chi_\varepsilon)},\chi_\varepsilon)$ are the same.

DEFINITION 6.3.18. For an arbitrary strategy $\mu \in \overline{\Sigma}_{\text{Min}}$ of player Min, a positive real $\varepsilon > 0$, a type-preserving boundary strategy $\chi \in \Xi_{\text{Max}}$ of player Max, an ε-close strategy $\chi_\varepsilon \in \widetilde{\Sigma}^{(\chi,\varepsilon)}_{\text{Max}}$, and a configuration $q = (s,R) \in \overline{Q}$, we define $\mu^{(q,\chi_\varepsilon)} \in \Xi_{\text{Min}}$ to be a type-preserving boundary strategy which satisfy the following conditions:

(1) $[\![\text{Run}(q,\mu^{(q,\chi_\varepsilon)},\chi_\varepsilon)]\!]_{\mathcal{R}} = [\![\text{Run}(q,\mu,\chi_\varepsilon)]\!]_{\mathcal{R}}$, and
(2) $[\![r]\!]_{\mathcal{R}} = [\![r']\!]_{\mathcal{R}}$ implies $\mu^{(q,\chi_\varepsilon)}_{\mathsf{bta}}(r) = \mu^{(q,\chi_\varepsilon)}_{\mathsf{bta}}(r')$ for all runs $r, r' \in \mathsf{PreRuns}_{\text{fin}}$.

Combining it with Definition 6.3.17 we get that $[\![\text{Run}(q,\mu^{(q,\chi_\varepsilon)},\chi)]\!]_{\mathcal{R}} = [\![\text{Run}(q,\mu,\chi_\varepsilon)]\!]_{\mathcal{R}}$.
For $\chi \in \overline{\Sigma}_{\text{Max}}$, $\chi_\varepsilon \in \widetilde{\Sigma}^{(\chi,\varepsilon)}_{\text{Max}}$, $\mu \in \Xi_{\text{Min}}$, and $q \in \overline{Q}$ the strategy $\chi^{(q,\mu_\varepsilon)} \in \Xi_{\text{Max}}$ is defined analogously.

We need the following property of $\mu^{(q,\chi_\varepsilon)}$ and $\chi^{(q,\mu_\varepsilon)}$ strategies.

PROPOSITION 6.3.19. For every arbitrary strategy $\mu \in \overline{\Sigma}_{\text{Min}}$, positive real $\varepsilon > 0$, type-preserving boundary strategy $\chi \in \Xi_{\text{Max}}$ of player Max, ε-close strategy $\chi_\varepsilon \in \widetilde{\Sigma}^{(\chi,\varepsilon)}_{\text{Max}}$ of player Max, and $q \in \overline{Q}$ we have

$$\mathsf{Time}(\text{Run}_n(q,\mu,\chi_\varepsilon)) \geq \mathsf{Time}(\text{Run}_n(q,\mu^{(q,\chi_\varepsilon)},\chi)) - n \cdot \varepsilon,$$

for every $n \in \mathbb{N}$. Similarly for every arbitrary strategy $\chi \in \overline{\Sigma}_{\text{Max}}$, positive real $\varepsilon > 0$, type-preserving boundary strategy $\mu \in \Xi_{\text{Min}}$ of player Max, ε-close strategy $\mu_\varepsilon \in \widetilde{\Sigma}^{(\mu,\varepsilon)}_{\text{Min}}$ of player Min, and $q \in \overline{Q}$ we have

$$\mathsf{Time}(\text{Run}_n(q,\mu_\varepsilon,\chi)) \leq \mathsf{Time}(\text{Run}_n(q,\mu_\varepsilon,\chi^{(q,\mu_\varepsilon)})) + n \cdot \varepsilon,$$

for every $n \in \mathbb{N}$.

PROOF. The proof is by induction on n. The base case, when $n = 0$, is trivial. In the rest of the proof we show that for $\chi \in \Xi_{\text{Max}}$, $\mu \in \overline{\Sigma}_{\text{Min}}$, $\varepsilon > 0$, $\chi_\varepsilon \in \widetilde{\Sigma}^{(\chi,\varepsilon)}_{\text{Max}}$, and a configuration $q = (s,R) \in \overline{Q}$, we have that $\mathsf{Time}(\text{Run}_{k+1}(q,\mu,\chi_\varepsilon)) \geq \mathsf{Time}(\text{Run}_{k+1}(q,\mu^{(q,\chi_\varepsilon)},\chi)) - k \cdot \varepsilon$, assuming that the proposition holds for $n = k$.

Let us fix $\chi \in \Xi_{\text{Max}}$, $\mu \in \overline{\Sigma}_{\text{Min}}$, $\varepsilon > 0$, and $\chi_\varepsilon \in \widetilde{\Sigma}^{(\chi,\varepsilon)}_{\text{Max}}$. Let the run $\text{Run}_{k+1}(q,\mu,\chi_\varepsilon)$ and the run $\text{Run}_{k+1}(q,\mu^{(q,\chi_\varepsilon)},\chi)$ be $\langle q_0,\tau_1,q_1,\ldots,q_{k+1}\rangle$ and $\langle q'_0,\tau'_1,q'_1,\ldots,q'_{k+1}\rangle$, respectively,

where $q_0 = q'_0 = q$. Notice that by definition the run types of both runs are the same. Hence for every index $i \leq k+1$ we have $q_i = (s_i, R_i)$ and $q'_i = (s'_i, R_i)$, and for every index $i \leq k+1$ we have $\tau_i = (t_i, R'_i, a_i)$ and $\tau'_i = (t'_i, R'_i, a_i)$.

Let $X \in \Xi_{\mathrm{Max}}$ and $M \in \overline{\Sigma}_{\mathrm{Min}}$ be such that the run $\mathrm{Run}_k(q_1, M, X_\varepsilon)$ be length k suffix of the run $\mathrm{Run}_{k+1}(q, \mu, \chi_\varepsilon)$. Notice that we assume that X is type-preserving. It is easy to see that

$$\mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu, \chi_\varepsilon)) = t_1 + \mathsf{Time}(\mathrm{Run}_k(q_1, M, X_\varepsilon)).$$

From inductive hypothesis, we get that

$$\mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu, \chi_\varepsilon)) \geq t_1 + \mathsf{Time}(\mathrm{Run}_k(q_1, M^{(q_1, X_\varepsilon)}, X)) - k \cdot \varepsilon. \tag{6.3.4}$$

Since the strategies $M^{(q_1, X_\varepsilon)} \in \Xi_{\mathrm{Min}}$ and $X \in \Xi_{\mathrm{Max}}$ are type-preserving boundary strategies, from Proposition 6.3.11 we get that $\mathsf{Time}(\mathrm{Run}_k(\cdot, M^{(q_1, X_\varepsilon)}, X))$ is regionally simple. Let us denote the restriction of this function on domain $\overline{Q}(R_1)$ by $\mathcal{F} : \overline{Q}(R_1) \to \mathbb{R}$. Let us define the partial function $\mathcal{F}^{\oplus}_{(q,R'_1,a)} : \mathbb{R}_\oplus \rightharpoonup \mathbb{R}$ as $t \mapsto t + \mathcal{F}(\mathsf{Succ}(q, (t, R'', a)))$, for all $t \in \mathbb{R}_\oplus$, such that $(s+t) \in \mathsf{clos}(R'_1)$. The following inequality follows from (6.3.4):

$$\mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu, \chi_\varepsilon)) \geq t_1 + \mathcal{F}(q_1) - k \cdot \varepsilon \geq \inf_t \left\{ \mathcal{F}^{\oplus}_{(q,R'_1,a)}(t) \ : \ s + t \in \mathsf{clos}(R'_1) \right\} - k \cdot \varepsilon.$$

We need to consider two cases: $q \in \overline{Q}_{\mathrm{Min}}$ and $q \in \overline{Q}_{\mathrm{Max}}$.

- Assume that $q \in \overline{Q}_{\mathrm{Min}}$. Since $\mu^{(q,\chi_\varepsilon)}$ is a type-preserving boundary strategy of player Min, from equation (6.3.1), we know that $t'_1 = \inf\{t \ : \ s + t \in \mathsf{clos}(R'_1)\}$. Moreover from Proposition 6.3.10 we have that $\mathcal{F}^{\oplus}_{(q,R'_1,a)}$ is continuous and non-decreasing on the domain $\{t \in \mathbb{R}_\oplus \ : \ (s+t) \in \mathsf{clos}(R'')\}$. Hence $\mathcal{F}^{\oplus}_{(q,R'_1,a)}(t'_1) = \inf_t \left\{ \mathcal{F}^{\oplus}_{(q,R'_1,a)}(t) \ : \ s + t \in \mathsf{clos}(R'_1) \right\}$. Combining these facts, we get the following inequalities:

  $$\mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu, \chi_\varepsilon)) \geq t'_1 + \mathsf{Time}(\mathrm{Run}_k(q'_1, M^{(q_1, X_\varepsilon)}, X)) - k \cdot \varepsilon.$$

  Since the run $\mathrm{Run}_k(q'_1, M^{(q_1, X_\varepsilon)}, X)$ is length k suffix of the run $\mathrm{Run}_{k+1}(q, \mu^{(q,\chi_\varepsilon)}, \chi)$, we get the following inequality:

  $$\begin{aligned} \mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu, \chi_\varepsilon)) &\geq \mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu^{(q_1,\chi_\varepsilon)}, \chi)) - k \cdot \varepsilon \\ &\geq \mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu^{(q_1,\chi_\varepsilon)}, \chi)) - (k+1) \cdot \varepsilon, \end{aligned}$$

  as required.
- Assume that $q \in \overline{Q}_{\mathrm{Max}}$. So far we have shown that

  $$\mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu, \chi_\varepsilon)) \geq t_1 + \mathcal{F}(q_1) - k \cdot \varepsilon. \tag{6.3.5}$$

  Since $\mathcal{F}$ is a simple function let $\mathcal{F}((s_1, R_1)) = b - s_1(c)$ for all $(s_1, R_1) \in \overline{Q}(R_1)$. For all $t \in \mathbb{R}_\oplus$ such that $s + t \in R'_1$ we have the following observation.

  $$t + \mathcal{F}((\mathsf{Succ}(s, (t, a_1))) = \begin{cases} t + b & \text{if } c \in \xi(a_1) \\ b - s(c) & \text{otherwise.} \end{cases} \tag{6.3.6}$$

By Definition 6.3.17 we know that $t_1 \geq t_1' - \varepsilon$. Combining this with (6.3.6) we get that

$$t_1 + \mathcal{F}(q_1) \geq t_1' + \mathcal{F}(q_1') - \varepsilon.$$

We can then rewrite (6.3.5) as the following:

$$\mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu, \chi_\varepsilon)) \geq t_1' + \mathcal{F}(q_1') - (k+1) \cdot \varepsilon.$$

The term $\mathcal{F}(q_1')$ represents the sum of the times of the run $\mathrm{Run}_k(q_1', M^{(q_1, X_\varepsilon)}, X)$. Since the run $\mathrm{Run}_k(q_1', M^{(q_1, X_\varepsilon)}, X)$ is length k suffix of the run $\mathrm{Run}_{k+1}(q, \mu^{(q, \chi_\varepsilon)}, \chi)$, we get the inequality

$$\mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu, \chi_\varepsilon)) \geq \mathsf{Time}(\mathrm{Run}_{k+1}(q, \mu^{(q, \chi_\varepsilon)}, \chi)) - (k+1) \cdot \varepsilon,$$

as required.

∎

The following result is an easy corollary of Proposition 6.3.19.

COROLLARY 6.3.20. For every $\chi \in \Xi_{\mathrm{Max}}$, $\mu \in \overline{\Sigma}_{\mathrm{Min}}$, $\varepsilon > 0$, $\chi_\varepsilon \in \widetilde{\Sigma}_{\mathrm{Max}}^{(\chi, \varepsilon)}$, and $q \in \overline{Q}$ we have that

$$\mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi_\varepsilon)) \geq \mathsf{AT}_{\mathrm{Max}}(q, \mu^{(q, \chi_\varepsilon)}, \chi)) - \varepsilon.$$

Similarly for every $\mu \in \Xi_{\mathrm{Min}}$, $\chi \in \overline{\Sigma}_{\mathrm{Max}}$, $\varepsilon > 0$, $\mu_\varepsilon \in \widetilde{\Sigma}_{\mathrm{Min}}^{(\mu, \varepsilon)}$, and $q \in \overline{Q}$ we have that

$$\mathsf{AT}_{\mathrm{Min}}(q, \mu_\varepsilon, \chi) \leq \mathsf{AT}_{\mathrm{Min}}(q, \mu, \chi^{(q, \mu_\varepsilon)}) + \varepsilon.$$

To summarise the relations between various strategies, note that the following inclusions hold:

$$\begin{array}{lll} \Xi_{\mathrm{Min}} \subseteq \widehat{\Sigma}_{\mathrm{Min}} \subseteq \overline{\Sigma}_{\mathrm{Min}} \subseteq \Sigma_{\mathrm{Min}}^{\mathrm{pre}} & \text{and} & \widetilde{\Sigma}_{\mathrm{Min}} \subseteq \overline{\Sigma}_{\mathrm{Min}} \subseteq \Sigma_{\mathrm{Min}}^{\mathrm{pre}}, \quad \text{and} \\ \Xi_{\mathrm{Max}} \subseteq \widehat{\Sigma}_{\mathrm{Max}} \subseteq \overline{\Sigma}_{\mathrm{Max}} \subseteq \Sigma_{\mathrm{Max}}^{\mathrm{pre}} & \text{and} & \widetilde{\Sigma}_{\mathrm{Max}} \subseteq \overline{\Sigma}_{\mathrm{Max}} \subseteq \Sigma_{\mathrm{Max}}^{\mathrm{pre}}. \end{array}$$

6.4. Average-Time Games on Region Graphs

We define the functions $\mathsf{AT}_{\mathrm{Min}} : \Omega \times \Sigma_{\mathrm{Min}}^{\mathrm{pre}} \times \Sigma_{\mathrm{Max}}^{\mathrm{pre}} \to \mathbb{R}_\oplus$ and $\mathsf{AT}_{\mathrm{Max}} : \Omega \times \Sigma_{\mathrm{Min}}^{\mathrm{pre}} \times \Sigma_{\mathrm{Max}}^{\mathrm{pre}} \to \mathbb{R}_\oplus$ in the following manner:

$$\begin{aligned} \mathsf{AT}_{\mathrm{Min}}(q, \mu, \chi) &= \limsup_{n \to \infty} \frac{1}{n} \cdot \mathsf{Time}(\mathrm{Run}_n(q, \mu, \chi)), \text{ and} \\ \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi) &= \liminf_{n \to \infty} \frac{1}{n} \cdot \mathsf{Time}(\mathrm{Run}_n(q, \mu, \chi)), \end{aligned}$$

where $\mu \in \Sigma_{\mathrm{Min}}^{\mathrm{pre}}$, $\chi \in \Sigma_{\mathrm{Max}}^{\mathrm{pre}}$ and $q \in \Omega$. For average-time games on a graph $\mathcal{G} \in \{\overline{\mathcal{T}}, \widehat{\mathcal{T}}, \widetilde{\mathcal{T}}\}$ we define the lower-value $\underline{\mathsf{Val}}^{\mathcal{G}}(q)$, the upper-value $\overline{\mathsf{Val}}^{\mathcal{G}}(q)$ and the value $\mathsf{Val}^{\mathcal{G}}(q)$ of a configuration $q \in \overline{Q}$ in a straightforward manner (for details, see Subsection 6.1.2).

From Proposition 3.7.5 it is clear that the difference between an average-time game on a timed automaton and the average-time game on corresponding region graph is purely syntactical. Hence if the average-time game on region graph $\widetilde{\mathcal{T}}$ is determined then average-time game on timed automaton $\mathcal{T}$ is determined as well.

PROPOSITION 6.4.1. An average-time game on timed automaton $\mathcal{T}$ is determined, if the corresponding average-time game on region graph $\widetilde{\mathcal{T}}$ is determined. Moreover for all $s \in S$ we have that $\mathsf{Val}(s) = \mathsf{Val}^{\widetilde{\mathcal{T}}}(s, [s])$.

The following is the main result of this section.

THEOREM 6.4.2. Let $\mathcal{T}$ be a timed automaton. Average-time games on the timed automaton $\mathcal{T}$, the closed region graph $\overline{\mathcal{T}}$, the region graph $\widetilde{\mathcal{T}}$, and the boundary region graph $\widehat{\mathcal{T}}$ are determined. Moreover for every $s \in S$ in a timed automaton $\mathcal{T}$, we have:

$$\mathsf{Val}^{\mathcal{T}}(s) = \mathsf{Val}^{\widetilde{\mathcal{T}}}(s, [s]) = \mathsf{Val}^{\overline{\mathcal{T}}}(s, [s]) = \mathsf{Val}^{\widehat{\mathcal{T}}}(s, [s]).$$

This theorem follows from Theorem 6.4.4, Theorem 6.4.8, Theorem 6.4.10, and Proposition 6.4.1.

Moreover Theorem 6.4.2 and Proposition 6.3.12 let us conclude the following lemma about the value of average-time games on timed automata.

LEMMA 6.4.3. The value of every average-time game is regionally constant.

An interesting implication of Lemma 6.4.3 is that corner-point abstraction is sufficient to solve average-time games with an arbitrary initial state.

6.4.1. Determinacy of Average-Time Games on the Boundary Region Graph

Positional determinacy of average-time games on the boundary region graph is immediate from Proposition 3.7.10 and Theorem 2.3.3.

THEOREM 6.4.4. The average-time game on $\widehat{\mathcal{T}}$ is determined, and there are optimal positional strategies in $\widehat{\mathcal{T}}$, i.e., for every $q \in \overline{Q}$, we have:

$$\mathsf{Val}^{\widehat{\mathcal{T}}}(q) = \inf_{\mu \in \widehat{\Pi}_{\mathrm{Min}}} \sup_{\chi \in \widehat{\Sigma}_{\mathrm{Max}}} \mathsf{AT}_{\mathrm{Min}}(q, \mu, \chi) = \sup_{\chi \in \widehat{\Pi}_{\mathrm{Max}}} \inf_{\mu \in \widehat{\Sigma}_{\mathrm{Min}}} \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi).$$

LEMMA 6.4.5. In $\widehat{\mathcal{T}}$, if $\mu \in \widehat{\Sigma}_{\mathrm{Min}}$ and $\chi \in \widehat{\Sigma}_{\mathrm{Max}}$ are mutual best responses from $q \in \overline{Q}$, then $\mu^{\downarrow q} \in \Xi_{\mathrm{Min}}$ and $\chi^{\downarrow q} \in \Xi_{\mathrm{Max}}$ are mutual best responses from every $q' \in \overline{Q}([q])$.

PROOF. We argue that $\chi^{\downarrow q}$ is a best response to $\mu^{\downarrow q}$ from $q' \in \overline{Q}([q])$ in $\widehat{\mathcal{T}}$; the other case is analogous. For all $X \in \widehat{\Sigma}_{\mathrm{Max}}$, we have the following:

$$\mathsf{AT}_{\mathrm{Min}}(q', \mu^{\downarrow q}, \chi^{\downarrow q}) = \mathsf{AT}_{\mathrm{Min}}(q, \mu^{\downarrow q}, \chi^{\downarrow q}) \geq \mathsf{AT}_{\mathrm{Min}}(q, \mu^{\downarrow q}, X^{\downarrow q'}) = \mathsf{AT}_{\mathrm{Min}}(q', \mu^{\downarrow q}, X^{\downarrow q'}) = \mathsf{AT}_{\mathrm{Min}}(q', \mu^{\downarrow q}, X).$$

The first equality follows from Proposition 6.3.12; the inequality follows because χ is a best response to μ from q; the second equality follows from Proposition 6.3.12 again; and the last equality is straightforward. ∎

THEOREM 6.4.6. There are optimal type-preserving boundary strategies in $\widehat{\mathcal{T}}$, i.e., for every $q \in \overline{Q}$, we have:

$$\mathsf{Val}^{\widehat{\mathcal{T}}}(q) = \inf_{\mu \in \Xi_{\mathrm{Min}}} \sup_{\chi \in \widehat{\Sigma}_{\mathrm{Max}}} \mathsf{AT}_{\mathrm{Min}}(q, \mu, \chi) = \sup_{\chi \in \Xi_{\mathrm{Max}}} \inf_{\mu \in \widehat{\Sigma}_{\mathrm{Min}}} \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi).$$

PROOF. Let $\mu^* \in \Xi_{\mathrm{Min}}$ and $\chi^* \in \Xi_{\mathrm{Max}}$ be mutual best responses in $\widehat{\mathcal{T}}$; existence of such strategies follows from Lemma 6.4.5. Moreover, we can assume that the strategies μ^* and χ^* have finite memory; this can be achieved by taking positional strategies $\mu \in \widehat{\Sigma}_{\mathrm{Min}}$ and $\chi \in \widehat{\Sigma}_{\mathrm{Max}}$ in Lemma 6.4.5. We then have the following:

$$\inf_{\mu \in \Xi_{\mathrm{Min}}} \sup_{\chi \in \widehat{\Sigma}_{\mathrm{Max}}} \mathsf{AT}_{\mathrm{Min}}(q, \mu, \chi) \leq \sup_{\chi \in \widehat{\Sigma}_{\mathrm{Max}}} \mathsf{AT}_{\mathrm{Min}}(q, \mu^*, \chi) = \mathsf{AT}_{\mathrm{Min}}(q, \mu^*, \chi^*) =$$
$$\mathsf{AT}_{\mathrm{Max}}(q, \mu^*, \chi^*) = \inf_{\mu \in \widehat{\Sigma}_{\mathrm{Min}}} \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi^*) \leq \sup_{\chi \in \Xi_{\mathrm{Max}}} \inf_{\mu \in \widehat{\Sigma}_{\mathrm{Min}}} \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi).$$

The first and last inequalities are straightforward because $\mu^* \in \Xi_{\mathrm{Min}}$ and $\chi^* \in \Xi_{\mathrm{Max}}$. The first equality holds because χ^* is a best response to μ^* in $\widehat{\mathcal{T}}$, and the third equality holds because μ^* is a best response to χ^* in $\widehat{\mathcal{T}}$. Finally, the second equality holds because strategies μ^* and χ^* have finite memory. ∎

6.4.2. Determinacy of Average-Time Games on the Closed Region Graph

LEMMA 6.4.7. In $\overline{\mathcal{T}}$, for every strategy in Ξ_{Min} there is a best response in Ξ_{Max}, and for every strategy in Ξ_{Max} there is a best response in Ξ_{Min}.

PROOF. We argue that if $\mu \in \overline{\Sigma}_{\mathrm{Min}}$ is best-response to $\chi \in \Xi_{\mathrm{Max}}$ from $q \in \overline{Q}$ then the strategy $\mu^{(q,\chi)}$ is best-response to χ from every $q' \in \overline{Q}([q])$. For all $M \in \overline{\Sigma}_{\mathrm{Min}}$ we have the following:

$$\mathsf{AT}_{\mathrm{Min}}(q', \mu^{(q,\chi)}, \chi) = \mathsf{AT}_{\mathrm{Min}}(q, \mu^{(q,\chi)}, \chi) \leq \mathsf{AT}_{\mathrm{Min}}(q, \mu, \chi) \leq \mathsf{AT}_{\mathrm{Min}}(q, M^{(q',\chi)}, \chi) =$$
$$\mathsf{AT}_{\mathrm{Min}}(q', M^{(q',\chi)}, \chi) \leq \mathsf{AT}_{\mathrm{Min}}(q', \mu', \chi).$$

The first and the second equalities follow from Proposition 6.3.12; the second inequality follows because μ is a best response to χ from q; and the first and the third inequalities follow from the the Corollary 6.3.16. It follows that in $\overline{\mathcal{T}}$ for every strategy $\chi \in \Xi_{\mathrm{Max}}$ there is a best response in Ξ_{Min}. Similarly we prove that in $\overline{\mathcal{T}}$ for every strategy $\mu \in \Xi_{\mathrm{Min}}$ there is a best response in Ξ_{Max}. ∎

THEOREM 6.4.8. The average-time game on $\overline{\mathcal{T}}$ is determined, and there are optimal type-preserving boundary strategies in $\overline{\mathcal{T}}$, i.e., for every $q \in \overline{Q}$, we have:

$$\mathsf{Val}^{\overline{\mathcal{T}}}(q) = \inf_{\mu \in \Xi_{\mathrm{Min}}} \sup_{\chi \in \overline{\Sigma}_{\mathrm{Max}}} \mathsf{AT}_{\mathrm{Min}}(q, \mu, \chi) = \sup_{\chi \in \Xi_{\mathrm{Max}}} \inf_{\mu \in \overline{\Sigma}_{\mathrm{Min}}} \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi) = \mathsf{Val}^{\widehat{\mathcal{T}}}(q).$$

PROOF. We have the following:

$$\inf_{\mu \in \Xi_{\mathrm{Min}}} \sup_{\chi \in \overline{\Sigma}_{\mathrm{Max}}} \mathsf{AT}_{\mathrm{Min}}(q, \mu, \chi) = \inf_{\mu \in \Xi_{\mathrm{Min}}} \sup_{\chi \in \Xi_{\mathrm{Max}}} \mathsf{AT}_{\mathrm{Min}}(q, \mu, \chi) = \sup_{\chi \in \Xi_{\mathrm{Max}}} \inf_{\mu \in \Xi_{\mathrm{Min}}} \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi) = \sup_{\chi \in \Xi_{\mathrm{Max}}} \inf_{\mu \in \overline{\Sigma}_{\mathrm{Min}}} \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi),$$

where the first and last equalities follow from Lemma 6.4.7, and the second equality follows from Theorem 6.4.6.

Now we show that $\underline{\mathsf{Val}}^{\overline{\mathcal{T}}}(q) \geq \underline{\mathsf{Val}}^{\widehat{\mathcal{T}}}(q)$. The proof that $\overline{\mathsf{Val}}^{\overline{\mathcal{T}}}(q) \leq \overline{\mathsf{Val}}^{\widehat{\mathcal{T}}}(q)$ is similar and hence omitted.

$$\begin{aligned}\underline{\mathsf{Val}}^{\overline{\mathcal{T}}}(q) = \sup_{\chi \in \overline{\Sigma}_{\mathrm{Max}}} \inf_{\mu \in \overline{\Sigma}_{\mathrm{Min}}} \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi) &\geq \sup_{\chi \in \Xi_{\mathrm{Max}}} \inf_{\mu \in \overline{\Sigma}_{\mathrm{Min}}} \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi) \\ &= \sup_{\chi \in \Xi_{\mathrm{Max}}} \inf_{\mu \in \Xi_{\mathrm{Min}}} \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi) = \underline{\mathsf{Val}}^{\widehat{\mathcal{T}}}(q).\end{aligned}$$

The first inequality follows as $\Xi_{\mathrm{Max}} \subseteq \overline{\Sigma}_{\mathrm{Max}}$. The first equality holds by definition, the second equality is proved in the first paragraph of this proof, and the third equality follows from Theorem 6.4.6. From Lemma 6.4.4 we know that $\underline{\mathsf{Val}}^{\widehat{\mathcal{T}}}(q) = \overline{\mathsf{Val}}^{\widehat{\mathcal{T}}}(q)$. It follows that the average-time game on $\overline{\mathcal{T}}$ is determined, and there are optimal type-preserving boundary strategies in $\overline{\mathcal{T}}$. ∎

6.4.3. Determinacy of Average-Time Games on the Region Graph

LEMMA 6.4.9. If the strategies $\mu^* \in \Xi_{\mathrm{Min}}$ and $\chi^* \in \Xi_{\mathrm{Max}}$ are optimal for respective players in $\overline{\mathcal{T}}$ then for every $\varepsilon > 0$, we have that

$$\sup_{\chi \in \overline{\Sigma}_{\mathrm{Max}}} \mathsf{AT}_{\mathrm{Min}}(q, \mu^*_\varepsilon, \chi) \leq \mathsf{Val}^{\overline{\mathcal{T}}}(q) + \varepsilon \text{ and } \inf_{\mu \in \overline{\Sigma}_{\mathrm{Min}}} \mathsf{AT}_{\mathrm{Max}}(q, \mu, \chi^*_\varepsilon) \geq \mathsf{Val}^{\overline{\mathcal{T}}}(q) - \varepsilon,$$

for all $\mu^*_\varepsilon \in \widetilde{\Sigma}^{(\mu^*, \varepsilon)}_{\mathrm{Min}}$ and $\chi^*_\varepsilon \in \widetilde{\Sigma}^{(\chi^*, \varepsilon)}_{\mathrm{Max}}$.

PROOF. Let $\mu^* \in \Xi_{\mathrm{Min}}$ and $\chi^* \in \Xi_{\mathrm{Max}}$ are optimal for respective players in $\overline{\mathcal{T}}$. For all $\chi \in \overline{\Sigma}_{\mathrm{Max}}$, $\varepsilon > 0$, and $\mu^*_\varepsilon \in \widetilde{\Sigma}^{(\mu^*, \varepsilon)}_{\mathrm{Min}}$, we have the following:

$$\mathsf{AT}_{\mathrm{Min}}(q, \mu^*_\varepsilon, \chi) \;\leq\; \mathsf{AT}_{\mathrm{Min}}(q, \mu^*, \chi^{(q, \mu^*_\varepsilon)}) + \varepsilon \;\leq\; \mathsf{AT}_{\mathrm{Min}}(q, \mu^*, \chi^*) + \varepsilon \;=\; \mathsf{Val}^{\overline{\mathcal{T}}}(q) + \varepsilon.$$

The first inequality is by Corollary 6.3.20. The second inequality holds because χ^* is an optimal strategy and the equality is due to the fact that μ^* and χ^* are optimal. ∎

THEOREM 6.4.10. The average-time game on $\widetilde{\mathcal{T}}$ is determined, and for every $q \in \overline{Q}$, we have $\mathsf{Val}^{\widetilde{\mathcal{T}}}(q) = \mathsf{Val}^{\overline{\mathcal{T}}}(q)$.

PROOF. Let $\mu^* \in \Xi_{\text{Min}}$ be an optimal strategy of player Min in $\overline{\mathcal{T}}$. Let us fix an $\varepsilon > 0$ and $\mu^*_\varepsilon \in \widetilde{\Sigma}^{(\mu^*,\varepsilon)}_{\text{Min}}$.

$$\overline{\mathsf{Val}}^{\widetilde{\mathcal{T}}}(q) = \inf_{\mu \in \widetilde{\Sigma}_{\text{Min}}} \sup_{\chi \in \widetilde{\Sigma}_{\text{Max}}} \mathsf{AT}_{\text{Min}}(q, \mu, \chi) \leq \sup_{\chi \in \widetilde{\Sigma}_{\text{Max}}} \mathsf{AT}_{\text{Min}}(q, \mu^*_\varepsilon, \chi) \leq$$

$$\sup_{\chi \in \overline{\Sigma}_{\text{Max}}} \mathsf{AT}_{\text{Min}}(q, \mu^*_\varepsilon, \chi) \leq \mathsf{Val}^{\overline{\mathcal{T}}}(q) + \varepsilon.$$

The second inequality follows because $\mu^*_\varepsilon \in \widetilde{\Sigma}_{\text{Min}}$ and the third inequality follows because $\widetilde{\Sigma}_{\text{Max}} \subseteq \overline{\Sigma}_{\text{Max}}$. The last inequality follows from Lemma 6.4.9 because $\mu^* \in \Xi_{\text{Min}}$ is an optimal strategy in $\overline{\mathcal{T}}$. Similarly we show that for every $\varepsilon > 0$ we have that $\underline{\mathsf{Val}}^{\widetilde{\mathcal{T}}}(q) \geq \mathsf{Val}^{\overline{\mathcal{T}}}(q) - \varepsilon$. Hence it follows that $\mathsf{Val}^{\widetilde{\mathcal{T}}}(q)$ exists and its value is equal to $\mathsf{Val}^{\overline{\mathcal{T}}}(q)$. ∎

6.5. Complexity

The main decision problem for average-time game is as follows: given an average-time game $\Gamma = (\mathcal{T}, L_{\text{Min}}, L_{\text{Max}})$, a state $s \in S$, and a number $B \in \mathbb{R}_\oplus$, decide whether $\mathsf{Val}(s) \leq B$.

THEOREM 6.5.1. Average-time games are EXPTIME-complete on timed automata with at least two clocks.

PROOF. From Theorem 6.4.2 we know that in order to solve an average-time game starting from an initial state of a timed automaton, it is sufficient to solve the average-time game on the set of states of the boundary region graph of the automaton that are reachable from the initial state. Observe that every region, and hence also every configuration of the game, can be represented in space polynomial in the size of the encoding of the timed automaton and of the encoding of the initial state, and that every move of the game can be simulated in polynomial time. Therefore, the value of the game can be computed by a straightforward alternating PSPACE algorithm, and hence the problem is in EXPTIME because APSPACE = EXPTIME.

In order to prove EXPTIME-hardness of solving average-time games on timed automata with two clocks, we reduce the EXPTIME-complete problem of solving countdown games [**JLS07**] to it. Let $G = (N, M, \pi, n_0, B_0)$ be a countdown game, where N is a finite set of nodes, $M \subseteq N \times N$ is a set of moves, $\pi : M \to \mathbb{N}_+$ assigns a positive integer number to every move, and $(n_0, B_0) \in N \times \mathbb{N}_+$ is the initial configuration.

Remark. W.l.o.g we assume that there is an integer W such that $\pi(n_1, n_2) \geq W$ for every move $(n_1, n_2) \in M$.

In every move of the game from a configuration $(n, B) \in N \times \mathbb{N}_+$, first player 1 chooses a number $p \in \mathbb{N}_+$, such that $p \leq B$ and $\pi(n, n') = p$ for some move $(n, n') \in M$, and then player 2 chooses a move $(n, n'') \in M$, such that $\pi(n, n'') = p$; the new configuration is then

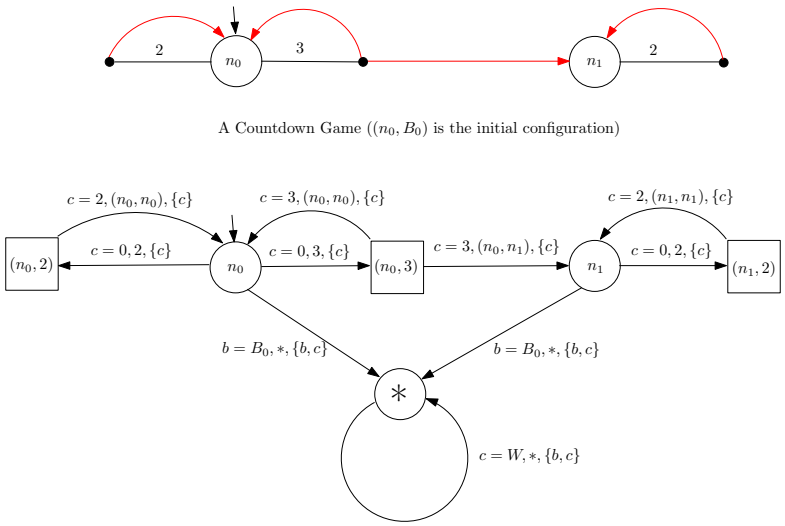


FIGURE 6.1. A Reduction from a Countdown Game to an Average-Time Game.

$(n'', B - p)$. Player 1 wins a play of the game when a configuration $(n, 0)$ is reached, and he loses (i.e., player 2 wins) when a configuration (n, B) is reached in which player 1 is stuck, i.e., for all moves $(n, n') \in M$, we have $\pi(n, n') > B$.

We define the timed automaton $\mathcal{T}_G = (L, C, S, A, E, \delta, \xi, F)$ by setting $C = \{ b, c \}$; $S = L \times (\llbracket B_0 \rrbracket_{\mathbb{R}})^2$; $A = \{ * \} \cup P \cup M$, where $P = \pi(M)$, the image of the function $\pi : M \to \mathbb{N}_+$;

$$L = \{ * \} \cup N \cup \big\{ (n, p) \,:\, \text{there is } (n, n') \in M, \text{s.t. } \pi(n, n') = p \big\};$$

$$E(a) = \begin{cases} \{ (n, \nu) \,:\, n \in N \text{ and } \nu(b) = B_0 \} \text{ if } a = *, \\ \{ (*, \nu) \,:\, \nu(c) = W \} \text{ if } a = *, \\ \big\{ (n, \nu) \,:\, \text{there is } (n, n') \in M, \text{s.t. } \pi(n, n') = p \text{ and } \nu(c) = 0 \big\} \text{ if } a = p \in P, \\ \big\{ \big((n, p), \nu \big) \,:\, \pi(n, n') = p \text{ and } \nu(c) = p \big\} \text{ if } a = (n, n') \in M, \end{cases}$$

$$\delta(\ell, a) = \begin{cases} * & \text{if } \ell = n \in N \text{ and } a = *, \\ * & \text{if } \ell = * \text{ and } a = *, \\ (n, p) & \text{if } \ell = n \in N \text{ and } a = p \in P, \\ n' & \text{if } \ell = (n, p) \in N \times P \text{ and } a = (n, n') \in M; \end{cases}$$

$\xi(a) = \{ c \}$, for every $a \in A \setminus \{ * \}$ and $\xi(*) = \{ b, c \}$. Note that the timed automaton $\mathcal{T}_G$ has only two clocks and that the clock b is reset only in the special location $*$.

Finally, we define the average-time game on timed game automaton $\Gamma_G = (\mathcal{T}_G, L_1, L_2)$ by setting $L_1 = N$ and $L_2 = L \setminus L_1$. It is routine to verify that value of the average-time game at the state $(n_0, (0, 0)) \in S$ is W in the average-time game on Γ_G if and only if player 1 has a winning strategy (from the initial configuration (n_0, B_0)) in the countdown game G. An example of such reduction is shown in Figure 6.1. ∎

Part 3

Conclusion

7

Summary and Future Work

7.1. Summary

In this thesis we presented a study of competitive optimisation on timed automata. We introduced an important class of strategies, boundary strategies, that suggest to a player a symbolic timed move of the form (b, c, a)— "wait until the value of the clock c is in very close proximity of the integer b, and then execute a transition labelled with the action a". The existence of optimal boundary strategies for a competitive optimisation problem allows us to work with the boundary region graph abstraction of the timed automata, which—for a fixed initial state—is a finite graph. Hence, to solve competitive optimisation problems on timed automata, it is sufficient to prove that there exist optimal boundary strategies for both players, and to then solve the corresponding optimisation problems on finite subgraphs of reachable states of the boundary region graph.

We showed that for a number of non-competitive optimisation problems and for competitive optimisation problems with reachability time and average time objectives there exist optimal boundary strategies.

For a noncompetitive optimisation problem on concavely-priced timed automata optimal boundary strategies exists if the corresponding cost function is a concave-regular function. Concave-regularity is satisfied by a number of cost functions including reachability-price, average-time, average-price, and average price-per-time-unit. The decision version of optimisation problems for a concave-regular cost function is PSPACE-complete on timed automata with at least three clocks.

Using Bellman's optimality equations, we showed that reachability-time games are positionally determined, and that in reachability-time games optimal boundary strategies exist for both players. We presented a strategy improvement algorithm to solve these optimality equations. We showed that the complexity of solving reachability-time games is EXPTIME-complete for timed automata with at least two clocks.

Using closed region graph, an abstraction of timed automata, we showed that average-time games are determined, and that in average-time games optimal boundary strategies exist for both players. We showed that the complexity of solving average-time games is EXPTIME-complete for timed automata with at least two clocks.

7.2. Future Work

7.2.1. Implementation

We have implemented the algorithm proposed in Chapter 5 for computing the optimal reachability-time of a timed automaton. The Flex code (lexical grammar) and the Bison code (parser grammar) for the specification of a timed game automaton is given in the Appendix D. A self-explanatory example of our specification language for timed (game) automata is given below.

EXAMPLE 7.2.1 (Lightbulb example)**.** The following is a specification of a light bulb from the Example 3.1. The bound of this timed automaton is 2.

```
system light Bulb
begin
  automaton light
  begin
    locations_n : {off, bright, dim};  /* locations of Min */
    locations_x : {};  /* locations of Max */
    clocks : x[2];    /* Declaration of clock variables, the number
                       * in square brackets denote the bound for
                       * that clock.*/
    actions : {press, unpress}; /* set of actions */

    invar(off, {x<=2});    /* Invariant for location off */
    invar(dim, {x<=2});    /* Invariant for location dim */
    invar(bright, {x<=2});/* Invariant for location bright */

    trans(off, dim, press, {x >= 0}, {x});

                     /* specification of a guarded transition.
                      * e.g., trans(l1, l2, a, g, rl) denotes a
                      * transition from locations l1 to l2,
                      * labelled with the action a, the guard of
                      * the transition is g and the set of clocks
                      * to be reset after the transition is rl.*/

    trans(dim, bright, press, {x <= 1}, {x});
    trans(dim, off, press, {x >= 1}, {x});
    trans(bright, off, press, {x >= 0}, {x});

    init(dim, {x=0});     /* Set of initial states */
    final(bright, {x>0}); /* Set of final states */
  end
end
```

The computational complexity of reachability-time games (EXPTIME-complete) indicates that it is unlikely to have an efficient algorithm for this problem. Our implementation of the strategy improvement algorithm for the reachability-time games is inefficient: it constructs and stores the whole boundary region graph of the timed automaton under consideration. More work is needed to make this implementation of practical interest.

A possible future direction of research work is to investigate symbolic zone-based algorithms [**UPP**] for competitive optimisation problems studied in this thesis.

7.2.2. Complexity of Competitive Optimisation Problems for Timed Automata with Two Clocks

The exact complexity of the reachability problem on timed automata with two clocks is unknown. By the time-abstract bisimulation property of region equivalence relation, it follows that the problem is in PSPACE. Laroussinie, Markey, and Schnoebelen [**LMS04**] proved the NP-hardness by giving a reduction from NP-complete subset-sum problem to reachability problem on timed automata with two clocks.

Using countdown games [**JLS07**], we proved that reachability games on timed automata with two clocks are EXPTIME-hard. Our initial guess was to use a similar approach—first prove that one-player countdown games are PSPACE-complete and then give a reduction from one-player countdown games to reachability problem on timed automata with two clocks—to prove PSPACE-hardness of this problem. However, one player countdown games turned out to be NP-complete.

The question of the exact complexity of the reachability problem, and hence of all the other one-player optimisation problems (reachability price, discounted price, average price, price-per-reward average, etc.) on concavely-priced timed automata with two clocks remains open.

However, for timed automata with one clock we can generalise the construction of Laroussinie, Markey, and Schnoebelen [**LMS04**]—which they used to show NLOGSPACE-membership of the reachability problem for one clock timed automata—to obtain an abstraction similar to the boundary region graph whose size is polynomial in the size of timed automata. Using such abstraction and the techniques presented in this thesis, it can be shown that for one clock timed automata noncompetitive optimisation problems for concave-regular cost functions are in NLOGSPACE. Similarly it is easy to show that for one-clock timed automata reachability-time games are in PTIME, and average-time games are in NP $\cap$ co-NP.

7.2.3. Maximisation Problem on Concave-Priced Timed Automata

In Chapter 4 we showed that minimisation problems on concavely-priced timed automata for concave-regular cost functions are PSPACE-complete. Using similar arguments it can be shown that maximisation problems on convexly-priced timed automata for convex-regular cost functions are PSPACE-complete.

We do not know much about maximisation problem on concavely-priced timed automata. However, for the following class of cost functions, it can easily be shown that

both maximisation and minimisation problems on concavely-priced timed automata are PSPACE-complete, as a monotonic function is both quasiconcave and quasiconvex [**BV04**].

DEFINITION 7.2.2 (Monotonic-Regular Cost Function)**.** A cost function $\mathsf{Cost} : \mathsf{PreRuns} \to \mathbb{R}$ is *monotonic-regular* if it satisfies the following properties.

(1) *(Monotonicity).* For every region $R \in \mathcal{R}$ and for every run type $\Lambda \in \mathrm{Types}(R)$, there is $N \in \mathbb{N}$, such that for every state $s \in R$ and for every $n \geq N$, the function $\mathsf{Cost}^{\Lambda}_{n,s}$ is monotonic on $\Delta^{\Lambda}_{n,s}$.

(2) *(Regular Lipschitz-continuity).* There is a constant $\kappa \geq 0$, such that for every region $R \in \mathcal{R}$ and for every *positional run type* $\Lambda \in \mathrm{Types}(R)$, there is $N \in \mathbb{N}$, such that for every state $s \in R$ and for every $n \geq N$, the function $\mathsf{Cost}^{\Lambda}_{n,s}$ is κ-continuous on $\Delta^{\Lambda}_{n,s}$.

(3) *(Positional optimality).* There is a positional optimal strategy for Cost in $\widehat{\mathcal{T}}$.

(4) *(Uniform convergence).* For every $s \in S$ we have that

$$\mathsf{Cost}^{\widehat{\mathcal{T}}}_{*}(s,[s]) = \lim_{n\to\infty} \mathsf{Cost}^{\widehat{\mathcal{T}}}_{n,*}(s,[s]).$$

7.2.4. Optimisation on Probabilistic Timed Automata

In this thesis, we considered competitive optimisation problem for the systems having time-critical behaviours modelled using timed automata. One possible direction for the future work in this line of research is to extend these techniques to solve competitive optimisation problems for systems having both time-critical and probabilistic behaviours.

Markov decision processes (MDPs) extend finite automata by providing a probability distribution on the successor states for every transition. Timed automata, as we discussed in this thesis, extend finite automata by providing a mechanism to constrain the transitions with continuous time. Probabilistic timed automata—a hybrid between MDPs and timed automata—were suggested by Kwiatkowska et al. [**KNSS99**] as a modelling formalism for system exhibiting both timed and probabilistic characteristics. We propose the study of optimisation problems (expected reachability-price, expected discounted-price, and expected average-price) on *concavely-priced* probabilistic timed automata which, arguably, can be used to model a larger class of scheduling problems than MDPs, concavely-priced timed automata and probabilistic timed automata.

We conjecture that optimisation problems for these three objectives are decidable. We claim that an abstraction—let us call it boundary region MDP—similar to boundary region graph extended with probability distribution on the edges may be suitable to solve these optimisation problems on probabilistic timed automata.

Part 4

Appendix

A

Notations and Acronyms

> It strikes me that mathematical writing is similar to using a language. To be understood you have to follow some grammatical rules. However, in our case, nobody has taken the trouble of writing down the grammar; we get it as a baby does from parents, by imitation of others. Some mathematicians have a good ear; some not. That's life.
>
> Jean-Pierre Serre

A.1 General Notations

Throughout the thesis we assume that the reader has some background in set theory and real analysis. However to keep the thesis self-contained, in Table 1, we provide some explanation of the notations used.

General Notations	
$\mathbb{Z}$	set of Integers
$\mathbb{N}$	set of nonnegative Integers
$\mathbb{N}_+$	set of positive Integers
$\mathbb{Q}$	set of rational numbers, that is the set $\{\frac{m}{n} : m \in \mathbb{Z}, n \in \mathbb{Z}, n \neq 0\}$
$\mathbb{R}$	set of Reals containing a maximal elemet ∞
$\mathbb{R}_\oplus$	set of non-negative Reals
$[\![n]\!]_{\mathbb{N}}$	set $\{i \in \mathbb{N} : i \leq n\}$
$[\![n]\!]_{\mathbb{R}}$	set $\{r \in \mathbb{R} : 0 \leq r \leq n\}$
	continued on next page

continued from previous page	
$x \in X$	x is an element of the set X
$x \notin X$	x is not an element of the set X
$\{x \in X \,:\, P(x)\}$	the set of all $x \in X$ for which the predicate $P(x)$ is true
$X \subseteq Y$	the set X is a subset of the set Y
$X \cup Y$	the union of the sets X and Y
$X \cap Y$	the intersection of the sets X and Y
$X \times Y$	Cartesian product of the sets X and Y
$X \setminus Y$	subset of elements of X which are not contained in Y
$f : X \to Y$	f is a function from the set X to the set Y
$f : X \rightharpoonup Y$	f is a partial function from the set X to the set Y
$[X \to Y]$	set of functions $f : X \to Y$
$[X \rightharpoonup Y]$	set of partial functions $f : X \rightharpoonup Y$
$\mathsf{dom}(f)$	domain of the function f
ε	a small quantity
$\mathsf{Choose}(X)$	an arbitrary element of the set X
$\lvert r \rvert$	absolute value of r
$\lfloor r \rfloor$	floor of r, i.e., largest Integer $n \in \mathbb{N}$ such that $n \leq r$
$\wr r \int$	fractional part of r, i.e. $r - \lfloor r \rfloor$
$\mathbb{R}^n$	n-dimensional Euclidean space
$\lVert x \rVert_\infty$	$\lVert x \rVert_\infty = \max\{\lvert x_i \rvert\}$, where $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$
$\lVert x \rVert_\infty$	$\lVert x \rVert_\infty = \sup_{i \in \mathbb{N}}\{\lvert x_i \rvert\}$, where $x = (x_1, x_2, \ldots,) \in \mathbb{R}^\omega$
$\mathsf{clos}(\mathcal{D})$	topological closure of the set $\mathcal{D}$, i.e., the set $\mathbb{R}^n \setminus \mathsf{int}(\mathbb{R}^n \setminus \mathcal{D})$
$\overline{\mathcal{D}}$	set $\mathsf{clos}(\mathcal{D})$
∎	completion of a proof

TABLE 1. General Notations

In Table 2 we explain notations related to the minimum and the infimum of a set or a function. For a function we also define the operator "argmin" which returns the set of arguments that minimise a function. For the sake of succinctness of technical presentation, we sometimes use certain non-standard versions of minimum, infimum, and argmin and they are defined in Table 3. For Tables 2 and 3, let Y be a subset of a *partially ordered set* $(Z, \leq)$ and let $f : X \to Y$. The operators max, sup, and argmax are defined in an analogous manner.

Minimum, Infimum, and Argmin	
$y_* = \min Y$	$y_* \in Y$ and $y_* \leq y$ for all $y \in Y$
$y_* = \inf Y$	$y_* \leq y$ for all $y \in Y$, and for all $z \in Z$ if $z \leq y$ for all $y \in Y$, then $z \leq y_*$
$y_* = \min_{x \in X} f(x)$	$y_* = \min\{f(x) \,:\, x \in X\}$
$y_* = \inf_{x \in X} f(x)$	$y_* = \inf\{f(x) \,:\, x \in X\}$
$\operatorname{argmin}_{x \in X} f(x)$	set $\{x_* \in X \,:\, f(x_*) = \min_{x \in X} f(x)\}$

TABLE 2. Standard Notations related to minmum, infimum, and argmin.

Minimum, Infimum, and Argmin (non-standard notations)	
$y_* = \min_{x\in X}\{f(x) : P(x)\}$	$y_* = \min\{f(y) : y \in \{x \in X : P(x)\}\}$
$\operatorname{argmin}_{x\in X}\{f(x) : P(x)\}$	set $\{x_* \in X : f(x_*) = \min_{x\in X}\{f(x) : P(x)\}\}$
$\min_x\{f(x) : P(x)\}$	$\min_{x\in X}\{f(x) : P(x)\}$, when X is clear from context
$\operatorname{argmin}_x\{f(x) : P(x)\}$	$\operatorname{argmin}_{x\in X}\{f(x) : P(x)\}$, when X is clear from context

TABLE 3. Non-standard notations related to minimum, infimum, and argmin

Given two partially ordered sets $(Y_1, \leq)$ and $(Y_2, \leq)$ we define lexicographic ordering $\leq^{\text{lex}}$ on $Y_1 \times Y_2$ as $(y_1, y_2) \leq^{\text{lex}} (y_1', y_2')$ if $y_1 < y_1'$, or $y_1 = y_1'$ and $y_2 \leq y_2'$. We write $(y_1, y_2) <^{\text{lex}} (y_1', y_2')$ if we have that $y_1 < y_1'$, or $y_1 = y_1'$ and $y_2 < y_2'$. The operators $>^{\text{lex}}$ and $\geq^{\text{lex}}$ are defined analogously. For a function $f : X \to Y$, where Y is a subset of partially ordered set $(Y_1 \times Y_2, \leq^{\text{lex}})$, we write $\min^{\text{lex}}$ and $\operatorname{argmin}^{\text{lex}}$ for min and argmin, respectively. The notations for $\max^{\text{lex}}$ and $\operatorname{argmax}^{\text{lex}}$ are defined analogously.

A.2 Timed Automata Specific Notations

For easy reference we present the notations related to timed automata in Table 4.

timed Automata	
$\mathcal{T}$	a timed automaton
Γ	a timed game automaton
C	finite set of clocks
$\mathcal{V}_C$	set of clock valuations $[C \to [\![k]\!]_{\mathbb{R}}]$ over C
ν, ν' etc.	denote clock valuations
$\nu' = \nu + t$	if $\nu'(c) = \nu(c)$ for all $c \in C$
$\nu' = \mathsf{Reset}(\nu, C')$	if $\nu'(c) = \nu(c)$, for all $c \notin C'$; and $\nu'(c) = 0$, for all $c \in C'$.
$\nu' = \lfloor \nu \rfloor$	if $\nu'(c) = \lfloor \nu(c) \rfloor$, for every clock $c \in C$
$\wr \nu \wr$	*fractional signature* (see index) of ν
$(\!\lvert \nu \rvert\!)$	*cellular signature* (see index) of ν
$\mathsf{SCC}(\nu)$	*simple clock constraints* (see index) which hold in ν
$[\nu]$	*clock region* (see index) of ν
L	finite set of locations
L_{Min}	set of locations controlled by player Min
L_{Max}	set of locations controlled by player Max
Q	set of configurations $Q = L \times \mathcal{V}_C$
S	the set of states of a timed automaton
s, s' etc.	denote a configuration or a state
A	finite set of actions
$E : A \to 2^S$	action enabledness function
$\delta : L \times A \to L$	transition function

continued on next page

continued from previous page	
$\xi : A \to 2^C$	clock reset function
$F \subseteq S$	set of final states
$s' = s + t$	if $s = (\ell, \nu)$ then $s' = (\ell, \nu + t)$
$s \rightharpoonup_t s'$	same as $s' = s + t$
$s \to_t s'$	$s \rightharpoonup_t s'$ and $s + t' \in S$ for all $t' \in [0, t]$
$s' = \mathsf{Succ}(s, a)$	if $s = (\ell, \nu)$ then $s' = (\delta(\ell, a), \mathsf{Reset}(\nu, \xi(a)))$
$s \overset{a}{\rightharpoonup} s'$	same as $s' = \mathsf{Succ}(s, a)$
$s \xrightarrow{a} s'$	if $s \overset{a}{\rightharpoonup} s'$; $s, s' \in S$; and $s \in E(a)$
$s \to_* s'$	if there exists $t \in \mathbb{R}_\oplus$ s.t. $s \to_t s'$
$s' = \mathsf{Succ}(s, (t, a))$	if $s' = \mathsf{Reset}(s + t, a)$, i.e. $s \rightharpoonup_t s'' \overset{a}{\rightharpoonup} s'$
$s \overset{a}{\rightharpoonup}_t s'$	same as $s' = \mathsf{Succ}(s, (t, a))$
$s \xrightarrow{a}_t s'$	if $s \to_t s'' \xrightarrow{a} s'$
r, r' etc.	runs, i.e., sequence of valid state and transition pairs
Runs	set of runs of $\mathcal{T}$
$\mathrm{Runs}(s)$	set of runs of $\mathcal{T}$ starting from s
$\mathsf{Last}(r)$	last state of the finite run r
$\mathsf{Stop}(r)$	the index of first final state in the run r
$\mathsf{Time}(r)$	$\sum_{i=1}^{n} t_i$ where $r = \langle s_0, (t_1, a_1), \ldots, (t_n, a_n), s_{n+1} \rangle$
R, R' etc.	denote a *region* (see index)
$\mathcal{Z}$	set of *clock zones*(see index)
$\mathcal{R}$	set of regions
$s \in \mathsf{clos}(R)$	see index, *closure of the region*
$s \in \mathsf{bd}(R)$	see index, *boundary of the region*
$R \to_* R'$	if for all $s \in R$ there is $s' \in R'$ s.t. $s \to_* s'$
$R \to_{+1} R'$	R' is time successor of R (see index, *region*)
$R' \leftarrow_{+1} R$	same as $R \to_{+1} R'$
$R \xrightarrow{a} R'$	if there is $s \in R$ and $s' \in R'$, such that $s \xrightarrow{a} s'$
$R \to_{b,c} R'$	$b \in \mathbb{N}$ and $c \in C$ are s.t. for all $s \in R$ we have $s + b - s(c) \in R'$
$\mathcal{R}_{\text{Thin}}$	set of thin regions (see index, *region*)
$\mathcal{R}_{\text{Thick}}$	set of thick regions (see index, *region*)
$\mathcal{A}$	set $[\![\mathcal{K}]\!]_{\mathbb{N}} \times C \times A$ of *boundary timed actions* (see index)
$\alpha = (b, c, a)$	a boundary timed action
$R \xrightarrow{\alpha} R'$	there is $R'' \in \mathcal{R}$ s.t. $R \to_{b,c} R'' \xrightarrow{a} R'$
$R \overset{\alpha}{\rightsquigarrow} R'$	$R \xrightarrow{\alpha} R'$; or $R \to_{b,c} R'' \to_{+1} R'' \xrightarrow{a} R'$; or $R \to_{b,c} R'' \leftarrow_{+1} R'' \xrightarrow{a} R'$
$\pi : S \times \mathbb{R}_\oplus \times A \to \mathbb{R}$	$\pi(s, (t, a))$ is the price for the timed action (t, a) from s
$\varrho : S \times \mathbb{R}_\oplus \times A \to \mathbb{R}$	$\varrho(s, (t, a))$ is the reward for the timed action (t, a) from s
Σ	set of strategies
Σ_{Max}	set of strategies of player Max
Σ_{Min}	set of strategies of player Min

continued on next page

continued from previous page	
Π	set of positional strategies
Π_{Max}	set of positional strategies of player Max
Π_{Min}	set of positional strategies of player Min
μ, μ' etc.	denote a strategy of player Min
χ, χ' etc.	denote a strategy of player Max
$\mathcal{T}_{\mathrm{RA}}$	*region automaton* (see index) of $\mathcal{T}$
$\mathcal{T}_{\mathrm{CP}}$	*corner-point abstraction* (see index) of $\mathcal{T}$
$\mathcal{T}_{\mathrm{BR}}$	*boundary region automaton* (see index) of $\mathcal{T}$
$\widetilde{\mathcal{T}}$	*region graph* (see index) of $\mathcal{T}$
$\widehat{\mathcal{T}}$	*boundary region graph* (see index) of $\mathcal{T}$
$\overline{\mathcal{T}}$	*closed region graph* (see index) of $\mathcal{T}$
$\mathrm{Runs}^{\mathcal{G}}$	set of runs of $\mathcal{G} \in \{\widetilde{\mathcal{T}}, \widehat{\mathcal{T}}, \overline{\mathcal{T}}\}$
$\mathrm{Runs}^{\mathcal{G}}(s, R)$	set of runs of $\mathcal{G} \in \{\widetilde{\mathcal{T}}, \widehat{\mathcal{T}}, \overline{\mathcal{T}}\}$ starting from (s, R)
$[\![r]\!]_{\mathcal{R}}$	*type* (see index) of the run r
$\mathsf{PreRuns}$	set of *pre-runs* (see index) of $\mathcal{T}$
$\mathsf{PreRuns}_{\mathrm{fin}}$	set of finite *pre-runs* (see index) of $\mathcal{T}$
$\mathsf{PreRuns}(s, R)$	set of *pre-runs* starting from (s, R)
$\mathsf{PreRuns}_{\mathrm{fin}}(s, R)$	set of finite *pre-runs* starting from (s, R)
$\Sigma^{\mathrm{pre}}_{\mathrm{Min}}$	set of *pre-strategies* (see index) of player Min in $\mathcal{T}$
$\Sigma^{\mathrm{pre}}_{\mathrm{Max}}$	set of pre-strategies of player Max in $\mathcal{T}$
$\widetilde{\Sigma}_{\mathrm{Min}}$	set of strategies of player Min in $\widetilde{\mathcal{T}}$
$\widetilde{\Sigma}_{\mathrm{Max}}$	set of strategies of player Max in $\widetilde{\mathcal{T}}$
$\overline{\Sigma}_{\mathrm{Min}}$	set of strategies of player Min in $\overline{\mathcal{T}}$
$\overline{\Sigma}_{\mathrm{Max}}$	set of strategies of player Max in $\overline{\mathcal{T}}$
$\widehat{\Sigma}_{\mathrm{Min}}$	set of strategies of player Min in $\widehat{\mathcal{T}}$
$\widehat{\Sigma}_{\mathrm{Max}}$	set of strategies of player Max in $\widehat{\mathcal{T}}$
Ξ_{Min}	set of *type-preserving boundary strategies* (see index) of player Min
Ξ_{Max}	set of *type-preserving boundary strategies* (see index) of player Max
$\mu^{\downarrow q}, \chi^{\downarrow q}$, etc.	see Definition 6.3.13
$\chi^{(q,\mu)}, \mu^{(q,\chi)}$, etc.	see Definition 6.3.14
$\widetilde{\Sigma}^{(\mu,\varepsilon)}_{\mathrm{Min}}, \widetilde{\Sigma}^{(\chi,\varepsilon)}_{\mathrm{Max}}$, etc.	see Definition 6.3.17
$p \models \mathrm{Opt}$	the function p satisfies the optimality equations Opt
$\mathcal{OE}^{\mathsf{Cost}}_{\mathrm{Min}}(G)$	optimality eqns. for minimisation of cost function Cost on G
$\mathcal{OE}^{\mathsf{Cost}}_{\mathrm{Max}}(G)$	optimality eqns. for maximisation of cost function Cost on G
$\mathcal{OE}^{\mathsf{Cost}}_{\mathrm{MinMax}}(G)$	optimality eqns. for cost game of payoff function Cost on G

TABLE 4. Notations specific to timed automata

A.3 Acronyms

List of commonly used acronyms is given in Table 5.

Acronyms	
w.l.o.g	without loss of generality
s.t.	such that
eqns.	equations
iff	if and only if
RT	Reachability time
RP	Reachability price
DP	Discounted price
AT	Average time per transition
AP	Average price per transition
PTAvg	Average price per time-unit
PRAvg	Price-per-reward average

TABLE 5. Acronyms

B

Results From Real Analysis

Indeed, I have found a very nice way of expressing continuity ...

Rudolph Lipschitz

B.1. Lipschitz-Continuous Functions

A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is Lipschitz-continuous **[EEJ04]** on its domain $\mathsf{dom}(f)$, if there exists a constant $K \geq 0$, called a Lipschitz constant of f, such that $\|f(x) - f(y)\| \leq K\|x - y\|$ for all $x, y \in \mathsf{dom}(f)$; we then also say that f is K-continuous. A K-continuous function is *contraction* if $K < 1$.

The following properties **[EEJ04]** of Lipschitz-continuous functions are of interest in this thesis.

LEMMA B.1.1. If for every $i = 1, 2, \ldots, k$, the function $f_i : \mathbb{R}^n \to \mathbb{R}^m$ is K_i-continuous and $w_i \in \mathbb{R}$, then the function $\mathbb{R}^n \ni x \mapsto \sum_{i=1}^{n} w_i f_i(x)$ is K-continuous for $K = \sum_{i=1}^{k} |w_i| K_i$.

PROOF. Let $x, y \in \mathbb{R}^n$ and assume that for every $i = 1, 2, \ldots, k$, the function $f_i : \mathbb{R}^n \to \mathbb{R}^m$ is K_i-continuous.

$$
\begin{aligned}
|f(x) - f(y)| &= \left| \sum_{i=0}^{k} w_i \cdot f_i(x) - \sum_{i=0}^{k} w_i \cdot f_i(y) \right| = \left| \sum_{i=0}^{k} (w_i \cdot (f_i(x) - f_i(y))) \right| \\
&\leq \sum_{i=0}^{k} (|w_i \cdot (f_i(x) - f_i(y))|) \leq \sum_{i=0}^{k} (|w_i| \cdot K_i \cdot |x - y|) = \sum_{i=0}^{k} (|w_i| \cdot K_i) \cdot |x - y|
\end{aligned}
$$

The first, the second, and the last equalities are straightforward. The first inequality follows as $|a + b| \leq |a| + |b|$, and the last inequality holds as the function f_i is K_i-continuous for all $i = 1, 2, \ldots, k$. ∎

LEMMA B.1.2. If $f_1 : \mathbb{R}^n \to \mathbb{R}^m$ and $f_2 : \mathbb{R}^m \to \mathbb{R}^k$ are K_1-continuous and K_2-continuous, respectively, then their composition, $\mathbb{R}^n \ni x \mapsto f_2(f_1(x))$, is K-continuous for $K = K_1 K_2$.

PROOF. Let $x, y \in \mathbb{R}^n$, and assume that f_1 and f_2 are K_1-continuous and K_2-continuous, respectively.

$$
|f(x) - f(y)| \;=\; |f_2(f_1(x)) - f_2(f_1(y))| \;\leq\; K_2 \cdot |f_1(x) - f_1(y)| \;\leq\; K_2 \cdot K_1 \cdot |x - y|
$$

The first inequality follows from K_2-continuity of f_2 and the last inequality follows from K_1-continuity of f_1. ∎

LEMMA B.1.3. Let $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ be K_1-continuous and K_2-continuous, respectively; let f_1 and f_2 be bounded, i.e., there is a constant $M \geq 0$, such that for all $x \in \mathsf{dom}(f_1) \cap \mathsf{dom}(f_2)$, we have $|f_1(x)|, |f_2(x)| \leq M$; and let f_2 be bounded from below, i.e., there is a constant $N > 0$, such that for all $x \in \mathsf{dom}(f_2)$, we have $f_2(x) \geq N$. Then the function $\mathbb{R}^n \ni x \mapsto f_1(x)/f_2(x)$ is K-continuous for $K = (NK_1 + MK_2)/N^2$.

PROOF. Let $x, y \in \mathbb{R}^n$, and assume that f_1 and f_2 are K_1-continuous and K_2-continuous, respectively. Further assume that f_1 and f_2 are bounded by M, and that f_2 is bounded from below by $N > 0$.

$$\begin{aligned}
|f(x) - f(y)| &= \left| \frac{f_1(x)}{f_2(x)} - \frac{f_1(y)}{f_2(y)} \right| = \left| \frac{f_1(x)f_2(y) - f_1(y)f_2(x)}{f_2(x)f_2(y)} \right| \\
&= \left| \frac{f_1(x)f_2(y) - f_2(y)f_1(y) + f_2(y)f_1(y) - f_1(y)f_2(x)}{f_2(x)f_2(y)} \right| \\
&= \left| \frac{f_2(y)(f_1(x) - f_1(y)) + f_1(y)(f_2(y) - f_2(x))}{f_2(x)f_2(y)} \right| \\
&\leq \left| \frac{1}{f_2(x)} \right| \cdot K_1 \cdot |x - y| + \left| \frac{f_1(y)}{f_2(x)f_2(y)} \right| \cdot K_2 \cdot |x - y| \\
&\leq \frac{K_1}{N} \cdot |x - y| + \frac{K_2 \cdot M}{N^2} \cdot |x - y| = \left(\frac{K_1 \cdot N + K_2 \cdot M}{N^2} \right) \cdot |x - y|
\end{aligned}$$

The first inequality follows from Lipschitz-continuity of f_1 and f_2 and the second inequality follows from boundedness assumptions. All the equalities are trivial. ∎

B.2. Concave and Quasiconcave Functions

B.2.1. Concave Functions

A set $\mathcal{D} \subseteq \mathbb{R}^n$ is *convex* if for all $x, y \in \mathcal{D}$ and $\lambda \in [0, 1]$, we have $\lambda x + (1 - \lambda)y \in \mathcal{D}$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is *concave* (on its domain $\mathsf{dom}(f) \subseteq \mathbb{R}^n$), if $\mathsf{dom}(f) \subseteq \mathbb{R}^n$ is a convex set, and for all $x, y \in \mathbb{R}^n$ and $\lambda \in [0, 1]$, we have $f(\lambda x + (1 - \lambda)y) \geq \lambda f(x) + (1 - \lambda)f(y)$. A function f is *convex* if the function $-f$ is concave. A function is *affine* if it is both convex and concave.

The *α-superlevel set* of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as $S^\alpha(f) = \{x \in \mathsf{dom}(f) : f(x) \geq \alpha\}$, and the *$\alpha$-sublevel set* of f is defined as $S_\alpha(f) = \{x \in \mathsf{dom}(f) : f(x) \leq \alpha\}$.

PROPOSITION B.2.1 ([BV04]). If a function is concave then its superlevel sets are convex; and if it is convex then its sublevel sets are convex.

The following properties **[BV04]** of concave functions are of interest in this thesis.

LEMMA B.2.2 (Non-negative weighted sum)**.** If $f_1, f_2, \ldots, f_k : \mathbb{R}^n \to \mathbb{R}$ are concave and $w_1, w_2, \ldots, w_k \geq 0$, then their weighted sum $\mathbb{R}^n \ni x \mapsto \sum_{i=1}^k w_i \cdot f_i$ is concave on its domain $\bigcap_{i=1}^k \mathsf{dom}(f_i)$.

PROOF. Let $x, y \in \mathbb{R}^n$, $\lambda \in [0,1]$, and $f_1, f_2, \ldots, f_k$ are concave.

$$\begin{aligned} f(\lambda x + (1-\lambda)y)) &= \sum_{i=1}^n w_i \cdot f_i(\lambda x + (1-\lambda)y) \geq \sum_{i=1}^n w_i \cdot (\lambda f_i(x) + (1-\lambda) f_i(y)) \\ &= \lambda \sum_{i=1}^n w_i \cdot f_i(x) + (1-\lambda) \sum_{i=1}^n w_i \cdot f_i(y) = \lambda f(x) + (1-\lambda) f(y). \end{aligned}$$

The inequality follows from the concavity assumption, and the equalities are straightforward. ∎

LEMMA B.2.3 (Composition with affine function)**.** If $f : \mathbb{R}^n \to \mathbb{R}$ is concave and $g : \mathbb{R}^m \to \mathbb{R}^n$ is affine, then their composition $\mathbb{R}^n \ni x \mapsto f(g(x))$ if concave.

PROOF. Let $x, y \in \mathbb{R}^n$ and $\lambda \in [0,1]$. Also, assume that f is concave and g is affine.

$$f(g(\lambda x + (1-\lambda)y)) \;=\; f(\lambda g(x) + (1-\lambda) g(y)) \;\geq\; \lambda f(g(x)) + (1-\lambda) f(g(y)).$$

The equality follows as g is affine, and the inequality is by concavity of f. ∎

LEMMA B.2.4 (Pointwise minimum)**.** If functions $f_1, f_2, \ldots f_k : \mathbb{R}^n \to \mathbb{R}$ are concave, then their *pointwise minimum* $\mathbb{R}^n \ni x \mapsto \min_{i=1}^k f_i(x)$ is concave (on its domain $\bigcap_{i=1}^k \mathsf{dom}(f_i)$).

PROOF. Let us assume that $f_1, f_2, \ldots, f_k$ are concave. Let $x, y \in \mathbb{R}^n$ and $\lambda \in [0,1]$.

$$\begin{aligned} f(\lambda x + (1-\lambda)y)) &= \min_{i=1}^n f_i(\lambda x + (1-\lambda)y)) \geq \min_{i=1}^n (\lambda f_i(x) + (1-\lambda) f_i(y)) \\ &\geq \lambda \min_{i=1}^n f_i(x) + (1-\lambda) \min_{i=1}^n f_i(y) = \lambda f(x) + (1-\lambda) f(y) \end{aligned}$$

The first inequality holds since, for all $i = 1, 2, \ldots, k$, the function f_i is concave, and the second inequality follows from that fact that for arbitrary sequences $\langle a_i \rangle_{i=1}^n$ and $\langle b_i \rangle_{i=1}^n$ we have $\min_{i \leq n}(a_i + b_i) \geq \min_{i \leq n} a_i + \min_{i \leq n} b_i$. ∎

The proof of the following lemma is similar to the proof of Lemma B.2.4, and hence omitted.

LEMMA B.2.5 (Pointwise infimum)**.** Let $f : \mathbb{R}^n \times Z \to \mathbb{R}$, where Z is an arbitrary (infinite) set. If for all $z \in Z$, the function $\mathbb{R}^n \ni x \mapsto f(x, z)$ is concave, then the function $\mathbb{R}^n \ni x \mapsto \inf_{z \in Z} f(x, z)$ is concave (on its domain $\bigcap_{z \in Z} \mathsf{dom}(f(\cdot, z))$).

B.2.2. Quasiconcave Functions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is *quasiconcave* (on its domain $\mathsf{dom}(f) \subseteq \mathbb{R}^n$), if $\mathsf{dom}(f)$ is a convex set, and for all $x, y \in \mathsf{dom}(f)$ and $\lambda \in [0,1]$, we have $f(\lambda x + (1-\lambda)y) \geq \min\{f(x), f(y)\}$. A function f is *quasiconvex* if the function $-f$ is quasiconcave.

PROPOSITION B.2.6 ([**BV04**])**.** A function is quasiconcave if and only if its superlevel sets are convex; and it is quasiconvex if and only if its sublevel sets are convex.

The following properties (see, Mangasarian [**Man69**]) of quasiconcave functions are of interest in this thesis.

LEMMA B.2.7 (Quasiconcavity of ratio functions)**.** For $h_1, h_2 : \mathbb{R}^n \to \mathbb{R}$, the function $\mathbb{R}^n \ni x \mapsto h_1(x)/h_2(x)$ is quasiconcave (on its domain $\mathsf{dom}(h_1) \cap \mathsf{dom}(h_2)$) if at least one of the following holds:

(1) h_1 is nonnegative and concave, h_2 is positive and convex.
(2) h_1 is nonpositive and concave, h_2 is positive and concave
(3) h_1 is nonnegative and convex, h_2 is negative and convex.
(4) h_1 is nonpositive and convex, h_2 is negative and concave.
(5) h_1 is affine and h_2 is non-zero and affine;
(6) h_1 is concave, and h_2 is positive and affine;
(7) h_1 is convex, and h_2 is negative and affine.

PROOF. To prove that the function f is quasiconcave, it is sufficient to show that the superlevel sets of f are convex, i.e. for all $\alpha \in \mathbb{R}$ we have that the set $\{x : \frac{h_1(x)}{h_2(x)} \geq \alpha\}$ or equivalently the set $\{x : h_1(x) - \alpha \cdot h_2(x) \geq 0\}$ is convex. It is easy to see that is the case when either of the above assumptions hold. ∎

B.3. Fixed Point Theorems

We say that $x \in X$ is a fixed point of a function $F : X \to X$ if $F(x) = x$. Before we state two important fixed point theorems, we need to define some concepts.

A metric space is an order pair (X, d) where X is a set and d is a metric on X. Given a metric space (X, d), a sequence $\langle x_0, x_1, \ldots \rangle$ is called *Cauchy*, if for every positive real number $\varepsilon > 0$, there exists $N \in \mathbb{N}$ such that for all natural numbers $m, n > N$ we have $d(x_m, x_n) < \varepsilon$. A metric space (X, d) in which every Cauchy sequence has a limit in X is called *complete*.

The following well-known fixed point theorems [**KK01**] are used in this thesis.

THEOREM B.3.1 (Banach Fixed Point Theorem)**.** Let the function $f : X \to X$ be a contraction over a complete metric space (X, d). Then f has a unique fixed-point $x_0 \in X$. Moreover, for an $y \in X$, $d(f^n(y), x_0) \to 0$ as $n \to \infty$.

THEOREM B.3.2 (Brouwer's Fixed Point Theorem)**.** Let X be an nonempty, compact (closed and bounded), convex set in a finite dimensional Euclidean space. Then a continuous mapping $f : X \to X$ has at least one fixed point.

C

Some Determinacy Results

It's not hard to make decisions when you know what your values are.

Roy Disney

C.1. Matrix Games

A Matrix game $\Gamma = (M)$ is played on an $(m \times n)$ matrix M between two players Max and Min. In this game, player Max chooses a row i of the matrix and player Min chooses a column j of the matrix and thus player Max wins [1] $M(i,j)$ from player Min.

Let the set of pure strategies of the player Max be $\{1,2,\ldots,m\}$ and the set of pure strategies set of player Min is $\{1,2,\ldots,n\}$. The set $\Sigma_{\text{Max}} \subseteq \mathbb{R}^m$ of mixed strategies of player Max as the set of probability distributions over the set of pure strategies. Hence we have $(p_1,p_2,\ldots,p_m) \in \Sigma_{\text{Max}}$ if and only if $\sum_{i=1}^{m} p_i = 1$ and $p_i \geq 0$ for all $1 \leq i \leq m$. We define the set Σ_{Min} of mixed strategies of the player Min in an analogous manner.

PROPOSITION C.1.1. In a matrix game the product set $\Sigma_{\text{Min}} \times \Sigma_{\text{Max}}$ of the mixed strategy sets of players Min and Max is nonempty, compact and convex.

For every index $1 \leq i \leq m$, we define the strategy $\chi^i \in \Sigma_{\text{Max}}$ such that for every index $1 \leq j \leq m$, we have $\chi^i(j) = 1$ if $j = i$, and $\chi^i(j) = 0$ otherwise. The strategies μ^i for player Min, for every $1 \leq i \leq n$, are defined analogously.

If player Max plays according to mixed strategy $\chi \in \Sigma_{\text{Max}}$ and the player Min plays according to mixed strategy $\mu \in \Sigma_{\text{Min}}$, then the expected win of player Max is given by the following payoff function $\mathbb{P} : \Sigma_{\text{Min}} \times \Sigma_{\text{Max}} \to \mathbb{R}$:

$$\mathbb{P}(\mu,\chi) = \sum_{i=1}^{m}\sum_{j=1}^{n} \chi(i) \cdot M(i,j) \cdot \mu(j).$$

The following fundamental result about the determinacy of Matrix games is due to John von Neumann **[vN28]**:

[1] We always say that player Max wins the amount $M(i,j)$ from player Min upon the selection of $(i,j)^{th}$ entry of the matrix. However, in practice, if $M(i,j)$ is positive then player Max wins that amount from Min; and if $M(i,j)$ is negative player Min wins $|M(i,j)|$ from player Max.

THEOREM C.1.2 (Determinacy theorem for matrix games [**vN28**])**.** Every Matrix game with mixed strategies is determined.

John von Neumann's 1928 proof of determinacy is quite sophisticated, and makes use of Brouwer's fixed point theorem. We present an alternative, conceptually simpler, proof due to John Nash [**Nas51**]. Although the this proof predates the theory of dynamic programming [**Bel57**], it is interesting to notice that the proof uses a strategy improvement function and its properties. Before we present this strategy improvement function, we need to define a few concepts.

For all $1 \leq i \leq m$, let us define the function $\mathsf{Adv}^i_{\mathrm{Max}} : \Sigma_{\mathrm{Min}} \times \Sigma_{\mathrm{Max}} \to \mathbb{R}$ in the following way:

$$\mathsf{Adv}^i_{\mathrm{Max}}(\mu, \chi) = \begin{cases} \mathbb{P}(\mu, \chi^i) - \mathbb{P}(\mu, \chi) & \text{if this quantity is positive,} \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, the function $\mathsf{Adv}^i_{\mathrm{Max}}$ gives the the advantage to player Max if he chooses to play pure strategy χ^i against player Min's strategy μ instead of playing strategy χ. Similarly we define $\mathsf{Adv}^i_{\mathrm{Min}}$ for $1 \leq i \leq n$ as

$$\mathsf{Adv}^i_{\mathrm{Min}}(\mu, \chi) = \begin{cases} \mathbb{P}(\mu, \chi) - \mathbb{P}(\mu^i, \chi) & \text{if this quantity is positive,} \\ 0 & \text{otherwise.} \end{cases}$$

Let us define a mapping Improve : $\Sigma_{\mathrm{Min}} \times \Sigma_{\mathrm{Max}} \to \Sigma_{\mathrm{Min}} \times \Sigma_{\mathrm{Max}}$ in the following manner. If $(\mu', \chi') = \mathrm{Improve}(\mu, \chi)$ then for all $1 \leq i \leq m$ we have that

$$\chi'(i) = \frac{\chi(i) + \mathsf{Adv}^i_{\mathrm{Max}}(\mu, \chi)}{1 + \sum_{i=1}^m \mathsf{Adv}^i_{\mathrm{Max}}(\mu, \chi)};$$

and for all $1 \leq i \leq n$ we have that

$$\mu'(i) = \frac{\mu(i) + \mathsf{Adv}^i_{\mathrm{Min}}(\mu, \chi)}{1 + \sum_{i=1}^n \mathsf{Adv}^i_{\mathrm{Min}}(\mu, \chi)}.$$

The following proposition together with Lemma C.1.4 proves the determinacy of Matrix games.

PROPOSITION C.1.3. Matrix games are determined if and only if there exists a fixed point for the strategy improvement function Improve.

PROOF. For the "if" part, suppose that (μ_*, χ_*) is a fixed point for function Improve. We first show that there exists a positive integer $i \leq n$ such that $\mu_*(i) \geq 0$ and $\mathsf{Adv}^i_{\mathrm{Min}}(\mu_*, \chi_*) = 0$. By definition, we have that $\mathbb{P}(\mu_*, \chi_*) = \sum_{i=1}^n \mu_*(i) \cdot \mathbb{P}(\mu^i_*, \chi_*)$. It is easy to see that $\mathbb{P}(\mu_*, \chi_*) < \mathbb{P}(\mu^i_*, \chi_*)$ can not be true for all i having $\mu_*(i) > 0$.

Now let us assume that $i \leq n$ is such that $\mu_*(i) \geq 0$ and $\mathbb{P}(\mu^i_*, \chi_*) \geq \mathbb{P}(\mu_*, \chi_*)$, i.e. $\mathsf{Adv}^i_{\mathrm{Min}}(\mu_*, \chi_*) = 0$. Since (μ_*, χ_*) is a fixed point, we have the following:

$$\mu_*(i) = \frac{\mu_*(i) + \mathsf{Adv}^i_{\mathrm{Min}}(\mu_*, \chi_*)}{1 + \sum_{i=1}^n \mathsf{Adv}^i_{\mathrm{Min}}(\mu_*, \chi_*)} = \frac{\mu_*(i)}{1 + \sum_{i=1}^n \mathsf{Adv}^i_{\mathrm{Min}}(\mu_*, \chi_*)}$$

It implies that $\sum_{i=1}^{n} \mathsf{Adv}^i_{\mathrm{Min}}(\mu_*, \chi_*) = 0$, and since by definition $\mathsf{Adv}^i_{\mathrm{Min}}(\mu_*, \chi_*)$ can not be negative, it implies that for all $1 \leq i \leq n$ we have that $\mathsf{Adv}^i_{\mathrm{Min}}(\mu_*, \chi_*) = 0$. Thus no positional strategy of Min player is better than μ_* against χ_*. Similarly we can prove that no positional strategy of Max is better than χ_* against μ_*. Hence (μ_*, χ_*) are optimal strategies.

For the "only if" part suppose that matrix games are determined, it means that there exists a pair of strategies (μ_*, χ_*) with the property that no positional strategy of Min is better than μ_* against χ_*, and vice-versa. Hence for all $1 \leq i \leq n$ we have $\mathsf{Adv}^i_{\mathrm{Min}}(\mu_*, \chi_*) = 0$. Similarly we can show that for all $1 \leq i \leq m$ we have $\mathsf{Adv}^i_{\mathrm{Max}}(\mu_*, \chi_*) = 0$. By definition of Improve function, it is clear that (μ_*, χ_*) is a fixed point. ∎

LEMMA C.1.4. There exists a fixed point of the function Improve.

PROOF. From Proposition C.1.1 the set $\Sigma_{\mathrm{Min}} \times \Sigma_{\mathrm{Max}}$ is a nonempty, compact (closed and bounded), and convex set, and Improve is a continuous function, the existence of a fixed point for follows from Theorem B.3.2 (Brouwer's fixed point theorem). ∎

In a later result, von Neumann generalised the result of determinacy of Matrix games to the following *minimax theorem*.

THEOREM C.1.5 (von Neumann's minimax theorem **[vN37]**)**.** Let X and Y be nonempty, convex subsets of Euclidean spaces, and $\mathbb{P} : X \times Y \to \mathbb{R}$ be a continuous function. If $\mathbb{P}$ is quasiconcave on X and quasiconvex on Y then

$$\sup_{x \in X} \inf_{y \in Y} \mathbb{P}(x, y) = \inf_{y \in Y} \sup_{x \in X} \mathbb{P}(x, y).$$

C.2. Stopping Stochastic Games

Stochastic games, introduced by Lloyd Shapley **[Sha53]**, are a generalisation of repeated matrix games. A stochastic game Γ is played, by two players Max and Min, on a finite graph with vertices $V = \{v_1, v_2, \ldots, v_N, v_{N+1} = s\}$, where s is a special vertex called the *stop vertex*. A matrix game M_k is associated with each vertex $v \in V \setminus \{s\}$.

The game start with a token in some vertex v_k other than s. Players Max and Min choose a row i and a column j, respectively, of the matrix M_k. As a result player Max wins $M(i, j)$ from player Min, and the token is moved to the vertex v_ℓ with probability $P(v_\ell | v_k, i, j)$. Note that the following restrictions are imposed on the probabilistic transition function P.

$$\begin{aligned} P(s|v_k, i, j) &> 0, \text{ for all } i, j, \text{ and } 1 \leq k \leq N, \\ \sum_{\ell=1}^{N+1} P(v_\ell|v_k, i, j) &= 1 \text{ for every } 1 \leq k \leq N, \text{ and} \\ P(s|s, i, j) &= 1, \text{ for all } i, j. \end{aligned} \tag{C.2.1}$$

If the token is moved to the stop vertex then the game terminates. Otherwise players play the Matrix game associated with the new vertex, player Max receives the payoff and token is moved to a new vertex according to the probabilistic transition function P.

The game continues in this fashion until the token reaches the stop vertex. Note that restriction (C.2.1) ensures that the game reaches the stop vertex in finitely many steps. We denote the game starting from vertex v_k by Γ^k, for all $1 \leq k \leq N$. To solve a stochastic game Γ^k, we need to compute the optimal strategies for the players and value of the game $\mathsf{Val}(\Gamma^k)$, if exists. Shapley showed that the value of every stochastic game always exists.

THEOREM C.2.1 (Determinacy theorem for stochastic games **[Sha53]**)**.** Every stochastic game is determined.

To prove determinacy of the stochastic games, Shapley designed the following set of optimality equations $\mathrm{Opt}(\Gamma)$, whose solution, if exists, gives a solution of the stochastic games (Lemma C.2.2). For all $1 \leq k \leq N$ we have:

$$\mathcal{V}_k \quad = \quad \mathsf{Val}\left(M_k(i,j) + \sum_{\ell=1}^{N} P(v_\ell | v_k, i, j) \cdot \mathcal{V}_\ell \right), \tag{C.2.2}$$

where $\mathsf{Val}(M_k(i,j) + \sum_{\ell=1}^{N} P(v_\ell|v_k,i,j) \cdot \mathcal{V}_\ell)$ denotes the value of a matrix game whose matrix has $(i,j)^{th}$ entry as $M_k(i,j) + \sum_{\ell=1}^{N} P(v_\ell|v_k,i,j) \cdot \mathcal{V}_\ell$. In our terminology, we say that a function $F : V \to \mathbb{R}$ satisfies the optimality equations and we write $F \models \mathrm{Opt}(\Gamma)$ if substituting $\mathcal{V}_k$ by $F(v_k)$ gives equality in all of the equations.

The following lemma states that a solution of optimality equations $\mathrm{Opt}(\Gamma)$ gives value of the game for all starting states, in a uniform manner, and optimal positional strategies for both players.

LEMMA C.2.2. **[Sha53]** If $F \models \mathrm{Opt}(\Gamma)$ then for all $1 \leq i \leq N$ the value of the game Γ^k exists and is equal to $F(k)$. Moreover there exist optimal positional (mixed) strategies.

The value improvement function Improve : $\mathbb{R}^N \to \mathbb{R}^N$ is defined as follows:

$$\mathrm{Improve}(F)(k) = \mathsf{Val}\left(M_k(i,j) + \sum_{\ell=1}^{N} P(v_\ell|v_k,i,j) \cdot F(\ell) \right), \text{ for all } 1 \leq k \leq N.$$

PROPOSITION C.2.3 (**[Sha53]**)**.** The value improvement function Improve : $\mathbb{R}^N \to \mathbb{R}^N$ is a contraction.

The proof of Theorem C.2.1 follows from Lemma C.2.2 and the following lemma.

LEMMA C.2.4. For every stochastic game Γ a solution of $\mathrm{Opt}(\Gamma)$ always exists.

PROOF. It is straightforward to verify that a fixed point of the Improve function gives the solution of the optimality equations $\mathrm{Opt}(\Gamma)$. From Proposition C.2.3 we know that Improve is a contraction over the metric space $(\mathbb{R}^N, \|\cdot\|_\infty)$. It follows from the Banach's fixed point theorem B.3.1 that fixed point of Improve function exists, and can be approximated using a straightforward value iteration algorithm. ∎

D

Implementation Details

D.1. Lexer

```
%{
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "parser.hh"
%}
alpha     [a-zA-Z]
alphanum  [a-zA-Z0-9_]
white     [ \t]+
digit     [0-9]
integer   {digit}+
iden      {alpha}{alphanum}*
%%
{white}  { /* We ignore white characters */ }
system    return SYSTEM;
begin     return IBEGIN;
end       return END;
automaton return AUTOMATON;
locations_x return LOC_X;
locations_n return LOC_N;
clocks    return CLOCKS;
actions   return ACTIONS;
invar     return INVAR;
trans     return TRANS;
init      return INIT;
final     return FINAL;
"("       return LEFT;
")"       return RIGHT;
"["       return SLEFT;
"]"       return SRIGHT;
"{"       return CLEFT;
"}"       return CRIGHT;
";"       return SEMI;
```

```
":"        return COLON;
","        return COMMA;
"-"        return MINUS;
"="        return EQ;
"<"        return LT;
">"        return GT;
"<="       return LE;
">="       return GE;
{iden} {
  yylval.id = strdup(yytext);
  return(IDENT);
}
{integer} {
    yylval.num=atoi(yytext);
    return(INT);
       }
```

D.2. Parser

In the following Bison code, we have removed all the C++ programming lines of code to build a model of the timed game automata.

```
%{
%token <num>INT
%token <id>IDENT
%token SYSTEM IBEGIN END AUTOMATON LOC_X LOC_N CLOCKS
%token ACTIONS INVAR TRANS INIT FINAL LEFT RIGHT
%token SLEFT SRIGHT CLEFT CRIGHT SEMI COLON
%token COMMA MINUS EQ LT GT LE GE

top : SYSTEM IDENT IBEGIN automata END
    ;

automata : automata automaton
         | automaton
         ;

automaton : AUTOMATON IDENT IBEGIN automaton_body END
          ;

automaton_body : declarations invar_transition_list init_final_list
               | declarations invar_transition_list
               ;
```

```
declarations : declarations locations
             | declarations clocks
             | declarations actions
             |
             ;

locations : LOC_X COLON CLEFT loc_x_list CRIGHT SEMI
          |  LOC_N COLON CLEFT loc_n_list CRIGHT SEMI
          ;

loc_x_list : IDENT COMMA loc_x_list
           | IDENT
           ;

loc_n_list : IDENT COMMA loc_n_list
           | IDENT
           ;

actions : ACTIONS COLON CLEFT act_list CRIGHT SEMI
        ;

act_list : IDENT COMMA act_list
         | IDENT
         ;

clocks : CLOCKS COLON clock_list SEMI
       ;

clock_list : IDENT SLEFT INT SRIGHT COMMA clock_list
           | IDENT SLEFT INT SRIGHT
           ;


invar_transition_list : transition invar_transition_list
                      | invars invar_transition_list
                      | transition
                      | invars
                      ;

transition : TRANS LEFT IDENT COMMA IDENT COMMA IDENT COMMA
              CLEFT cons_list CRIGHT COMMA
              CLEFT reset_list CRIGHT RIGHT SEMI
           ;
```

```
invars : INVAR LEFT IDENT COMMA
          CLEFT cons_list CRIGHT RIGHT SEMI
       ;

cons_list : cons_list COMMA cons
          | cons
          |
          ;

cons : IDENT GT INT
     | IDENT LT INT
     | IDENT EQ INT
     | IDENT LE INT
     | IDENT GE INT
     | IDENT MINUS IDENT GT INT
     | IDENT MINUS IDENT LT INT
     | IDENT MINUS IDENT EQ INT
     | IDENT MINUS IDENT LE INT
     | IDENT MINUS IDENT GE INT
     ;

reset_list :IDENT COMMA reset_list
           | IDENT
           |
           ;

init_final_list : init init_final_list
                | final init_final_list
                |
                ;

init : INIT LEFT IDENT COMMA
        CLEFT cons_list CRIGHT RIGHT SEMI
     ;

final : FINAL LEFT IDENT COMMA
         CLEFT cons_list CRIGHT RIGHT SEMI
      ;
```

Bibliography

[ABM04] R. Alur, M. Bernadsky, and P. Madhusudan. Optimal reachability for weighted timed games. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3142 of *LNCS*, pages 122–133. Springer, 2004.

[ACH97] R. Alur, C. Courcoubetis, and T. A. Henzinger. Computing accumulated delays in real-time systems. *Formal Methods in System Design*, 11(2):137–155, 1997.

[AD90] R. Alur and D. Dill. Automata for modeling real-time systems. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 443 of *LNCS*, pages 322 – 335. Springer, 1990.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[ALTP01] R. Alur, S. La Torre, and G. J Pappas. Optimal paths in weighted timed automata. In *International Workshop on Hybrid Systems: Computation and Control (HSCC)*, pages 49 – 62. Springer-Verlag, 2001.

[ALTP04] R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. *Theoretical Computer Science*, 318(3):297–322, 2004.

[Alu91] R. Alur. *Techniques for Automatic Verification of Real-Time Systems*. PhD thesis, Standford University, August 1991.

[AM99] E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In F. W. Vaandrager and J. H. van Schuppen, editors, *International Workshop on Hybrid Systems: Computation and Control (HSCC)*, volume 1569 of *LNCS*, pages 19–30. Springer-Verlag, 1999.

[AM01] Y. Abdeddaïm and O. Maler. Job-shop scheduling using timed automata. In G. Berry, H. Comon, and A. Finkel, editors, *International Conference on Computer Aided Verification (CAV)*, volume 2102 of *LNCS*, pages 478–492, Heidelberg, 2001. Springer.

[AM04] R. Alur and P. Madhusudan. Decision problems for timed automata: A survey. *Formal Methods for the Design of Real-Time Systems*, 3185:1–24, 2004.

[AMP95] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In P. Antsaklis, W. Kohn, A. Nerode, and Sastry S., editors, *Hybrid Systems II*, 1995.

[AMPS98] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In P. Antsaklis, W. Kohn, A. Nerode, and Sastry S., editors, *IFAC Symposium on System Structure and Control*, 1998.

[BBBR07] P. Bouyer, T. Brihaye, V. Bruyère, and J. Raskin. On the optimal reachability problem on 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 *International Workshop on Hybrid Systems: Computation and Control (HSCC)*, volume 2993 of *LNCS*, pages 203–218. Springer, 2004.

[BBL08] P. Bouyer, E. Brinksma, and K. G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 32(1):3–23, 2008.

[BBM06] P. Bouyer, T. Brihaye, and N. Markey. Improved undecidability results on weighted timed automata. *Information Processing Letters*, 98:188–194, 2006.

[BBR05] Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On optimal timed strategies. In *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 3829 of *LNCS*, pages 49–64. Springer, 2005.

[BCFL04] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 3328 of *LNCS*, pages 148–160. Springer, 2004.

[BCFL05] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Synthesis of optimal strategies using hytech. In *Workshop on Games in Design and Verification (GDV)*, volume 119 of *Electronic Notes Theoretical Computer Science*, pages 11–31. Elsevier, 2005.

[Bel57] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.

[Bel84] R. Bellman. *Eye of the Hurricane: an Autobiography*. World Scientific, Singapore, 1984.

[Ber95] D. P. Bertsekas. *Dynamic Programming and Optimal Control (Volume 1)*. Athena Scientific, Belmont, MA, 1995.

[Ber01] D. P. Bertsekas. *Dynamic Programming and Optimal Control (Volume 2)*. Athena Scientific, Belmont, MA, second edition, 2001. First edition was published in 1995.

[BFH+01] G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. W. Vaandrager. Minimum-cost reachability for priced timed automata. In M. D. Di Benedetto and A. L. Sangiovanni-Vincentelli, editors, *International Workshop on Hybrid Systems: Computation and Control (HSCC)*, volume 2034 of *LNCS*, pages 147–161, Heidelberg, 2001. Springer.

[BFHL01] G. Behrmann, A. Fehnker, T. Hune, and K. G. Larsen. Efficient guiding towards cost-optimality in UPPAAL. In *International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 2031 of *LNCS*, pages 174–188. Springer, 2001.

[BGS00] S. Bornot, G. Gössler, and J. Sifakis. On the construction of live timed systems. In *International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 1785 of *LNCS*, pages 109 – 126. Springer-Verlag, 2000.

[BHPR07] T. Brihaye, T. A. Henzinger, V. S. Prabhu, and J. Raskin. Minimum-time reachability in timed games. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of *LNCS*, pages 825–837. Springer, 2007.

[BLMR06] P. Bouyer, K. L. Larsen, N. Markey, and J. L. Rasmussen. Almost optimal strategies in one clock priced timed games. In S. Arun-Kumar and N. Garg, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 4337 of *LNCS*, pages 345 –356. Springer, 2006.

[BNO03] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, Belmont, MA, 2003.

[Bou06] P. Bouyer. Weighted timed automata: Model-checking and games. In *Annual Conference on Mathematical Foundations of Programming Semantics (MFPS)*, volume 158 of *Electronic Notes Theoretical Computer Science*, pages 3–17, 2006.

[Bou09] P. Bouyer. *From Qualitative to Quantitative Analysis of Timed Systems*. Mémoire d'habilitation, Université Paris 7, Paris, France, January 2009.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

[CHR02] F. Cassez, T. Henzinger, and J. F. Raskin. A comparison of control problems for timed and hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control (HSCC)*, volume 2289 of *LNCS*, pages 134–148. Springer-Verlag, 2002.

[CY92] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. In *Formal Methods in Computer Science*, volume 1, pages 385–415, Dordrecht, 1992. Kluwer.

[dA03] L. de Alfaro. Quantitative verification and control via the mu-calculus. In *International Conference on Concurrency Theory (CONCUR)*, volume 2761 of *LNCS*, pages 102–126. Springer, 2003.

[dAFH+03] L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *International Conference on Concurrency Theory (CONCUR)*, volume 2761 of *LNCS*, pages 144–158, 2003.

[EEJ04] K. Eriksson, D. Estep, and C. Johnson. *Applied Mathematics: Body and Soul Volume 1: Derivatives and Geometry in* $\mathbb{R}^3$. Springer, 2004.

[FH72] J. E. Falk and J. L. Horowitz. Critical path problems with concave cost-time curves. *Management Science*, 19(4):446–455, 1972.

[FV97] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1997.

[GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games. A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

[HK99] T. A. Henzinger and P. W. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221(1-2):369–392, 1999.

[HMP92] T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 623 of *Lecture Notes in Computer Science*, pages 545–558, London, UK, 1992. Springer-Verlag.

[HNSY92] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:394–406, 1992.

[How60] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.

[HWT92] G. Hoffmann and H Wong-Toi. The input-output control of real-time discrete event systems. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 256–265, 1992.

[JLS07] M. Jurdziński, F. Laroussinie, and J. Sproston. Model checking probabilistic timed automata with one or two clocks. In *International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 4424 of *LNCS*, pages 170 –184. Springer, 2007.

[JT07] M. Jurdziński and A. Trivedi. Reachability-time games on timed automata. In L. Arge, C. Cachin, T. Jurdziński, and A. Tarlecki, editors, *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of *LNCS*, pages 838–849. Springer, 2007.

[JT08a] M. Jurdziński and A. Trivedi. Average-time games. In R. Hariharan, M. Mukund, and V. Vinay, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 08004 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2008. `http://drops.dagstuhl.de/opus/volltexte/2008/1765`.

[JT08b] M. Jurdziński and A. Trivedi. Concavely-priced timed automata. In F. Cassez and C. Jard, editors, *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 5215 of *LNCS*, pages 48–62. Springer, 2008.

[KK01] M. A. Khamsi and W. A. Kirk. *An introduction to metric spaces and fixed point theory*. Wiley-IEEE, 2001.

[KNSS99] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. In *AMAST Workshop on Real-time Systems (ARTS)*, pages 75–95, 1999.

[KPSS99] Y. Kesten, A. Pnueli, J. Sifakis, and Yovine. S. Decidable integration graphs. *Information and Computation*, 150(2):209–243, 1999.

[LBB+01] K. G. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson, and J. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In G. Berry, H. Comon, and A. Finkel, editors, *International Conference on Computer Aided Verification (CAV)*, volume 2102 of *LNCS*, pages 493–505, Heidelberg, 2001. Springer.

[LBB+04] K. G. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson, and J. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *International Workshop on Hybrid Systems: Computation and Control (HSCC)*, volume 2993 of *LNCS*, pages 203–218. Springer, 2004.

[LMS04] F. Laroussinie, N. Markey, and P. Schnoebelen. Model checking timed automata with one or two clocks. In Philippa Gardner and Nobuko Yoshida, editors, *International Conference on Concurrency Theory (CONCUR)*, volume 3170 of *LNCS*, pages 387–401, London, UK, August 2004. Springer.

[LTMM02] S. La Torre, S. Mukhopadhyay, and A. Murano. Optimal-reachability and control for acyclic weighted timed automata. *Theoretical Computer Science*, 223:485–497, 2002. IFIP Conference Proceedings.

[Man69] O. L. Mangasarian. *Nonlinear Programming*. McGraw-Hill Series in Systems Science. McGraw-Hill, New York, 1969.

[Mar65] B. Martos. The direct power of adjacent vertex programming methods. *Management Science*, 12(3):241–252, 1965.

[Mar75] D. A. Martin. Borel determinacy. *Annals of Mathematics*, 102:363–371, 1975.

[Mar98] D. A. Martin. The determinacy of Blackwell games. *Journal of Symbolic Logic*, 63(4):1565–1581, 1998.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 900 of *LNCS*, pages 229–242. Springer-Verlag, 1995.

[Nas51] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[NTY00] P. Niebert, S. Tripakis, and S. Yovine. Minimum-time reachability for timed automata. In *IEEE Mediteranean Control Conference (MED)*, Los Alamitos, 2000. IEEE Comp. Soc. Press.

[Put94] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[Roh99] C. H. Rohwer. The babylonian method and higher order approximation to square roots. In *The Delta 99 Symposium on undergraduate Mathematics*, pages 183–188, Queensland, Australia, 1999.

[RW89] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. In *IEEE*, volume 77, pages 81–98, 1989.

[Sha53] L. S. Shapley. Stochastic games. *Proc. Nat. Acad. Sci. U.S.A.*, 39:1095–1100, 1953.

[Tho95] W. Thomas. On the synthesis of strategies in infinite games. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 900 of *LNCS*, pages 1–13. Springer, 1995.

[Tri99] S. Tripakis. Verifying progress in timed systems. In *AMAST Workshop on Real-time Systems (ARTS)*, pages 299 – 314. Springer-Verlag, 1999.

[UPP] Uppaal. `http://www.uppaal.com/`.

[VJ00] J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games (Extended abstract). In *International Conference on Computer Aided Verification (CAV)*, volume 1855 of *LNCS*, pages 202–215. Springer, 2000.

[vN28] J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[vN37] J. von Neumann. Über ein ökonomisches gleichungssystem und eine verallgemeinerung des brouwerschen fixpunktsatzes. *Ergebn. Math. Koloq. Wein*, 8:73–83, 1937.

[Wik09] Wikipedia. Life-critical system — wikipedia, the free encyclopedia, 2009. [Online; accessed 19-April-2009].

[WTH92] H Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. Technical report, Stanford University, 1992.

Index