The Controller Synthesis Problem: Implementability and robustness

Thèse d'Habilitation à Diriger des Recherches

École Doctorale MSTIC
Spécialité Informatique
Unité de recherche 4219

Présentée et soutenue publiquement
à Créteil,
le 28 / 11 / 2024,
par

Youssouf Oualhadj

Composition du jury:

Résumé :

L'étude algorithmique de l'existence de contrôleurs optimaux pour une spécification donnée a pour but de pouvoir obtenir, de manière automatisée, des programmes corrects pour le contrôle de systèmes complexes. Outre le défi de prouver l'existence d'un tel contrôleur et d'un algorithme le construisant, un défi à ne pas négliger est de réaliser l'implémentation du contrôleur obtenu. Il est facile de trouver, dans la littérature adéquate, des exemples de modèles pour lesquels un contrôleur optimal nécessite une mémoire infinie pour pouvoir être implémenté, ce qui n'est pas réaliste. Si l'on se place dans un cadre d'étude de systèmes à temps réel, certains modèles ont besoin de contrôleurs nécessitant une précision d'horloge infinie. Le problème de *l'implémentabilité* a pour objectif d'étudier l'existence de contrôleurs optimaux et physiquement implémentables.

Dans ce manuscrit, je reviens sur des travaux et contributions liés à ce problème.

Abstract:

The controller synthesis problem lays the ground for a formal toolbox allowing a "correct by design" paradigm. Nevertheless bringing the theory and application together implies at the least the possibility of a realistic physical implementation of the designed controller. However, one has to keep in mind that this controller is nothing but a mathematical idealization of a correct solution in where (some) physical constraints are often abstracted away. For instance, the obtained controller does not have memory constraint, nor it is concerned with energy consumption, or hardware limitation such as clock speed. Sometimes, the time guarantees brought by the computed solution are simply not realistic. For example, in the case where a specification requires repeated behavior, a controller inducing a behavior where each repetition requires arbitrary longer period of time to take place is deemed correct. In this case, the controller can be implemented but its behavior is still not completely satisfying. This comes from the fact that usually the specification formalism used allows, from a mathematical stand point, such behaviors. One can obviously say it suffices to use a more precise specification, i.e., a language where such phenomena are ruled out. It turns out that this is not a simple task. Indeed, one has to bear in mind that we are aiming for an automatic procedure, thus decidability (and tractability) is paramount. Therefore, fine tuning the specification language while maintaining algorithmic properties is an ongoing challenge.

In this manuscript we present several contributions that fall in the scope of automatically designed implementable controllers.

Contents

Introduction

An open reactive system is a term used to describe cyber-physical systems involving interaction between multiple entities that are either controllable (the controller) or uncontrollable (the environment). When designing such a system, a natural requirement to ask for is to ensure that it runs according to a desired behavior without mischief. This requirement can be formulated either as a model-checking problem or a controller synthesis problem. In the former case, the question asked is whether any run of the system behaves well, i.e. is the system bug-free? In the latter case, the question asked is weather it is possible to "control the system" without changing the actions of the environment such that the obtained behavior is correct, i.e., enforce the system to be well-behaved.

The mathematical interpretation of the controller synthesis, also known as Church synthesis problem, is usually modeled by a game played on a graph. Such a game involves two players, the first one is called $\mathcal{P}_1$ (for player one) represents the controller and the second one is called $\mathcal{P}_2$ and represents the environment. The set of states of the game graph is divided into two disjoint subsets. The first one belongs to $\mathcal{P}_1$ and the second belongs to $\mathcal{P}_2$. A play is an infinite path in the game graph, it is obtained using the following interaction between $\mathcal{P}_1$ and $\mathcal{P}_2$. Let s_{ini} be a fixed initial state s_{ini} in the game graph. The player controlling this state chooses a successor s of s_{ini} such that (s_{ini}, s) forms an edge in the graph, the play moves to this newly chosen state and the player controlling s picks a new state from all the possible successors of s. This turn-based interaction is repeated infinitely creating an infinite play in the graph (assuming that all the states always have at least a successor). The players rely on a strategy for picking the new state of the play, a strategy is simply a mapping from finite plays to states. The behavior we want $\mathcal{P}_1$ to ensure, is called the objective, it is given by a set of infinite path over the graph of the game. $\mathcal{P}_1$ aims at enforcing the objective, and $\mathcal{P}_2$ aims at falsifying it. Using this game theoretic translation, solving the controller synthesis problem amounts to checking whether $\mathcal{P}_1$ has a strategy that allows him to ensure objective against any strategy of $\mathcal{P}_2$. A strategy of $\mathcal{P}_1$ is called winning if it ensures the objective against any strategy of $\mathcal{P}_2$. This winning strategy is a mathematical formulation of a correct controller.

This game theoretic metaphor lays the ground for a nice formal framework of a "correct by design" system. Nevertheless bringing the theory and application together implies at the least the possibility of a realistic physical implementation of the designed controller. But one has to keep in mind that this controller is nothing but a mathematical idealization of a correct solution in where (some) physical constraints are often abstracted away. For instance, the obtained controller does not have memory constraint, nor it is concerned with energy consumption, neither hardware limitation such as clock speed. Sometimes, the time guarantees brought by the computed solution are simply not realistic. For example, in the case where a specification requires repeated behavior, a controller inducing a behavior where each repetition requires arbitrary longer period of time to take place is deemed correct. In this case, the controller can be implemented but its behavior is still not completely satisfying. This comes from the fact that usually the specification formalism used allows, from a mathematical stand point, such behaviors. One can obviously say it suffices to use a more precise specification language where such phenomena are ruled out. It turns out that this is not a simple task. Indeed, one has to bear in mind that we are aiming for an automatic procedure, thus decidability (and tractability) is paramount. Therefore, fine tuning the specification language while maintaining algorithmic properties is an ongoing challenge.

Motivated by these observations, several line of research have taken up the task of developing formal tools investigating the theory and algorithms for implementable controllers. Recall that a correct controller can be acquainted with a winning strategy in some game. Thus to understand implementability issues of correct controllers, a natural starting point is to investigate how complex winning strategies need to be with respect to a given specification. When thinking about strategy complexity, one usually thinks about memory requirements, e.g., [GZ05, Kop06, FHKM15, AR17, RPR18]. Among these related work we highlight the results from [GZ05] where a full characterization of memoryless specifications is obtained. However, the complexity of the cyber-physical systems to model keeps increasing giving rise to more complex specifications where memoryless controllers are simply not sufficient [CHP07, CD12a, CRR14, VCD$^+$15a, JLS15a]. Extending the understanding of memory requirements in optimal strategies will be the focus the Chapter 1. We show the limits of the techniques developed in [GZ05] and generalize their approach to handle memory optimal strategies and stochastic arenas.

An other aspect that we already mentioned is the specification itself. Increasing the accuracy of the specification while maintaining good algorithmic properties has already been explored, for example introducing a timing frame to obtain more predictable behavior in long term specifications using the so-called "window mechanism" was introduced in [CDRR15, BHR16a], but these new specifications were only considered in the context of games and never in the context of stochastic arenas. This is what we present in Chapter 2, where we develop a unified approach extending the results of both paper to the context of Markov decision processes.

In Chapter 3, we shift the focus to real-time systems. Here, the formalism of choice is the one of timed automata. This consists of an abstract mathematical semantics offering arbitrarily precise clocks and time delays. However, real-world digital systems have response times that may not be negligible, and control software cannot ensure timing constraints exactly, but only up to some error caused by clock imprecision, measurement errors, and communication delays. A major challenge is thus to ensure that the synthesized control software is robust, i.e., ensures the specification even in the presence of imprecision [HS06].

Following these shortcomings, there has been a growing interest in lifting the theory of verification and synthesis to take into account robustness. Model-checking problems were re-visited by considering an unknown perturbation parameter to be synthesized for several kinds of properties [Pur00, DDMR08, BMR08, BMS13]. However, little has been achieved in the setting of timed game. Actually, due to the infinitely precise abstract semantics, synthesized strategies may not be realizable in a finitely-precise environment; the controlled systems obtained using the timed game formalism may not satisfy the seeked properties at all. In particular, due to perturbations in the time delays, some infinite behaviors may disappear completely.

The above problem is formalized as the study of the existence of robust strategies in timed games, namely, those that guarantee winning despite an imprecision bounded by a parameter.

Finally in chapter 4 we study multi-player games. So far we considered a mathematical framing where the controller and the system are adversaries. This entails a pessimistic models and rather conservative solutions. In an effort to circumvent this (contrived) antagonistic assumptions, Ummels [Umm08] considered the synthesis problem in a setting where each agent is assigned an individual specification together with a list of the specification that "must-hold". In this case, one aims at constructing a global controller that ensures a rational behavior for the global system such that its outcome ensures the "must-hold" specifications. Usually rationality is captured with the concept of Nash equilibrium. A well known issue with this model is that the proposed solution is nothing but a suggestion. For instance, in an equilibrium, a player might be bound to loose, hence he does not have any clear interest in following the proposed rational solution. This lack of robustness in the concept of Nash equilibria has drawn the attention in the community where refinement were proposed aiming at designing solutions with more guarantees [CHJ04, FKL10, CDFR14, BMR14]. In chapter 4, we propose a setting where the player have their own selfish objective but also have to share a common resource. We extend the setting of [FKL10] to the setting with common resources, we develop a framework where a rational solutions that maintain the common resource are preferred over the ones depleting it. this way, we view solutions where common resources are saved as more robust solutions.

In this manuscript we present several contributions that fall in the scope of the context presented above.

Chapter 1

Memory for synthesis

1.1 Outline of the chapter

In this chapter we are interested in the memory requirement of a correct strategy. Our main objective is to understand when finite memory strategy are sufficient, with respect to a given specification. We start with a very simple example showcasing what we intend by the notion of (finite) memory strategy.

Example 1.1. Consider the (one-player) arena depicted in Figure 1.1. In this example, from every state the player decides the next edge to pick, inducing an infinite sequence of vertices. We want to design a control policy that enforces an infinite amount of visits of state b ***while*** ensuring that $\liminf$ of the average accumulation is non-negative. Notice that this specification consists of two goal; i) visiting b infinitely often, and ii) the $\liminf$ condition. In order to ensure this specification, a correct controller has to take transition (a, b) from time to time to ensure the first part of the specification. But at the same time it has to spend long periods of time in state a to ensure that the average accumulation asymptotically goes to 0. In order to combine these two behaviors and ensure the full specification, our controller has to create long and consecutive visits of state a followed by one visit of state b followed by longer sequences of state a and repeat this behavior infinitely. Implementing this behavior requires from the controller to store the required number of visits of state a. Now, notice how this quantity grows unbounded. This causes the memory needed by the controller to be ***unbounded***. We refer to such memories as infinite. It is easy to see that bounding the number of visits of state a will cause the inferior limit to be negative. $\triangleleft$

The above example shows a very simple instance where the correct behavior can only be enforced by an infinite memory, but such a strategy cannot be implemented in a real-life scenario since the memory space is a finite resource.

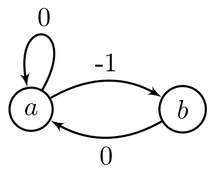


Figure 1.1: Infinite memory is unavoidable.

In this chapter we describe a better understanding of when a system is controllable by a finite memory policy. As usual we will address the control problem through the game theoretic prism. A system will be viewed as a game played between two competitive entities. The first one models the controller, and the second models the environment. The control policy we build is referred to as an optimal (or winning) strategy. In this context we want to identify the frontier between when a finite memory optimal strategy exists and when it does not.

We base our presentation on a characterization by Gimbert and Zielonka [GZ05]. In their work, a characterization of all the preference relations that admit memoryless optimal strategies is presented. Their approach is built upon necessary and sufficient conditions. These conditions are agnostic to the game graph. Thus from a theoretic point view it allows one to pinpoint exactly what properties of a preference relation are required to get memoryless determinacy, i.e. existence of memoryless optimal strategies for both players. Moreover, not only they characterize the existence of memoryless optimal strategies, they also design a mathematical tool to design them through what they called the lifting corollary. This tool allows one to bring the above problem to the simpler setting of graphs and obtain results in the setting of two-player games.

In this chapter, we overview contributions obtained from a collaboration with Patricia Bouyer, Stéphane Le Roux, Mickael Randour, and Pierre Vandenhove [BRO+20, BORV21, BRO+22]. The main contributions are:

- We show that the lifting corollary as intended in [GZ05] does not hold when memory is involved, cf. the game of Figure 1.8;
- we introduce the notion of arena-independent memory through the concept of a memory skeleton, cf. Definition 1.16;
- we generalize the characterization of [GZ05] to the more complex setting of arena-independent memory strategies;
- we show that the lifting corollary holds for arena-independent memory strategies;
- we extend our understanding to the case of stochastic arenas.

This chapter is organized as follows:

- In Section 1.2, we present the formalism of two-player games together with the necessary formal notions and notations.
- In Section 1.3, we present related work from the literature regarding existing characterization.
- In Section 1.4, we preview our results regarding the characterization of preference relation that are finite memory determined. In particular we show how we generalize the techniques from Gimbert and Zielonka. The results previewed in this section are:
  - A generalization of the central notions of monotony and selectivity.
  - A characterization of arena-independent finite memory determined preference relations.
  - A lifting corollary in the context of arena-independent finite memory strategies.
- In Section 1.5, we present extensions to stochastic arenas in particular we:
  - Show a counter part to the lifting corollary.
  - We also show a characterization of preference relations admitting arena-independent finite memory in one-player arenas.

1.2 Two-player games

Arenas and plays

We consider games played between two players; player 1 ($\mathcal{P}_1$) and player 2 ($\mathcal{P}_2$). A two-player arena or arena for short is a tuple $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E})$ such that $\mathsf{S} = \mathsf{S}_1 \uplus \mathsf{S}_2$ is a finite set of states partitioned into states of $\mathcal{P}_1$, i.e., S_1 and states of $\mathcal{P}_2$, i.e., S_2, and $\mathsf{E} \subseteq \mathsf{S} \times \mathsf{S}$ is a finite set of edges.

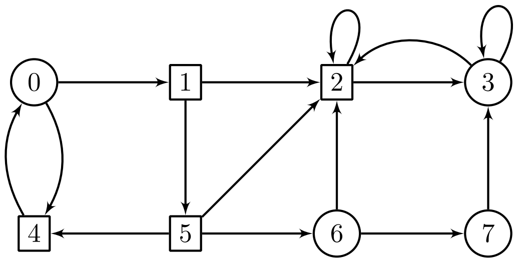


Figure 1.2: A two-player arena $\mathcal{A}_0$

Example 1.2. Consider the arena depicted in Figure 1.2, the edge relation is depicted as ***arrows***, S_1 the set of states controlled by $\mathcal{P}_1$ are depicted as ***circles***, and S_2 is the set of states controlled by $\mathcal{P}_2$. These are depicted as ***squares***. $\triangleleft$

With each arena, we associate a coloring function $\mathsf{col}\colon \mathsf{E} \to \mathsf{C}$ that maps edges to an arbitrary set of colors C. we denote by $\widehat{\mathsf{col}}$ its natural extension over sequences of edges.

Example 1.3. Considering the arena of Figure 1.2, a set of colors could be $\mathsf{C} = \{●, ●\}$ and a coloring function could be

$$\mathsf{col}\colon e \mapsto \begin{cases} ● \text{ if } \mathsf{trgt}(e) = 3 & , \\ ● \text{ otherwise.} \end{cases}$$

$\triangleleft$

For an edge $e \in \mathsf{E}$, we use $\mathsf{src}(e)$ and $\mathsf{trgt}(e)$ to denote its source and its target respectively, i.e., $e = (\mathsf{src}(e), \mathsf{trgt}(e))$. We write $s \xrightarrow{c} t$ the edge e such that $\mathsf{src}(e) = s$, $\mathsf{trgt}(e) = t$, and $\mathsf{col}(e) = c$. We assume all arenas to be non-blocking, i.e., for all $s \in \mathsf{S}$, there exists $e \in \mathsf{E}$ such that $\mathsf{src}(e) = s$.

For $i \in \{1, 2\}$, we call an arena $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E})$ a $\mathcal{P}_i$'s one-player arena if for all $s \in \mathsf{S}_{3-i}$, $|\{e \in \mathsf{E} \mid \mathsf{src}(e) = s\}| = 1$, i.e., $\mathcal{P}_{3-i}$ has no choice.

Let $\mathsf{Hists}(\mathcal{A}, s)$ be the set of histories in $\mathcal{A}$ from the initial state $s \in \mathsf{S}$, i.e., finite sequences of edges $\rho = e_1 \dots e_n \in \mathsf{E}^+$ such that $\mathsf{src}(e_1) = s$ and for all i, $1 \le i < n$, $\mathsf{trgt}(e_i) = \mathsf{src}(e_{i+1})$.

Let $\mathsf{Plays}(\mathcal{A}, s)$ be the set of plays in $\mathcal{A}$ from the initial state $s \in \mathsf{S}$, i.e., infinite sequences of edges $\pi = e_1 e_2 \dots \in \mathsf{E}^\omega$ such that $\mathsf{src}(e_1) = s$ and for all $i \ge 1$, $\mathsf{trgt}(e_i) = \mathsf{src}(e_{i+1})$.

We write $\mathsf{Hists}(\mathcal{A}, S')$ and $\mathsf{Plays}(\mathcal{A}, S')$ for the unions over subsets of initial states $S' \subseteq \mathsf{S}$, and write $\mathsf{Hists}(\mathcal{A})$ and $\mathsf{Plays}(\mathcal{A})$ for the unions over all states of $\mathcal{A}$.

Let $\rho = e_1 \dots e_n \in \mathsf{Hists}(\mathcal{A})$ (resp. $\pi = e_1 e_2 \dots \in \mathsf{Plays}(\mathcal{A})$): we extend the operator src to histories (resp. plays) by identifying $\mathsf{src}(\rho)$ (resp. $\mathsf{src}(\pi)$) to $\mathsf{src}(e_1)$. We proceed similarly for trgt and histories: $\mathsf{trgt}(\rho) = \mathsf{trgt}(e_n)$.

For the sake of convenience, we consider that any set $\mathsf{Hists}(\mathcal{A}, s)$ contains the empty history λ_s such that $\mathsf{src}(\lambda_s) = \mathsf{trgt}(\lambda_s) = s$. We write $\mathsf{Hists}_i(\mathcal{A}, s)$ and $\mathsf{Hists}_i(\mathcal{A})$ for the subsets of histories ρ such that $\mathsf{trgt}(\rho) \in \mathsf{S}_i$, $i \in \{1, 2\}$, i.e., histories whose last state belongs to $\mathcal{P}_i$.

For any set of histories $H \subseteq \mathsf{Hists}(\mathcal{A})$, we write $\widehat{\mathsf{col}}(H)$ for its projection on colors, i.e., $\widehat{\mathsf{col}}(H) = \{\widehat{\mathsf{col}}(\rho) \mid \rho \in H\}$. We do the same for sets of plays.

Preferences and games

Let $\sqsubseteq$ be a total preorder on C^ω, called preference relation. We consider zero-sum games, where the objective of $\mathcal{P}_1$ is to enforce the best possible play with respect to $\sqsubseteq$ while the objective of $\mathcal{P}_2$ is to obtain the worst possible one. That is, $\mathcal{P}_2$ is interested in the inverse relation $\sqsubseteq^{-1}$.

Example 1.4. Consider the arena from Figure 1.2 together with the coloring function from Example 1.3. Now consider the simple reachability objective for $\mathcal{P}_1$ consisting in reaching state 3. This preference relation can be expressed with respect to the coloring from Example 1.3 as follows:

$$\forall \rho, \rho' \in \mathsf{Plays}(\mathcal{A}_0),\ \mathsf{col}(\rho) \sqsubseteq \mathsf{col}(\rho') \text{ if there exists a position } i \text{ in } \rho' \text{ such that } \mathsf{col}(\mathsf{trgt}(\rho'[i])) = \bullet \ .$$

◁

Given $w, w' \in \mathsf{C}^\omega$, we write $w \sqsubset w'$ if we have $\neg(w' \sqsubseteq w)$. We also extend the relation $\sqsubseteq$ over subsets of C^ω as follows: for $W, W' \subseteq \mathsf{C}^\omega$,

$$W \sqsubseteq W' \iff \forall w \in W, \exists w' \in W', w \sqsubseteq w' \ .$$

We also write

$$W \sqsubset W' \iff \exists w' \in W', \forall w \in W, w \sqsubset w' \ .$$

Note that $W \sqsubset W'$ if and only if $\neg(W' \sqsubseteq W)$.

For the sake of convenience, we compare words $w \in \mathsf{C}^\omega$ with languages $K \subseteq \mathsf{C}^\omega$, by simply identifying the word w with the singleton $\{w\}$.

A (deterministic turn-based two-player) game is a tuple $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$ where $\mathcal{A}$ is an arena and $\sqsubseteq$ is a preference relation. For $i \in \{1, 2\}$, a $\mathcal{P}_i$'s one-player game is a game $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$ such that $\mathcal{A}$ is a $\mathcal{P}_i$'s one-player arena.

Strategies

A strategy σ_i for $\mathcal{P}_i$, $i \in \{1, 2\}$, on arena $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E})$, is a function $\sigma_i \colon \mathsf{Hists}_i(\mathcal{A}) \to \mathsf{E}$ such that for all $\rho \in \mathsf{Hists}_i(\mathcal{A})$, $\mathsf{src}(\sigma_i(\rho)) = \mathsf{trgt}(\rho)$. Let $\Sigma_i(\mathcal{A})$ be the set of all strategies of $\mathcal{P}_i$ on $\mathcal{A}$.

Example 1.5. Considering the arena from Figure 1.2, a strategy σ_1 for $\mathcal{P}_1$ could be:

$$\forall\ h \in \mathsf{Hists}_i(\mathcal{A}_0, 0),\ \sigma_1 \colon h\ \mapsto \begin{cases} (0,1) \text{ if } \mathsf{trgt}(h) = 0 \\ (3,3) \text{ if } \mathsf{trgt}(h) = 3 \\ (7,3) \text{ if } \mathsf{trgt}(h) = 7 \\ (7,6) \text{ if } \mathsf{trgt}(h) = 6 \end{cases}$$

If we associate $\mathcal{A}_0$ with the preference relation from Example 1.4, we will see later that such strategy is ***optimal*** for this preference relation from state 0 although it does not ensure that state 3 is visited. ◁

Remark 1.6. We highlight the fact that this strategy does not rely on the entire history but only on the last state of the history. Such strategies are called ***memoryless***.

A strategy σ_i is memoryless if it is a function $\sigma_i \colon \mathsf{S}_i \to \mathsf{E}$ (cf. Example 1.5). We denote by $\Sigma_i^{\mathsf{ML}}(\mathcal{A})$ the set of all memoryless strategies of $\mathcal{P}_i$ on $\mathcal{A}$.

We denote by $\mathsf{Plays}(\mathcal{A}, s, \sigma_i)$ the set of plays consistent with a strategy σ_i of $\mathcal{P}_i$ from an initial state s, i.e., all plays $\pi = e_1 e_2 \ldots \in \mathsf{Plays}(\mathcal{A}, s)$ such that for all prefixes $\rho = e_1 \ldots e_n$, $\mathsf{trgt}(\rho) \in \mathsf{S}_i \implies \sigma_i(\rho) = e_{n+1}$. We write $\mathsf{Plays}(\mathcal{A}, s, \sigma_1, \sigma_2)$ for the singleton set containing the unique play consistent with a couple of strategies for the two players. We use similar notations for histories.

Optimal strategies

Let $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$ be a game on arena $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E})$. Given a $\mathcal{P}_i$-strategy $\sigma_i \in \Sigma_i(\mathcal{A})$ and a state $s \in \mathsf{S}$, we define

$$\mathsf{UCol}_{\sqsubseteq}(\mathcal{A}, s, \sigma_i) = \{w \in \mathsf{C}^\omega \mid \exists\, \sigma_{3-i} \in \Sigma_{3-i}(\mathcal{A}),\ \widehat{\mathsf{col}}(\mathsf{Plays}(\mathcal{A}, s, \sigma_1, \sigma_2)) \sqsubseteq w\}\ ,$$
$$\mathsf{DCol}_{\sqsubseteq}(\mathcal{A}, s, \sigma_i) = \{w \in \mathsf{C}^\omega \mid \exists\, \sigma_{3-i} \in \Sigma_{3-i}(\mathcal{A}),\ w \sqsubseteq \widehat{\mathsf{col}}(\mathsf{Plays}(\mathcal{A}, s, \sigma_1, \sigma_2))\}\ .$$

Note that $\mathsf{DCol}_{\sqsubseteq}(\mathcal{A}, s, \sigma_i) = \mathsf{UCol}_{\sqsubseteq^{-1}}(\mathcal{A}, s, \sigma_i)$.

Taking the standpoint of $\mathcal{P}_1$, we say that a strategy $\sigma_1 \in \Sigma_1(\mathcal{A})$ is at least as good as a strategy $\sigma_1' \in \Sigma_1(\mathcal{A})$ from a state $s \in \mathsf{S}$ if

$$\mathsf{UCol}_{\sqsubseteq}(\mathcal{A}, s, \sigma_1) \subseteq \mathsf{UCol}_{\sqsubseteq}(\mathcal{A}, s, \sigma_1')\ .$$

Intuitively, σ_1 is at least as good as σ_1' if the "worst-case" plays consistent with σ_1 are at least as good as the ones consistent with σ_1'. The UCol operator is useful to define this notion properly even in the case where there is no "worst-case" play for a strategy (i.e., if the infimum used in the classical quantitative setting is not reached). Similar notions have been used before, e.g., in [Rou13].

Symmetrically, for $\mathcal{P}_2$, we say that a strategy $\sigma_2 \in \Sigma_2(\mathcal{A})$ is at least as good as a strategy $\sigma_2' \in \Sigma_2(\mathcal{A})$ from a state $s \in \mathsf{S}$ if

$$\mathsf{DCol}_{\sqsubseteq}(\mathcal{A}, s, \sigma_2) \subseteq \mathsf{DCol}_{\sqsubseteq}(\mathcal{A}, s, \sigma_2')\ .$$

Now, we say that a strategy $\sigma_i \in \Sigma_i(\mathcal{A})$ of $\mathcal{P}_i$ is optimal from a state $s \in \mathsf{S}$, s-optimal, if it is at least as good as every other strategy $\sigma_i' \in \Sigma_i(\mathcal{A})$ from s. We extend this notation to subsets of states in the natural way, and we say that a strategy σ_i is **uniformly-optimal** if it is S-optimal.

Example 1.7. Consider once more the strategy from Example 1.5. In this example the strategy described does not ensure a visit of the target state 3, thus it is not winning. However, notice that according to the above definition of optimal strategies, the outcome ensured by the described strategy cannot be bested by any other, i.e., $\mathcal{P}_2$ can always avoid state 3. This shows that the considered strategy is at least as good as any other strategy of $\mathcal{P}_1$, hence it is optimal. $\triangleleft$

We recall that our goal is to characterize the preference relations that admit finite-memory optimal strategies in all arenas. Next we give an example of an arena where optimal strategies need memory.

Example 1.8. Consider the arena in Figure 1.3, for the sake of the example we drop the coloring function and consider that the preference relation where any run that visits both states 2 and 0 infinitely often is preferred over any run which does not, this preference relation is usually known

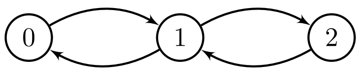


Figure 1.3: Arenas where $\mathcal{P}_1$ needs memory to play optimally.

as generalized Büchi objective [BK08]. In this game from any state, $\mathcal{P}_1$ has a ***winning*** strategy with exactly 2 memory states, in particular it is sufficient to use one state to remember the last visit of state 0, and a second state to remember the last visit of state 2. $\triangleleft$

As already mentioned, we want to better understand the preferences relation admitting finite memory optimal strategies. We base our approach on an existing work regarding the case of uniformly-optimal memoryless (UML) strategies, i,e., a characterization of Gimbert and Zielonka [GZ05]. We first present their approach in this simpler case with the purpuse to offer better insight on our result.

1.3 Playing optimally with no memory

In this section we recall the characterization of preference relations that admit UML strategies proposed by Gimbert and Zielonka. Their characterization uses two technical notions; monotony and selectivity. These notions intuitively describe how sequences of colors behave with respect to a fixed preference relation.

From a technical point of view, these two properties are built upon automata theoretic notions, therefore we introduce the necessary definitions and notations regarding automata and regular languages.

Automata and languages of colors.

We recall classical notions on automata on finite words. A non-deterministic finite-state automaton (NFA) is a tuple $\mathcal{N} = (Q, \mathsf{B}, \delta, Q_{\mathsf{init}}, Q_{\mathsf{fin}})$, where Q is a finite set of states, $\mathsf{B} \subseteq \mathsf{C}$ is a finite alphabet of colors, $\delta \subseteq Q \times \mathsf{B} \times Q$ is a set of transitions, $Q_{\mathsf{init}} \subseteq Q$ is a set of initial states, and $Q_{\mathsf{fin}} \subseteq Q$ is a set of final states. Given a state $q \in Q$ and a word $w \in \mathsf{B}^*$, we denote by $\widehat{\delta}(q, w)$ the set of states that can be reached from q after reading w. Without loss of generality, we assume all NFA to be co-accessible, i.e., for all $q \in Q$, there exists $w \in \mathsf{B}^*$, such that $\widehat{\delta}(q, w) \cap Q_{\mathsf{fin}} \neq \emptyset$. Recall that NFA precisely recognize regular languages [BK08].

For any finite subset $\mathsf{B} \subseteq \mathsf{C}$, we denote by $\mathsf{Reg}(\mathsf{B})$ the set of all regular languages over B. Let $\mathcal{R}(\mathsf{C}) = \bigcup_{\mathsf{B} \subseteq \mathsf{C},\, |\mathsf{B}| < \infty} \mathsf{Reg}(\mathsf{B})$, that is, all the regular languages built over C.

Let $K \subseteq \mathsf{C}^*$ be a language of finite words. We denote by $\mathsf{Prefs}(K)$ the set of all prefixes of the words in K. We define the set of infinite words

$$[K] = \{w = c_1 c_2 \ldots \in \mathsf{C}^\omega \mid \forall\, n \geq 1,\, c_1 \ldots c_n \in \mathsf{Prefs}(K)\} \ ,$$

which contains all infinite words for which every finite prefix is a prefix of a word in K. Intuitively, if K is regular, $[K]$ is the language of infinite words that correspond to infinite paths that can always branch and reach a final state, on an automaton for K. Given a finite word $w \in \mathsf{C}^*$ and a language $K \subseteq \mathsf{C}^*$, we write wK for their concatenation, i.e., the language $wK = \{w' = ww'' \mid w'' \in K\} \subseteq C^*$.

We now introduce two core properties used in the characterization, i.e., the monotony and the selectivity.

Monotony

Definition 1.9 (Monotony)**.** A preference relation $\sqsubseteq$ is ***monotone*** if for all, for all $K_1, K_2 \in \mathcal{R}(\mathsf{C})$,

$$\exists w \in \mathsf{C}^*,\ [wK_1] \sqsubset [wK_2] \implies \forall w' \in \mathsf{C}^*,\ [w'K_1] \sqsubseteq [w'K_2]\ . \tag{1.1}$$

This property roughly captures the ability of $\sqsubseteq$ to remain stable with respect to any prefix addition. To grasp a better idea, consider the depiction in Figure 1.4. The bullets in the far left of the image stand for a prefix w, the continuations of w are depicted in different colors, the topmost one is a continuation in K_2 and the bottom-most one is a continuation in K_1. Assume now that $\mathcal{P}_1$ prefers an outcome $w\rho_2$ in $[wK_2]$ over any outcome $w\rho_1$ in $[wK_1]$, then the monotony states that, it is not possible to find a different prefix w' such that $\mathcal{P}_1$ will prefers an outcome in $[wK_1]$. This property can be understood as a formal mean to capture preference relations for which the order between outcomes cannot be switched over using a different prefix constructed with the help of more memory.

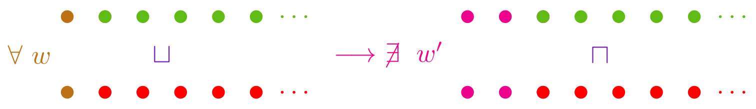


Figure 1.4: Monotonicity

As an example of preference relations enjoying monotony, one could think of any prefix-independent payoff, e.g., Büchi, parity, mean payoff. But one could also consider other conditions, for instance safety is monotone but not prefix-independent. To see this, consider a finite subset B of C, and remember that the preference relation induced by a safety objectives will prefer sequences in B^ω over sequences in $(\mathsf{C} \cup \mathsf{B})^\omega$. For the sake of simplicity, consider two sequences u and v in C^ω and a finite sequence w in C^ω. Assume that $wu \sqsubset wv$, then necessarily w and v consists of colors in B, therefore for any w' in C^*, it will be the case that $w'u \sqsubseteq w'v$.

On the other hand, the following preference relation is not monotone. Let $\mathsf{C} = \mathbb{N}$ and consider the payoff:

$$c_0 c_1 \ldots \mapsto \sup_{n \geq 0} \sum_{i \geq 0} \frac{c_i}{n+1}\ .$$

This payoff maps sequence of naturals to the supremum natural seen. This payoff does not induce a monotone preference relation. Indeed consider the sequences $u = 1^\omega, v = 20^\omega$, and $w = 00$, then $(wv = 0020^\omega) \sqsubset (wu = 001^\omega)$, however for $w' = 11$ we have $(w'u = 11^\omega) \sqsubset (w'v = 1120^\omega)$.

We now switch our attention to selectivity.

Selectivity

Definition 1.10 (Selectivity)**.** A preference relation $\sqsubseteq$ is ***selective*** if for all $w \in \mathsf{C}^*$, and for all $K_1, K_2, K_3 \in \mathcal{R}(\mathsf{C})$

$$[w(K_1 \cup K_2)^* K_3] \sqsubseteq [wK_1^*] \cup [wK_2^*] \cup [wK_3]\ . \tag{1.2}$$

This property states that $\sqsubseteq$ is somehow stable under shuffling of plays. Consider the sequence of colors depicted in Figure 1.5. In the left-hand side two colored sequences, the selectivity implies that one cannot obtain a better pay-off by interleaving the two sequences. At this stage, the reader may wonder how this very notion relates to memory strategies. It is sufficient to notice that such interleaving requires extra memory, thus this property could be understood as the impossibility of improving a payoff by using memory by mixing plays.

We also mention that finding non selective payoffs is easy since common payoffs such as Muller or generalized Büchi are not selective.

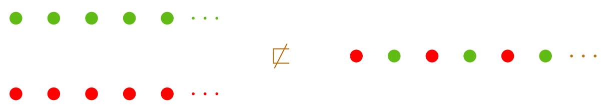


Figure 1.5: Selectivity

The characterization

We now have the technical background to state the characterization of memoryless preference relations.

Theorem 1.11 (Memoryless characterization)**.** *Let $\sqsubseteq$ be a preference relation. Both players have* **UML** *strategies in all games $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$* **if and only if** *$\sqsubseteq$ and $\sqsubseteq^{-1}$ are* **monotone** *and* **selective***.*

The first part of the proof (left-to-right implication) uses mainly automata theoretic arguments cf. Lemma 1.12. We present the main idea of the proof to help the reader grasp the essence of both monotony and selectivity. It should also makes the second part of this chapter (finite memory characterization) easier to follows since we will follow the same recipe.

Lemma 1.12. *Assume that $\mathcal{P}_1$ has UML strategies in all games $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$ where $\mathsf{S}_2 = \emptyset$, then $\sqsubseteq$ is monotone and selective.*

The very rough idea to establish the monotony is as follows. Build four automata: $\mathcal{N}_w$, $\mathcal{N}_{w'}$, $\mathcal{N}_{K_1}$, and $\mathcal{N}_{K_2}$. The two first automata recognize respectively the languages $\{w\}$ and $\{w'\}$. The two last recognize respectively the languages K_1, and K_2. This four languages are the parameters from Equation (1.1). The main trick in the proof is deployed in two steps:

Step 1. Glue the above automata using a fresh state t such that the final states of $\mathcal{N}_w$, $\mathcal{N}_{w'}$ are ignored and any ingoing transition to these states is now glued to the fresh state t.
The initial states of $\mathcal{N}_{K_1}$, and $\mathcal{N}_{K_2}$ are ignored and the outgoing transitions from these states outgo from state t. This construction is displayed in Figure 1.6.

Step 2. Interpret the obtained automaton as a one-player game, and fix an optimal strategy σ^* for $\sqsubseteq$. By assumption σ^* can be chosen UML. The trick now is to notice that any infinite play has to cross t and that in t a UML strategy will always make the same choice whether the history is w or w'.

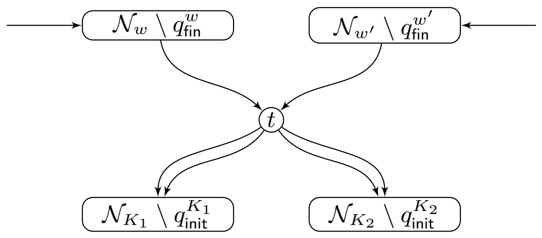
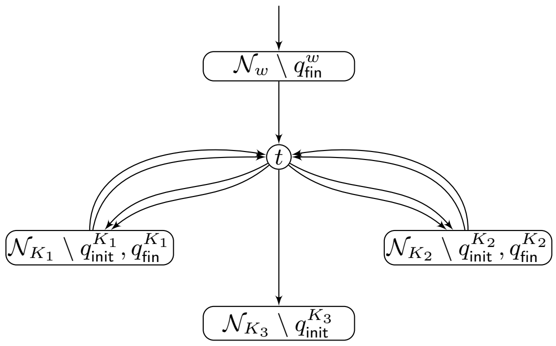

$\mathcal{N}_w \setminus q^w_{\mathsf{fin}}$
$\mathcal{N}_{w'} \setminus q^{w'}_{\mathsf{fin}}$
t
$\mathcal{N}_{K_1} \setminus q^{K_1}_{\mathsf{init}}$
$\mathcal{N}_{K_2} \setminus q^{K_2}_{\mathsf{init}}$

Figure 1.6: Automaton $\mathcal{N}$ built to establish monotony.

Establishing the selectivity follows the same spirit but has to deal with cycles. The intuition of this part of the proof can be explained as follows. We start by building automata $\mathcal{N}_w$, $\mathcal{N}_{K_1}$,

$\mathcal{N}_{K_2}$, and $\mathcal{N}_{K_3}$. Using the same trick as before, we glue them using a fresh state t such in the obtained automaton, the initial state is the initial state of $\mathcal{N}_w$, and the final state is the final state of $\mathcal{N}_{K_3}$, cf. Figure 1.7. To conclude, we interpret the new automaton as a one-player game and notice that a UML strategy σ^* (which exists by assumption) will always make the same choice in t.



$\mathcal{N}_w \setminus q_{\mathsf{fin}}^w$
t
$\mathcal{N}_{K_1} \setminus q_{\mathsf{init}}^{K_1}, q_{\mathsf{fin}}^{K_1}$
$\mathcal{N}_{K_2} \setminus q_{\mathsf{init}}^{K_2}, q_{\mathsf{fin}}^{K_2}$
$\mathcal{N}_{K_3} \setminus q_{\mathsf{init}}^{K_3}$

Figure 1.7: Automaton $\mathcal{N}$ built to establish selectivity.

The same lemma holds for $\mathcal{P}_2$ and $\sqsubseteq^{-1}$ using similar arguments.

Lemma 1.13. *Assume that $\sqsubseteq$ and $\sqsubseteq^{-1}$ are monotone and selective, then both players have UML strategies in all games $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$.*

The proof of the above lemma is obtained through an induction over the number of choices the players have. In order to convey some intuition we need to introduce some notations. For an arena $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E})$, we write $n_{\mathcal{A}} = |\mathsf{E}| - |\mathsf{S}|$ for its number of choices. We also define the notion of sub-arena: we say that an arena $\mathcal{A}' = (\mathsf{S}'_1, \mathsf{S}'_2, \mathsf{E}')$ is a sub-arena of an arena $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E})$ if $\mathsf{S}_1 = \mathsf{S}'_1$, $\mathsf{S}_2 = \mathsf{S}'_2$, and $\mathsf{E}' \subseteq \mathsf{E}$. That is, arena $\mathcal{A}'$ is a sub-arena of $\mathcal{A}$ if it can be obtained from $\mathcal{A}$ by removing some edges of $\mathcal{A}$ (while keeping it non-blocking). We say that a set of arenas $\mathfrak{A}$ is closed under sub-arena operation if for all $\mathcal{A} \in \mathfrak{A}$, for all sub-arenas $\mathcal{A}'$ of $\mathcal{A}$, $\mathcal{A}' \in \mathfrak{A}$. Now fix a set of arenas $\mathfrak{A}$ is closed under sub-arena operation and an integer n. Assume that for any arena $\mathcal{A}'$ in $\mathfrak{A}$ such that $n'_{\mathcal{A}} < n$, the lemma holds. Then we show that the lemma holds for any $\mathcal{A}$ such that $n_{\mathcal{A}} = n$. It should be clear that if $\mathcal{P}_1$ has no choices, i.e., from any state in S_1 there is only one outgoing edge, then $\mathcal{P}_1$ has an UML strategy.

Now let $\mathcal{A}$ be such that $n_{\mathcal{A}} = n$ and that $\mathcal{P}_1$ has at least a choice. Since $\mathcal{P}_1$ has at least a choice, there must exists a state t with at least two outgoing edges. The proof follows the following steps:

Step 1. Let $t\mathsf{E}$ be the set of outgoing edges from t,

Step 2. choose a partition $(t\mathsf{E}_1, t\mathsf{E}_2)$ of $t\mathsf{E}$,

Step 3. build two sub-arenas $\mathcal{A}_1$ and $\mathcal{A}_2$ such that in $\mathcal{A}_1$ the set of outgoing edges from t is $t\mathsf{E}_1$ and in $\mathcal{A}_2$ the set of outgoing edges from t is $t\mathsf{E}_2$.

The induction hypothesis implies that $\mathcal{P}_1$ has two UML optimal strategies, σ_1^* in $\mathcal{A}_1$, and σ_2^* in $\mathcal{A}_2$. Notice that both these strategies are well defined over $\mathcal{A}$. Now using the monotony of $\sqsubseteq$, the plays crossing t in one of the two sub-arenas are always at least as good as the one in the other sub-arena, w.l.o.g. $\mathcal{A}_1$. Hence, the outcomes of σ_1^* are always at least as good as the outcomes of σ_2^*. In other words $\mathcal{P}_1$ has to rather play in $\mathcal{A}_1$ which he can force by playing according to σ_1^*

in $\mathcal{A}$. However, it could be the case that switching in between plays in $\mathcal{A}_1$ and $\mathcal{A}_2$ is even better, but the selectivity of $\sqsubseteq$ allows as to rule out this scenario and conclude that σ_1^* is indeed a UML optimal strategy in $\mathcal{A}$.

The lifting corollary

Gimbert and Zielonka not only characterized memoryless preference relations, but they also provided an elegant tool allowing to study these relations. This tools takes its form in the following corollary called the the lifting corollary. This corollary is stated as follows:

Corollary 1.14. *Let $\sqsubseteq$ be a preference relation such that:*

- *$\sqsubseteq$ is memoryless for any $\mathcal{P}_1$'s arena,*
- *$\sqsubseteq^{-1}$ is memoryless for any $\mathcal{P}_2$'s arena,*

then $\sqsubseteq$ is memoryless for both players in any arena.

The lifting corollary follows quite easily indeed, it is sufficient to notice that if $\sqsubseteq$ and $\sqsubseteq^{-1}$ are positional on all one player arenas then they must be monotone and selective by Lemma 1.12, therefore according to Theorem 1.11 (right-to-left), $\sqsubseteq$ is memoryless for both players in any arena.

Not only the above corollary is easy to prove, it also provides a simpler way for establishing the existence of UML strategies since it relies on the study of optimal strategies in one-player arenas which in turn amounts to graph reasoning. This is usually a lot simpler than studying two-players arenas.

1.4 Playing optimally with memory

In this section, we present the main result of this chapter which is a characterization of the preference relations that admit uniformly-optimal finite-memory (UFM) strategies based on a given skeleton $\mathcal{M}$ in all arenas.

The lifting corollary: a counter example

Example 1.15. Consider the arena depicted in Figure 1.8, consider also the preference relation where $\mathcal{P}_1$ wins whenever one of the two following conditions holds:

- The running sum of weights grows up to infinity.
- The running sum of weights takes value zero infinitely often.

It should be clear enough that an optimal strategy for $\mathcal{P}_1$ cannot remain in state b, thus $\mathcal{P}_1$ should move the play to state a. From that state, if $\mathcal{P}_2$ chooses to remain forever in state a, then the running sum will diverge fulfilling the first condition. If he chooses to remain in state a for some time then move the play again to state b again then $\mathcal{P}_1$ can reply by decreasing the running sum to 0 then moving to state a. This latter reply of $\mathcal{P}_1$ shows that he will fulfill the second condition of the winning condition. However, if $\mathcal{P}_2$ decides to stay for arbitrary long periods of time in state a before moving the play to state b, this will force $\mathcal{P}_1$ to use an arbitrarily large memory in order to implement his winning strategy. This shows that $\mathcal{P}_1$ has an infinite memory optimal strategy. Now, we argue that $\mathcal{P}_1$, is bound to use infinite memory to win. This is rather simple to witness, assume that $\mathcal{P}_1$ plays according to a finite memory strategy, then $\mathcal{P}_2$ can always loop in state a long enough as to exceed the memory chosen by $\mathcal{P}_1$. At this stage $\mathcal{P}_1$ cannot track the running sum anymore thus failing to reset this value to 0 infinitely often.

We now turn our attention to the one-player game version of the same preference condition. This means that $\mathcal{P}_1$ wants to build a play such that either:

- The running sum of weights grows up to infinity.
- The running sum of weights takes value zero infinitely often.

and $\mathcal{P}_2$ wants to build a play (in an arena that he controls) such that ***both*** the following conditions hold:

- The running sum of weights ***does not*** grow up to infinity.
- The running sum of weights takes value zero ***finitely*** often.

A study of the underlying finite graph structure of each one-player arena shows that both players need only to reach ***good cycles*** and repeat them, this behavior can be implemented thanks to a finite memory.

This simple example exhibits a two-player arena and a preference relation where both players have ***finite memory optimal strategies*** in their respective one-player arena, however $\mathcal{P}_1$ needs ***infinite memory*** in a two-player arenas. $\lhd$

In the light of the above example, one might think that the lifting corollary in the context of finite memory strategies cannot exist. However, a closer look at the memory structure used by $\mathcal{P}_1$ shows that the notion of finite memory is quite rich. In fact, $\mathcal{P}_1$ needs infinite memory because he needs to track the exact value of the running sum along the current play, but this value can be forced by $\mathcal{P}_2$, to be arbitrarily high.This shows that the memory size $\mathcal{P}_1$ needs change subject to the behavior of $\mathcal{P}_2$. Actually, consider any one-player arena of $\mathcal{P}_1$. In order to force the running sum to be 0 infinitely often, $\mathcal{P}_1$ builds a cycle whose weights sum up to 0, but the length of such cycle depends on the colors in the arena (weights in our case).

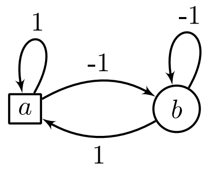


Figure 1.8: A counter example

Let us compare such a memory structure to a memory in an arena where the preference relation is given by a generalized Büchi condition, e.g. Figure 1.3. The memory consists in boolean flags helping $\mathcal{P}_1$ switch between memoryless behaviors. The crucial remark resides in the fact that the number of these flags is function of the preference relation and not the arena. We shall call such a memory structure arena independent. We argue next that the lifting corollary holds when restricted to this simpler family of finite memory strategies.

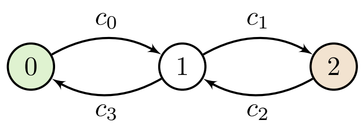
Arena independent finite memory

Strategies and memory skeletons

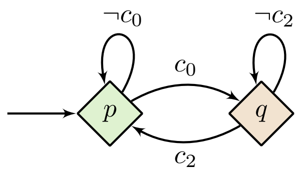
A memory skeleton, is an automaton like formalism, it updates the memory state given the current state and color of a finite play. The crucial property of the skeletons we consider is that they can be defined on the sole knowledge of the preference relation whiteout priors on the arena structure. Formally,

Definition 1.16. A memory skeleton is a tuple $\mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$ where M is a finite set of states, $m_{\mathsf{init}} \in \mathsf{M}$ is a fixed initial state and $\mathsf{upd}\colon \mathsf{M} \times \mathsf{C} \to \mathsf{M}$ is an update function. We write $\widehat{\mathsf{upd}}$ for the natural extension of upd to sequences of colors in C^*.

Note that memory skeletons are deterministic and might have an infinite number of transitions.



(a) $\mathcal{P}_1$ wants to visits states 0 and 2 infinitely often.



(b) A memory skeleton for the preference relation described in Figure 1.9a.

Figure 1.9: An arena independent memory skeleton.

Example 1.17. In Figure 1.9 an arena and a skeleton are depicted. The arena is the one from Figure 1.3, here $\mathcal{P}_1$ aims at visiting states 0 and 2 infinitely many times (this is a generalized Büchi objective with two conditions). In order to achieve such objective, a player can switch between a winning strategy for the first condition, and a winning strategy for the second condition. In our example it amounts to switching back and forth between cycling around state 1 through state 0 and through state 2. A finite memory strategy in this case should inform the player when to operate this switch. With this in mind, consider the skeleton in Figure 1.9b is a memory for a Generalized Büchi condition with two conditions. In its initial state, p in our case, it awaits for the a state in the first condition to be visited, then it switches to a new state, q in our example. As soon as a state in the second condition is visited the memory is updated to its initial state. $\triangleleft$

We highlight the fact that in the previous example, the skeleton used is independent of the arena, it suffices to know the preference relation. For instance the preference relation used in Example 1.17 can be written as follows:

$$\forall w, w' \in \mathsf{C}^\omega, w \sqsubseteq w' \iff w' \in W \wedge w \in \overline{W} \ ,$$

where

$$W = \{w \in \mathsf{C}^\omega \mid \forall i \geq 0,\ \exists j > i, w[j] = c_0 \wedge \exists k > i, w[k] = c_2\} \ ,$$
$$\overline{W} = \mathsf{C}^\omega \setminus W \ .$$

We define the trivial memory skeleton with only one state as:

$$\mathcal{M}_{\mathsf{triv}} = (\mathsf{M} = \{m_{\mathsf{init}}\}, m_{\mathsf{init}}, \mathsf{upd}\colon \{m_{\mathsf{init}}\} \times \mathsf{C} \to \{m_{\mathsf{init}}\}) \ .$$

This skeleton corresponds to the notion of memoryless strategies discussed earlier in the previous section.

A finite-memory strategy σ_i is a strategy that can be encoded as a Mealy machine, i.e., a memory skeleton $\mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$ with transitions over a finite subset of colors $\mathsf{B} \subseteq \mathsf{C}$, enriched with a next-action function $\mathsf{nxt}\colon \mathsf{M} \times \mathsf{S}_i \to \mathsf{E}$ such that for all $m \in \mathsf{M}$, $s \in \mathsf{S}_i$, $\mathsf{src}(\mathsf{nxt}(m, s)) = s$. Given a Mealy machine $\Gamma_{\sigma_i} = (\mathcal{M}, \mathsf{nxt})$, strategy σ_i is defined as follows:

- $\forall\, s \in \mathsf{S}_i$, $\sigma_i(\lambda_s) = \mathsf{nxt}(m_{\mathsf{init}}, s)$,
- $\forall\, \rho \cdot e \in \mathsf{Hists}_i(\mathcal{A})$, $e \in \mathsf{E}$, $\sigma_i(\rho \cdot e) = \mathsf{nxt}\left(\widehat{\mathsf{upd}}\left(m_{\mathsf{init}}, \widehat{\mathsf{col}}\,(\rho \cdot e)\right), \mathsf{trgt}(e)\right)$.

We denote by $\Sigma_i^{\mathsf{FM}}(\mathcal{A})$ the set of all finite-memory strategies of $\mathcal{P}_i$ on $\mathcal{A}$. We say that a strategy $\sigma_i \in \Sigma_i^{\mathsf{FM}}(\mathcal{A})$ is based on memory skeleton $\mathcal{M}$ if it can be encoded as a Mealy machine $\Gamma_{\sigma_i} = (\mathcal{M}, \mathsf{nxt})$, as above. We always implicitly assume that strategies of $\Sigma_i^{\mathsf{FM}}(\mathcal{A})$ are built by restricting the transitions of their skeleton $\mathcal{M}$ to the actual subset of colors appearing in $\mathcal{A}$.

Characterization

The elegance of the characterization presented by Gimbert and Zielonka lies in the fact that the conditions a preference relation should fulfill are completely agnostic to the arena, indeed monotony and selectivity are defined over the sequence of colors. However, when a player needs memory to play optimally, he uses this memory to discriminate between sequence of colors. Intuitively speaking, using the appropriate memory skeleton, sequences of colors ending leading to the same memory state should behave following some structure, but two sequences leading to different memory state should not. In the case of memoryless optimal strategies, this observation holds since the skeleton used consists of a single memory state and all the sequences lead to this very state. Thus we do not discriminate between any sequence of colors.

With the above remark in mind, we augment the notions of monotony and selectivity with some sort of compliance with memory states. Thus, we will require monotony and selectivity over sequences that are indistinguishable by a memory skeleton $\mathcal{M}$. We will call these properties $\mathcal{M}$-monotony and $\mathcal{M}$-selectivity. Therefore, we introduce the following notation:

Let $\mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$ be a memory skeleton, for $m, m' \in \mathsf{M}$, we define the language $L_{m,m'} = \left\{ w \in \mathsf{C}^* \mid \widehat{\mathsf{upd}}(m, w) = m' \right\}$ that contains all words that can be read from m to m' in $\mathcal{M}$.

Definition 1.18 ($\mathcal{M}$-monotony)**.** Let $\mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$ be a memory skeleton. A preference relation $\sqsubseteq$ is ***$\mathcal{M}$-monotone*** if for all $m \in \mathsf{M}$, for all $K_1, K_2 \in \mathcal{R}(\mathsf{C})$;

$$(\exists\, w \in L_{m_{\mathsf{init}},m},\ [wK_1] \sqsubset [wK_2]) \implies (\forall\, w' \in L_{m_{\mathsf{init}},m},\ [w'K_1] \sqsubseteq [w'K_2])\,. \tag{1.3}$$

Definition 1.19 ($\mathcal{M}$-selectivity)**.** Let $\mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$ be a memory skeleton. A preference relation $\sqsubseteq$ is ***$\mathcal{M}$-selective*** if for all $w \in \mathsf{C}^*$, $m = \widehat{\mathsf{upd}}(m_{\mathsf{init}}, w)$; for all $K_1, K_2 \in \mathcal{R}(\mathsf{C})$ such that $K_1, K_2 \subseteq L_{m,m}$, for all $K_3 \in \mathcal{R}(\mathsf{C})$,

$$[w(K_1 \cup K_2)^* K_3] \sqsubseteq [wK_1^*] \cup [wK_2^*] \cup [wK_3]. \tag{1.4}$$

Example 1.20. In Figure 1.10, are depicted two memory skeletons $\mathcal{M}^{\mathsf{p}} = (\mathsf{M}^{\mathsf{p}}, m^{\mathsf{p}}_{\mathsf{init}}, \mathsf{upd}^{\mathsf{p}})$ and $\mathcal{M}^{\mathsf{c}} = (\mathsf{M}^{\mathsf{c}}, m^{\mathsf{c}}_{\mathsf{init}}, \mathsf{upd}^{\mathsf{c}})$. $\sqsubseteq$ in this example is a two-target reachability games T_1 and T_2. We use two separate skeletons. One for $\mathcal{M}$-monotony and one, for $\mathcal{M}$-selectivity. In the subsequent development, we follow this compositional approach. A strategy using a joint skeleton can be obtained by simply combining the two structures. We claim that $\sqsubseteq$ is $\mathcal{M}^{\mathsf{p}}$-monotone and $\mathcal{M}^{\mathsf{c}}$-selective.

For instance, we show as an example that for any word w in $L_{m^{\mathsf{p}}_{\mathsf{init}}, m^{\mathsf{p}}_{\mathsf{init}}}$ such that $[wK_1] \sqsubset [wK_2]$ Equation (1.3) holds. First notice that:

i. $[wK_1] \sqsubset [wK_2]$ means that all words of $[wK_1]$ are losing, and that there exists a winning word in $[wK_2]$.

ii. w and w' in $L_{m^{\mathsf{p}}_{\mathsf{init}}, m^{\mathsf{p}}_{\mathsf{init}}}$ means that non of them reaches T_1.

If T_2 is not visited along w, then item *i.* implies that $[K_2]$ contains a winning word w'', now noticing that $w'w''$ is still winning implies that $[w'K_1] \sqsubseteq [w'K_2]$. If T_2 is not visited along w, then by item *i.* we know that $[K_1]$ does not contain a word reaching T_1. Item *ii.* yield again that all words in $[w'K_1]$ are loosing, which in turn entails that $[w'K_1] \sqsubseteq [w'K_2]$. ◁

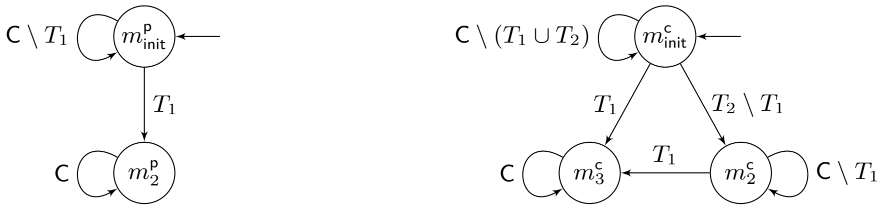

$\mathsf{C} \setminus T_1$
$m^{\mathsf{p}}_{\mathsf{init}}$
T_1
C
m^{p}_2
$\mathsf{C} \setminus (T_1 \cup T_2)$
$m^{\mathsf{c}}_{\mathsf{init}}$
T_1
$T_2 \setminus T_1$
C
m^{c}_3
T_1
m^{c}_2
$\mathsf{C} \setminus T_1$

Figure 1.10: Memory skeletons $\mathcal{M}^{\mathsf{p}}$ (left) and $\mathcal{M}^{\mathsf{c}}$ (right).

We now present a characterization of the preference relations that admit arena independent uniformly-optimal finite memory (UFM) strategies.

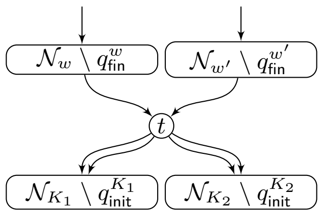
Theorem 1.21. *Let $\sqsubseteq$ be a preference relation and let $\mathcal{M}$ be a memory skeleton. Then, both players have UFM strategies based on memory skeleton $\mathcal{M}$ in all games $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$ if and only if $\sqsubseteq$ and $\sqsubseteq^{-1}$ are $\mathcal{M}$-monotone and $\mathcal{M}$-selective.*

The proof of this theorem follows the same idea as the proof from Gimbert and Zielonka.

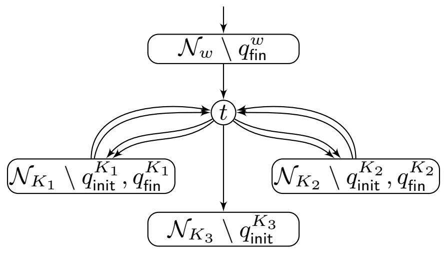
Left-to-right implication The first part tackles the left-to-right implication which is the matter in the next two lemmas. Lemma 1.22 establishes $\mathcal{M}$-monotony, and Lemma 1.23 establishes $\mathcal{M}$-selectivity.

Lemma 1.22. *Let $\mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$ be a memory skeleton and $\sqsubseteq$ be a preference relation. Assume that for all one-player arenas $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2 = \emptyset, E)$, for all $s, s' \in \mathsf{S}$, $\mathcal{P}_1$ has an s-optimal and s'-optimal strategy $\sigma \in \Sigma_1^{\mathsf{FM}}(\mathcal{A})$, encoded as a Mealy machine $\Gamma_\sigma = (\mathcal{M}, \mathsf{nxt})$, in $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$. Then $\sqsubseteq$ is $\mathcal{M}$-monotone.*

Lemma 1.23. *Let $\mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$ be a memory skeleton and $\sqsubseteq$ be a preference relation. Assume that for all one-player arenas $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2 = \emptyset, E)$, for all $s \in \mathsf{S}$, $\mathcal{P}_1$ has an s-optimal strategy $\sigma \in \Sigma_1^{\mathsf{FM}}(\mathcal{A})$, encoded as a Mealy machine $\Gamma_\sigma = (\mathcal{M}, \mathsf{nxt})$, in $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$. Then $\sqsubseteq$ is $\mathcal{M}$-selective.*


$\mathcal{N}_w \setminus q^w_{\mathsf{fin}}$
$\mathcal{N}_{w'} \setminus q^{w'}_{\mathsf{fin}}$
t
$\mathcal{N}_{K_1} \setminus q^{K_1}_{\mathsf{init}}$
$\mathcal{N}_{K_2} \setminus q^{K_2}_{\mathsf{init}}$

(a) Automaton $\mathcal{N}$ built to establish $\mathcal{M}$-monotony.


$\mathcal{N}_w \setminus q^w_{\mathsf{fin}}$
t
$\mathcal{N}_{K_1} \setminus q^{K_1}_{\mathsf{init}}, q^{K_1}_{\mathsf{fin}}$
$\mathcal{N}_{K_2} \setminus q^{K_2}_{\mathsf{init}}, q^{K_2}_{\mathsf{fin}}$
$\mathcal{N}_{K_3} \setminus q^{K_3}_{\mathsf{init}}$

(b) Automaton $\mathcal{N}$ built to establish $\mathcal{M}$-selectivity.

These two lemmas are established along the same techniques of their memoryless counterparts. Actually in Figure 1.11a and Figure 1.11b one can already guess that the same technique applies. The main difference being the fact that the plays induced by w and w' lead to the same memory state. Moreover in the case of $\mathcal{M}$-selectivity, the automata $\mathcal{N}_{K_1}$ and $\mathcal{N}_{K_2}$ recognize plays forming cycles in the memory skeleton. Thanks to these extra properties, the intuition of the proof works exactly the same; once the built automata viewed as one-player arena, any finite memory will behave the same from the crucial state t. Thus using the arguments from the memoryless case yields the $\mathcal{M}$-monotony and $\mathcal{M}$-selectivity. This tends to show that the proposed generalizations in Definition 1.9 and Definition 1.10 fit exactly our needs.

Right-to-left implication For this part, we need the following notation:

Let $\mathcal{M}^1 = (\mathsf{M}^1, m^1_{\mathsf{init}}, \mathsf{upd}^1)$ and $\mathcal{M}^2 = (\mathsf{M}^2, m^2_{\mathsf{init}}, \mathsf{upd}^2)$ be two memory skeletons. We define their product $\mathcal{M}^1 \otimes \mathcal{M}^2$ as the memory skeleton $\mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$ obtained as follows: $\mathsf{M} = \mathsf{M}^1 \times \mathsf{M}^2$, $m_{\mathsf{init}} = (m^1_{\mathsf{init}}, m^2_{\mathsf{init}})$, and, for all $m^1 \in \mathsf{M}^1$, $m^2 \in \mathsf{M}^2$, $c \in \mathsf{C}$, $\mathsf{upd}((m^1, m^2), c) = (\mathsf{upd}^1(m^1, c), \mathsf{upd}^2(m^2, c))$. That is, the memories are updated in parallel when a color is read.

Our goal now is the prove the following lemma:

Lemma 1.24. *Let $\sqsubseteq$ be a preference relation and $\mathcal{M}^{\mathsf{p}}_1$, $\mathcal{M}^{\mathsf{p}}_2$, $\mathcal{M}^{\mathsf{c}}_1$ and $\mathcal{M}^{\mathsf{c}}_2$ be four memory skeletons. Assume that $\sqsubseteq$ is $\mathcal{M}^{\mathsf{p}}_1$-monotone and $\mathcal{M}^{\mathsf{c}}_1$-selective, and that $\sqsubseteq^{-1}$ is $\mathcal{M}^{\mathsf{p}}_2$-monotone and $\mathcal{M}^{\mathsf{c}}_2$-selective. Then, for all arenas $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E})$, there exists uniform finite-memory optimal strategies $(\sigma_1, \sigma_2) \in \Sigma^{\mathsf{FM}}_1(\mathcal{A}) \times \Sigma^{\mathsf{FM}}_2(\mathcal{A})$ in $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$, such that strategies σ_i are encoded as Mealy machines $\Gamma_{\sigma_i} = (\mathcal{M}, \mathsf{nxt}^i)$ based on the joint memory skeleton $\mathcal{M} = \mathcal{M}^{\mathsf{p}}_1 \otimes \mathcal{M}^{\mathsf{p}}_2 \otimes \mathcal{M}^{\mathsf{c}}_1 \otimes \mathcal{M}^{\mathsf{c}}_2$.*

For the sake of decomposition, we use four memory skeletons and build strategies based on their product memory. However, if there exists a skeleton $\mathcal{M}$ that is already such that both $\sqsubseteq$ and $\sqsubseteq^{-1}$ are $\mathcal{M}$-monotone and $\mathcal{M}$-selective, this skeleton suffices to build both strategies (this is transparent in the following proof).

We plan to follow the proof technique of Gimbert and Zielonka. However, in the memoryless case the inductive argument is deployed over the number of choices available in an arena. Now notice that in the context of memoryless strategies, the arena itself contains already all the information about de memory structure. In our case this argument breaks since the arena displays only one aspect of a finite memory strategy, but the memory states are not present. Thus we need to extend the reasoning and build an inductive argument over the product of the arena with the memory structure. Remember that such a strategy uses this information to decide the next state where the play should move.

We overcome this difficulty using a very simple idea; Both players have a finite memory optimal strategy if they have memoryless optimal strategies for $\sqsubseteq$ and $\sqsubseteq^{-1}$ when playing in any arena which is already synchronized with the memory skeleton $\mathcal{M}$. However, one should pay attention to the fact that characterization of Gimbert and Zielonka does not apply. This follows by observing that the quantifiers are not applied over the same objects. In the former case (the memory case), only arenas that are synchronized with $\mathcal{M}$ are considered. In the latter case (the memoryless case), all the arenas are considered. Thus the first technical challenge to overcome is to identify and formalize the properties of an arena which is already synchronized with $\mathcal{M}$. We shall call any arena that enjoys these properties a covered arena.

Covered arenas

In this section, we present the notion of covered arena which is a core notion in building our inductive argument for obtaining the proof of Theorem 1.21. But we can already touch on the intuition. As displayed in Lemma 1.22 and Lemma 1.23, the arguments were carried over plays that contains information about the memory structure. In particular, the memory state reached after a finite (cyclic) play. For an arena to be covered, it should exhibit the two following properties:

Definition 1.25 (Prefix-covers)**.** Let $\mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$ be a memory skeleton and $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E})$ be an arena. Let $\mathsf{S}_{\mathsf{cov}} \subseteq \mathsf{S}$.

We say that $\mathcal{M}$ is a *prefix-cover* of $\mathsf{S}_{\mathsf{cov}}$ in $\mathcal{A}$ if for all $s \in \mathsf{S}$, there exists $m_s \in \mathsf{M}$ such that, for all $\rho \in \mathsf{Hists}(\mathcal{A})$ such that $\mathsf{src}(\rho) \in \mathsf{S}_{\mathsf{cov}}$, $\mathsf{trgt}(\rho) = s$ and such that for all ρ' proper prefix of ρ, $\mathsf{trgt}(\rho') \neq s$, we have $\widehat{\mathsf{upd}}(m_{\mathsf{init}}, \widehat{\mathsf{col}}(\rho)) = m_s$.

Definition 1.26 (Cyclic-covers)**.** Let $\mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$ be a memory skeleton and $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E})$ be an arena. Let $\mathsf{S}_{\mathsf{cov}} \subseteq \mathsf{S}$.

We say that $\mathcal{M}$ is a *cyclic-cover* of $\mathsf{S}_{\mathsf{cov}}$ in $\mathcal{A}$ if for all $\rho \in \mathsf{Hists}(\mathcal{A})$ such that $\mathsf{src}(\rho) \in \mathsf{S}_{\mathsf{cov}}$, if $s = \mathsf{trgt}(\rho)$ and $m = \widehat{\mathsf{upd}}(m_{\mathsf{init}}, \widehat{\mathsf{col}}(\rho))$, for all $\rho' \in \mathsf{Hists}(\mathcal{A})$ such that $\mathsf{src}(\rho') = \mathsf{trgt}(\rho') = s$, $\widehat{\mathsf{upd}}(m, \widehat{\mathsf{col}}(\rho')) = m$.

Intuitively, $\mathcal{M}$ is a prefix-cover for a set of states $\mathsf{S}_{\mathsf{cov}}$ if the histories starting in $\mathsf{S}_{\mathsf{cov}}$ and visiting a given state $s \in \mathsf{S}$ for the first time are read up to the same memory state in the memory skeleton. Similarly, $\mathcal{M}$ is a cyclic-cover of $\mathcal{A}$ if the cycles[1] of $\mathcal{A}$ are read as cycles in the memory skeleton, once the memory has been initialized properly.

The most natural example of a covered arena is the product of $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E})$ and the associated memory skeleton $\mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$. For instance, $\mathcal{M}$ is both a prefix-cover and a cyclic-cover of the set $\mathsf{S} \times \{m_{\mathsf{init}}\}$ in the product arena.

With these new notions in mind, we can now claim the following lemma on which the proof of Lemma 1.24 revolves:

Lemma 1.27. *Let $\sqsubseteq$ be a preference relation and $\mathcal{M}_1^{\mathsf{p}}$, $\mathcal{M}_2^{\mathsf{p}}$, $\mathcal{M}_1^{\mathsf{c}}$ and $\mathcal{M}_2^{\mathsf{c}}$ be four memory skeletons. Assume that $\sqsubseteq$ is $\mathcal{M}_1^{\mathsf{p}}$-monotone and $\mathcal{M}_1^{\mathsf{c}}$-selective, and that $\sqsubseteq^{-1}$ is $\mathcal{M}_2^{\mathsf{p}}$-monotone and $\mathcal{M}_2^{\mathsf{c}}$-selective. Then, for all arenas $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E})$, for all subsets of states $\mathsf{S}_{\mathsf{cov}} \subseteq \mathsf{S}$ for which $\mathcal{M}_1^{\mathsf{p}}$ and $\mathcal{M}_2^{\mathsf{p}}$ are prefix-covers, and $\mathcal{M}_1^{\mathsf{c}}$ and $\mathcal{M}_2^{\mathsf{c}}$ are cyclic-covers, there exists memoryless optimal strategies $(\sigma_1, \sigma_2) \in \Sigma_1^{\mathsf{ML}}(\mathcal{A}) \times \Sigma_2^{\mathsf{ML}}(\mathcal{A})$ from $\mathsf{S}_{\mathsf{cov}}$ in $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$.*

Note that the above lemma is slightly more general. This statement captures arenas that are covered by design.

Lemma 1.27 relies on the usual inductive argument revisited in the context of covered arena, i.e., if a player can optimally using memoryless strategies in small and covered arenas, then the same hold in larger covered arenas. The crucial induction step is based on the following lemma:

[1]can be equivalently stated by considering simple cycles only.

Lemma 1.28. *Let $\sqsubseteq$ be a preference relation, $\mathcal{M}^{\mathsf{p}}$ and $\mathcal{M}^{\mathsf{c}}$ be two memory skeletons, and $\mathfrak{A}$ be a set of arenas closed under sub-arena operation. Assume that $\sqsubseteq$ is $\mathcal{M}^{\mathsf{p}}$-monotone and $\mathcal{M}^{\mathsf{c}}$-selective, and that for all $\mathcal{P}_2$'s one-player arenas $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E}) \in \mathfrak{A}$, for all subsets of states $\mathsf{S}_{\mathsf{cov}} \subseteq \mathsf{S}$ for which $\mathcal{M}^{\mathsf{p}}$ is a prefix-cover and $\mathcal{M}^{\mathsf{c}}$ is a cyclic-cover, $\mathcal{P}_2$ has an optimal strategy from $\mathsf{S}_{\mathsf{cov}}$.*

Let $n \in \mathbb{N}$. Assume that for all arenas $\mathcal{A}' = (\mathsf{S}'_1, \mathsf{S}'_2, \mathsf{E}') \in \mathfrak{A}$ such that $n_{\mathcal{A}'} < n$, for all subsets of states $\mathsf{S}'_{\mathsf{cov}} \subseteq \mathsf{S}'$ for which $\mathcal{M}^{\mathsf{p}}$ is a prefix-cover and $\mathcal{M}^{\mathsf{c}}$ is a cyclic-cover, there exists memoryless optimal strategies $(\sigma'_1, \sigma'_2) \in \Sigma_1^{\mathsf{ML}}(\mathcal{A}') \times \Sigma_2^{\mathsf{ML}}(\mathcal{A}')$ from $\mathsf{S}'_{\mathsf{cov}}$ in $\mathcal{G}' = (\mathcal{A}', \sqsubseteq)$.

Then, for all arenas $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{E}) \in \mathfrak{A}$ such that $n_{\mathcal{A}} = n$, for all subsets of states $\mathsf{S}_{\mathsf{cov}} \subseteq \mathsf{S}$ for which $\mathcal{M}^{\mathsf{p}}$ is a prefix-cover and $\mathcal{M}^{\mathsf{c}}$ is a cyclic-cover, there exists an optimal strategy $\sigma_1 \in \Sigma_1^{\mathsf{ML}}(\mathcal{A})$ from $\mathsf{S}_{\mathsf{cov}}$ in $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$ such that σ_1 is memoryless.

To wrap up, it remains to apply Lemma 1.27 over the arena $\mathcal{A} \times \mathcal{M}_1^{\mathsf{p}} \otimes \mathcal{M}_2^{\mathsf{p}} \otimes \mathcal{M}_1^{\mathsf{c}} \otimes \mathcal{M}_2^{\mathsf{c}}$. This arena is covered since the joint skeleton $\mathcal{M}_1^{\mathsf{p}} \otimes \mathcal{M}_2^{\mathsf{p}} \otimes \mathcal{M}_1^{\mathsf{c}} \otimes \mathcal{M}_2^{\mathsf{c}}$ is a valid skeleton. Thus we build a memoryless optimal strategy from all the states tagged with the initial state of the joint memory skeleton. This yield a UFM strategy for both players since Lemma 1.27 applies symmetrically for $\mathcal{P}_2$ and $\sqsubseteq^{-1}$.

Lifting corollary

We conclude this part by stating the lifting corollary in the context of arena-independent finite memory strategies.

Corollary 1.29. *Let $\sqsubseteq$ be a preference relation and $\mathcal{M}_1, \mathcal{M}_2$ be two memory skeletons. Assume that*

1. *for all one-player arenas $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2 = \emptyset, E)$, $\mathcal{P}_1$ has a UFM strategy $\sigma_1 \in \Sigma_1^{\mathsf{FM}}(\mathcal{A})$ based on memory skeleton $\mathcal{M}_1$ in $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$;*
2. *for all one-player arenas $\mathcal{A} = (\mathsf{S}_1 = \emptyset, \mathsf{S}_2, E)$, $\mathcal{P}_2$ has a UFM strategy $\sigma_2 \in \Sigma_2^{\mathsf{FM}}(\mathcal{A})$ based on memory skeleton $\mathcal{M}_2$ in $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$.*

Then, for all two-player arenas $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, E)$, both $\mathcal{P}_1$ and $\mathcal{P}_2$ have UFM strategies $\sigma_i \in \Sigma_i^{\mathsf{FM}}(\mathcal{A})$ based on memory skeleton $\mathcal{M} = \mathcal{M}_1 \otimes \mathcal{M}_2$ in $\mathcal{G} = (\mathcal{A}, \sqsubseteq)$.

The lifting corollary follows almost readily from the previous lemmas. First notice that applying the lemma used in left-to-right implication of Theorem 1.21, i.e., Lemma 1.22 and Lemma 1.23 implies that $\sqsubseteq$ and $\sqsubseteq^{-1}$ are $\mathcal{M}$-monotone and $\mathcal{M}$-selective. Now using the right-to-left implication of Theorem 1.21, i.e., Lemma 1.24 yields the corollary.

1.5 Playing optimally with memory in stochastic games

In this section, we discuss extensions of the above results in the setting of stochastic games. That is, what are the preference relations that admit a uniform optimal strategy with arena independent memory over any stochastic arena. We also study the possible liftings from one to two players stochastic games.

Before reporting the actual extensions, we introduce some notions and notations in order to extend the previous framework.

The first being stochastic arenas and games. For a measurable space $(\Omega, \mathcal{F})$ (resp. a finite set Ω), we write $\mathsf{Dist}(\Omega, \mathcal{F})$ (resp. $\mathsf{Dist}(\Omega)$) for the set of probability distributions on $(\Omega, \mathcal{F})$ (resp. on Ω). For Ω a finite set and $\mu \in \mathsf{Dist}(\Omega)$, we write $\mathsf{Supp}(\mu) = \{\omega \in \Omega \mid \mu(\omega) > 0\}$ for the support of μ.

A (two-player stochastic turn-based) arena is a tuple $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{A}, \delta, \mathsf{col})$, where: S_1, S_2, and col remain unchanged, cf. Section 1.2. A is a finite set of actions; $\delta \colon \mathsf{S} \times \mathsf{A} \to \mathsf{Dist}(\mathsf{S})$ is a partial function called probabilistic transition function; For a state $s \in \mathsf{S}$, we write $\mathsf{A}(s)$ for the set of actions that are available in s, that is, the set of actions for which $\delta(s, a)$ is defined. For $s \in \mathsf{S}$, function col must be defined for all pairs (s, a) such that a is available in s. We require that for all $s \in \mathsf{S}$, $\mathsf{A}(s) \neq \emptyset$ (i.e., arenas are non-blocking).

A one-player arena of $\mathcal{P}_i$ in this case is an arena $\mathcal{A} = (\mathsf{S}_1, \mathsf{S}_2, \mathsf{A}, \delta, \mathsf{col})$ such that for all $s \in \mathsf{S}_{3-i}$, $|\mathsf{A}(s)| = 1$. A one-player arena corresponds to a Markov decision process (MDP). The interested reader my refer to [Put14, BK08] or to later chapters for specific notations. However, for the time being we do not require formal details in the overview presented in the sequel.

The notion of preferences used so far is extended over $\mathsf{Dist}(\mathsf{C}^\omega, \mathcal{F})$, thus $\sqsubseteq$ is a total preorder over distributions in $\mathsf{Dist}(\mathsf{C}^\omega, \mathcal{F})$. For instance, in the context of two-player arenas defined in the beginning of this chapter, elements of $\mathsf{Dist}(\mathsf{C}^\omega, \mathcal{F})$ coincide with Dirac distributions and thus with infinite words over C.

Strategies are extended as follows: for $i \in \{1, 2\}$, a strategy of $\mathcal{P}_i$ on $\mathcal{A}$ from a state s is a function $\sigma_i \colon s(\mathsf{AS})^* \to \mathsf{Dist}(\mathsf{A})$ such that for all $\rho \in s(\mathsf{AS})^*$ we have that $\mathsf{Last}(\rho) \in \mathsf{S}_i$, and $\mathsf{Supp}(\sigma_i(\rho)) \subseteq \mathsf{A}(\mathsf{Last}(\rho))$, where $\mathsf{Last}(\rho)$ denotes the last state of ρ. According to this new definition, notice that we allow randomization in the choice of the next action. For $i \in \{1, 2\}$, we denote by $\Sigma_i^{\mathsf{G}}(\mathcal{A}, s)$ the set of all strategies of $\mathcal{P}_i$ on $\mathcal{A}$ from s.

The notion of Memory remains unchanged and does not involve any sort of randomization. We denote by $\Sigma_i^{\mathsf{PFM}}(\mathcal{A}, s)$ (resp. $\Sigma_i^{\mathsf{P}}(\mathcal{A}, s)$, $\Sigma_i^{\mathsf{GFM}}(\mathcal{A}, s)$, $\Sigma_i^{\mathsf{G}}(\mathcal{A}, s)$) the set of pure finite-memory (resp. pure, finite-memory, general) strategies of $\mathcal{P}_i$ on $(\mathcal{A}, s)$. A type of strategies is an element $\mathsf{X} \in \{\mathsf{PFM}, \mathsf{P}, \mathsf{GFM}, \mathsf{G}\}$ corresponding to these subsets.

One-to-two-player lift

As explained earlier, results establishing bridges between the one-player case and the two player case are much appreciated. Indeed, it is usually a much simpler case to study a one-player game rather than a two-player one. In [GZ05], the Lifting corollary for memoryless optimal strategies in deterministic games was established. In [Gim07] a version for MDPs was presented. In the previous section, we presented an extension of [GZ05] to arena-independent memory case. Next, we present a generalization that unifies all the latter mentioned results. Moreover, our unified lift captures more instances as it deals with both stochastic arena and arena-independent memory. The proof techniques originates from an adaption of the proof technique of [Gim07] to the case of covered arena by using the knowledge gained by the new insights provided by the previous section.

Theorem 1.30 (Pure AIFM one-to-two-player lift)**.** *Let $\sqsubseteq$ be a preference relation, $\mathcal{M}_1$ and $\mathcal{M}_2$ be two memory skeletons, and $\mathsf{X} \in \{\mathsf{PFM}, \mathsf{P}, \mathsf{GFM}, \mathsf{G}\}$ be a type of strategies. Let $\mathfrak{A}$ be the class of all stochastic or deterministic arenas.*

Assume that:

i. in all one-player arenas of $\mathcal{P}_1$ in $\mathfrak{A}$, $\mathcal{P}_1$ can play X-optimally with a pure uniform strategy based on $\mathcal{M}_1$,

ii. in all one-player arenas of $\mathcal{P}_2$ in $\mathfrak{A}$, $\mathcal{P}_2$ can play X-optimally with a pure uniform strategy based on $\mathcal{M}_2$.

Then in all two-player arenas in $\mathfrak{A}$, both players have a pure uniform X strategy based on $\mathcal{M}_1 \otimes \mathcal{M}_2$.

Characterization

For this last part, we fix $\sqsubseteq$ a preference relation, $\mathsf{X} \in \{\mathsf{PFM}, \mathsf{P}, \mathsf{GFM}, \mathsf{G}\}$ a type of strategies, and $\mathcal{M} = \mathcal{M} = (\mathsf{M}, m_{\mathsf{init}}, \mathsf{upd})$ a memory skeleton. We distinguish two classes of arenas: the class $\mathfrak{A}^{\mathsf{D}}_{\mathcal{P}_1}$ of all one-player deterministic arenas of $\mathcal{P}_1$, and the class $\mathfrak{A}^{\mathsf{S}}_{\mathcal{P}_1}$ of all one-player stochastic arenas of $\mathcal{P}_1$.

A class of arenas will therefore be specified by a letter $\mathsf{Y} \in \{\mathsf{D}, \mathsf{S}\}$, which we fix for the whole section.

We denote by $\mathbb{P}^{\sigma_1}_{\mathcal{A},s}$ and $\mathsf{Pc}^{\sigma_1}_{\mathcal{A},s}$ the distributions over plays and colors induced by a strategy σ_1 of $\mathcal{P}_1$ in $\mathcal{A}$ from a state s, where $\mathcal{P}_2$ is not involved.

For $\mathcal{A} \in \mathfrak{A}^{\mathsf{Y}}_{\mathcal{P}_1}$ and s a state of $\mathcal{A}$, we write

$$[\mathcal{A}]^{\mathsf{X}}_s = \{\mathsf{Pc}^{\sigma_1}_{\mathcal{A},s} \mid \sigma_1 \in \Sigma^{\mathsf{X}}_1(\mathcal{A}, s)\}$$

for the set of all distributions over $(\mathsf{C}^\omega, \mathcal{F})$ induced by strategies of type X in $\mathcal{A}$ from s.

For a distribution μ and a word w, we call $w\mu$ the shifted distribution, i.e., a distribution such that for an event $E \in \mathcal{F}$, $w\mu(E) = \mu(\{w' \in \mathsf{C}^\omega \mid ww' \in E\})$. We extend this notation to sets of distributions as follows: for $w \in \mathsf{C}^*$, for $\Lambda \subseteq \mathsf{Dist}(\mathsf{C}^\omega, \mathcal{F})$, we write $w\Lambda$ for the set $\{w\mu \mid \mu \in \Lambda\}$.

Definition 1.31 (General monotony)**.** We say that $\sqsubseteq$ is X-Y-$\mathcal{M}$-***monotone*** if for all $m \in \mathsf{M}$, for all $(\mathcal{A}_1, s_1), (\mathcal{A}_2, s_2) \in \mathfrak{A}^{\mathsf{Y}}_{\mathcal{P}_1}$, there exists $i \in \{1, 2\}$ s.t. for all

$$w \in L_{m_{\mathsf{init}},m}, w[\mathcal{A}_{3-i}]^{\mathsf{X}}_{s_{3-i}} \sqsubseteq w[\mathcal{A}_i]^{\mathsf{X}}_{s_i} \enspace .$$

Definition 1.32 (General selectivity)**.** We say that $\sqsubseteq$ is X-Y-$\mathcal{M}$-***selective*** if for all $m \in \mathsf{M}$, for all $(\mathcal{A}_1, s_1), (\mathcal{A}_2, s_2) \in \mathfrak{A}^{\mathsf{Y}}_{\mathcal{P}_1}$ such that for $i \in \{1, 2\}$, $\widehat{\mathsf{colHists}}(\mathcal{A}_i, s_i, s_i) \subseteq L_{m,m}$, for all $w \in L_{m_{\mathsf{init}},m}$,

$$w[(\mathcal{A}_1, s_1) \sqcup (\mathcal{A}_2, s_2)]^{\mathsf{X}}_t \sqsubseteq w[\mathcal{A}_1]^{\mathsf{X}}_{s_1} \cup w[\mathcal{A}_2]^{\mathsf{X}}_{s_2} \enspace .$$

We use the following property that could essentially be understood as a sufficiency of pure strategies in the sequel.

Definition 1.33 (Mixing is useless)**.** We say that ***mixing is useless for*** $\sqsubseteq$ if for any at most countable set I, for all reals $(\lambda_i)_{i \in I}$ such that $\sum_{i \in I} \lambda_i = 1$, for all families $(\mu_i)_{i \in I}$, $(\mu'_i)_{i \in I}$ of distributions in $\mathsf{Dist}(\mathsf{C}^\omega, \mathcal{F})$:

$$\forall i \in I,\ \mu_i \sqsubseteq \mu'_i,\ \Longrightarrow\ \sum_{i \in I} \lambda_i \mu_i \sqsubseteq \sum_{i \in I} \lambda_i \mu'_i \enspace .$$

Note that usual payoff functions and preferences enjoy the above property.

Theorem 1.34. *For $\mathsf{X} \in \{\mathsf{P}, \mathsf{PFM}\}$ and $\mathsf{Y} = \mathsf{D}$, or assuming that mixing is useless for $\sqsubseteq$. Then pure strategies based on $\mathcal{M}$ suffice to play X-optimally in all one-player arenas in $\mathfrak{A}^{\mathsf{Y}}_{\mathcal{P}_1}$ for $\mathcal{P}_1$ if and only if $\sqsubseteq$ is X-Y-$\mathcal{M}$-monotone and X-Y-$\mathcal{M}$-selective.*

1.6 Discussion

Wrap-up

In this chapter we concerned ourselves with the memory requirement in an optimal controller in a zero-sum game. The main result, cf. Theorem 1.21, exhibits a characterization of the preference relations admitting optimal controllers with an arena-independent finite memory. This result falls in the same line of results already initiated by Gimbert and Zielonka [GZ05], where they introduced two properties that allow the same kind of characterization but in the context of memoryless strategies, cf. Definition 1.9, and Definition 1.10. We followed the footsteps of Gimbert and Zielonka and defined two properties that are independent of the arena that allow a characterization in the more general setting of memory strategy. However, due to the unavoidable coupling between a strategy and the memory it uses, we had to introduce properties that account for this link with the memory, cf., Definition 1.18, and Definition 1.19. On the top of this characterization, we showed that the formal tool introduced by Gimbert and Zielonka, the so-called lifting corollary holds in the setting of arena-independent finite memory strategies. We recall that thanks to this corollary, the difficulty of establishing finite memory drops from two-player to one-player games, cf. Corollary 1.29. We also presented an example establishing the impossibility of having a one-to-two-player lift in the setting of general finite memory strategies, cf. Example 1.15.

Finally, we extended the above results to the more general setting of stochastic arenas. In the frame work of stochastic games, we presented a general notion of monotony, selectivity, and characterization in the setting Markov decision processes, cf. Theorem 1.34. We also managed to establish a general lift, cf. 1.30.

Perspectives

A natural perspective would be to consider arena-dependent memories. Obviously, one should not expect any positive results regarding lifts from one-player arenas. Nevertheless, Kozachinskiy investigates the exact frontier of when the lift breaks. Does it break for any memory or it is possible to find some classification in there? In [Koz22], Kozachinskiy introduce the notion of mildly growing memory strategies, these are strategies that use a memory of size at most sublinear in the size of the arena (let us call this property, the sublinearity hypothesis) Roughly speaking, the results he obtains are an attempt to explain why the lift breaks in the setting of arena-dependent memories. He shows that when the sublinearity hypothesis holds, then the lift holds. Unfortunately, he fails to show that the lift necessarily fails when the sublinearity hypothesis is not guaranteed, actually known examples from the literature present payoff function that do not fulfil the sublinearity hypothesis and admit finite memory strategies [JLS15b, VCD$^+$15b]. This tends to show that there still is room to improve our understanding in this direction.

Another direction that one could head towards consists in investigating lifts only from the perspective of one player. Let us provide some context here, in [GK14] the following lift is shown:

- If $\mathcal{P}_2$ has finite memory strategies in all his one-player arena.
- If the preference relation is prefix-independent and concave[2].

Then $\mathcal{P}_2$ has a finite memory in all arenas.

Possible future works are to investigate generalizations of the above one-sided lift.

[2]also called submixing, a condition that deals with the shuffle of plays

Chapter 2

Synthesis through the window

2.1 Outline of the chapter

In this chapter we are interested in Markov decision processes (MDP) equipped with either parity objectives or mean-payoff objectives. In the former objective; we label the state of an MDP with non negative integers (called priorities), and the goal is to build infinite runs where the minimal integer seen infinitely often is even. In the latter objective; we label the action with integers, and the goal is to build infinite runs where the limit of mean accumulated value is non negative. These objective are expressive enough to capture a wide variety of specification [BK08]. We study their robustness properties in the long term. In fact both these behaviors display some shortcomings. Indeed, they exhibit some unwanted behavior in practice due to the fact that they are evaluated asymptotically without ensuring a good behavior locally or even in the short term. To grasp some intuition of this, consider the following example.

Example 2.1. In Figure 2.1 we depict a very simple MDP, actually a Markov chain, equipped with a parity objective. In this example, almost-any infinite run satisfies the parity objective. Notice however, that for any fixed frame of time t, there is a positive probability that an even priority is not seen for t consecutive steps in state s_1. This actually holds true for an arbitrary large time frame t. ◁

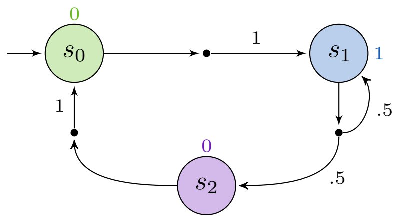


Figure 2.1: A Markov chain with parity objective.

The previous example shows a behavior that could be interpreted as not robust, the window mechanism was introduced to deal with these issues. This mechanism is defined with respect to two variants. In the fixed variant, one considers a window of size bounded by $\lambda \in \mathbb{N}_0$ (given as a parameter) sliding over an infinite run. A run is winning if, in all positions, the window is such that the (mean-payoff or parity) objective is locally satisfied. In the bounded variant, the window size is not fixed a priori, but a run is winning if there exists a bound λ for which the condition holds. Window mechanism have been considered both in what is called direct versions, where the window property must hold from the start of the run, and prefix-independent versions, where it must hold from some point on. Window games were initially studied for mean-payoff [CDRR15] and parity [BHR16a]. They have since seen diverse extensions and applications: e.g., [BKKW14, Bai15, BFKN16, BHR16b, HPR18, RPR18].

Example 2.2. Consider again the Markov chain from figure 2.1. Now, consider the ***window parity*** objective that informally asks for the minimum priority inside a window of size bounded by λ to be even, with this window sliding all along the infinite run. Fix any $\lambda \in \mathbb{N}_0$. It is clear that every time s_1 is visited, there will be a positive probability $\varepsilon > 0$ of not seeing 0 before λ steps: this probability is $1/2^{\lambda-1}$. Let us call this seeing a ***bad window***. Since we are in a bottom strongly connected component of the Markov chain, we will almost-surely visit s_1 infinitely often [BK08]. Using classical probability arguments (Borel-Cantelli), one can easily be convinced that the probability to see bad windows infinitely often is one. Hence the probability to win the window parity objective is zero. This example illustrates the difference between traditional parity and window parity: the latter is more restrictive since it asks for a strict bound on the time frame in which each odd priority should be answered by a smaller even priority. Hence enforcing desirable and robust behaviors. $\triangleleft$

In this chapter we overview contributions from a collaboration with Thomas Brihaye, Florent Delgrange, and Mickael Randour [BDOR19, BDOR20], we present the following results:

- In Lemma 2.12, we relate the probability of satisfying a direct (resp. prefix independent) fixed window objective with the probability of satisfying a safety (resp. co-Büchi) objective in a larger Markov decision process.
- In Theorem 2.13, we derive memory requirement for winning strategies.
- In Theorem 2.14, we derive complexity upper-bounds.
- In Theorem 2.16, we present complexity lower-bounds.
- We study properties of the window mechanism inside end-components. Precisely we establish a zero-one law in Lemma 2.21 and introduce a classification for end-components.
- In Theorem 2.28, we establish complexity bounds for this classification.
- Finally in Theorem 2.30, we solve the general case of Markov decision processes and give complexity and memory bounds for winning strategies.

This chapter is organized as follows:

- In Section 2.2, we present notations and notions related to the formalism of interest in this chapter, i.e., Markov decision processes.
- In Section 2.3, we present the window mechanism and introduce the statement of problems related to this mechanism in Markov decision processes.
- In Section 2.4, we develop a solution to fixed window mechanism. Our solution is based on an unfolding technique that solves the direct and the prefix-independent version of the fixed window mechanism.
- In Section 2.5, we study the special case of end-components and establish their properties.
- In Section 2.6, we take advantage of the properties of end-components and design algorithms for Markov decision processes with window mechanism.

2.2 Markov decision processes

Recall that for a given set S, $\mathsf{Dist}(S)$ denotes the set of rational probability distributions over S. Given a distribution $\iota \in \mathsf{Dist}(S)$, $\mathsf{Supp}(\iota) = \{s \in S \mid \iota(s) > 0\}$ denotes its support.

A finite Markov decision process (MDP) is a tuple $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$ where S is a finite set of states, A is a finite set of actions and $\delta \colon \mathsf{S} \times \mathsf{A} \to \mathsf{Dist}(\mathsf{S})$ is a partial function called the probabilistic transition function.

The set of actions that are available in a state $s \in \mathsf{S}$ is denoted by $\mathsf{A}(s)$. We use $\delta(s, a, s')$ as a shorthand for $\delta(s, a)(s')$. We assume w.l.o.g. that MDPs are deadlock-free: for all $s \in \mathsf{S}$, $\mathsf{A}(s) \neq \emptyset$. An MDP where for all $s \in \mathsf{S}$, $|\mathsf{A}(s)| = 1$ is a fully-stochastic process called a Markov chain (MC).

A play of $\mathcal{M}$ is an infinite sequence $\pi = s_0 a_0 \dots a_{n-1} s_n \dots$ of states and actions such that $\delta(s_i, a_i, s_{i+1}) > 0$ for all $i \geq 0$. The prefix up to the n-th state of π is the finite sequence $\pi[..n] = s_0 a_0 \dots a_{n-1} s_n$. The suffix of π starting from the n-th state of π is the run $\pi[n..] = s_n a_n s_{n+1} a_{n+1} \dots$. Moreover, we denote by $\pi[n]$ the n-th state s_n of π. Finite prefixes of plays of the form $h = s_0 a_0 \dots a_{n-1} s_n$ are called histories. We sometimes denote the last state of history h by $\mathsf{Last}(h)$. We resp. denote the sets of runs and histories of an MDP $\mathcal{M}$ by $\mathsf{Plays}(\mathcal{M})$ and $\mathsf{Hists}(\mathcal{M})$.

Fix an MDP $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$. A sub-MDP of $\mathcal{M}$ is an MDP $\mathcal{M}' = (\mathsf{S}', \mathsf{A}', \delta')$ with $\mathsf{S}' \subseteq \mathsf{S}$, $\emptyset \neq \mathsf{A}'(s) \subseteq \mathsf{A}(s)$ for all $s \in \mathsf{S}'$, $\mathsf{Supp}(\delta(s, a)) \subseteq \mathsf{S}'$ for all $s \in \mathsf{S}', a \in \mathsf{A}'(s)$, $\delta' = \delta|_{\mathsf{S}' \times \mathsf{A}'}$. Such a sub-MDP $\mathcal{M}'$ is an end-component (EC) of $\mathcal{M}$ if and only if the underlying graph of $\mathcal{M}'$ is strongly connected, i.e., there is a run between any pair of states in S'. Given an EC $\mathcal{M}' = (\mathsf{S}', \mathsf{A}', \delta')$ of $\mathcal{M}$, we say that its sub-MDP $\mathcal{M}'' = (\mathsf{S}'', \mathsf{A}'', \delta'')$, $\mathsf{S}'' \subseteq \mathsf{S}'$, $\mathsf{A}'' \subseteq \mathsf{A}'$, is a sub-EC of $\mathcal{M}'$ if $\mathcal{M}''$ is also an EC. We let $\mathsf{EC}(\mathcal{M})$ denote the set of ECs of $\mathcal{M}$, which may be of exponential size as ECs need not be disjoint.

The union of two ECs with non-empty intersection is itself an EC: hence we can define the maximal ECs (MECs) of an MDP, i.e., the ECs that cannot be extended. We let $\mathsf{MEC}(\mathcal{M})$ denote the set of MECs of $\mathcal{M}$, of polynomial size (because MECs are pair-wise disjoints) and computable in polynomial time [CH14].

The counterparts of ECs in MCs are bottom strongly-connected components (BSCCs). In our formalism, an MC is simply an MDP $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$ with $|\mathsf{A}(s)| = 1$ for all $s \in \mathsf{S}$, thus BSCCs are exactly the ECs of such an MDP $\mathcal{M}$.

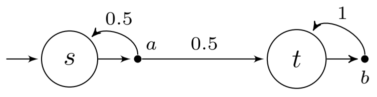


Figure 2.2: A simple MDP.

Example 2.3. Consider the MDP depicted in Figure 2.2. This very simple MDP consists of two states, s and t. The singleton $\{t\}$ is the only EC which is obviously maximal. Notice also that $|\mathsf{A}(s)| = 1$ for all $s \in \mathsf{S}$. Thus, this particular instance is in fact an MC. In particular, as explained, the ECs correspond to BSCCs. $\triangleleft$

Strategies, probability measure, and events

We use the same notion of strategies as the previous chapter, we allow the memory strategies to be arena dependent. For the sake of clarity, we restate the necessary definitions using the formalism of MDPs.

A strategy σ is a function $\mathsf{Hists}(\mathcal{M}) \to \mathsf{Dist}(\mathsf{A})$ such that for all $h \in \mathsf{Hists}(\mathcal{M})$ ending in s, we have $\mathsf{Supp}(\sigma(h)) \subseteq \mathsf{A}(s)$. The set of all strategies is Σ. Recall that a strategy is pure if all histories are mapped to Dirac distributions, i.e., the support is a singleton.

A strategy σ can be encoded by a Moore machine $(Q, \sigma_n, \sigma_u, \iota)$ where Q is a finite or infinite set of memory elements, ι the initial distribution on Q, σ_n the next action function $\sigma_n : \mathsf{S} \times Q \to \mathsf{Dist}(\mathsf{A})$ where $\mathsf{Supp}(\sigma_n(s,q)) \subseteq \mathsf{A}(s)$ for any $s \in \mathsf{S}$ and $q \in Q$, and σ_u the memory update function $\sigma_u : \mathsf{A} \times \mathsf{S} \times Q \to Q$.

We say that σ is finite-memory if $|Q| < \infty$, and K-memory if $|Q| = K$; it is memoryless if $K = 1$, thus only depends on the last state of the history. We see such strategies as functions $s \mapsto \mathsf{Dist}(\mathsf{A}(s))$ for $s \in \mathsf{S}$. When Q is infinite, σ is infinite-memory. The entity choosing the strategy is often called the controller.

An MDP $\mathcal{M}$, a strategy σ encoded by $(Q, \sigma_a, \sigma_u, \iota)$, and a state s induce a Markov chain $\mathcal{M}_s^\sigma$ and a probability measure $\mathbb{P}^\sigma_{\mathcal{M},s}$.

$\mathcal{M}_s^\sigma$ is defined as follows:

- State space is $\mathsf{S} \times Q$.
- The initial distribution $\iota^\sigma_{\mathcal{M},s}$ equals $\iota(q)$ for any state (s,q) where $q \in Q$ and 0 otherwise.
- The probabilistic transition function $\delta^\sigma_{\mathcal{M},s} : \mathsf{S} \times Q \times \mathsf{A} \to \mathsf{Dist}(\mathsf{S} \times Q)$ is defined as follows

$$\forall (s,q),(s',q') \in \mathsf{S} \times Q,\ \delta^\sigma_{\mathcal{M},s}((s,q),(s',q'),a) = \begin{cases} \sigma_n(s,q)(a) \cdot \delta(s,a,s') \text{ if } q' = \sigma_u(s,q,a) \\ 0 \text{ otherwise.} \end{cases} ,$$

A run of $\mathcal{M}_s^\sigma$ is an infinite sequence of the form $(s_0,q_0)a_0(s_1,q_1)a_1\ldots$, where each $(s_i,q_i) \xrightarrow{a_i} (s_{i+1},q_{i+1})$ is a transition with non-zero probability in $\mathcal{M}_s^\sigma$, and $s_0 = s$.

$\mathbb{P}^\sigma_{\mathcal{M},s}$ is defined over the Borel σ-algebra induced by the cylinders of $(\mathsf{SA})^\omega$. It is uniquely defined thanks to Carathéodory's extension theorem [ADD99].

Given $E \subseteq (\mathsf{SA})^\omega$, $\mathbb{P}^\sigma_{\mathcal{M},s}[E]$ is the probability of the runs of $\mathcal{M}_s^\sigma$ whose projection[1] over $\mathcal{M}$ is in E, i.e., the probability of event E when $\mathcal{M}$ is executed with initial state s and strategy σ. We may drop some of the subscripts $\mathcal{M}$, s, and σ when the context is clear.

Let $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$ be an MDP, $\sigma \in \Sigma$ be a strategy, and $E \subseteq (\mathsf{SA})^\omega$ be an event, We say that:

- E is sure, denoted $\mathsf{Sure}^\sigma_{\mathcal{M},s}[E]$, if $\mathsf{Plays}(\mathcal{M}_s^\sigma) \subseteq E$.
- E is almost-sure, denoted $\mathsf{ASure}^\sigma_{\mathcal{M},s}[E]$, if $\mathbb{P}^\sigma_{\mathcal{M},s}[E] = 1$.

Let π be a play and E a measurable set of plays, we say that E is prefix-independent if for every $n > 0$ we have that $\pi[n..] \in E$ if and only if $\pi \in E$.

Asymptotic properties of MDPs

Given a run $\rho = s_0a_0s_1a_1\ldots \in \mathsf{Plays}(\mathcal{M})$, let

$$\mathsf{Inf}(\rho) = \{s \in \mathsf{S} \mid \forall\, i \geq 0, \exists\, j > i, s_j = s\} ,$$

denote the set of states visited infinitely-often along ρ, and let

$$\mathsf{infAct}(\rho) = \{a \in \mathsf{A} \mid \forall\, i \geq 0, \exists\, j > i, a_j = a\} ,$$

similarly denote the actions taken infinitely-often along ρ.

Let $\mathsf{limitSet}(\rho)$ denote the pair $(\mathsf{Inf}(\rho), \mathsf{infAct}(\rho))$. Note that this pair may induce a well-defined sub-MDP $\mathcal{M}' = (\mathsf{Inf}(\rho), \mathsf{infAct}(\rho), \delta|_{\mathsf{Inf}(\rho)\times\mathsf{infAct}(\rho)})$. A folk result in MDPs (cf., [BK08]) is the following:

[1]The projection of a run $(s_0,q_0)a_0(s_1,q_1)a_1\ldots$ in $\mathcal{M}_s^\sigma$ to $\mathcal{M}$ is simply the run $s_0a_0s_1a_1\ldots$ in $\mathcal{M}$.

For any state s of an MDP $\mathcal{M}$, for any strategy $\sigma \in \Sigma$, we have:

$$\mathsf{ASure}^{\sigma}_{\mathcal{M},s}[\{\rho \in \mathsf{Plays}(\mathcal{M}^{\sigma}_{s}) \mid \mathsf{limitSet}(\rho) \in \mathsf{EC}(\mathcal{M})\}],$$

that is, under any strategy, the limit behavior of the MDP almost-surely coincides with an EC. This property is a key tool in the analysis of MDPs with prefix-independent events, as it essentially says that we only need to identify the "best" ECs and maximize the probability to reach them.

2.3 Window objectives in Markov decision processes

Objectives

An objective for an MDP $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$ is a measurable set of runs $E \subseteq (\mathsf{SA})^{\omega}$. Given an MDP $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$, an initial state s, a threshold $\alpha \in [0,1] \cap \mathbb{Q}$, and an objective E, the threshold probability problem is to decide whether there exists a strategy $\sigma \in \Sigma$ such that $\mathbb{P}^{\sigma}_{\mathcal{M},s}[E] \geq \alpha$. Furthermore, if it exists, we want to build such a strategy. In the remaining of this chapter, we always assume an MDP $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$ with either

- a weight function $\mathsf{wght} \colon \mathsf{A} \to \mathbb{Z}$ where the largest absolute weight is denoted W, or
- a priority function $\mathsf{prty} \colon \mathsf{S} \to \{0, 1, \dots, d\}$, with $d \leq |\mathsf{S}| + 1$ (w.l.o.g.).

This choice is left implicit when the context is clear.

When studying the complexity of decision problems, we make the classical assumptions:

- The size of the model $|\mathcal{M}|$ is polynomial in $|S|$.
- The weights and probabilities are encoded in binary.
- The largest priority d, as well as the upcoming window size λ, are encoded in unary.
- When a problem is polynomial in W, we say that it is pseudo-polynomial, i.e., it would be polynomial if weights were given in unary.

Parity objectives

Consider an MDP $\mathcal{M}$ with a priority function prty. The parity objective requires that the smallest priority seen infinitely often along a run be even, i.e.:

$$\mathsf{Parity} = \{\rho \in \mathsf{Plays}(\mathcal{M}) \mid \min_{s \in \mathsf{Inf}(\rho)} \mathsf{prty}(s) = 0 \pmod 2\} \ .$$

The corresponding threshold probability problem is in PTIME and pure memoryless strategies suffice [CJH04].

Mean-payoff objectives

Consider a weighted MDP $\mathcal{M}$. Let $\rho \in \mathsf{Plays}(\mathcal{M})$ be a run of $\mathcal{M}$. The mean payoff of a run $\rho = s_0 a_0 s_1 a_1 \ldots \in \mathsf{Plays}(\mathcal{M})$ denoted $\mathsf{MP}(\rho)$ is the limit of the average of the accumulation of weights along ρ, i.e.:

$$\mathsf{MP}(\rho) = \liminf_{n} \frac{1}{n+1} \sum_{i=0}^{n} \mathsf{wght}(a_i) \ .$$

Given a threshold $\nu \in \mathbb{Q}$, the mean-payoff objective accepts all runs whose mean-payoff is above ν, i.e.:

$$\mathsf{MeanPayoff}_\nu = \{\rho \in \mathsf{Plays}(\mathcal{M}) \mid \mathsf{MP}(\rho) \geq \nu\} \ .$$

The corresponding threshold probability problem is in PTIME using linear programming, and pure memoryless strategies suffice (see, e.g., [RRS17]). Note that ν can be taken equal to zero w.l.o.g., and the mean-payoff function can be equivalently defined using $\limsup$[2].

Window objectives

Good window

Given an MDP $\mathcal{M}$ with priority function p, we define the good window parity objective,

$$\mathsf{GW}_{\mathsf{par}}(\lambda) = \Big\{\rho \in \mathsf{Plays}(\mathcal{M}) \mid \exists\, l < \lambda,\ \big(p(\rho[l]) \bmod 2 = 0 \wedge \forall\, k < l,\ p(\rho[l]) < p(\rho[k])\big)\Big\}$$

requiring the existence of a window of size bounded by λ and starting at the first position of the run, for which the last priority is even and is the smallest within the window.

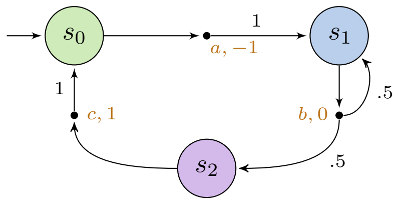
Similarly, given a weighted MDP $\mathcal{M}$ and $\lambda > 0$, we define the good window mean-payoff objective

$$\mathsf{GW}_{\mathsf{mp}}(\lambda) = \{\rho \in \mathsf{Plays}(\mathcal{M}) \mid \exists\, l < \lambda,\ \mathsf{MP}\big(\rho[0, l+1]\big) \geq 0\}$$

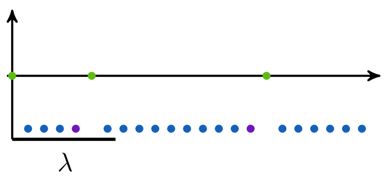
requiring the existence of a window of size bounded by λ and starting at the first position of the run, over which the mean-payoff is at least equal to zero (w.l.o.g.).

For the sake of factorization, we use subscripts mp and par for mean-payoff and parity variants respectively. Using this new notation, let Ω in $\{\mathsf{mp}, \mathsf{par}\}$ and a run $\rho \in \mathsf{Plays}(\mathcal{M})$, we say that an Ω-window is closed in at most λ steps from $\rho[i]$ if $\rho[i, \infty]$ is in $\mathsf{GW}_\Omega(\lambda)$. If a window is not yet closed, we call it open.

Fixed windows



(a) Mean-payoff MDP.



(b) Accumulation of costs along a play.

Figure 2.3: For any λ, both $\mathsf{DFW}_{\mathsf{mp}}(\lambda)$ and $\mathsf{FW}_{\mathsf{mp}}(\lambda)$ are not satisfied.

Given $\lambda > 0$ and Ω in $\{\mathsf{mp}, \mathsf{par}\}$, the direct fixed window objective, denoted $\mathsf{DFW}_\Omega(\lambda)$, is:

$$\mathsf{DFW}_\Omega(\lambda) = \{\rho \in \mathsf{Plays}(\mathcal{M}) \mid \forall\, j \geq 0,\ \rho[j, \infty] \in \mathsf{GW}_\Omega(\lambda)\} \ ,$$

that is, all the Ω-windows have to be closed within λ steps along the run.

We also define the fixed window objective, denoted $\mathsf{FW}_\Omega(\lambda)$, as follows:

$$\mathsf{FW}_\Omega(\lambda) = \{\rho \in \mathsf{Plays}(\mathcal{M}) \mid \exists\, i \geq 0,\ \rho[i, \infty] \in \mathsf{DFW}_\Omega(\lambda)\} \ ,$$

that is the prefix-independent version of the previous one, i.e., $\mathsf{DFW}_\Omega(\lambda)$ is eventually satisfied along every run.

[2]in the classical one-dimension setting for finite MDPs

Example 2.4. Consider the arena from Figure 2.3a. This is the same Markov chain from Figure 2.1 but this time equipped with weight function. We are dealing with mean-payoff objective now. Let us start by considering the fixed window setting. Fix a window size λ, then by definition of $\mathsf{DFW}_{\mathsf{mp}}(\lambda)$ our goal is to build a play π such that inside every sliding window the mean-payoff is positive at some point. Consider Figure 2.3b where we display the accumulation of weights along a play. From this figure we can get the intuition that in order to close each window, the run has to leave state s_1 before consuming the entire time frame λ. Using classical argument on Markov chains involving Borel Cantelli Lemma, one can show that almost-surely, that a run will stay in state s_1 for a long enough period of time leading to window that never closes. This tends to show that $\mathsf{DFW}_{\mathsf{mp}}(\lambda)$ cannot be satisfied. Moreover, this never closing window can occur at any point in any play, hence $\mathsf{FW}_{\mathsf{mp}}(\lambda)$ cannot be satisfied neither. $\triangleleft$

Bounded windows

The bounded window objective, denoted BW_Ω, is:

$$\mathsf{BW}_\Omega = \{\rho \in \mathsf{Plays}(\mathcal{M}) \mid \exists\, \lambda > 0,\ \rho \in \mathsf{FW}_\Omega(\lambda)\}\ .$$

This objective requires the existence of a bound λ for which the fixed window objective is satisfied. Notice that this bound need not be uniform along all runs. A direct variant may also be defined, but turns out to be ill-behaved in the stochastic context, we exhibit an example, later in the chapter, displaying such behaviors. Hence we focus on the prefix-independent version.

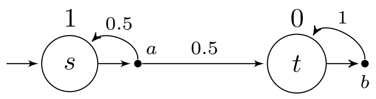


Figure 2.4: A parity MDP with no uniform bound over all the runs.

Example 2.5. Consider the MDP depicted in Figure 2.4, the MC from Figure 2.2 with priorities assigned to states. In this MPD, for any window size $\lambda > 0$, there is probability $1/2^{\lambda-1}$ that objective $\mathsf{DFW}_{\mathsf{par}}(\lambda)$ is not satisfied. Hence,

$$\forall\lambda > 0,\ \mathbb{P}_{\mathcal{M},s}[\mathsf{DFW}_{\mathsf{par}}(\lambda)] < 1\ .$$

Now let $\mathsf{DBW}_{\mathsf{par}}$ be the ***direct*** bounded window objective evoked above and defined as follows:

$$\mathsf{DBW}_{\mathsf{par}} = \bigcup_{\lambda>0} \mathsf{DFW}_{\mathsf{par}}(\lambda)\ .$$

We claim that $\mathbb{P}_{\mathcal{M},s}[\mathsf{DBW}_{\mathsf{par}}] = 1$. Indeed, any run ending in t belongs to $\mathsf{DBW}_{\mathsf{par}}$, since it belongs to $\mathsf{DFW}_{\mathsf{par}}(\lambda)$ for λ equal to the length of the prefix outside t plus one. Since t is almost-surely reached (since it is the only BSCC of the MC), we conclude that $\mathsf{DBW}_{\mathsf{par}}$ is indeed satisfied almost-surely. $\triangleleft$

2.4 Solving fixed window objectives

Our approach for solving fixed window objectives is rather straight forward. We take advantage of the fact that the size of the window is fixed and unfold our MDP using the size of the window as a bound for the unfolding. We will show that in the direct setting, this naive approach is optimal. However in the prefix-independent setting, we can improve over this unfolding, but we will nevertheless establish some useful properties of an optimal strategy. These properties will be used to obtain more efficient bound. Before, we give some insights of this so-called unfolding.

Let $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$ be an MDP, $\Omega = \{\mathsf{par}, \mathsf{mp}\}$, and $\lambda > 0$ be the window size, we build a new MDP denoted $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$. The obtained MDP is unweighted, it keeps track in each of its states of the current state of $\mathcal{M}$, the size of the current open window as well as the current sum of weights (or the minimal seen priority) along the window: these two values are reset whenever a window is closed (left-hand side of the disjunction) or stays open for λ steps (right hand-side). A key underlying property used here is the so-called inductive property of windows [CDRR15, BHR16a].

Definition 2.6 (Inductive property of windows)**.** Let $\rho = s_0a_0s_1a_1\dots$ be a run of an MDP. Fix a window starting in position $i \geq 0$. Let j be the position in which this window closes (assuming it does). Then, all windows in positions from i to j also close in j.

Remark 2.7. The validity of this property is easy to check by contradiction; if it would not be the case, then the window would close before j.

Remark 2.8. This property is fundamental in our reduction: without it we would have to keep track of all open windows in parallel, which would result in a blow-up exponential in λ.

λ-MP-unfolding

Let $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$ be an MDP with weight function wght, and $\lambda > 0$ be the window size. The λ-MP-unfolding of $\mathcal{M}$ is the MDP $\widetilde{\mathcal{M}}^{\lambda}_{\mathsf{mp}} = (\tilde{\mathsf{S}}, \mathsf{A}, \tilde{\delta})$ is defined as follows:

- $\tilde{\mathsf{S}} = S \times \{0, \dots, \lambda\} \times \{-\lambda \cdot W, \dots, 0\}$.
- $\tilde{\delta} \colon \tilde{\mathsf{S}} \times \mathsf{A} \to \mathsf{Dist}(\tilde{\mathsf{S}})$ is defined as follows for all a in A:

$$\tilde{\delta}\left((s,l,z),a\right)(t,l+1,z+w(a)) = \nu \text{ if } (\delta(s,a)(t) = \nu) \ \wedge \ (l < \lambda) \ \wedge \ (z + w(a) < 0)\,,$$
$$\tilde{\delta}\left((s,l,z),a\right)(t,0,0) = \nu \text{ if } (\delta(s,a)(t) = \nu) \ \wedge \ [(z + w(a) \geq 0) \vee ((l = \lambda) \ \wedge \ (z < 0))]\,.$$

- Once an initial state $s_{\mathsf{ini}} \in \mathsf{S}$ is fixed in $\mathcal{M}$, the matching initial state in $\widetilde{\mathcal{M}}^{\lambda}_{\mathsf{mp}}$ is $\tilde{s}_{\mathsf{init}} = (s_{\mathsf{ini}}, 0, 0)$.

Let B_{mp} be the following set:

$$\mathsf{B}_{\mathsf{mp}} = \{(s,l,z) \mid (l = \lambda) \wedge (z < 0)\}\ . \tag{2.1}$$

By construction of $\widetilde{\mathcal{M}}^{\lambda}_{\mathsf{mp}}$, runs visiting B_{mp} correspond to runs containing windows staying open for λ steps.

λ-Parity-unfolding

Let $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$ be an MDP with priority function prty, and $\lambda > 0$ be the window size. The λ-Parity-unfolding of $\mathcal{M}$ is the MDP $\widetilde{\mathcal{M}}^{\lambda}_{\mathsf{par}} = (\tilde{\mathsf{S}}, \mathsf{A}, \tilde{\delta})$ defined as follows:

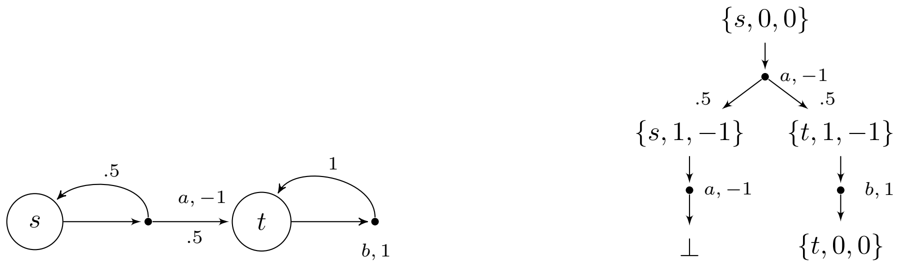
- $\tilde{\mathsf{S}} = \mathsf{S} \times \{0, \dots, \lambda\} \times \{0, 1, \dots, d\}$.
- $\tilde{\delta} \colon \tilde{\mathsf{S}} \times \mathsf{A} \to \mathsf{Dist}(\tilde{\mathsf{S}})$ is defined for all a in A as follows:

$$\tilde{\delta}\left((s,l,c),a\right)(t,l+1,\min(c,\mathsf{prty}(t))) = \nu \text{ if } (\delta(s,a)(t) = \nu) \ \wedge \ (l < \lambda - 1) \ \wedge \ (c \bmod 2 = 1)\,,$$
$$\tilde{\delta}\left((s,l,c),a\right)(t,0,\mathsf{prty}(t)) = \nu \text{ if } \big(\delta(s,a)(t) = \nu\big) \ \wedge \ \big(l = \lambda - 1\big) \ \wedge \ \big(c \bmod 2 = 1\big),$$
$$\tilde{\delta}\left((s,l,c),a\right)(t,0,\mathsf{prty}(t)) = \nu \text{ if } \big(\delta(s,a)(t) = \nu\big) \ \wedge \ \big(c \bmod 2 = 0\big).$$

- Once an initial state s_{ini} in S is fixed in $\mathcal{M}$, the associated one in $\widetilde{\mathcal{M}}^{\lambda}_{\mathsf{par}}$ is $\tilde{s}_{\mathsf{init}} = (s_{\mathsf{ini}}, 0, \mathsf{prty}(s_{\mathsf{ini}}))$.

Let $\mathsf{B}_{\mathsf{par}}$ be the following set:

$$\mathsf{B}_{\mathsf{par}} = \{(s,l,c) \mid (l = \lambda - 1) \wedge (c \bmod 2 = 1)\} \ . \tag{2.2}$$



(a) Mean payoff MDP with objective $\mathsf{DFW}_{\mathsf{mp}}(2)$. (b) The associated λ-MP-unfolding

Figure 2.5: An MDP and its associated unfolding.

Example 2.9. In Figure 2.5, an arena together with its associated unfolding are depicted. For the sake of clarity the states $\perp$ and $\{t,0,0\}$ are sinks but the outgoing edges are not drawn. In this example, the window size λ is 2. Notice that the second components keeps track of the period of time the window stays ***open***. From state $\{s,1,-1\}$, the play moves to state $\perp$. This is due to the following facts:

- at this stage, the window has to be closed in one step,
- the only possible action weighs -1,
- the accumulation of weights is -1.

Thus, the current opened window cannot be closed in one step. Once a window didn't close before the appropriate duration, i.e., the state $\perp$ is reached, this window remains open forever, this is detected by the fact that $\perp$ is a sink. $\triangleleft$

Properties of the unfolding

Let $\mathcal{M}$ be an MDP and $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$ its associated unfolding with $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, define the following sets of plays of $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$:

$$\mathsf{Reach}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}) = (\tilde{\mathsf{S}}\mathsf{A})^*\mathsf{B}_{\Omega}\mathsf{A}(\tilde{\mathsf{S}}\mathsf{A})^{\omega}, \qquad \mathsf{Safety}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}) = (\tilde{\mathsf{S}}\mathsf{A})^{\omega} \setminus \mathsf{Reach}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}),$$
$$\mathsf{Buchi}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}) = ((\tilde{\mathsf{S}}\mathsf{A})^*\mathsf{B}_{\Omega}\mathsf{A})^{\omega}, \qquad \mathsf{coBuchi}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}) = (\tilde{\mathsf{S}}\mathsf{A})^{\omega} \setminus \mathsf{Buchi}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}).$$

We will use the above sets as objective in the MDPs obtained by unfolding to detect winning strategies. In particular we establish that:

- $\mathsf{DFW}_{\Omega}(\lambda)$ can be solved by simply solving a safety objective over the unfolding.
- $\mathsf{FW}_{\Omega}(\lambda)$ can be solved by simply solving a coBüchi objective over the unfolding.

In order to obtain the above properties, we establish a mapping whose key property is to preserve the probability of winning in both MDPs (the original and the unfolded).

Mapping.

Fix an initial state s_{ini} in $\mathcal{M}$ and let $\tilde{s}_{\mathsf{init}}$ be its corresponding initial state in $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$. Let $\mathsf{Hists}(\mathcal{M}, s_{\mathsf{ini}})$ denote the histories of $\mathcal{M}$ starting in $\tilde{s}_{\mathsf{init}}$.
Define the mapping $\pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}}$, and its inverse $\pi_{\mathcal{M}}$, between histories of $\mathsf{Hists}(\mathcal{M}, \tilde{s}_{\mathsf{init}})$ and $\mathsf{Hists}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}, \tilde{s}_{\mathsf{init}})$. We use

$$\pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}} \colon \mathsf{Hists}(\mathcal{M}, s_{\mathsf{ini}}) \to \mathsf{Hists}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}, \tilde{s}_{\mathsf{init}})$$

for the $\mathcal{M}$-to-$\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$ direction, and

$$\pi_{\mathcal{M}} \colon \mathsf{Hists}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}, \tilde{s}_{\mathsf{init}}) \to \mathsf{Hists}(\mathcal{M}, s_{\mathsf{ini}})$$

for the opposite one. We define $\pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}}$ inductively as follows:

- $\pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}}(\tilde{s}_{\mathsf{init}}) = \tilde{s}_{\mathsf{init}}$.
- Let $h \in \mathsf{Hists}(\mathcal{M}, s_{\mathsf{ini}})$, $\tilde{h} = \pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}}(h)$, $a \in \mathsf{A}$, $s \in \mathsf{S}$. Then, $\pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}}(h \cdot a \cdot s) = \tilde{h} \cdot a \cdot \tilde{s}$, where $\tilde{s}$ is obtained from $\mathsf{Last}(\tilde{h})$, a and s following the unfolding construction.

We define $\pi_{\mathcal{M}}$ as its inverse, i.e., the function projecting histories of $\mathsf{Hists}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}, \tilde{s}_{\mathsf{init}})$ to $(\mathsf{SA})^*\mathsf{S}$. We naturally extend these mappings to runs based on this inductive construction. We also extend these mappings over strategies. Let σ be a strategy in $\mathcal{M}$. We define its twin strategy $\tilde{\sigma} = \pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}}(\sigma)$ in $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$ as follows:

$$\forall \tilde{h} \in \mathsf{Hists}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}, s_{\mathsf{ini}}),\ \tilde{\sigma}(\tilde{h}) = \sigma(\pi_{\mathcal{M}}(\tilde{h}))\ .$$

Note that this strategy is well-defined as $\mathcal{M}$ and $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$ share the same actions and $\pi_{\mathcal{M}}$ is well-defined over $\mathsf{Hists}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}, \tilde{s}_{\mathsf{init}})$ (i.e., each history $\tilde{h}$ has an image in $\mathsf{Hists}(\mathcal{M}, s_{\mathsf{ini}})$). Similarly, given a strategy $\tilde{\sigma}$ in $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$, we build a twin strategy $\sigma = \pi_{\mathcal{M}}(\tilde{\sigma})$ using $\pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}}$. Hence we also have a one-to-one mapping over strategies.

We say that two objects (histories, runs, strategies) are π-twin if they are the image of one another through mappings $\pi_{\mathcal{M}}$ and $\pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}}$.

Probability-wise equivalence.

For any any history h, we denote by $\mathsf{Cyl}(h)$ the cylinder set spanned by h, i.e., the set of all possible extensions of h. Cylinder sets are the building blocks of probability measures in MCs, as all measurable sets belong to the σ-algebra built upon them [BK08].

Lemma 2.10. *Let $\mathcal{M}$, $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$, $\pi_{\mathcal{M}}$ and $\pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}}$ be defined as above. Fix any couple of π-twin strategies $(\sigma, \tilde{\sigma})$ for $\mathcal{M}$ and $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$ respectively. Then, for any couple of π-twin histories $(h, \tilde{h})$ in $\mathsf{Hists}(\mathcal{M}, s_{\mathsf{ini}}) \times \mathsf{Hists}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}, \tilde{s}_{\mathsf{init}})$, we have that*

$$\mathbb{P}^{\sigma}_{\mathcal{M}, s_{\mathsf{ini}}}[\mathsf{Cyl}(h)] = \mathbb{P}^{\tilde{\sigma}}_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}, \tilde{s}_{\mathsf{init}}}[\mathsf{Cyl}(\tilde{h})]. \tag{2.3}$$

Since the previous lemma holds for all cylinders, we may extend its claim to any event.

Corollary 2.11. *Let $\mathcal{M}$, $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$, $\pi_{\mathcal{M}}$ and $\pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}}$ be defined as above. Fix any couple of π-twin strategies $(\sigma, \tilde{\sigma})$. Then, for any couple of π-twin events $(E, \tilde{E}) \subseteq \mathsf{Plays}(\mathcal{M}, s_{\mathsf{ini}}) \times \mathsf{Plays}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega}, \tilde{s}_{\mathsf{init}})$, we have:*

$$\mathbb{P}^{\sigma}_{\mathcal{M}, s_{init}}[E] = \mathbb{P}^{\tilde{\sigma}}_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}, \tilde{s}_{\mathsf{init}}}[\tilde{E}].$$

Correctness of the reductions.

Lemma 2.12. *Let $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$ be an MDP, $\lambda > 0$ be the window size, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, $\widetilde{\mathcal{M}}^{\lambda}_{\Omega} = (\tilde{\mathsf{S}}, \mathsf{A}, \tilde{\delta})$ be the λ-Ω-unfolding of $\mathcal{M}$, $s \in \mathsf{S}$ be a state of $\mathcal{M}$, and $\tilde{s} \in \tilde{\mathsf{S}}$ be its π-twin state in $\mathcal{M}_{\lambda}$. The following assertions hold.*

1. *For any strategy σ in $\mathcal{M}$, there exists a strategy $\tilde{\sigma}$ in $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$ such that*

$$\mathbb{P}^{\tilde{\sigma}}_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega},\tilde{s}}[\mathsf{Safety}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega})] = \mathbb{P}^{\sigma}_{\mathcal{M},s}[\mathsf{DFW}_{\Omega}(\lambda)] \qquad \wedge \qquad \mathbb{P}^{\tilde{\sigma}}_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega},\tilde{s}}[\mathsf{coBuchi}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega})] = \mathbb{P}^{\sigma}_{\mathcal{M},s}[\mathsf{FW}_{\Omega}(\lambda)].$$

2. *For any strategy $\tilde{\sigma}$ in $\widetilde{\mathcal{M}}^{\lambda}_{\Omega}$, there exists a strategy σ in $\mathcal{M}$ such that*

$$\mathbb{P}^{\sigma}_{\mathcal{M},s}[\mathsf{DFW}_{\Omega}(\lambda)] = \mathbb{P}^{\tilde{\sigma}}_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega},\tilde{s}}[\mathsf{Safety}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega})] \qquad \wedge \qquad \mathbb{P}^{\sigma}_{\mathcal{M},s}[\mathsf{FW}_{\Omega}(\lambda)] = \mathbb{P}^{\tilde{\sigma}}_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega},\tilde{s}}[\mathsf{coBuchi}(\widetilde{\mathcal{M}}^{\lambda}_{\Omega})].$$

Moreover, such strategies can be obtained through mappings $\pi_{\mathcal{M}}$ and $\pi_{\widetilde{\mathcal{M}}^{\lambda}_{\Omega}}$.

Memory requirements and complexity

Thanks to the reductions established in Lemma 2.12, along with the fact that pure memoryless strategies suffice for safety and co-Büchi objectives in MDPs [BK08], we obtain the following result.

Theorem 2.13. *Pure finite-memory strategies suffice for the threshold probability problem for all fixed window objectives. That is, given MDP $\mathcal{M} = (S, A, \delta)$, initial state $s \in S$, window size $\lambda > 0$, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, objective $O \in \{\mathsf{DFW}_{\Omega}(\lambda), \mathsf{FW}_{\Omega}(\lambda)\}$ and threshold probability $\alpha \in [0, 1] \cap \mathbb{Q}$, the following assertions are equivalent:*

- *There exists a strategy $\sigma \in \Sigma$ such that $\mathbb{P}^{\sigma}_{\mathcal{M},s}[O] \geq \alpha$.*
- *There exists a pure finite-memory strategy σ' such that $\mathbb{P}^{\sigma'}_{\mathcal{M},s}[O] \geq \alpha$.*

Complexity-wise, these reductions also yield algorithms for the threshold probability problem in the fixed window case.

Theorem 2.14. *The threshold probability problem is*

1. *in* PTIME *for direct fixed window parity objectives, and pure polynomial-memory optimal strategies can be constructed in polynomial time.*
2. *in* EXPTIME *for direct fixed window mean-payoff objectives, and pure pseudo-polynomial-memory optimal strategies can be constructed in pseudo-polynomial time.*

Remark 2.15. Theorem 2.14 does not claim anything about the prefix-independent case. This is postponed to a latter analysis where we take advantage the asymptotic properties of MDPs and prefix-independence. We develop a specific technique to solve our problem when restricted to an ***end-component***. This tailor-made approach yields better algorithms.

We also obtain these almost matching lower bounds. Details are skipped and can be found in [BDOR19, BDOR20].

Theorem 2.16. *The threshold probability problem is*

1. PTIME*-hard for direct fixed window parity objectives, and polynomial-memory strategies are in general necessary;*

2. PSPACE*-hard for direct fixed window mean-payoff objectives (even for acyclic MDPs), and pseudo-polynomial-memory strategies are in general necessary.*

Remark 2.17. An interesting feature in the setting of ***direct fixed*** window objectives is that almost-surely winning coincides with surely winning. Therefore, the threshold probability problem for $\mathsf{DFW}_{\mathsf{mp}}(\lambda)$ collapses to P when the threshold is 1 [CDRR15].

2.5 Detour through end-components

In this section, we restrict our focus on the end-components. We will exhibit several properties that allow us to classify end-component. We will in particular show that in order to maximize the probability of a prefix-independent window objective, it is sufficient to maximize the probability to reach a "good" end-component.

In order to formalize the notion of a good end-component, we establish a strong link between ECs and two-player games; either the probability to win a window objective in an end-component C is zero, or it is one and there exists a sub-end-component C where the controller can actually win surely, i.e., in a two-player game played on this sub-end-component.

We start by defining the notion of λ-safety.

Definition 2.18 (λ-safety)**.** Let $\mathcal{M}$ be an MDP, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, $\lambda > 0$, and $\mathcal{C} = (\mathsf{S}_{\mathcal{C}}, \mathsf{A}_{\mathcal{C}}, \delta_{\mathcal{C}}) \in \mathsf{EC}(\mathcal{M})$, we say that $\mathcal{C}$ is λ-safe$_\Omega$ if there exists a strategy $\sigma \in \Sigma$ in $\mathcal{C}$ such that:

$$\forall s \in \mathsf{S}_{\mathcal{C}},\ \mathsf{Sure}^{\sigma}_{\mathcal{C},s}[\mathsf{DFW}_{\Omega}(\lambda)]\ .$$

The above definition essentially says that an EC $\mathcal{C}$ is λ-safe if we leave the control of randomness to some antagonistic entity, the controller still has a strategy to achieve its objective. With this intuition in mind and existing results in two-player games with window objectives (cf. [CDRR15] and [BHR16a]), we obtain the following proposition.

Proposition 2.19. *Let* $\mathcal{M}$ *be an MDP,* $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$*,* $\lambda > 0$*, and* $\mathcal{C} = (S_{\mathcal{C}}, A_{\mathcal{C}}, \delta_{\mathcal{C}}) \in \mathsf{EC}(\mathcal{M})$ *be* λ*-safe*$_\Omega$*. Then, there exists a pure polynomial-memory strategy* $\sigma^{\Omega,\lambda,\mathcal{C}}_{safe}$ *in* $\mathcal{C}$ *such that* $\mathsf{Sure}^{\sigma^{\Omega,\lambda,\mathcal{C}}_{safe}}_{\mathcal{C},s}[\mathsf{DFW}_{\Omega}(\lambda)]$ *for all* $s \in S_{\mathcal{C}}$*.*

We now introduce the notion of good ECs.

Definition 2.20. Let $\mathcal{M}$ be an MDP, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, and $\mathcal{C} \in \mathsf{EC}(\mathcal{M})$, we say that

- $\mathcal{C}$ is λ-good$_\Omega$, for $\lambda > 0$, if it contains a sub-EC $\mathcal{C}'$ which is λ-safe$_\Omega$.

- $\mathcal{C}$ is BW-good$_\Omega$ if it contains a sub-EC $\mathcal{C}'$ which is λ-safe$_\Omega$ for some $\lambda > 0$.

The good ECs enjoy the following useful property, they correspond exactly to where window objectives can be satisfied with non-zero probability, and actually, with probability one. We call this property zero-one law and it is established in the following lemma:

Lemma 2.21 (Zero-one law)**.** *Let* $\mathcal{M}$ *be an MDP,* $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$ *and* $\mathcal{C} = (S_{\mathcal{C}}, A_{\mathcal{C}}, \delta_{\mathcal{C}}) \in \mathsf{EC}(\mathcal{M})$*. The following assertions hold.*

1. *For all $\lambda > 0$,*
   (a) *either $\mathcal{C}$ is λ-good$_\Omega$ and there exists a strategy σ in $\mathcal{C}$ such that $\mathsf{ASure}^{\sigma}_{\mathcal{C},s}[\mathsf{FW}_\Omega(\lambda)]$ for all $s \in S_\mathcal{C}$,*
   (b) *or for all $s \in S_\mathcal{C}$, for all strategy σ in $\mathcal{C}$, $\mathbb{P}^{\sigma}_{\mathcal{C},s}[\mathsf{FW}_\Omega(\lambda)] = 0$.*
2. (a) *Either $\mathcal{C}$ is BW-good$_\Omega$ and there exists a strategy σ in $\mathcal{C}$ such that $\mathsf{ASure}^{\sigma}_{\mathcal{C},s}[\mathsf{BW}_\Omega]$ for all $s \in S_\mathcal{C}$,*
   (b) *or for all $s \in S_\mathcal{C}$, for all strategy σ in $\mathcal{C}$, $\mathbb{P}^{\sigma}_{\mathcal{C},s}[\mathsf{BW}_\Omega] = 0$.*

We present the proof of the above lemma since it helps understand how we use the different notions presented so-far, in particular how we use the results from the previous section.

Proof. We begin with the fixed variant 1, i.e., Case 1a.

Fix $\lambda > 0$ and assume there exists a sub-EC $\mathcal{C}'$ with state space $\mathsf{S}_{\mathcal{C}'}$ that is λ-safe$_\Omega$. By Prop. 2.19, there exists a strategy $\sigma^{\Omega,\lambda,\mathcal{C}'}_{\text{safe}}$ in $\mathcal{C}'$ such that for all $s \in S_{\mathcal{C}'}$, we have

$$\mathsf{Sure}^{\sigma^{\Omega,\lambda,\mathcal{C}'}_{\text{safe}}}_{\mathcal{C}',s}[\mathsf{DFW}_\Omega(\lambda)] \ .$$

Now, since $\mathcal{C}$ is an EC, there exists a (pure memoryless) strategy σ_{reach} in $\mathcal{C}$ that ensures eventually reaching $\mathcal{C}'$ almost-surely from any state $s \in S_\mathcal{C}$ [BK08]. Hence, the desired strategy σ can be defined as follows:

- play according to σ_{reach} until $\mathcal{C}'$ is reached,
- then switch to $\sigma^{\Omega,\lambda,\mathcal{C}'}_{\text{safe}}$ forever.

σ clearly satisfies $\mathsf{FW}_\Omega(\lambda)$ from anywhere in $\mathcal{C}$ almost-surely, thanks to prefix-independence. Note that it does not ensure it surely in general, as σ_{reach} does not guarantee to reach $\mathcal{C}'$ surely either.

Case 1b. Now assume that such a λ-safe$_\Omega$ sub-EC does not exist. Recall that finite-memory strategies suffice for $\mathsf{FW}_\Omega(\lambda)$ objectives by Theorem. 2.13, hence we can restrict our study to such strategies without loss of generality. Fix any finite-memory strategy σ in $\mathcal{C}$ and state $s \in \mathsf{S}_\mathcal{C}$. The induced MC $\mathcal{M}^\sigma_s$ is finite, hence its runs almost-surely end up in a BSCC [BK08]. Let $\mathcal{B}$ be any BSCC of $\mathcal{M}^\sigma_s$ reached with positive probability. Since there exists no λ-safe$_\Omega$ sub-EC in $\mathcal{C}$, there must exist a run $\widehat{\rho}$ in $\mathcal{B}$ such that $\widehat{\rho} \notin \mathsf{DFW}_\Omega(\lambda)$, otherwise, σ would be witness that the EC obtained by projecting $\mathcal{B}$ over $S_\mathcal{C}$ is λ-safe$_\Omega$. From $\widehat{\rho}$, we extract a history $\widehat{h}$ that contains a window open for λ steps (it exists otherwise $\widehat{\rho}$ would be in $\mathsf{DFW}_\Omega(\lambda)$). This history is finite: it has a probability lower-bounded by some $\varepsilon > 0$ to occur whenever its starting state is visited. Now, since all the states in $\mathcal{B}$ are almost-surely visited infinitely often, we conclude that this history also happens infinitely often with probability one. Therefore, the probability to win the prefix-independent objective $\mathsf{FW}_\Omega(\lambda)$ when reaching $\mathcal{B}$ is zero. Since this holds for any BSCC induced by σ, the claim follows.

The bounded case 2. Case 2a is trivial thanks to 11a and $\mathsf{FW}_\Omega(\lambda) \subseteq \mathsf{BW}_\Omega$. Now consider case 2b. By 11b, we have that $\mathbb{P}^{\sigma}_{\mathcal{C},s}[\mathsf{FW}_\Omega(\lambda)] = 0$ for all $s \in \mathsf{S}_\mathcal{C}$, $\lambda > 0$ and σ in $\mathcal{C}$. Observe that by definition, we have

$$\mathsf{BW}_\Omega = \bigcup_{\lambda > 0} \mathsf{FW}_\Omega(\lambda).$$

Fix any strategy σ in $\mathcal{C}$ and $s \in S_\mathcal{C}$. By the additivity rule, we obtain

$$\mathbb{P}^{\sigma}_{\mathcal{C},s}[\mathsf{BW}_\Omega] \ \leq \ \sum_{\lambda > 0} \mathbb{P}^{\sigma}_{\mathcal{C},s}[\mathsf{FW}_\Omega(\lambda)] = 0 \ ,$$

the claim follows. □

Remark 2.22. An interesting consequence of Lemma 2.21 is the existence of uniform bounds on λ in ECs, in contrast to the general MDP case, as seen in Example 2.5. This is natural, since we established that winning with positive probability within an EC coincides with winning surely in a sub-EC; sub-EC that can be seen as a two-player zero-sum game where uniform bounds are granted by [CDRR15, BHR16a].

We are now able to claim nice properties of winning strategies using the knowledge we gained about good ECs.

Proposition 2.23. *Let $\mathcal{M}$ be an MDP, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, and $\mathcal{C} = (\mathsf{S}_\mathcal{C}, A_\mathcal{C}, \delta_\mathcal{C}) \in \mathsf{EC}(\mathcal{M})$.*

- *If $\mathcal{C}$ is λ-good$_\Omega$, for some $\lambda > 0$, there exists a pure polynomial-memory strategy $\sigma_{good}^{\Omega,\lambda,\mathcal{C}}$ such that:*
$$\forall s \in \mathsf{S}_\mathcal{C},\ \mathsf{ASure}_{\mathcal{C},s}^{\sigma_{good}^{\Omega,\lambda,\mathcal{C}}}[\mathsf{FW}_\Omega(\lambda)]\ .$$
- *If $\mathcal{C}$ is BW-good$_\Omega$, there exists a pure memoryless strategy $\sigma_{good}^{\Omega,\mathsf{BW},\mathcal{C}}$ such that:*
$$\forall s \in S_\mathcal{C},\ \mathsf{ASure}_{\mathcal{C},s}^{\sigma_{good}^{\Omega,\mathsf{BW},\mathcal{C}}}[\mathsf{BW}_\Omega]\ .$$

Remark 2.24. Intuitively, such strategies first mimic a pure memoryless strategy reaching a safe$_\Omega$ sub-EC almost-surely, then switch to a strategy surely winning in this sub-EC, which is lifted from the game interpretation.

Remark 2.25. In the above claim, the strategies we consider are uniform in the game-theoretic sense. That is, if an objective is achievable from a set of states, we do not need to have a different strategy for each starting state, and we may instead use the very same strategy from all initial states. This uniformity is not needed but it is rather obtained for free for both the classical MDP strategies (reachability, Büchi, etc) [BK08] and the two-player window games [CDRR15, BHR16a]. We use it for the sake of readability and convenience, since it allows to have one strategy per EC instead of one per state of the EC for example.

The aftermath of Proposition 2.23 is that in order to build an efficient algorithm, one needs to classify ECs efficiently despite the fact an MDP my contain an exponential number of ECs. We circumvent this thanks to the following lemma:

Lemma 2.26. *Let $\mathcal{M}$ be an MDP and $\mathcal{C} \in \mathsf{EC}(\mathcal{M})$. If $\mathcal{C}$ is λ-good$_\Omega$ (resp. BW-good$_\Omega$), then it is also the case of any super-EC $\mathcal{C}' \in \mathsf{EC}(\mathcal{M})$ containing $\mathcal{C}$.*

And its corollary,

Corollary 2.27. *Let $\mathcal{M}$ be an MDP and $\mathcal{C} \in \mathsf{MEC}(\mathcal{M})$ be a maximal EC. If $\mathcal{C}$ is not λ-good$_\Omega$ (resp. BW-good$_\Omega$), then neither is any of its sub-EC $\mathcal{C}' \in \mathsf{EC}(\mathcal{M})$.*

The interest of Corollary 2.27 is that the number of MECs is bounded by $|S|$ for any MDP $\mathcal{M} = (S, A, \delta)$ because they are all disjoints. Furthermore, the MEC decomposition can be done efficiently (e.g., quadratic time [CH14]). It remains to discuss how to classify a MEC as good$_\Omega$ or not.

Let $\mathcal{M} = (S, A, \delta)$. Recall that a MEC $\mathcal{C} = (\mathsf{S}_\mathcal{C}, A_\mathcal{C}, \delta_\mathcal{C}) \in \mathsf{MEC}(\mathcal{M})$ is λ-good$_\Omega$ (resp. BW-good$_\Omega$) if and only if it contains a λ-safe$_\Omega$ sub-EC. By definition of λ-safety, this is equivalent to having a non-empty winning set for the controller in the two-player zero-sum game over $\mathcal{C}$ This winning set contains all the states in $S_\mathcal{C}$ from which the controller has a surely winning strategy. This winning set, if non-empty, necessarily contains at least one sub-EC of $\mathcal{C}$, otherwise the opponent could force the controller to leave it and win the game by prefix-independence. Thus, testing if a MEC is good$_\Omega$ consists in solving its two-player game interpretation.

Finally we can state the following main theorem:

Theorem 2.28 (MEC classification)**.** *Let $\mathcal{M}$ be an MDP and $\mathcal{C} \in \mathsf{MEC}(\mathcal{M})$. The following assertions hold.*

1. *Deciding if $\mathcal{C}$ is λ-good$_\Omega$, for $\lambda > 0$, is in* P *for $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$. Furthermore, a corresponding pure polynomial-memory strategy $\sigma^{\Omega,\lambda,\mathcal{C}}_{good}$ can be constructed in polynomial time.*
2. *Deciding if $\mathcal{C}$ is* BW*-good$_{\mathsf{mp}}$ is in* NP $\cap$ co-NP *and a corresponding pure memoryless strategy $\sigma^{\mathsf{mp},\mathsf{BW},\mathcal{C}}_{good}$ can be constructed in pseudo-polynomial time.*
3. *Deciding if $\mathcal{C}$ is* BW*-good$_{\mathsf{par}}$ is in* PTIME *and a corresponding pure memoryless strategy $\sigma^{\mathsf{par},\mathsf{BW},\mathcal{C}}_{good}$ can be constructed in polynomial time.*

2.6 General MDPs

In this section, we take advantage of the results yielded by Theorem 2.28 and combine them with classical reachability results in MDPs. We obtain two algorithms 1 and 2 where we respectively solve the fixed window and bounded window objectives. In these algorithms, we use $\mathsf{MaxReachability}(s, T)$ as a sub-routine that computes in an MDP the maximal probability to reach a subset of states T from an initial state s, cf. [Put14] for the interested reader.

```
Data: MDP $\mathcal{M}$, state $s$, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, $\lambda > 0$
Result: Maximum probability of $\mathsf{FW}_\Omega(\lambda)$ from $s$
$T \leftarrow \emptyset$;
/* Classification of MECs                                        */
forall $\mathcal{C} = (\mathsf{S}_\mathcal{C}, \mathsf{A}_\mathcal{C}, \delta_\mathcal{C}) \in \mathsf{MEC}(\mathcal{M})$ do
    if $\mathcal{C}$ is $\lambda$-good$_\Omega$ then
        $T \leftarrow T \uplus \mathsf{S}_\mathcal{C}$;
/* Computing the maximal probability of reaching a $\lambda$-MEC   */
$\nu = \mathsf{MaxReachability}(s, T)$;
return $\nu$;
```

Algorithm 1: $\mathsf{FixedWindow}(\mathcal{M}, s, \Omega, \lambda)$

```
Data: MDP $\mathcal{M}$, state $s$, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$
Result: Maximum probability of $\mathsf{BW}_\Omega$ from $s$
$T \leftarrow \emptyset$;
/* Classification of MECs                                        */
forall $\mathcal{C} = (\mathsf{S}_\mathcal{C}, \mathsf{A}_\mathcal{C}, \delta_\mathcal{C}) \in \mathsf{MEC}(\mathcal{M})$ do
    if $\mathcal{C}$ is BW-good$_\Omega$ then
        $T \leftarrow T \uplus \mathsf{S}_\mathcal{C}$;
/* Computing the maximal probability of reaching a BW-good MEC   */
$\nu = \mathsf{MaxReachability}(s, T)$;
return $\nu$;
```

Algorithm 2: $\mathsf{BoundedWindow}(\mathcal{M}, s, \Omega)$

These two algorithms have the following property:

Lemma 2.29. *Given an MDP $\mathcal{M} = (\mathsf{S}, \mathsf{A}, \delta)$, an initial state $s \in \mathsf{S}$, $\Omega \in \{\mathsf{mp}, \mathsf{par}\}$, $\lambda > 0$, then*

$$\mathsf{FixedWindow}(\mathcal{M}, s, \Omega, \lambda) = \max_{\sigma \in \Sigma} \mathbb{P}^{\sigma}_{\mathcal{M},s}[\mathsf{FW}_{\Omega}(\lambda)],$$
$$\mathsf{BoundedWindow}(\mathcal{M}, s, \Omega) = \max_{\sigma \in \Sigma} \mathbb{P}^{\sigma}_{\mathcal{M},s}[\mathsf{BW}_{\Omega}].$$

Using the above lemma, we claim the following bounds for winning strategies.

Theorem 2.30. *The threshold probability problem is*

- *in* PTIME *for fixed window parity objectives and fixed window mean-payoff objectives, and pure polynomial-memory optimal strategies can be constructed in polynomial time;*
- *in* PTIME *for bounded window parity objectives, and pure memoryless optimal strategies can be constructed in polynomial time;*
- *in* NP ∩ co-NP *for bounded window mean-payoff objectives, and pure memoryless optimal strategies can be constructed in pseudo-polynomial time.*

Sketch of the proof. These complexity results are obtained as follows,

- the MEC decomposition takes quadratic time [CH14] and yields at most $|\mathsf{S}|$ MECs,
- classifying a MEC is in PTIME for fixed variants and bounded window parity, and in NP ∩ co-NP for bounded window mean-payoff, cf. Theorem 2.28.
- $\mathsf{MaxReachability}(s, T)$ (Line 1) takes polynomial time [Put14].

Overall we have PTIME-membership for all variants except bounded window mean-payoff, where we are in $\mathsf{PTIME}^{\mathsf{NP} \cap \mathsf{co\text{-}NP}}$, which is equal to NP ∩ co-NP [Bra79].

Concerning optimal strategies, they follow from Algorithm. 1 and Algorithm. 2. In particular,

- Inside good_{Ω} MECs, we play the strategy guaranteed by Theorem. 2.28,
- outside, we simply play a pure memoryless optimal reachability strategy obtained through the sub-routine $\mathsf{MaxReachability}(s, T)$ [Put14].

□

We also have the following matching lower bounds:

Theorem 2.31. *The threshold probability problem is*

- PTIME*-hard for fixed window parity objectives and fixed window mean-payoff objectives, and polynomial-memory strategies are in general necessary;*
- PTIME*-hard for bounded window parity objectives;*
- *as hard as mean-payoff games for bounded window mean-payoff objectives.*

2.7 Discussion

Wrap-up

In this chapter, we focused on some properties of Markov decision processes equipped with parity or mean-payoff objectives. In order to promote robust behavior in the long term, we equipped this model with the window mechanism. We recall that this mechanism has been used already in two-player games in [CDRR15, BHR16a, HPR18]. In this case, this mechanism was able to reduce the complexity of computing winning strategies.

Let us comment on the complexity bounds achieved in this chapter. Indeed, the computational complexity is already polynomial. When we apply the widow mechanism, this complexity significantly rises in the case of mean-payoff with direct fixed window. This is a consequence of the expressive power of this objective, precisely it can encode complex combinatorial problem such as the shortest paths in Markov decision processes [HK15, RRS17, BGMR18, HJKQ18]. For the case of bounded window mean-payoff we match the complexity bound for two-player game. This is the best one can achieve since we show that essentially solving this problem for Markov decision processes amounts to solving a two-player games with the same objective, c.f. Section 2.6. However, we argue that our main motivation for Markov decision processes does not concern the complexity bounds but lies elsewhere. It actually concerns the modeling power, as shown in Example 2.1 and Example 2.4 the window mechanism ensures a better behaved system.

Perspectives

Several extension of the windows mechanism arise:

- One could think of an optimization problem where one computes a strategy that minimizes the expected window size.
- Study the window mechanism in the setting of simple stochastic games, this has been achieved in [DGG23] for mean-payoff objectives. The main technical tool is a reduction to the setting of two-player games.
- An other exciting direction consists in designing a unified framework to described specifications using the window mechanism. Actually, if such setting exists already in the case of model checking, one would, for instance, discard a system only in the presence of a robust counter-example.

A natural candidate for the last item would be the temporal logic prompt LTL [KPV09]. This formalism has some flavor of the window mechanism but does not coincide with it. In particular, the inductive property of windows presented in Definition 2.6 does not hold in prompt LTL causing the formalism to be a lot more complex from a combinatorial point of view. Another difference is that prompt LTL does not allow for a fixed version.

Nevertheless, interesting fragment of prompt LTL were investigated in the PhD of Léo Tible. The initial motivation was to compare two views of the model checking. On the one hand, the universal model checking where one considers all the run of a system. On the other hand, the fair model checking where one introduces a randomization in the system and considers almost-all the runs of a system. Such comparison was presented for LTL without promptness constraint, and for the so-called Muller fragment, the fair model checking was shown to be faster [SVV07].

An extension to prompt LTL was considered and the reported results are that the fair and universal model checking problems coincide in this case, they both are co-NP-complete. We also introduced prefix-independent fragment of this fragment, and for this variant the fair model checking drops to polynomial while the universal one remains co-NP-complete.

Chapter 3

Robustness in timed synthesis

3.1 Outline of the chapter

In this chapter we are interested in the problem of synthesis in the realm of real time systems. For such systems, timed games are a standard mathematical tool which can model controller synthesis problems under timing constraints. These consist in zero-sum games played on arenas, defined by timed automata, whose state space consists in discrete locations and continuous clock values. However, In timed automata, the abstract mathematical semantics offers arbitrarily precise clocks and time delays, while real-world digital systems have response times that may not be negligible, and the control policy cannot ensure timing constraints exactly, but only up to some error, caused by clock's imprecision, measurement errors, and communication delays.

In the present chapter, the main focus is thus to ensure that the synthesized control software is robust, i.e., ensures the specification even in presence of imprecisions [HS06]. In fact, due to the infinite precision of the semantics, synthesized strategies may not be implemented in a finite-precision environment; the controlled systems synthesized using timed games formalism may not satisfy the desired specification at all. In particular, due to perturbations in timings, some infinite behaviors may disappear completely. We develop algorithms for robust controller synthesis in timed games: we consider this problem by studying the existence of robust strategies in timed games, namely, those guaranteeing winning despite an imprecision bounded by a parameter.

In this chapter we overview a series of results obtained in collaboration with Ocan Sankur and Pierre-Alain Reynier [ORS14]. We present the following contributions:

- We develop a game semantics to capture the problem of robust controller synthesis in timed automata.
- We develop an abstraction capturing robust strategies, cf. Lemma 3.15.
- We establish complexity bound for the robust control of timed automata and match the complexity of synthesis without robustness constraints, cf. Theorem 3.16.
- We develop a stochastic setting of the robust controller synthesis problem.
- We study two variation of the stochastic version of the problem and establish complexity bounds, cf. Theorem 3.17 and Theorem 3.24.

This chapter follows the following organization:

- In Section 3.2, we recall some basic notion on timed automata, timed games, and present a game semantics for robust timed games.
- In Section 3.3, we present an abstraction well suited for robustness related questions.
- In Section 3.4, we present an algorithm for solving the problem of robust controller synthesis in timed games.
- In Section 3.5, we present a setting where the environment is stochastic.

3.2 Timed games

Timed automata

Given a finite set of clocks $\mathcal{C}$, we call valuations the elements of $\mathbb{R}_{\geq 0}^{\mathcal{C}}$. For a subset $R \subseteq \mathcal{C}$ and a valuation ν, $\nu[R \leftarrow 0]$ is the valuation defined by $\nu[R \leftarrow 0](x) = 0$ if $x \in R$ and $\nu[R \leftarrow 0](x) = \nu(x)$ otherwise. Given $d \in \mathbb{R}_{\geq 0}$ and a valuation ν, the valuation $\nu + d$ is defined by $(\nu+d)(x) = \nu(x) + d$ for all $x \in \mathcal{C}$. We extend these operations to sets of valuations in the obvious way. We write $\vec{0}$ for the valuation that assigns 0 to every clock.

An atomic clock constraint is a formula of the form $k \preceq x \preceq' l$ or $k \preceq x - y \preceq' l$ where $x, y \in \mathcal{C}$, $k, l \in \mathbb{Z} \cup \{-\infty, \infty\}$ and $\preceq, \preceq' \in \{<, \leq\}$. A guard is a conjunction of atomic clock constraints. A valuation ν satisfies a guard g, denoted $\nu \models g$, if all constraints are satisfied when each $x \in \mathcal{C}$ is replaced with $\nu(x)$. We write $\Phi_{\mathcal{C}}$ for the set of guards built on $\mathcal{C}$. A zone is a subset of $\mathbb{R}_{\geq 0}^{\mathcal{C}}$ defined by a guard.

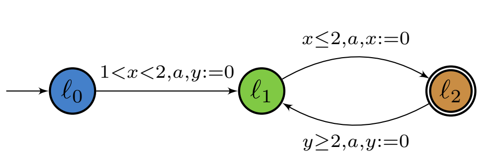
Definition 3.1 (Timed automata)**.** A ***timed automaton*** $\mathcal{A}$ over a finite alphabet of actions A is a tuple $(\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{A}, \mathsf{E})$, where $\mathcal{L}$ is a finite set of locations, $\mathcal{C}$ is a finite set of clocks, $\mathsf{E} \subseteq \mathcal{L} \times \Phi_{\mathcal{C}} \times \mathsf{A} \times 2^{\mathcal{C}} \times \mathcal{L}$ is a set of edges, and $\ell_0 \in \mathcal{L}$ is the initial location. An edge $e = (\ell, g, a, R, \ell')$ is also written as $\ell \xrightarrow{g,a,R} \ell'$. A state is a pair $q = (\ell, \nu) \in \mathcal{L} \times \mathbb{R}_{\geq 0}^{\mathcal{C}}$. An edge $e = (\ell, g, a, R, \ell')$ is enabled in a state (ℓ, ν) if ν satisfies the guard g.

The set of possible behaviors of a timed automaton can be described by the set of its runs $\mathsf{Runs}(\mathcal{A})$, as follows. A run in $\mathsf{Runs}(\mathcal{A})$ is a sequence $q_1 e_1 q_2 e_2 \ldots$ where $q_i \in \mathcal{L} \times \mathbb{R}_{\geq 0}^{\mathcal{C}}$. For $q_i = (\ell, \nu)$, either:

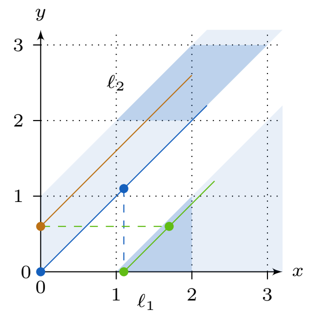
- $e_i \in \mathbb{R}_{>0}$, in which case $q_{i+1} = (\ell, \nu + e_i)$ (delay), or
- $e_i = (\ell, g, a, R, \ell') \in \mathsf{E}$, in which case $\nu \models g$ and $q_{i+1} = (\ell', \nu[R \leftarrow 0])$ (discrete move).

Example 3.2. Consider the timed automaton of Figure 3.1. On the left-hand side a timed automaton, cf. Figure 3.1a, and on the right-hand side a run over this automaton, cf. Figure 3.1b. At the initial location ℓ_0 a time lapse of at least 1 time unit but no more than 2 time units is necessary in order to enable the guard. This is depicted by the solid diagonal starting at $(0, 0)$. Once inside the guard, the darker gray triangle, the transition is taken and clock y is rest to 0, cf. the dashed vertical line. The run continues from the new configuration, i.e., $(\ell_1, (1.1, 0))$ and so on and so forth. $\triangleleft$

Remark 3.3. In Figure 3.1b, the light gray areas correspond to the possibles trajectories, the darker ones correspond to the moment where guards are enabled. These darker places are the basic intuition of the ***region based abstraction***.



(a)



(b)

Figure 3.1: A run of a timed automaton.

Region automaton

Following [DDMR08, Pur00, BA11], we assume that the clocks are bounded above by a known constant[1] in all timed automata we consider.

Fix a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{A}, \mathsf{E})$, we define regions following the definition of [AD94].

Let C be the largest integer appearing in $\mathcal{A}$. Alur et. al., consider an equivalence relation $\mathcal{R}$ of finite index defined over $\mathbb{R}^{\mathcal{C}}_{\geq 0} \times \mathbb{R}^{\mathcal{C}}_{\geq 0}$ as follows:
Let ν and ν' two valuations in $\mathbb{R}^{\mathcal{C}}_{\geq 0}$, then (ν, ν') is in $\mathcal{R}$ if:

- For all c in $\mathcal{C}$, either both $\nu(c)$ and $\nu'(c)$ are greater than C, or $\lfloor \nu(c) \rfloor = \lfloor \nu'(c) \rfloor$.
- For all c, c' in $\mathcal{C}$, if both $\nu(c)$ and $\nu(c')$ are lower than C, then:

$$\begin{aligned} \mathsf{frac}(\nu(c)) \leq \mathsf{frac}(\nu(c')) &\iff \mathsf{frac}(\nu'(c)) \leq \mathsf{frac}(\nu'(c')) \ , \\ \mathsf{frac}(\nu(c)) = 0 &\iff \mathsf{frac}(\nu'(c)) = 0 \ . \end{aligned}$$

Definition 3.4 (Regions)**.** The set $\mathsf{Reg}(\mathcal{A})$ of ***regions*** of a timed automaton $\mathcal{A}$ is the set of all the equivalences classes induced by the relation $\mathcal{R}$.

For any valuation ν, let $[\nu]$ denote the region to which ν belongs.

A region r is non-punctual if it contains some $\nu \in r$ such that $\nu + [-\varepsilon, \varepsilon] \subseteq r$ for some $\varepsilon > 0$. It is punctual otherwise.

By extension, we say that (ℓ, r) is non-punctual if r is for some location ℓ.

Definition 3.5 (Region automaton)**.** Fix a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{A}, \mathsf{E})$, the ***region automaton*** is a finite automaton whose states are pairs (ℓ, r) where ℓ is in $\mathcal{L}$ and r is in $\mathsf{Reg}(\mathcal{A})$. A transition $(\ell, r) \xrightarrow{\Delta} (\ell, s)$ exists if s is non-punctual[2], and there exist $\nu \in r$, $\nu' \in s$ and $d > 0$ such that $\nu' = \nu + d$. A transition $(\ell, r) \xrightarrow{e} (\ell', s)$ with $e = (\ell, g, R, \ell')$ if $r \models g$ and $r[R \leftarrow 0] = s$.

Remark 3.6. The set of regions is ***finite*** and ***exponential*** in the number of clocks, cf. [AD94] for more details.

[1]Any timed automaton can be transformed to satisfy this property.
[2]Note this slight modification in the definition of the region automaton.

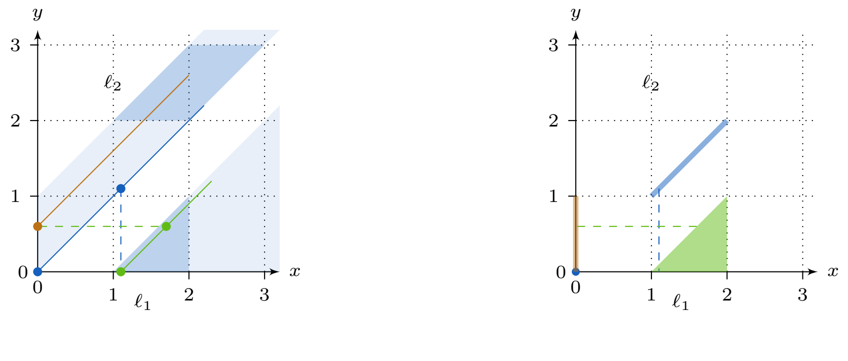


Figure 3.2: Regions of a timed automaton.

Example 3.7. In Figure 3.2a we recall the run from the automaton of Figure 3.1. We highlight the regions visited along this run. Notice how, the regions correspond exactly to the moment when the guards are enabled. ◁

We define paths over the region automaton as $\pi = q_1e_1q_2e_2\dots q_n$ where each q_i is a state, and $e_i \in \mathsf{E} \cup \{\Delta\}$, such that $q_i \xrightarrow{e_i} q_{i+1}$ for all $1 \leq i \leq n-1$. The length of the path is n, and is denoted by $|\pi|$.

The region automaton and the timed automaton it abstracts are bond through a relation called timed bisimulation [BK08]. Intuitively, this relation implies that the region automaton is a sound and complete abstraction for the timed automaton, that is:

Given a run $\rho = p_1e_1p_2e_2\dots p_n$ over $\mathcal{A}$ with $e_i \in \mathbb{R}_{>0} \cup \mathsf{E}$, its projection on regions is a path $\pi = q_1e'_1q_2e'_2\dots q_n$ in the region automaton such that $p_i \in q_i$, and $e'_i = e_i$ if $e_i \in \mathsf{E}$ and $e'_i = \Delta$ otherwise.

Conversely, given a path $\pi = q_1e'_1q_2e'_2\dots q_n$ in the region automaton, there exists a run $\rho = p_1e_1p_2e_2\dots p_n$ over $\mathcal{A}$ such that $p_i \in q_i$, and $e'_i = e_i$ if $e_i \in \mathsf{E}$ and if $e'_i = \Delta$ then there exist a delay δ such that $e_i = \delta$.

This last fact implies, that checking a reachability property over a timed automaton can be done over its region automaton. Actually any ω-regular property can be checked over the region automaton [BK08].

Now that we have introduced the main notions, we can move to the core of this chapter, this concerns the synthesis problem.

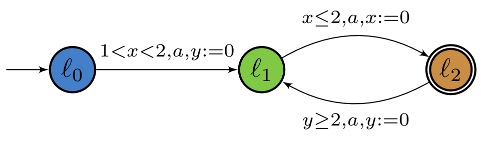
Robustness game for timed synthesis

Usually, a timed game played over a timed automaton involves two entities; a controller and an environment. In each location, the controller chooses a delay and an action such that at least one guard is satisfied. The environment solves the non-determinism and the play carries on.

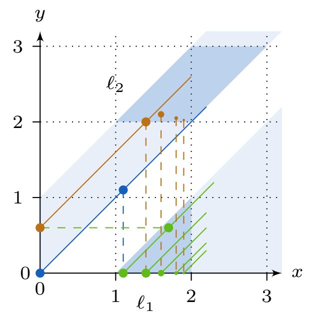
Example 3.8. In order to grasp some intuition, consider Figure 3.3, where the timed automaton from the previous example is viewed as a game. Since this automaton is deterministic, only controller plays. Clearly the controller can choose delays such that he induces an infinite run. This is displayed in Figure 3.3b, his strategy is straight-forward, he needs to choose delays satisfying the guard in ℓ_2 but he should not wait for too long otherwise he risks causing the play to be blocked in ℓ_1. At this point, one can just try to play "as soon as possible" in ℓ_1, but this will also cause the play to be blocking. This he needs to find a balance. However this balance is mathematically sound, it can never be implemented in a real life scenario. Let us sketch the issue: we first propose the following strategy σ:

- $\sigma(\ell_1, x, 0) \mapsto \frac{2-x}{2}$,
- $\sigma(\ell_2, 0, y) \mapsto y - 2$.

This strategy waits in ℓ_2 a time laps that ensures sufficient "room" in ℓ_1 to play. Nevertheless, there is an unavoidable progress of time causing the time to converge toward $(2, 0)$. The effect of this convergence is to play faster and faster in ℓ_1. This phenomenon translates in real life to devices requires ***infinitely small*** precision. $\triangleleft$



(a) Timed automaton seen as a timed game.



(b) in order to play infinitely, Controller has to enforce the convergence.

Figure 3.3: An infinite play in the timed game.

The main interest of this chapter is to develop algorithms for the timed synthesis problem where the designed solutions are implementable, i.e., solution that do not rely on convergence. We base our analysis on the theory of robustness for timed automata [Pur00, BA11]. We extend the framework of robustness in timed automata to the setting of two-player game. Especially, we define the following game semantics, where we ask the controller to ensure a specification without requiring to convergence phenomena.

Intuitively, let $\delta > 0$, Player 1, also called Controller chooses a delay $d > \delta$ and an action $a \in \mathsf{A}$ such that every a-labeled enabled edge is such that its guard is satisfied after any delay in the set $d + [-\delta, \delta]$ (and there exists at least one such edge). Then, Player 2, also called Perturbator chooses an actual delay $d' \in d + [-\delta, \delta]$ after which the edge is taken, and chooses one of the enabled a-labeled edges. Hence, Controller is required to always suggest delays that satisfy the guards whatever the perturbations are.

In the game of Figure 3.3, Controller cannot ensure the specification since Perturbator can always manage to bock the play. Indeed, being able to ensure a specification against a non-controllable perturbation can be seen as being able to ensure the same specification without requiring to an infinitely small precision hence the implementability. We now formalize the above game semantics. The automaton of this example was presented in [Pur00] where it was shown to be not robust.

Parameterized timed game

Formally, given a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{A}, E)$ and $\delta > 0$, we define the parameterized timed game of $\mathcal{A}$ w.r.t. δ as a two-player turn-based game $\mathcal{G}_\delta(\mathcal{A})$ between players Controller and Perturbator. The state space of $\mathcal{G}_\delta(\mathcal{A})$ is partitioned into $V_C \cup V_P$ where $V_C = \mathcal{L} \times \mathbb{R}_{\geq 0}^{\mathcal{C}}$ belong to Controller, and $V_P = \mathcal{L} \times \mathbb{R}_{\geq 0}^{\mathcal{C}} \times \mathbb{R}_{\geq 0} \times \mathsf{A}$ belong to Perturbator. The initial state is $(\ell_0, \vec{0}) \in V_C$. The transitions are defined as follows: from any state $(\ell, \nu) \in V_C$, there is a transition to $(\ell, \nu, d, a) \in V_P$ whenever $d > \delta$, for every edge $e = (\ell, g, a, R, \ell')$ such that $\nu + d \models g$, we have $\nu + d + \varepsilon \models g$ for all $\varepsilon \in [-\delta, \delta]$, and there exists at least one such edge e. Then, from any such state $(\ell, \nu, d, a) \in V_P$, there is a transition to $(\ell', \nu') \in V_C$ iff there exists an edge $e = (\ell, g, a, R, \ell')$ as before, and $\varepsilon \in [-\delta, \delta]$ such that $\nu' = (\nu + d + \varepsilon)[R \leftarrow 0])$.

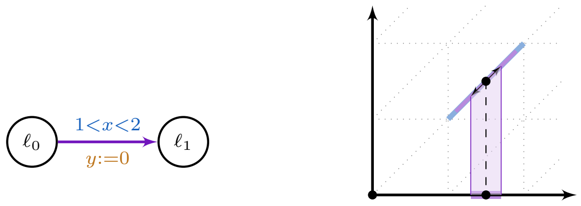


Figure 3.4: Interaction in the Parameterized timed game between Controller and Perturbator.

Example 3.9. Let $\delta > 0$, and consider Figure 3.4. In the left-hand side a transition in some timed automaton. Assume that it is Controller's turn to play and that the current configuration is $(\ell_0, 0, 0)$. Controller needs to pick a delay d such that $1 < x < 2$ is satisfied but remember that he has to allow a perturbation of d with at most $\pm\delta$. We depicted these possibilities in the right-hand side of Figure 3.4. Controller picks a delay inside the region $((1,1),(2,2))$, he makes sure that this delay is within a distance δ from the borders. We symbolize his pick with the bullet inside the region. It is now Perturbator's turn to make his move, he applies a perturbation ε bounded by δ and the play proceeds, i.e., clock y is reset, and we repeat the same interaction from $(\ell_1, \varepsilon, 0)$. $\lhd$

Plays and strategies

A play of $\mathcal{G}_\delta(\mathcal{A})$ is a finite or infinite sequence $q_1 e_1 q_2 e_2 \ldots$ of states and transitions of $\mathcal{G}_\delta(\mathcal{A})$, with $q_1 = (\ell_0, \vec{0})$, where e_i is a transition from q_i to q_{i+1}.It is said to be maximal if it is infinite or cannot be extended.

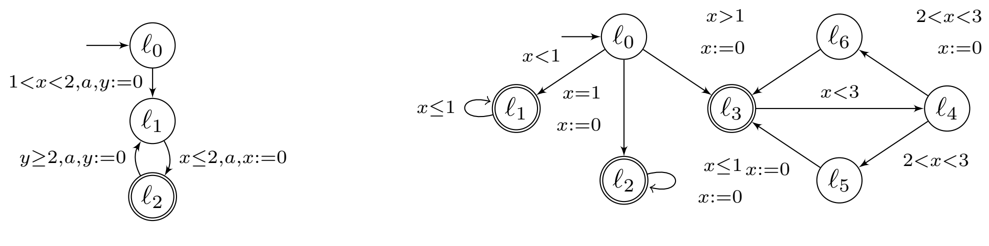
A strategy for Controller is a function that assigns to every non-maximal play ending in some $(\ell, \nu) \in V_C$, a pair (d, a) where $d > \delta$ and a is an action such that there exists a transition from (ℓ, ν) to (ℓ, ν, d, a).

Robust timed synthesis problem

Let $\delta > 0$, and let (σ, τ) be a pair of strategies, respectively for Controller and Perturbator, we denote ρ the unique maximal run that is compatible with both σ and τ. We will be interested in Büchi objectives, therefore we are given a subset of the locations of $\mathcal{A}$ that we call the Büchi set and denote B.

A Controller's strategy σ is winning w.r.t. B if against any Perturbator's strategy τ the run ρ that is compatible with the pair (σ, τ) is infinite and visits infinitely often a location of B.

The robust timed synthesis problem asks, given a timed automaton $\mathcal{A}$ and a Büchi set B, if there exists a $\delta > 0$ such that Controller has a winning strategy in $\mathcal{G}_\delta(\mathcal{A})$ w.r.t. B. When this holds, we say that Controller wins the robust timed game for $\mathcal{A}$, otherwise that Perturbator does.



(a) Not robustly controllable. (b) Robustly controllable.

Figure 3.5: Two timed games

We conclude this section by recapping the introduced game semantics in the following example.

Example 3.10. In Figure 3.5, two timed games are depicted. On the left, the timed game from the previous examples which is not robustly controllable for the Büchi objective $\{\ell_2\}$. Recall that we have seen that, Perturbator can enforce that the value of x be increased by δ at each arrival at ℓ_1, thus blocking the run eventually (cf. [SBMR13] for full details). The timed game on the right is robustly controllable for the Büchi objective $\{\ell_1, \ell_2, \ell_3\}$. For the sake of simplicity we assume that all transitions have the same label. The cycle around ℓ_1 cannot be taken forever, as value of x increases due to perturbations. The cycle around ℓ_2 can be taken forever, but Controller cannot reach ℓ_2 due to the equality $x = 1$. Controller's strategy is thus to loop forever around ℓ_3. This is possible as for both choices of Perturbator in location ℓ_4, clock x will be reset, and thus perturbations do not accumulate. If one of the two resets were absent, Perturbator could force the run to always take that branch, and would win the game. $\triangleleft$

3.3 An abstraction for solving the robust timed synthesis

Let us first give a flavor of the main challenge to overcome here. Recall the game of Figure 3.3, we have already sketched why a robust solution does not exist. Notice however that if we rely on the region automaton abstraction, it is possible to develop an infinite play. This is due to the fact that the region automaton does not offer sufficiently accurate knowledge on the evolution dynamics of configurations in the abstracted timed automaton. Thus, it cannot detect convergence, this is why we need to rely on different abstractions. Actually we will use regions to check the existence of an infinite play (as if there is no solution clearly there is no robust solution), then we will use the so-called orbit graph to check whether the witnessed solution is robust or not.

Folded orbit graphs

A vertex of a region r is any point of $\bar{r} \cap \mathbb{N}^{\mathcal{C}}$, where $\bar{r}$ denotes the topological closure of r. Let $\mathcal{V}(r)$ denote the set of vertices of r. We also extend this definition to $\mathcal{V}((\ell, r)) = \mathcal{V}(r)$.

With any path π of the region automaton, we associate a labeled bipartite graph $\Gamma(\pi)$ called the folded orbit graph of π [Pur00] (FOG for short). Intuitively, the FOG of a path gives the reachability relation between the vertices of the first and last regions. Formally, for a transition $\tau = q_1 e_1 q_2$, its orbit graph $\Gamma(\tau) = (V_1 \cup V_2, (q_1, q_2), E)$ is a bipartite graph where $V_1 = \{(1, v)\}_{v \in \mathcal{V}(q_1)}$, and $V_2 = \{(2, v)\}_{v \in \mathcal{V}(q_2)}$. For any $\big((1, u), (2, v)\big) \in V_1 \times V_2$, we have an edge $((1, u), (2, v)) \in E$, if, and only if $u \xrightarrow{\overline{e_1}} v$, where $\overline{e_1} = \Delta$ if $e_1 = \Delta$, and otherwise $\overline{e_1}$ is obtained by replacing the guard by its closed counterpart. Note that each vertex has at least one successor through $\overline{e_1}$ [AD94].

In order to extend $\Gamma(\cdot)$ to paths, we use a composition operator $\oplus$ between FOGs, defined as follows. If $G = (V_1 \cup V_2, (q_1, q_2), E)$ and $G' = (V_1' \cup V_2', (q_1', q_2'), E')$ denote two FOGs, then $G \oplus G'$ is defined if, and only if, $q_2 = q_1'$, that is, when the path defining the former graph ends in the first state of the path defining the latter graph. In this case, the graph $G'' = G \oplus G' = (V_1 \cup \ldots V_2', (q_1, q_2'), E'')$ is defined by taking the disjoint union of G and G', merging each node $(2, v)$ of V_2 with the node $(1, v)$ of V_1'. Now, we extend $\Gamma(\cdot)$ to paths by induction, as follows. Consider any path $\pi = q_1 e_1 \ldots q_{n-1} e_{n-1} q_n$, and let $G = (V_1 \cup V_2, (q_1, q_{n-1}), E)$ be the FOG $\Gamma(q_1 e_1 \ldots q_{n-1})$, given by induction. Let $G' = (U \cup U', (q_{n-1}, q_n), E')$ denote the bipartite graph of $q_{n-1} e_{n-1} q_n$. Then, we let $\Gamma(\pi) = G \oplus G'$. If the given path $\pi = q_1 \ldots q_1$ is a cycle, then we define $\Gamma(\pi)$ on the node set $\mathcal{V}(q_1)$, by merging the nodes of the bipartite graph corresponding to the same vertex. Note that delays of duration zero are allowed when defining orbit graphs.

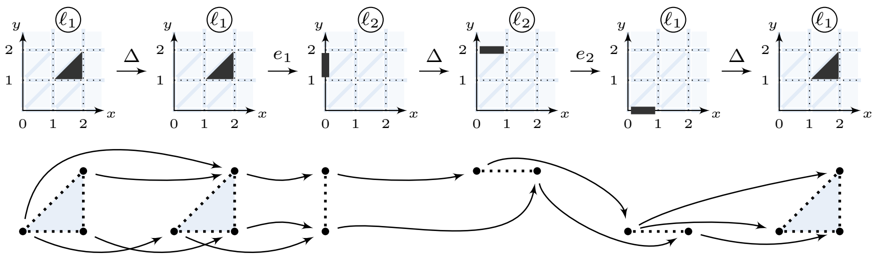


Figure 3.6: The orbit graph of a (cyclic) path of regions automaton of the automaton of Fig. 3.5a.

Example 3.11. In Figure 3.6, we give an example of an orbit graph. In the top part of the figure we display the regions visited along a path in the timed automaton of Figure 3.5a. In the bottom part we display the orbit graph induced by this path. Notice how in the orbit graph, the focus shifts from each region to its corners. Actually, we will see later that through the vertices of the orbit graph, and the way they are connected to each other, we will be able to detect convergence. In Figure 3.7 we depicted the fully factorized $\triangleleft$

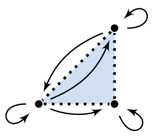


Figure 3.7: The FOG of the cycle of Fig. 3.6 ***"folded"*** over the corners of the region in ℓ_1.

Properties of FOGs

As explained in the beginning of the section we will use FOGs to discriminate between valid solutions and spurious ones in the regions automaton. In order to achieve this, we will first give some properties enjoyed by FOGs.

A FOG is said to be complete if the edge relation between its vertices is complete. Complete FOG are of particular interest to us as they exactly correspond to paths whose reachability relation, between valuations of the initial and last region, is complete [BA11]. This actually implies that there is no convergence phenomena along the path.

Another important notion that we will heavily rely on, is the one of aperiodic cycles. These are closely related to cycles whose FOG is a complete. In particular a long enough iteration of the former gives the latter and conversely, any cycle that owns some power with a complete FOG is aperiodic.

The following lemma formalizes the relation between aperiodic cycles and cycles with complete orbit graphs.

Lemma 3.12. *If π is an aperiodic cycle, then $\Gamma(\pi^n)$ is a complete graph for all $n \geq (|\mathcal{C}|+1) \times (|\mathcal{C}|+1)!$. Conversely, if π is a cycle such that $\Gamma(\pi^n)$ is a complete graph for some $n \geq 1$, then π is aperiodic.*

With the above lemma in mind we can sketch a characterization of "good" cycles . Consider a cycle ρ in the region automaton of some timed automaton $\mathcal{A}$. There is no convergence when ρ is iterated if and only if $\Gamma(\rho)$ is aperiodic. The intuition we can get from this last remark is that the goal of Controller will be to enforce his Büchi objective using an aperiodic cycle. Therefore we will design an algorithm that build a strategy for Controller to achieve this enhanced Büchi objective.

3.4 Robustness game

In this section, we will define an abstraction based on region automata in order to characterize winning in the robust timed game. We recall that the usual region automaton does not carry enough information for our purpose; for instance, the blocking behavior in Figure 3.5a cannot be detected in the region automaton, which does contain infinite runs. We therefore define, on top of the usual region construction, a complex winning condition $\mathcal{W}$ characterizing accepting runs along aperiodic cycles. In order to be able to transfer the condition $\mathcal{W}$ to the continuous semantics of the timed automaton, we study the properties of $\mathcal{W}$ on the abstract region game, and derive two necessary and sufficient conditions, $\mathcal{C}_C$ and $\mathcal{C}_P$, for winning which will be used to design an algorithm for solving the parameterized timed game.

Abstract arena and strategies

We fix a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \ell_0, \mathsf{A}, E)$. The region game automaton of $\mathcal{A}$ is a finite automaton $\mathcal{R}(\mathcal{A}) = (V, v_0, E)$, defined on the alphabet $\Lambda = \mathsf{Reg}(\mathcal{A}) \times \mathsf{A}$, where $V = \mathcal{L} \times \mathsf{Reg}(\mathcal{A})$ is the set of states (ℓ, r) of the region automaton, the initial state is $v_0 = (\ell_0, [\vec{0}])$, and the transitions are defined as follows. We have $(\ell, r) \xrightarrow{(r', a)} (\ell', s)$ if and only if the edges $(\ell, r) \xrightarrow{\Delta} (\ell, r')$ and $(\ell, r') \xrightarrow{a} (\ell', s)$ exist in the region automaton of $\mathcal{A}$. Thus $\mathcal{R}(\mathcal{A})$ is a non-deterministic finite automaton. In this game, Controller's strategy consists in choosing actions, while Perturbator's strategy consists in resolving non-determinism.

We consider standard notions of finite-memory, and memoryless strategies in this game and, given a finite-memory strategy σ, we denote by $\mathcal{R}(\mathcal{A})[\sigma]$ the automaton obtained under σ.

Winning condition in $\mathcal{R}(\mathcal{A})$

We define set $\mathcal{W}$ of winning plays in the game $\mathcal{R}(\mathcal{A})$: an infinite play is winning, i.e. it belongs to $\mathcal{W}$, if the following two conditions are satisfied:

1. an accepting state is visited infinitely often,
2. finite factors with complete folded orbit graphs are visited infinitely often.

The above game with the introduced winning condition is determined as state in the next proposition, it can also be solved in EXPTIME.

Proposition 3.13. *The game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ is determined, admits finite-memory strategies for both players, and wining strategies can be computed in* EXPTIME*.*

Proof. Sketch of the proof The above proposition is proved by showing that condition 2) of $\mathcal{W}$ can be rewritten as a Büchi condition: the set of folded orbit graphs constitute a finite monoid (of exponential size) which can be used to build a Büchi automaton encoding condition 2). Using a product construction for Büchi automata, one can define a Büchi game of exponential size where winning for any player is equivalent to winning in $(\mathcal{R}(\mathcal{A}), \mathcal{W})$. □

Now that we have a sketched an algorithm for solving $(\mathcal{R}(\mathcal{A}), \mathcal{W})$, it remains to show that $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ is an appropriate abstraction for our purpose. That is, there exists a robust controller in $\mathcal{A}$ if and only if there exists a winning strategy in $(\mathcal{R}(\mathcal{A}), \mathcal{W})$.

From the abstract game to the robust timed game

We introduce two conditions for Perturbator and Controller which are used to build concrete strategies in the parameterized timed game.

$\mathcal{C}_P$: There exists a finite memory strategy τ for Perturbator such that no cycle in $\mathcal{R}(\mathcal{A})[\tau]$ reachable from the initial state is winning aperiodic.

$\mathcal{C}_C$: There exists a finite-memory strategy σ for Controller such that every cycle in $\mathcal{R}(\mathcal{A})[\sigma]$ reachable from the initial state is winning aperiodic.

Remark 3.14. We relied on the determinacy of $(\mathcal{R}(\mathcal{A}), \mathcal{W})$, we write that either all cycles are aperiodic, or none is, respectively under each player's winning strategies.

The next lemma shows $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ enjoys the sought properties:

Lemma 3.15. *The winning condition $\mathcal{W}$ is equivalent to $\mathcal{C}_P$ and $\mathcal{C}_C$:*

- Perturbator *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ if and only if property $\mathcal{C}_P$ holds.*
- Controller *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$ if and only if property $\mathcal{C}_C$ holds.*

In both cases, a winning strategy for $\mathcal{W}$ is also a witness for $\mathcal{C}_C$ (resp. $\mathcal{C}_P$).

Sketch of the proof. The proof is obtained through the following facts:

- finite-memory strategies are sufficient to win the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$, thanks to the previous proposition;
- given a folded orbit graph γ, there exists n such that γ^n is complete if and only if γ is aperiodic;
- the concatenation of a complete FOG with an arbitrary FOG is complete.

□

This last lemma yields the following theorem

Theorem 3.16. *The robust timed synthesis problem is* EXPTIME*-complete.*

The hardness follows from classical techniques encoding a bounded tape alternating Turing machine.

3.5 Probabilistic Semantics

In some systems, considering the environment as a completely adversarial opponent is a bold assumption. In order to model an environment with less pessimistic behavior we consider two different models of timing imprecision.

The first one consists in considering random perturbations rather than adversarially chosen ones, we refer to this semantic as the stochastic game semantics. This semantics is defined as follows; Controller suggests delays and actions, Perturbator resolves non-determinism by choosing an edge, and the delay is perturbed uniformly at random (and independently at each step) in the interval $[-\delta, \delta]$.

In the second semantic the perturbations are randomized and the non-determinism in action is resolved according to a uniform distribution. We refer to this semantics as the MDP semantics. This semantics is defined as follows; Controller suggests delays and actions, the non-determinism is resolved by picking an edge according to a uniform distribution, and the delay is perturbed uniformly at random (and independently at each step) in the interval $[-\delta, \delta]$.

Stochastic Game Semantics

Formally, the state space is partitioned into $V_C \cup V_P$ as previously. Strategies for Controller determine a delay and an action $(d, a) \in \mathbb{R}_{\geq 0} \times \mathsf{A}$ at any step, and, given Controller's move, strategies for Perturbator simply choose an edge e with label a. The transition is then determined by choosing a delay at random in the interval $[d - \delta, d + \delta]$, and taking the edge e.

Once both players have chosen a strategy, we can define a probability. Our definitions follow closely [BBB+08a]. In particular, we define the probability measure on cylinders defined by the notion of constrained symbolic paths. This allows us to reuse the proofs of [BBB+08a] to derive the a well-defined measure, cf. [BBB+08a], and [BBB+08b, pages 40-42].

We define the probabilistic robust timed synthesis as the problem of deciding if Controller has a strategy ensuring the Büchi condition almost surely. It turns out that the same characterization from Lemma 3.15 holds in the probabilistic case.

Theorem 3.17. *Under the stochastic game semantics, if $\mathcal{C}_C$ holds then* Controller *wins almost-surely; if $\mathcal{C}_P$ holds then* Perturbator *wins almost-surely.*

Recall that $\mathcal{C}_C$ and $\mathcal{C}_P$ are complementary (Lemma 3.15). This is actually a quite powerful result since it yields the following corollary

Corollary 3.18. *Under the stochastic game semantics,* Controller *wins with either probability* 0 *or* 1.

MDP semantics

In the MDP semantics, for given $\delta > 0$, at each step, Controller picks a delay $d \geq \delta$, and an action a such that for every edge $e = (\ell, g, a, R, \ell')$ such that $\nu + d \models g$, we have $\nu + d + \varepsilon \models g$ for all $\varepsilon \in [-\delta, \delta]$, and there exists at least one such edge e. Then, a perturbation $\varepsilon \in [-\delta, \delta]$, and an edge satisfying the above conditions are chosen independently and uniformly at random. Thus in this semantics, Controller's strategy resolves all non-determinism, and defines a random process. We denote by $\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A})$ the resulting game, and $\mathbb{P}^{\sigma}_{\mathcal{G}_\delta^{\mathsf{MDP}}(\mathcal{A}),s}$ the probability measure on $\mathsf{Runs}(A, s)$ under strategy σ.

Remark 3.19. In the MDP semantics, Perturbator does not have any strategical power, in other word we can say that always plays the strategy where he resolves the non-determinism uniformly at random. Thus this semantics is a particular case of the stochastic game one. Therefore it follows that $\mathbb{P}^{\sigma}_{\mathcal{G}^{\mathsf{MDP}}_{\delta}(\mathcal{A}),s}$ is well defined for any Controller's strategy σ.

For a given timed Büchi automaton, we denote by ϕ the set of accepting runs. We are interested in the two following problems:

Definition 3.20 (Almost-sure winning)**.** Does there exist $\delta > 0$ and a strategy σ for Controller such that $\mathbb{P}^{\sigma}_{\mathcal{G}^{\mathsf{MDP}}_{\delta}(\mathcal{A}),s_0}(\phi) = 1$?

Definition 3.21 (Limit-sure winning)**.** Does there exist, for every $0 \leq \varepsilon \leq 1$, a perturbation upper bound δ, and a strategy σ for Controller such that $\mathbb{P}^{\sigma}_{\mathcal{G}^{\mathsf{MDP}}_{\delta}(\mathcal{A}),s_0}(\phi) \geq 1 - \varepsilon$?

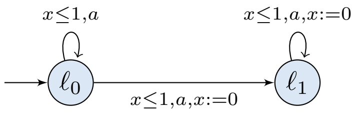


Figure 3.8: Limit-sure VS. almost-sure.

Example 3.22. Consider the automaton of Figure 3.8. This automaton is loosing under the MDP semantics for the almost-sure objective but winning under the same semantics for the limit-sure objective. Indeed, notice that as long as a play reaches state ℓ_1, then Controller is ensured to win since he can always play in the center of the region and be safe w.r.t. to any perturbation. However, from state ℓ_0, he has to ensure that the play leaves this state before it is blocked, but for any $\varepsilon > 0$, there always will be small portion of plays that will stay in state ℓ_0 for a duration long enough to allow the accumulation of the randomly chosen perturbations to block the play in the same state. Hence, this small proportion of plays will be loosing. $\triangleleft$

Remark 3.23. Observe that the second problem has a very concrete interpretation in terms of controller synthesis: given a quantitative constraint on the quality of the controller, what should be the precision on clocks measurements to be able to synthesize a correct controller?

The main result of this section is:

Theorem 3.24. *One can decide in* EXPTIME *whether* Controller *wins almost-surely (resp. limit-surely) in the MDP semantics of a timed Büchi automaton.*

In order to establish the above theorem, we define two conditions $\mathcal{W}'$ and $\mathcal{W}''$. The former will be used to witness almost-sure winning, the latter will be used to witness limit-sure winning.

Almost-sure winning

We consider the region game automaton $\mathcal{R}(\mathcal{A})$ of $\mathcal{A}$, viewed as a (finite-state) Markov decision process, in which the non-determinism of actions is resolved according to a uniform distribution. Given a strategy $\hat{\sigma}$ for Controller and a state v, we denote by $\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}),v}$ the resulting measure on $\mathsf{Runs}(\mathcal{R}(\mathcal{A}), v)$. The initial state of $\mathcal{R}(\mathcal{A})$ is v_0. We introduce A new winning condition:

Definition 3.25 (Condition $\mathcal{W}'$)**.** Consider the region game automaton $\mathcal{R}(\mathcal{A})$ of $\mathcal{A}$, viewed as a (finite-state) Markov decision process. Let $\hat{\sigma}$ be a Controller's strategy in $\mathcal{R}(\mathcal{A})$. $\hat{\sigma}$ is ***winning for*** $\mathcal{W}'$ from a state v if:

- $\mathbb{P}^{\hat{\sigma}}_{\mathcal{R}(\mathcal{A}),v}(\phi) = 1$,

- every outcome of $\mathcal{R}(\mathcal{A})[\hat{\sigma}]$ starting in v contains infinitely many factors whose FOG is complete.

Remark 3.26. Observe that the first item is an almost-sure requirement while the second is a sure requirement. This latter requirement can be seen as a condition in a game where random choices are made by an adversarial entity. Thus in following proposition, we slightly abuse the definition of an MDP.

Proposition 3.27. Controller *wins almost-surely in the MDP semantics of a timed Büchi automaton $\mathcal{A}$ if and only if* Controller *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$ from v_0.*

Thus it remains to solve the game $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$.

Lemma 3.28. *The game $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$ is determined, admits finite-memory strategies for both players, and wining strategies can be computed in* EXPTIME*.*

As observed before, the condition on FOGs can be expressed as a deterministic Büchi automaton. We thus obtain a winning condition expressed as the conjunction of an almost-sure Büchi condition with a sure Büchi condition. We can solve these games in polynomial time using standard attractors computation. The finite memory strategy essentially alternates between the strategies for the these two objectives.

Finally Proposition 3.27 follows, using similar ideas from the setting of Section 3.4 together with the fact that any almost-surely winning strategy $\hat{\sigma}$ in $(\mathcal{R}(\mathcal{A}), \mathcal{W}')$ yields an almost-surely winning strategy σ in $\mathcal{A}$.

Limit-sure winning

In order to solve the problem under this semantics, we again introduce a new winning condition denoted by $\mathcal{W}''$. This winning condition again mixes an almost-sure objective with a sure objective. Denote first by Win' the set of winning states of Controller in the game $(\mathcal{R}(\mathcal{A}), \mathcal{W})$.

Definition 3.29 (Condition $\mathcal{W}''$)**.** Then $\mathcal{W}''$ is defined as the almost-sure reachability of the set $\textsc{Win}'$.

With this last condition, we characterize a solution in this setting as follows:

Proposition 3.30. Controller *wins limit-surely in the MDP semantics of a timed Büchi automaton $\mathcal{A}$ if and only if* Controller *wins the game $(\mathcal{R}(\mathcal{A}), \mathcal{W}'')$ in v_0.*

The proof of this proposition relies on the following lemma, and uses techniques similar to those involved in the proof of Proposition 3.27.

Lemma 3.31. *The game $(\mathcal{R}(\mathcal{A}), \mathcal{W}'')$ is determined, admits finite-memory strategies for both players, and wining strategies can be computed in* EXPTIME*.*

3.6 Discussion

wrap-up

In this chapter we developed a formal framework for studying the problem of robust synthesis in real-time systems. Our approach relies on a game semantics used in [SBMR13]. This game semantics build a parameterized problem. A restriction of this parameterized problem to deterministic timed automata was already considered and solved in [SBMR13]. In the case where the automaton is deterministic, the power of Perturbator is restricted as it only modifies the delays suggested by Controller, but has no non-determinism to resolve. From a technically stand point, the algorithm consists in finding an aperiodic cycle, these are cycles that are "stable" against perturbations. This notion was defined in [BA11] to study entropy in timed languages and was also at the core of this chapter.

A variant of the semantics we considered was studied in [BMS12] for deterministic timed automata and shown to be EXPTIME-complete for reachability due to an implicit presence of alternation. Timed games, Büchi conditions, or stochastic environments were not considered.

Probabilistic timed automata (PTA) where the non-determinism is resolved following probability distributions have been studied [Jen96, Bea03, KNSS02]. Our results consist in deciding almost-sure and limit-sure Büchi objectives in PTAs subject to random perturbations in the delays. Note that PTAs are equipped with a possibly different probability distribution for each action. Although we only consider uniform distributions, these two settings are equivalent for almost-sure and limit-sure objectives. Games played on PTA were considered in [FKNT10] for minimizing expected time to reachability with NEXPTIME $\cap$ co-NEXPTIME algorithms.

To the best of our knowledge, the work presented in this chapter is the first to study a stochastic model of perturbations for synthesis in timed automata.

Our results on stochastic perturbations can also be seen as a new interpretation of robustness phenomena in timed automata. In fact, in the literature on robustness in timed automata, non-robust behaviors are due to the accumulation of the imprecision δ along long runs, and in the proofs, one exhibits non-robustness by artificially constructing such runs (e.g. [DDMR08, SBMR13]). In contrast, in the stochastic setting, we show that non-robust behaviors either occur almost-surely, or can be avoided almost-surely (Theorem 3.17).

Perspectives

So far most of the algorithmic results regarding robust synthesis of timed systems rely on region based techniques. A first natural direction to consider would be to design symbolic approaches. A first step has been made in [BMRS19] where the deterministic setting of [SBMR13], the next step would be to extend this symbolic approach to the non deterministic setting presented earlier in the present chapter.

The results partially answer the challenges posed in [CHP11] since we present a solution for the problem for unknown parameter δ. However, it still remains to extend our result to the more general setting of concurrent timed games introduced in [CHP11].

In the context of robust synthesis under the MDP semantics, we have only considered qualitative question, i.e., almost-sure winning, limit-sure winning. But it would be also interesting from a system design point of view to develop techniques quantifying the robustness of system. For instance one could try to define a threshold problem aiming at computing strategies ensuring that the measure of non-robust runs is bounded away from some probability $0 < p < 1$.

Finally, one can aim at designing a notion of "repair" for non robust systems in the sense that a system can be robust if a strict subset of the clocks is precise. As a first step one can aim at defining a problem where some clocks are subjected to perturbation while others are not. This could be used in a system where some clocks are used for synchronization purposes, those clocks should thus be precise by assumption. An even simpler version of this problem could be imagined where some transitions are perturbed and others are precise. We define all these latter problem as instance of a new notion of robustness that we call partial robustness. A first step towards this direction was made in the results presented in [BBGDO24].

Chapter 4

Rational synthesis

4.1 Outline of the chapter

In this chapter we depart from the setting of zero-sum games to the one of non zero-sum multiplayer games. In this setting, the synthesis problem aims at building a strategy profile, i.e., a strategy for each player. The computed profile is satisfactory if it ensures some rationality constraint that we will define later. This problem is called the rational synthesis.

There are two forms of rational synthesis: cooperative and non-cooperative [KS22]. Cooperative rational synthesis can be seen as the synthesis of a Nash equilibrium that satisfies a given specification, and this is akin to the problem of equilibrium checking [AGH+21]. One must practically convince every player that everyone else is playing their part of the Nash equilibrium. Since no one has an incentive to change their strategy, the Nash equilibrium should be played, and the specification of the system should be satisfied. Non-cooperative rational synthesis [KPV14] is different in that, one suggests a strategy for only a given player (considered to be a controlled entity). The problem becomes to automatically constructs a strategy for the controller such that against any Nash equilibrium which contains the controller's strategy, the specification of the system holds.

We introduce and study the problem of rational synthesis and its implementability with respect to common resources. These are resources like water, air, coal, pastures, or fish stocks [Ost90]. They are non-excludable: they are out there for the taking. They are rivalrous: one agent's consumption can limit or prevent another agent to consume it. We will focus on the resource of energy, as it is understood in the literature in Computer Science. In fact, it is both more abstract, and much more than "energy": it captures any common-pool resource that can be conveniently quantified by assigning a number to it, and where a bigger number indicates a greater amount of the resource. Energy also is a kind of resource that typically can be framed as a commons [MCB+17].

For a long time, it was believed that common-pool resources were bound to collapse due to the actions of self-interested agents, causing the resources to be depleted or spoiled. The great contribution of E. Ostrom [Ost90] was to evidence wide-ranging instances where this is not in fact the case, and to identify design principles of successful common-pool resource management. The models and algorithmic solutions presented here are contributions to the pursued efforts of engineering solutions for commons management. Specifically, we investigate the synthesis of a common controller which is compliant with rational behavior of all the player, modeled as a Nash equilibrium, but also sustainable with respect to the common resource.

Throughout this chapter, the player interact in turn-based fashion in a finite arena, which is a graph where in each state one player decides the next edge to follow. To each edge is associated an integer which corresponds to the energy cost incurred by the whole system when it follows this edge. Each player has a qualitative objective.

We propose two types of players: careless and careful. Careless player bother only about their objective. Careful player also pay attention to not deplete the system's resources. We will see that this two types of players induce two different notions of equilibria.

In what follows, we report a series of contribution obtained in collaboration with Rodica Condurache, Catalin Dima, and Nicolas Troquard [CDOT21, CDOT23]. The results that we preview are:

- We introduce the notions of cooperative careless and cooperative careful rational synthesis, cf., Definition 4.6 and Definition 4.11.
- We develop algorithm tool to study the above problem when the specification for all the players are given in term of parity objectives, cf., Proposition 4.9 and Proposition 4.16.
- We give tight complexity bounds for the both settings of the cooperative rational synthesis, cf., Theorem 4.10 and Theorem 4.18.
- We introduce the problem of non-cooperative rational synthesis in the context of common resources, cf., Equation 4.3.
- We develop an algorithmic framework for solving this latter problem cf., Proposition 4.20.
- We establish complexity bound for this latter problem, cf. Theorem 4.25.

This chapter is organized as follows:

- In Section 4.2, we present the model we are considering together with necessary formalisms and notations.
- In Section 4.3, we define and solve the problem of careless cooperative rational synthesis.
- In Section 4.5, we define and solve the problem of careful cooperative rational synthesis.
- In Section 4.6, we define and solve the problem of non-cooperative rational synthesis with common resources.

4.2 Multi-player games

Multi-player game, profiles, and payoffs

A multi-player game is a tuple $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E} \rangle$, where S is a finite set of states, $(\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n)$ is a partition of S, s_{ini} is an initial state, $\mathsf{P} = \{1, \ldots, n\}$ is the set of players, and E is in an edge relation in $\mathsf{S} \times \mathsf{S}$. For every edge $e = (s, t)$ in E, recall that $\mathsf{src}(e)$ is s and $\mathsf{trgt}(e)$ is t.

For an arena $\mathcal{G}$, we denote by $\mathsf{Plays}(\mathcal{G})$ the set of elements $s_{\mathsf{ini}} s_1 s_2 \ldots$ in S^ω such that for all $i \geq 0$, (s_i, s_{i+1}) is in E. The set $\mathsf{Hist}(\mathcal{G})$ is the set of finite and proper prefixes of elements in $\mathsf{Plays}(\mathcal{G})$. For a history h in $\mathsf{Hist}(\mathcal{G})$, $\mathsf{Last}(h)$ denotes the last element of h. Finally, $\mathsf{Hist}_i(\mathcal{G})$ for i in P is the set of elements in $\mathsf{Hist}(\mathcal{G})$ whose last element is in S_i i.e.,

$$\mathsf{Hist}_i(\mathcal{G}) = \{h \in \mathsf{Hist}(\mathcal{G}) \mid \mathsf{Last}(h) \in \mathsf{S}_i\} \ .$$

We use the same notion of strategies from Chapter 1 for two-player games, i.e., a strategy for player i is a function $\sigma_i \colon \mathsf{Hist}_i(\mathcal{G}) \to \mathsf{S}$ mapping a history whose last element is s to a state s' such that $(s, s') \in \mathsf{E}$.

For a strategy σ_i for player i, we define the set of outcomes, denoted by $\mathsf{out}\,(\sigma_i)$, as the set of plays that are compatible with σ_i i.e.,

$$\mathsf{out}\,(\sigma_i) = \{\pi \in \mathsf{Plays}(\mathcal{G}) \mid \forall j \geq 0,\ \pi[..j] \in \mathsf{Hist}_i(\mathcal{G}) \implies \sigma_i(\pi[..j]) = \pi[j+1]\}\ .$$

Once a strategy σ_i for each player i is chosen, we obtain a strategy profile $\overline{\sigma} = \langle \sigma_1, \ldots, \sigma_n \rangle$. $\overline{\sigma}_{\text{-}i}$ is the corresponding partial profile without the strategy for player i. For a strategy σ'_i for a player i, we write $\langle \overline{\sigma}_{\text{-}i}, \sigma'_i \rangle$ the profile $\langle \sigma_1, \ldots, \sigma'_i, \ldots, \sigma_n \rangle$. We denote by $\mathsf{out}\,(\overline{\sigma})$ the unique outcome of the strategy profile $\overline{\sigma}$.

Each player will be assigned an objective. An objective Obj is a subset of $\mathsf{Plays}(\mathcal{G})$. We write Obj_i to specify that it is the objective of player i. We define the payoff $\mathsf{Payoff}_i(\overline{\sigma})$ of player i w.r.t. the profile $\overline{\sigma}$ as follows:

$$\begin{cases} \mathsf{Payoff}_i(\overline{\sigma}) = 1 \text{ if } \mathsf{out}\,(\overline{\sigma}) \in \mathsf{Obj}_i, \\ \mathsf{Payoff}_i(\overline{\sigma}) = 0 \text{ otherwise.} \end{cases}$$

Example 4.1. In Figure 4.1, a three-player game is depicted. In this example, the initial state is a, it is controlled by the first player, the second plays controls state b, and the third player controls state c. The other states can belong to any players. The objective of the first player consists in the set of plays that reach $(1,1,0)$, the objective of the second player consists in the set of plays reaching state $(1,1,0)$ or state $(0,1,0)$, and the objective of the third player consists in the set of plays reaching state $(0,0,1)$. An example of a profile $\overline{\sigma}$, could be the one where every player plays to its right. In this case $\mathsf{out}\,(\overline{\sigma}) = abc(1,1,0)^\omega$, and the payoffs of $\overline{\sigma}$ are

$$\mathsf{Payoff}_1(\overline{\sigma}) = \mathsf{Payoff}_2(\overline{\sigma}) = 1\ ,\ \mathsf{Payoff}_3(\overline{\sigma}) = 0\ .$$

◁

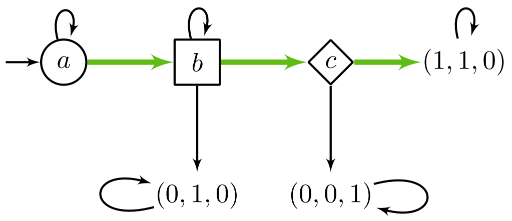


Figure 4.1: A multi-player game.

We recall that a zero-sum game is a game consisting of only two players with opposing objectives, i.e.,

$$\forall i \in \{1,2\},\ \mathsf{Obj}_{3-i} = \mathsf{Plays}(\mathcal{G}) \setminus \mathsf{Obj}_i\ .$$

Thanks to the zero-sum nature, we can define the notion of a winning strategy for player i, i.e., a strategy σ_i s.t. $\mathsf{out}\,(\sigma_i)$ is a subset of Obj_i.

Given a a multiplayer arena $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E} \rangle$, we write $\mathcal{G}^{i,\text{-}i}$ for the zero-sum game where player 1 is i, and player 2 is the coalition of the rest of players seen as one entity, i.e.,

$$\mathsf{S}_1 = \mathsf{S}_i,\ \mathsf{S}_2 = \bigcup_{j \neq i} \mathsf{S}_j,\ \mathsf{Obj}_1 = \mathsf{Obj}_i,\ \mathsf{Obj}_2 = \mathsf{Plays}(\mathcal{G}) \setminus \mathsf{Obj}_1\ .$$

Solution Concept

We define in our setting the notion of equilibrium introduced by Nash. A Nash equilibrium is a profile of strategies in which no player could do better by unilaterally changing his strategy, provided that the other players keep their strategies unchanged.

Formally, a strategy profile $\overline{\sigma}$ is a Nash equilibrium (NE) if for any strategy σ'_i for any player i in P we have:

$$\mathsf{Payoff}_i(\overline{\sigma}) \geq \mathsf{Payoff}_i(\langle \overline{\sigma}_{\text{-}i}, \sigma'_i \rangle) \ .$$

We write $\mathsf{NE}(\mathcal{G})$ for the set of all the profiles that are Nash equilibria in $\mathcal{G}$.

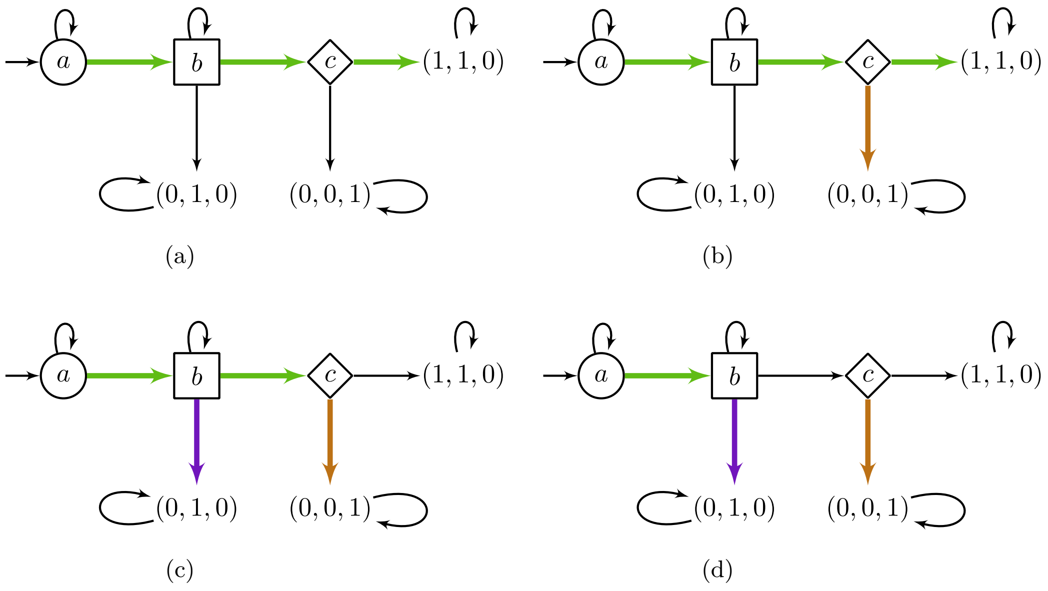


Figure 4.2: Nash equilibria in multi-player games

Example 4.2. Consider the games depicted in Figure 4.2. In Figure 4.2a, we highlight the profile from Example 4.1. We claim that this profile is not a NE. For instance, in Figure 4.2b we show that player 3 can unilaterally deviate and increase his payoff from 0 to 1. Once more this new profile is not an equilibrium since player 2 can deviate this time as it is shown in Figure 4.2c. The profile obtained and displayed in Figure 4.2d is a NE. It is not the only one, but in this example one can see it is not possible to obtain an equilibrium where player 3 gets a payoff of 1.◁

The rational synthesis problem

The goal of the rational synthesis is to check the possibility of building equilibria satisfying a given specification. We define the rational synthesis problem as follows:

Problem 4.3 (Rational synthesis)**.** Let $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E} \rangle$ be a multi-player game. Let $\mathsf{Obj}_1, \ldots, \mathsf{Obj}_n$ be objectives for players in P. Let also $\mathsf{Obj} \subseteq \mathsf{S}^\omega$ be a ***general objective***. The rational synthesis problem consists in deciding whether there exists $\overline{\sigma} \in \mathsf{NE}$ such that $\mathsf{out}\,(\overline{\sigma}) \in \mathsf{Obj}$.

Notice that in the above definition a valid solution assumes cooperation from all the players. Thus two notions of the rational synthesis were introduced, cooperative [FKL10] and non-cooperative [KPV14]. The cooperative version coincides with the latter defined problem, the non-cooperative will be introduced later in the chapter.

4.3 The cooperative and rational synthesis in the commons

Energy objectives

Let $\mathsf{cst}\colon \mathsf{E} \to \mathbb{Z}$ be a cost function. To lighten the notation, we write $\mathsf{cst}(s,t)$ instead of $\mathsf{cst}((s,t))$. Let $h = s_{\mathsf{ini}} \dots s_n$ be a history in $\mathsf{Hist}(\mathcal{G})$; we abusively write $\mathsf{cst}(h)$ to mean the extension of cst to histories that is:

$$\mathsf{cst}(h) = \mathsf{cst}(s_{\mathsf{ini}}, s_1) + \sum_{i=1}^{n-1} \mathsf{cst}(s_i, s_{i+1}) \enspace .$$

The energy objective for a game $\mathcal{G}$ equipped with a cost function cst is given by the set $\mathsf{Energy}(\mathcal{G})$ described as follows:

$$\mathsf{Energy} = \{\pi \in \mathsf{Plays}(\mathcal{G}) \mid \forall i \geq 1,\ \mathsf{cst}(\pi[..i]) \geq 0\} \enspace .$$

We denote by W the largest absolute value that appears in cst, i.e.

$$\mathsf{W} = \max\{|c| \in \mathbb{Z} \mid \exists e \in \mathsf{E},\ \mathsf{cst}(e) = c\} \enspace .$$

Throughout this chapter, values of cst are encoded in binary unless explicitly said otherwise.

Let $\mathcal{G}$ be a zero-sum game equipped with both a priority function prty and a cost function cst, the energy parity objective $\mathsf{EnergyParity}$ for this game is given by the set

$$\mathsf{EnergyParity} = \mathsf{Energy} \cap \mathsf{Parity} \enspace .$$

The following problem will be instrumental in our development. Given an energy-parity game and a state, the initial credit problem asks whether there exists an initial value for the energy such that the first player has a strategy to ensure both objectives. This problem was solved by Chatterjee et. al. [CD12b].

Theorem 4.4 ([CD12b]). *The initial credit problem can be solved in $O(|E|\ .\ D\ .\ |\mathsf{S}|^{d+2}\ .\ \mathsf{W})$ where d is the highest priority in the game.*

Synthesis in the common

Our goal here is to introduce a notion of rational synthesis where the players share a resource (the energy in our case). However, we differentiate between two type of players, careless and careful. A careless player is player who is not interested in the energy level while careful player is. With these two types of player in mind, we introduce two different notions of the cooperative rational synthesis; the careless cooperative rational synthesis, and the careful cooperative rational synthesis. In the former one, all the players are considered careless and only the general objective contains an energy condition. In the latter, all the players as careful and are concerned with not depleting the energy stored.

Example 4.5. Consider the arena depicted in Figure 4.3. Player 1 (circle) controls state a, his objective is given by the set of all the plays that ultimately reach state $(1, 1, 0)$. Player 2 (square)

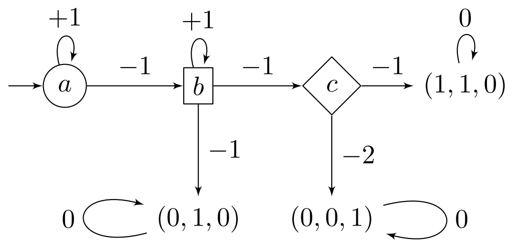


Figure 4.3: A multi-player game equipped with a cost function.

controls state b, his objective is given by the set of all the plays that ultimately reach either state $(1, 1, 0)$ or $(0, 1, 0)$. Player 3 (diamond) controls state c, his objective is given by the set of all the plays that ultimately reach state $(0, 0, 1)$.

Assume that the general specification in this game coincides with the objective of player 1, then a valid solution for the careless cooperation rational problem has to be an equilibrium that meets the objective of player 1 and is sustainable energy-wise.

Clearly from state a, player 1 has to move the play to state b, but since the cost of this edge is -1 he has to take the self-loop in a at least once. Suppose player 1 takes the self-loop in state a 3 times then moves the play to state b. Suppose further player 2 chooses to advance to state c. The current energy level is then 1. Now it is up to player 3 to decide where the play ends.

Despite the fact that he meets his objective by going to state $(0, 0, 1)$, in the careful setting player 3 cannot go to this state since the cost of that transition is -2 bringing the energy level below 0. Therefore, the only possible move is towards $(1, 1, 0)$. The resulting strategy profile is a fixed Nash equilibrium. Indeed, player 2 has no incentive to deviate, and since player 1 meets his objective, the described strategy is a solution to the careful rational synthesis problem.

Note that not all Nash equilibria are solutions for the cooperative rational synthesis. For instance, if in state b, player 2 moves the play to $(0, 1, 0)$, the resulting outcome is a Nash equilibrium but it is not in player 1's objective.

One can also see that the careless synthesis problem has no solution. This is due the fact that any path which is in player 1's objective would have to take the transition from c to $(1, 1, 0)$. But this is player 3's decision to make, who is unsatisfied by doing this and could deviate to state $(0, 0, 1)$ which is beneficial since the new play is in his objective.

4.4 Careless cooperative rational synthesis

We first focus on the case where all the players are careless, we start by formalizing our problem in the careless setting.

Definition 4.6. Let $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E} \rangle$ be a game, $\mathsf{cst}\colon \mathsf{E} \to \mathbb{Z}$ be a cost function, $\mathsf{Obj}_1, \ldots, \mathsf{Obj}_n$ be parity objectives (one for each player), Obj be a general parity objective, and let $\overline{\sigma}$ be a strategy profile. Then $\overline{\sigma}$ is a ***solution*** to the ***careless*** cooperative rational synthesis problem if:

$$\mathsf{out}\,(\overline{\sigma}) \in \mathsf{Energy} \cap \mathsf{Obj}, \text{and } \overline{\sigma} \in \mathsf{NE} \ .$$

We denote the set of all the solutions by $\underline{\mathsf{NE}}(\mathcal{G})$.

Logical Characterization

Our goal here is to provide a formal tool to recognize “good” outcomes, i.e. plays that are generated by profiles that are solutions to our problem. Our formal tool will heavily rely on a logical characterization using a quantitative version of the Linear Temporal Logic (LTL).

eLTL

Since we are considering quantitative extension for the rational synthesis problem, the following extension of LTL will come in handy. Introduced in [BBFR13], it is evaluated over plays as follows:

$$\rho \models_{\mathsf{Energy}} \phi \iff (\rho \models \phi) \wedge (\rho \in \mathsf{Energy}) \ .$$

Essentially, this logic is defined similarly to LTL but is evaluated over weighted runs and run ρ satisfies a formula ϕ if and only if it as a model for ϕ and $\mathsf{cst}(\rho[..n]) \geq 0$ for any $n \geq 1$.

Problem 4.7 (Energy LTL Model checking)**.** Given an arena $\mathcal{G}$, a cost function $\mathsf{cst}\colon \mathsf{E} \to \mathbb{Z}$, and an LTL formula ϕ, decide whether there exists a play π in $\mathsf{Plays}(\mathcal{G})$ such that $\pi \models_{\mathsf{Energy}} \phi$?

Theorem 4.8 ([BCHK14, DG09])**.** *Problem 4.7 is* PSPACE*-complete.*

Reduction to eLTL model-checking

Let $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E}\rangle$ be a multi-player game where each player i is given a parity objective Obj_i induced by a priority function prty_i and assume that $\mathcal{G}$ is endowed with a cost function $\mathsf{cst}\colon \mathsf{E} \to \mathbb{Z}$.

For each player i, we denote by $\underline{\mathsf{Win}}[\mathsf{Obj}_i]$ the set of states from where i has a winning strategy for his objective Obj_i against the coalition -i. This set is described by the propositional formula $\bigvee_{s \in \underline{\mathsf{Win}}[\mathsf{Obj}_i]} s$. For each player, the set $\underline{\mathsf{Win}}[\mathsf{Obj}_i]$ can be computed by any classical algorithm for the parity objectives, in non-deterministic polynomial time [Zie98].

For each player i, we denote by Φ_{prty_i} the LTL formula that defines the set of plays in $\mathsf{Parity}_i(\mathcal{G})$. This formula is defined as follows:

$$\Phi_{\mathsf{prty}_i} \equiv \bigwedge_{\substack{s \in \mathsf{S} \\ \mathsf{prty}_i(s) \text{ is odd}}} \left(\Box\Diamond s \to \bigvee_{\substack{s' \in \mathsf{S} \\ \mathsf{prty}_i(s') < \mathsf{prty}_i(s) \\ \mathsf{prty}_i(s') \text{ is even}}} \Box\Diamond s'\right)$$

Consider the following formula:

$$\Phi_{\mathsf{NE}} \equiv \bigwedge_{i \geq 1} \left(\neg\Phi_{\mathsf{prty}_i} \to \Box\neg\left(\bigvee_{s \in \underline{\mathsf{Win}}[\mathsf{Obj}_i]} s\right)\right)$$

The intuition behind the above formula is that along any play π a player that did not achieve his parity goal and never visited his winning region cannot deviate and improve his payoff. In [CFGR16a], it is shown that any play satisfying the above formula is the outcome of some profile in $\mathsf{NE}(\mathcal{G})$.

Fact 1 ([CFGR16a])**.** *Let* $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E}\rangle$ *be an arena where each player* i *is given an objective* Obj_i *induced by a priority function* prty_i *then:*

$$\forall \overline{\sigma},\ \mathsf{out}\,(\overline{\sigma}) \models \Phi_{\mathsf{NE}} \iff \overline{\sigma} \in \mathsf{NE}(\mathcal{G}) \ .$$

We now take advantage of both Fact 1 and Theorem 4.8 and get the following proposition.

Proposition 4.9. *Let $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E}\rangle$ be a multi-player game where each player i is given a parity objective Parity_i, $\mathsf{cst}\colon \mathsf{E} \to \mathbb{Z}$ be a cost function, and let Obj be a general objective. There exists a solution to the cooperative careless problem if and only if there exists π in $\mathsf{Plays}(\mathcal{G})$ such that:*

$$\pi \models_{\mathsf{Energy}} \mathsf{Obj} \wedge \Phi_{\mathsf{NE}} \enspace .$$

Theorem 4.10. *Let $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E}\rangle$ be a multi-player game where each player i is given a parity objective Obj_i induced by a priority function prty_i and assume that $\mathcal{G}$ is endowed with a cost function $\mathsf{cst}\colon \mathsf{E} \to \mathbb{Z}$. Then the careless cooperative rational synthesis is* NP*-complete for parity objectives.*

To decide if there is a strategy profile $\bar{\sigma}$ that is a solution to the careless cooperative synthesis problem,

we use the following non-deterministic procedure where we add an extra vacuous player that we call player 0. This player does not control any state but his objective is given by the general objective Obj.

- Guess a payoff $\bar{z} \in \{0,1\}^{n+1}$ for all the players and with $z_0 = 1$. We denote $I = \{i \leq n \mid z_i = 0\}$.
- For each player $i \in I$, guess a positional strategy $\tau_{\text{-}i}$ for the zero-sum game $\mathcal{G}^{i,-i}$.
- Guess a set $U \subseteq \bigcap_{i \in I}(S \setminus \underline{\mathsf{Win}}[\mathsf{Parity}_i])$. This set will serve for satisfying the parity conditions while maintaining the energy level.
- Guess the followings:
  i. A subset V s.t. $V \subseteq U$.
  ii. A cycle h_{par} of size $\leq 2 \cdot |I| \cdot |S|$ in V that visits all the states of V. h_{par} is meant to satisfy the parity conditions.
  iii. A cycle h_{ch} of size $\leq |S|$ in U. h_{ch} is meant to be a "charging cycle", useful when h_{par} has a negative cost.
  iv. Two paths h^1_{lnk} and h^2_{lnk} with $h^1_{lnk}[1] = h^2_{lnk}[|h^2_{lnk}|] = h_{par}[1]$ and $h^2_{lnk}[1] = h^1_{lnk}[|h^1_{lnk}|] = h_{ch}[1]$ in U. These two paths are meant to connect h_{ch} with h_{par}.
  v. A cycle h_{acc} and two paths h_1 and h_2 of size $\leq |S|$ s.t. $h_1[1] = s_{ini}$, $h_1[|h_1|] = h_{acc}[1] = h_2[1]$ and $h_2[|h_2|] = h_{par}[1]$. These three items are needed for reaching h_{par} with sufficient energy stores and ensuring that the energy level is still non-negative.

Then the algorithm performs the following steps – in which any "check" step which does not succeed blocks the algorithm:

1. Check whether $\forall i$, $\tau_{\text{-}i}$ is winning for -i in the game $\mathcal{G}^{i,\text{-}i}$.
2. Check whether the set V is a subset of U, is strongly connected, and $\forall i \leq n$, $\min\{\mathsf{prty}_i(s) \mid s \in U\}$ is odd if and only if $i \in I$.
3. Check that h_{par} satisfies the desired parity conditions: for each $i \notin I$, $\min\{\mathsf{prty}_i(h_{par}[j]) \mid 1 \leq j \leq |h_{par}|\}$ is even.
4. Compute the costs $\mathsf{cst}(h_{par})$ and $\mathsf{cst}(h_{ch})$.

5. If $\mathsf{cst}(h_{par}) \geq 0$, then check that at least one of the following properties hold:
   (a) For each $j \leq |h_1 h_2 h_{par}|$ check that $\mathsf{cst}(h_1 h_2 h_{par}[..j]) \geq 0$.
   (b) For each $j \leq |h_1 h_{acc}|$ check whether $\mathsf{cst}(h_1 h_{acc}[..j]) \geq 0$.
6. If $\mathsf{cst}(h_{par}) < 0$, check whether $\mathsf{cst}(h_{ch}) > 0$ and further perform the iterative checks (5.a) and (5.b) above. Further check that h_{lnk} has a nonempty intersection with both h_{par} and h_{ch}.

The hardness is due to the fact that the cooperative setting from [CFGR16a] is a particular case of our problem and it is already NP-hard for parity objectives.

4.5 Careful cooperative rational synthesis

We now consider the case where all the players are careful about not depleting the resource.

Definition 4.11. Let $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E}\rangle$ be a game, $\mathsf{cst}\colon \mathsf{E} \to \mathbb{Z}$ be a cost function, objectives $\mathsf{Obj}_1, \ldots, \mathsf{Obj}_n$, $\overline{\sigma}$ be a profile, and Obj a global objective.
$\overline{\sigma}$ is a ***solution*** to the ***careful*** cooperative rational problem if:

$$\begin{aligned}&\mathsf{out}\,(\overline{\sigma}) \in \mathsf{Energy} \cap \mathsf{Obj}, \text{and}\\ &\forall \sigma'_i \text{ a strategy for player } i \geq 1, \langle \sigma_{-i}, \sigma'_i\rangle \in \mathsf{Obj}_i \cap \mathsf{Energy} \implies \mathsf{out}\,(\overline{\sigma}) \in \mathsf{Obj}_i\ .\end{aligned}$$

We denote the set of all the solutions by $\overline{\mathsf{NE}}(\mathcal{G})$.

We propose a logical characterization in term of Constrained LTL model checking. This characterization is used to provide a PSPACE solution to the cooperative synthesis problem. The PSPACE lower bound is obtained by a reduction from the reachability problem in bounded one-counter automata. Finally, we establish that the complexity drops to NP when the weights are represented in unary.

A Logical Characterization

Let x be an integer variable, we will present an extension of LTL that interprets x as a counter value and allows to write formulas constrained by x. These constraints will be described using the following grammar:

$$\alpha ::= x \sim d \mid \neg\alpha \mid \alpha \wedge \alpha\ . \tag{4.1}$$

where x is the counter variable, $d \in \mathbb{Z}$, and $\sim \in \{<, >, \leq, \geq, =\}$.

The formulas of 1-CLTL are then:

$$\phi ::= \alpha \mid p \mid \neg\phi \mid \phi \vee \phi \mid \mathsf{X}\ \phi \mid \phi\ \mathsf{Until}\ \phi\ .$$

where α is a counter constraint.

Remark 4.12. Note that if we remove α from the above grammar the resulting logic is plain LTL.

Fix a set of atomic propositions PROP. The models of 1-CLTL(PROP) are pairs of mappings $\langle \mu_1, \mu_2\rangle$.

The mapping $\mu_1 : \mathbb{N} \to 2^{\mathsf{PROP}}$ indicates which are the atomic propositions true in every instant.

The mapping $\mu_2 : \mathbb{N} \to \mathbb{Z}$ indicates the counter value at all instants.

In these models, propositional variables are evaluated wrt. μ_1 and counter constraints are evaluated wrt. μ_2 as follows:

$$\langle \mu_1, \mu_2 \rangle, i \models p \text{ iff } p \in \mu_1(i), \qquad \langle \mu_1, \mu_2 \rangle, i \models x \sim d \text{ iff } \mu_2(i) \sim d \ .$$

The temporal operators are evaluated as in LTL.

The formula of this logic are evaluated over the runs of one-counter automata. A one-counter automaton is a tuple $\Gamma = (L, \delta, l_0)$, where L is a finite set of locations, δ is a set of transitions, and $l_0 \in L$ is the initial location. A transition in δ is a tuple (l, p, g, l'), where l and l' are locations, $p \in \mathbb{Z}$ is the weight of the transition, and g is a guard generated by the grammar of Equation (4.1). A run in a one-counter automaton is a pair $\langle \mu_1, \mu_2 \rangle$, where $\mu_1 : \mathbb{N} \to L$ and $\mu_2 : \mathbb{N} \to \mathbb{Z}$, and for every $i \geq 0$, if $\mu_1(i) = l$ and $\mu_2(i) = c$, $\mu_1(i+1) = l'$, $\mu_2(i+1) = c'$, then there is $(l, p, g, l') \in \delta$ such that $c' = c + p$ and $\langle \mu_1, \mu_2 \rangle, i \models g$.

Problem 4.13 (1-CLTL Model checking)**.** Given a one-counter automaton Γ and a 1-CLTL formula ϕ, decide whether there exists a run $\langle \mu_1, \mu_2 \rangle$ such that $\langle \mu_1, \mu_2 \rangle \models \phi$.

1-CLTL is a particular case of Constrained LTL with propositional variables, one integer variable, and one-step look-ahead, whose model checking is PSPACE-complete [DG09].

Theorem 4.14. *Problem 4.13 is* PSPACE*-complete.*

Reduction to 1-CLTL model checking

Similarly to the careless case, we will express the existence of a solution to the careful synthesis problem as a model checking question. First we define a mapping $\mathsf{Credit} : \mathsf{S} \times \mathsf{P} \to \mathbb{N} \cup \{\omega\}$, that associates with each couple of state and player the minimal initial credit to meet its objective against the coalition with a positive energy, ω means that the player cannot win with any initial credit. The minimal initial credit is computed thanks to the algorithm used in [CD12b] that computes for each state a minimal credit and a winning strategy in an energy-parity game for the first player (protagonist).

Let $\overline{\mathsf{Win}}[\mathsf{Obj}_i] = \{s \in \mathsf{S} \mid \mathsf{Credit}(s, i) \neq \omega\}$.

Let π be a play in $\mathsf{Plays}(\mathcal{G})$, define the extended play in the one-counter automaton:

$$\mu_{\pi_1}(i) = \pi[i+1], \qquad \mu_{\pi_2}(0) = 0, \qquad \mu_{\pi_2}(i+1) = \mathsf{cst}(\pi[..(i+1)]) \ .$$

We also define the following 1-CLTL formula:

$$\Phi_{\overline{\mathsf{NE}}} \equiv \bigwedge_{i \geq 1} \left(\neg \Phi_{\mathsf{prty}_i} \to \Box \neg \left(\bigvee_{s \in \overline{\mathsf{Win}}[\mathsf{Obj}_i]} s \wedge (x \geq \mathsf{Credit}(s, i)) \right) \right)$$

Now using a construction analogous to the one presented in the proof of Proposition 4.9, we can state the two following propositions.

Proposition 4.15. *Let $\langle \mu_{\pi_1}, \mu_{\pi_2} \rangle$ be the extended play of some play π in $\mathcal{G}$ such that*

$$\langle \mu_{\pi_1}, \mu_{\pi_2} \rangle \models \Phi_{\overline{\mathsf{NE}}} \ ,$$

then π is the outcome of an $\overline{\mathsf{NE}}$.

Proposition 4.16. *There exists a solution to the cooperative careful rational synthesis problem if and only if there exists a play π such that:*

$$\langle \mu_{\pi_1}, \mu_{\pi_2} \rangle \models \mathsf{Obj} \wedge \Phi_{\overline{\mathsf{NE}}} \wedge \Box(x \geq 0)$$

Proposition 4.17. *Careful cooperative rational synthesis is* PSPACE*-complete for Büchi and Parity objectives when the weights are encoded in binary.*

The PSPACE membership follows immediately from the poly-size logical characterization of solutions using 1-CLTL and Theorem 4.14. On the other hand, the hardness follows by encoding the problem of reachability in bounded one-counter automata [FJ15]. A bounded one-counter automaton Γ is a tuple (L, b, δ, l_0), where (L, δ, l_0) is a one-counter automaton, and $b \in \mathbb{N}$ is a counter bound. A transition in δ is of the form $(l, p, x \geq d_1 \wedge x \leq d_2, l')$ where, $p \in \{-b, \dots, b\}$, and $d_1, d_2 \in \{0, \dots, b\}$. The reachability problem in bounded one-counter automata asks, given a bounded one-counter automaton $\Gamma = (L, b, \delta, l_0)$ and a location $t \in L$, whether there is a run $\langle \mu_1, \mu_2 \rangle$ and an $i \in \mathbb{N}$ such that $\mu_1(i) = t$, and $\mu_2(i) = 0$. [FJ15, Corollary 12] states that the reachability problem in bounded one-counter automata is PSPACE-complete.

Theorem 4.18. *Careful cooperative rational synthesis is in* NP *for Parity objectives when weights are encoded in unary.*

4.6 Non-cooperative rational synthesis in the common

We will use the notion of fixed Nash equilibrium. A strategy profile $\overline{\sigma}$ is a fixed Nash equilibrium (f-NE) if for any strategy σ'_i for any player i in $\mathsf{P} \setminus \{1\}$ we have:

$$\mathsf{Payoff}_i(\overline{\sigma}) \geq \mathsf{Payoff}_i(\langle \overline{\sigma}_{\text{-}i}, \sigma'_i \rangle) \ .$$

We write f-NE($\mathcal{G}$) for the set of all the profiles that are fixed Nash equilibria in $\mathcal{G}$.

Given an arena $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \dots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E} \rangle$ and objectives $\mathsf{Obj}_1, \dots, \mathsf{Obj}_n$ the non-cooperative rational synthesis problem is to decide whether there exists a strategy σ_1 such that for all strategies $\sigma_2, \dots, \sigma_n$ the profile $\overline{\sigma} = \langle \sigma_1, \dots, \sigma_n \rangle$ satisfies the following:

$$\mathsf{out}\,(\overline{\sigma}) \in \text{f-NE}(\mathcal{G}) \implies \mathsf{out}\,(\overline{\sigma}) \in \mathsf{Obj}_1 \ . \tag{4.2}$$

We are interested in the setting where the controller is resource-aware, therefore we equip the arena $\mathcal{G}$ with a cost function cst. In this case the problem is to decide whether there exists a strategy σ_1 such that for all strategies $\sigma_2, \dots, \sigma_n$ the profile $\overline{\sigma} = \langle \sigma_1, \dots, \sigma_n \rangle$ satisfies the following:

$$\mathsf{out}\,(\overline{\sigma}) \in \text{f-NE}(\mathcal{G}) \implies \mathsf{out}\,(\overline{\sigma}) \in \mathsf{Energy} \ . \tag{4.3}$$

Since player 1 a.k.a. the controller is concerned with the resource, he wants to guarantee the feasibility of any rational behavior of the players. The objectives $\mathsf{Obj}_2, \dots, \mathsf{Obj}_n$ are induced by LTL formulas. We shall call the problem consisting in designing a controller satisfying (4.3) the non-cooperative feasible rational synthesis. We shall also call a controller that is a solution a resource-aware controller.

Given a multi-player arena $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \dots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E} \rangle$ and objectives $\mathsf{Obj}_2, \dots, \mathsf{Obj}_n$, the problem of non-cooperative feasible rational synthesis asks whether the controller has a strategy σ_1 such that for all strategies $\sigma_2, \dots, \sigma_n$ of the other players the profile $\overline{\sigma} = \langle \sigma_1, \dots, \sigma_n \rangle$ satisfies Equation 4.3.

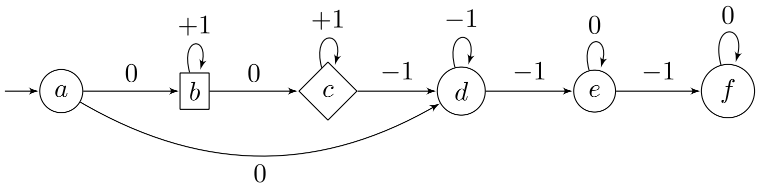


Figure 4.4: A three-player game. Player 1 (circle) has a vacuous qualitative objective and thus only cares about not depleting the resource.

Example 4.19. In the game illustrated on Figure 4.4, there is a solution to the problem of ***non-cooperative feasible rational synthesis***. Suppose that player 1 (controls circle states) is only interested in maintaining the energy of the system non-negative. Player 2 and player 3 control the square and diamond states, respectively, and want to reach the state labeled e and f, respectively, regardless of the energy level (i.e., their objective are induced by the LTL formulas $\mathsf{F}e$ and $\mathsf{F}f$).

Player 1 does not have a winning strategy to ensure that the energy remains above zero. Indeed, from a, he must go to b where player 2 could go straight to c and player 3 could go straight to d, bringing the level of energy below zero. The alternative move from a to go to d necessarily brings the energy below zero at the following step. A strategy σ_1 for player 1 which is a solution to the problem of non-cooperative feasible rational synthesis can be informally described as follows:

- in a go to b;
- in d go to the state labeled e if the energy is at least 1, and loop otherwise;
- in the state labeled e go to the state labeled f if the energy is at least 1, and loop otherwise;
- in the state labeled f, loop.

When player 1 plays σ_1, one possible Nash Equilibrium is when both player 1 and player 2 loop over c and d, respectively, for ever. This is a Nash equilibrium, because even though both player 2 and player 3 lose (they do not reach either e and f), they do not have a unilateral deviation to do so. Still, the energy remains above zero, and player 1 satisfies his objective. Another Nash Equilibrium is when player 2 and player 3 loop over c and d a finite number of times and for a total of at least 3 times. In these cases, the game reaches the state labeled f, where all the players satisfy their objectives, including player 1 who satisfy his energy objective. ◁

Before we move on to the resolution of our new synthesis problem, we emphasize that player 1 does not have a qualitative objective. We will later introduce the problem of non-cooperative feasible rational synthesis with rich specifications where the system's specification also includes a qualitative objective, and we will show that we can solve it in a uniform manner.

4.7 Computing a resource-aware controller

We adapt a proof technique initially proposed in [BBMU15] to synthesize Nash equilibria in concurrent games with ω-regular objectives. It was later successively adapted to solve the problem of non-cooperative rational synthesis in turn-based games [CFGR16b] and in concurrent games [COT18]. We will follow more closely the presentation of the latter.

The negotiation game

The technique consists in constructing a turn-based two-player game (we call it the “negotiation game”) which is an abstraction of the original multi-player game, in such a way that there is a winning strategy in the abstraction if and only if there a solution to the non-cooperative rational synthesis in the original game.

The construction results in a turn-based two-player game with objectives (a Boolean combination of two parity objectives and an energy objective) for which no known solution exists.

Given a game arena $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E}\rangle$ we construct a turn-based 2-player zero-sum game arena $\mathcal{H} = \langle \widehat{\mathsf{S}}, (\widehat{\mathsf{S}}_{\mathsf{C}} \uplus \widehat{\mathsf{S}}_{\mathsf{S}}), \widehat{s}_{\mathsf{ini}}, \widehat{\mathsf{E}}\rangle$ in which Constructor and Spoiler play, and where:

- $\widehat{\mathsf{S}}_{\mathsf{C}}$ is the set of states controlled by Constructor,
- $\widehat{\mathsf{S}}_{\mathsf{S}}$ is the set of states controlled by Spoiler,
- $\widehat{s}_{\mathsf{ini}}$ is the initial state,
- $\widehat{\mathsf{E}}$ is the transition table defined over $\widehat{\mathsf{S}} \times \widehat{\mathsf{S}}$.

The set $\widehat{\mathsf{S}}$ is:

$$\widehat{\mathsf{S}} = \left(\mathsf{S} \times (2^{\mathsf{P}})^2\right) \cup \left(\mathsf{S} \times (2^{\mathsf{P}})^2 \times (\mathsf{S} \cup \{-\})\right) \cup \left(\mathsf{S} \times (2^{\mathsf{P}})^2 \times (\mathsf{S} \cup \{-\})\right) \times \mathsf{S} \ .$$

The set $\widehat{\mathsf{S}}_{\mathsf{C}}$ is:

$$\widehat{\mathsf{S}}_{\mathsf{C}} = \{(s, W, D) \mid (s \in \mathsf{S}_1)\} \cup \{(s, W, D) \mid (s \in \mathsf{S}_{i \neq 1} \wedge i \in W)\} \cup \{(s, W, D, t) \mid (s \in \mathsf{S}_{i \neq 1}) \wedge (i \notin W \cup D)\} \ .$$

The set $\widehat{\mathsf{S}}_{\mathsf{S}}$ is $\widehat{\mathsf{S}} \setminus \widehat{\mathsf{S}}_{\mathsf{C}}$.
Let W and D be two subsets of P, $\widehat{\mathsf{E}}$ contains the following set of transitions:

$$\begin{aligned}
&(s, W, D) \mapsto (t, W, D) \text{ if } (s \in \mathsf{S}_1) \wedge ((s,t) \in \mathsf{E}) \ ,\\
&(s, W, D) \mapsto (t, W, D) \text{ if } (s \in \mathsf{S}_i) \wedge (i \in D) \wedge ((s,t) \in \mathsf{E}) \ ,\\
&(s, W, D) \mapsto (s, W, D, t) \text{ if } (s \in \mathsf{S}_{i \neq 1}) \wedge (i \in W) \wedge ((s,t) \in \mathsf{E}) \ ,\\
&(s, W, D) \mapsto (s, W, D, t) \text{ if } (s \in \mathsf{S}_i) \wedge \big((i \notin W \cup D) \wedge ((s,t) \in \mathsf{E})\big) \ ,\\
&(s, W, D, t) \mapsto (s, W, D, t, r) \text{ if } (s, r) \in \mathsf{E} \ ,\\
&(s, W, D, t) \mapsto (s, W, D, t, -) \ ,\\
&(s, W, D, t, -) \mapsto (t, W, D) \ ,\\
&(s, W, D, t, r) \mapsto (q, W', D') \ ,
\end{aligned}$$

where

$$\begin{aligned}
W' &= W \cup \{i\} \text{ if } (q = r) \wedge (s \in \mathsf{S}_i) \ ,\\
D' &= D \cup \{i\} \text{ if } (q = t) \wedge (s \in \mathsf{S}_i) \ .
\end{aligned}$$

$\widehat{\mathsf{cst}}$ is defined as follows:

$$\begin{aligned}
&\big((s, W, D), (t, W, D)\big) \mapsto \mathsf{cst}(s,t) \ ,\\
&\big((s, W, D, t, r), (r, W', D')\big) \mapsto \mathsf{cst}(s,r) \ ,\\
&\big((s, W, D, t, -), (t, W, D)\big) \mapsto \mathsf{cst}(s,t) \ ,\\
&0 \text{ in all the other cases.}
\end{aligned}$$

In the negotiation arena, we call the set of states in $\mathsf{S} \times \left(2^{\mathsf{P}}\right)^2 \times (\mathsf{S} \cup \{-\})$ negotiation states. The states in $\mathsf{S} \times \left(2^{\mathsf{P}}\right)^2$ are decision states.

Building a non cooperative solution

A play in the negotiation arena is a sequence of states. For the ease of notation we will consider the projection of this sequence over the decision states. We denote this set of sequences by $\widehat{\mathsf{S}}^\omega\restriction_{dec}$.

We equip $\widehat{\mathsf{S}}$ with the canonical projection proj_i that is the projection over the i-th component. In particular, for every $(s, W, D) \in \widehat{\mathsf{S}}$, we have $\mathsf{proj}_1((s, W, D)) = s$, $\mathsf{proj}_2((s, W, D)) = W$, and $\mathsf{proj}_3((s, W, D)) = D$. We also extend proj_i over $\widehat{\mathsf{S}}^+$ and $\widehat{\mathsf{S}}^\omega$ as expected.

The set $\overrightarrow{\mathsf{proj}}_2(\pi\restriction_{\widehat{\mathsf{S}}^1_\mathsf{C}})$ (resp. $\overrightarrow{\mathsf{proj}}_3(\pi\restriction_{\widehat{\mathsf{S}}^1_\mathsf{C}})$) is the set of agents in the limit of W's (resp. D's).

We define the following sets:

$$\widehat{\mathsf{Obj}_\mathsf{D}} = \{\pi \in \widehat{\mathsf{S}}^\omega \mid \exists p \in \overrightarrow{\mathsf{proj}}_2(\pi\restriction_{dec}), \mathsf{proj}_1(\pi\restriction_{dec}) \notin \mathsf{Obj}_p\} \ , \tag{4.4}$$

$$\widehat{\mathsf{Obj}_\mathsf{W}} = \{\pi \in \widehat{\mathsf{S}}^\omega \mid \forall p \in \overrightarrow{\mathsf{proj}}_3(\pi\restriction_{dec}), \mathsf{proj}_1(\pi\restriction_{dec}) \in \mathsf{Obj}_p\} \ , \tag{4.5}$$

$$\widehat{\mathsf{Energy}} = \{\pi \in \widehat{\mathsf{S}}^\omega \mid \widehat{\mathsf{cst}}(\pi) \geq 0\} \ . \tag{4.6}$$

Finally, we obtain the negotiation game by equipping the negotiation arena with the following winning condition:

$$\left((\widehat{\mathsf{Energy}} \cup \widehat{\mathsf{Obj}_\mathsf{D}}) \cap \widehat{\mathsf{Obj}_\mathsf{W}}\right) \ . \tag{4.7}$$

We briefly explain how it relates to analogous approaches in the literature. In [COT18], a very similar construction is presented in the case of concurrent arenas but with temporal objectives. Here we have adapted this construction to the simpler setting of turn-based games. It may appear that the case of turn-based games was already handled in [CFGR16b]. However, in the latter work, the approach consists in building a tree automaton. In our case, where the objective of player 1 is quantitative, this would lead to a new class of weighted tree automata for which we would need to solve the emptiness problem. In turn, solving the emptiness of this class of automata would require to solve a new class of turn-based games where the objective is $(\widehat{\mathsf{Energy}} \cup \widehat{\mathsf{Obj}_\mathsf{D}}) \cap \widehat{\mathsf{Obj}_\mathsf{W}}$. Instead, we directly build this turn-based game and extract a solution, by adapting the construction from [COT18].

The intuition behind the winning condition is the following: Constructor aims at building a solution. If the strategy he designs a strategy that ensures Energy then this strategy describes a correct solution. In the case where Energy is not achieved, Constructor has to prove that the players are not behaving rationally. This is where $\widehat{\mathsf{Obj}_\mathsf{D}}$ and $\widehat{\mathsf{Obj}_\mathsf{W}}$ come into play. Indeed $\widehat{\mathsf{Obj}_\mathsf{D}}$ ensures that Constructor has detected the possible deviators, while $\widehat{\mathsf{Obj}_\mathsf{W}}$ prevents him from detecting spurious deviations, i.e., allows Constructor to be sure the the player he is suspecting is following a profitable deviation.

These elements put together entail the following proposition.

Proposition 4.20. *There exists a solution to the problem of non-cooperative rational synthesis in the multi-player game $\mathcal{G}$ if and only if there exists a winning strategy for* Constructor *in $\mathcal{H}$.*

Solving the negotiation game

In order to solve the negotiation game we need to design an algorithm for a two-player game where the winning condition is given by Equation (4.7). We proceed as follows; we use results from [CD12b] on solving energy parity games, by first encoding the winning condition with an LTL formula. Then we transform this formula into a parity condition. At this stage one has to make sure that the formula does not blow up, actually the crux is to encode the winning condition of Equation (4.7) by a formula of size polynomial in $\mathcal{G}$. We recall that $\mathcal{G}$ is equipped with a labeling function $\mathsf{lbl}\colon \mathsf{S} \to 2^{\mathsf{AP}}$ where AP is a set of atomic propositions. We equip the Negotiation Game with a labeling function $\widehat{\mathsf{lbl}}$ which maps elements from $\widehat{\mathsf{S}}$ to subsets of $\widehat{\mathsf{AP}}$ a fresh set of atomic propositions. This new set of atomic propositions is :

$$\widehat{\mathsf{AP}} = \mathsf{AP} \cup \{p_W, p_D \mid p \in \mathsf{P}\} \ ,$$

and the mapping $\widehat{\mathsf{lbl}}$ is defined for each state $s \in \widehat{\mathsf{S}}$ as follows:

$$\widehat{\mathsf{lbl}}(s) = \mathsf{lbl}(\mathsf{proj}_1(s)) \cup \{p_W \mid p \in \mathsf{proj}_2(s)\} \cup \{p_D \mid p \in \mathsf{proj}_3(s)\} \ .$$

Then the set of plays defined in Equations (4.4), resp. (4.5), can be characterized by the following LTL formulas:

$$\widehat{\mathsf{Obj}_\mathsf{D}} \equiv \bigvee_{p \in \mathsf{P}} \left(\mathsf{FG}p_D \to \neg \mathsf{Obj}_{p_D}\right) \ , \tag{4.8}$$

$$\widehat{\mathsf{Obj}_\mathsf{W}} \equiv \bigvee_{p \in \mathsf{P}} \left(\mathsf{FG}p_W \leftrightarrow \neg \mathsf{Obj}_{p_W}\right) \ . \tag{4.9}$$

Note that the size of both these formulas is polynomial in the size of the original arena $\mathcal{G}$.

We apply classical results from [GTW02] to obtain two deterministic parity automata $\mathcal{A}_1$ and $\mathcal{A}_2$, recognizing the sets induced by $\widehat{\mathsf{Obj}_\mathsf{D}}$, resp $\widehat{\mathsf{Obj}_\mathsf{W}}$. The size of both these automata is doubly exponential in the size of the formulas for $\widehat{\mathsf{Obj}_\mathsf{D}}$ resp. $\widehat{\mathsf{Obj}_\mathsf{W}}$ that is, $O\left(2^{2^{|\mathcal{G}|}}\right)$, where $|\mathcal{G}|$ is the size of the description of $\mathcal{G}$. Finally, by a synchronous composition of both automata with the arena $\mathcal{G}$ (synchronized on the atomic propositions $\widehat{\mathsf{AP}}$), we get a new two-player game whose size is $O\left(2^{2^{|\mathcal{G}|}}\right)$ in which the winning condition for the Constructor can be expressed as:

$$(\mathsf{Energy} \cup \mathsf{Parity}_1) \cap \mathsf{Parity}_2 \ . \tag{4.10}$$

Remark 4.21. We highlight that in the above game, the sets Parity_1 and Parity_2 are induced by priority functions prty_1 and prty_2 from the automata $\mathcal{A}_1$ and $\mathcal{A}_2$. A crucial property of prty_1 and prty_2 is that their size is polynomial in the size of formulas of Equations (4.8) and (4.9).

We shall call a game where the winning condition is given by Equation 4.10 an ***EPP*** game.

EPP games

We design an algorithm for computing a winning strategy for player 1 in an EPP game when it exists.

Formally we are given a two-player arena $\mathcal{G} = (\mathsf{S}, (\mathsf{S}_1 \uplus \mathsf{S}_2), s_{\mathsf{ini}}, \mathsf{E})$ equipped with two priority functions prty_1 and prty_2 and a cost function cst. These functions induce three objectives respectively Parity_1, Parity_2, and Energy. As explained earlier, we aim at solving games where the objective is given by the set described in Equation (4.10).

Before designing a solution to these games, we establish a technical lemma. This lemma is instrumental. It describes a special property of winning strategies in EPP games. Later, we use this property to reduce any EPP game to a game where the winning condition is given by the set $\mathsf{Energy} \cup \mathsf{Parity}$ for some cost function and some priority function. The solution follows from the fact that winning strategy from the initial state in the new game for the objective $\mathsf{Energy} \cap \mathsf{Parity}$ witnesses the existence of a winning strategy in the original game. $\mathsf{Energy} \cap \mathsf{Parity}$ were originally introduced in [CD12b], under the name energy parity games, here they are called differently in order to remain consistent with our notation. We also highlight the fact that problem solved in [CD12b] is the initial credit problem that it, computing the least possible energy level in the initial state such that a winning strategy exists for player 1. Note that in this paper the initial credit is always 0, but this does not change the complexity results obtained in [CD12b].

Lemma 4.22. *Let $\mathcal{G}$ be a two-player arena, σ_1 be a strategy for player 1, and let $X = W(\mathsf{Parity}_1 \cap \mathsf{Parity}_2)$. Assume that σ_1 is winning for the objective $(\mathsf{Energy}(\mathcal{G}) \cup \mathsf{Parity}_1) \cap \mathsf{Parity}_2$. Then for each path $\pi \in \mathsf{out}\,(\sigma_1)$, the following holds:*

$$\pi \in \mathsf{out}\,(\sigma_1) \setminus S^* X S^\omega \implies \pi \in \mathsf{Energy} \cap \mathsf{Parity}_2 \enspace .$$

We now present a construction that reduces EPP games to $\mathsf{Energy} \cap \mathsf{Parity}$ games.

Let $\mathcal{G} = (\mathsf{S}, (\mathsf{S}_1 \uplus \mathsf{S}_2), s_{\mathsf{ini}}, \mathsf{E})$ be a two-player zero-sum game where the objective is given by Equation (4.10) above. We then build a fresh two-player game $\hat{\mathcal{G}} = (\mathsf{S}, \mathsf{S}_1 \uplus \mathsf{S}_2, s_{\mathsf{ini}}, \hat{\mathsf{E}})$. The construction builds the arena schematically depicted in Figure 4.5. Basically, the set of state is preserved, the edges in $\mathsf{S} \setminus X$ are also preserved but edges in X^2 are erased and all replaced by self loop with cost 0.

Before establishing the formal details of our construction, consider Figure 4.5 to build some intuition.

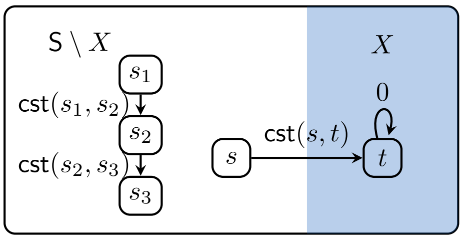


Figure 4.5: Illustration of the game $\hat{\mathcal{G}}$ arena construction. X is as in Lemma 4.22.

The game $\hat{\mathcal{G}}$ is formally designed as follows:

- The sets of states is S, S_1 and S_2 are unchanged and the initial state are the same.
- The set of edges $\hat{\mathsf{E}}$ is given by $\hat{\mathsf{E}}_1 \cup \hat{\mathsf{E}}_2$ where:

$$\begin{aligned} \hat{\mathsf{E}}_1 &= \{(s_1, s_2) \in \mathsf{E} \mid s_1 \notin W(\mathsf{Parity}_1 \cap \mathsf{Parity}_2)\} \enspace , \\ \hat{\mathsf{E}}_2 &= \{(s, s) \mid s \in W(\mathsf{Parity}_1 \cap \mathsf{Parity}_2)\} \enspace . \end{aligned}$$

- The new priority function over $\hat{\mathsf{S}}$ is obtained as follows:

$$\widehat{\mathsf{prty}}(s) = \begin{cases} 0 & \text{if } s \in W(\mathsf{Parity}_1 \cap \mathsf{Parity}_2) \enspace , \\ \mathsf{prty}(s) & \text{otherwise.} \end{cases}$$

- The new cost function over $\hat{\mathsf{E}}$ is defined as follows:

$$\widehat{\mathsf{cst}}(s_1, s_2) = \begin{cases} 0 & \text{when } (s_1, s_2) \in \hat{\mathsf{E}}_2 \\ \mathsf{cst}(s_1, s_2) & \text{otherwise.} \end{cases}$$

We will use the above construction to solve EPP games. We first state a couple of facts (X as in Lemma 4.22):

Fact 1. For any pair of states (s,t) in $(\mathsf{S} \setminus X)^2$ the priority of s and the cost of (s,t) are similar in $\mathcal{G}$ and $\hat{\mathcal{G}}$.

Fact 2. Histories and plays in $\mathcal{G}$ that never visit X are preserved in $\hat{\mathcal{G}}$. Moreover, if such a play in $\mathcal{G}$ is in $\mathsf{Energy} \cap \mathsf{Parity}_2$ then it is in $\mathsf{Energy} \cap \mathsf{Parity}$ induced by $\widehat{\mathsf{cst}}$ and $\widehat{\mathsf{prty}}$.

We want to show that the above construction preserves winning strategies in the original game $\mathcal{G}$. The preservation of strategies is formalized in the following sense:

Proposition 4.23. *Let $\mathcal{G}$ be an EPP game, then the following assertions are equivalent:*

a. Player 1 wins $\mathcal{G}$ (from the initial state),

b. Player 1 wins $\hat{\mathcal{G}}$ (from the initial state) for the objective $\mathsf{Energy} \cap \mathsf{Parity}$ induced by $\widehat{\mathsf{cst}}$ and $\widehat{\mathsf{prty}}$.

From the previous proposition and classical results, we also obtain a complexity upper-bound for solving EPP games.

Corollary 4.24. *The problem of deciding the existence of a winning strategy in an EPP game is in $\mathsf{NP} \cap \mathsf{co\text{-}NP}$.*

The Complexity of computing a resource-aware controller

With the previous results, we can finally establish the complexity of the problem of non-cooperative rational synthesis when the objectives of the players are induced by an LTL specification. Then, we extend it to the case where the system has a "rich specification", that is, in addition to having the objective of maintaining the resource non-negative, the controller must also ensure a qualitative objective.

Theorem 4.25. *The non-cooperative feasible rational synthesis problem is $\mathsf{2EXPTIME}$-complete.*

The hardness easily follows from classical LTL synthesis problem [GTW02]. The membership is as follows. From Proposition 4.20 computing a solution for the non-cooperative feasible rational synthesis amounts to solving a negotiation game where the winning condition is given by Equation 4.7. But solving a game with this objective according entails solving a game where the objective is given by Equation 4.10 (which is an EPP game). However the built arena is double exponential in the size of the original game but has priority functions of polynomial size. Thanks to Proposition 4.23 we can invoke, the algorithm from [CD12b] where they solve games where the objective is $\mathsf{Energy} \cap \mathsf{Parity}$. Finally, notice that the algorithm in [CD12b] runs in time polynomial in the size of the input arena and exponential in the size of the priority function, thus using Remark 4.21 commenting the bounds of the negotiation game entails the upper bound.

The Non-Cooperative Feasible Rational Synthesis with Rich Specifications

Now we consider a multi-player arena $\mathcal{G} = \langle \mathsf{S}, (\mathsf{S}_1 \uplus \ldots \uplus \mathsf{S}_n), s_{\mathsf{ini}}, \mathsf{P}, \mathsf{E} \rangle$ with objectives $\mathsf{Obj}_1, \ldots, \mathsf{Obj}_n$ induced by LTL formulas and a cost function cst. We aim at designing a strategy σ_1 for player 1 such that against any strategies for players $2, \ldots, n$, the profile $\overline{\sigma} = \langle \sigma_1, \ldots, \sigma_n \rangle$ satisfies

$$\mathsf{out}\,(\overline{\sigma}) \in \mathsf{f\text{-}NE}(\mathcal{G}) \implies \mathsf{out}\,(\overline{\sigma}) \in \mathsf{Energy} \cap \mathsf{Obj}_1 \ . \tag{4.11}$$

We shall call this problem the non-cooperative feasible rational synthesis with rich objectives.

Theorem 4.26. *The non-cooperative feasible rational synthesis with rich specifications is* 2EXPTIME*-complete.*

Again the hardness follows from the classical LTL synthesis problem. To obtain the upper bound, one can build a negotiation arena for the input, and solve the negotiation with the following objective

$$\left((\widehat{\mathsf{Energy}} \cap \widehat{\mathsf{Obj}_1}) \cup \widehat{\mathsf{Obj}_\mathsf{D}}\right) \cap \widehat{\mathsf{Obj}_\mathsf{W}} \ , \tag{4.12}$$

where

$$\widehat{\mathsf{Obj}_1} = \{\pi \in \widehat{\mathsf{S}}^\omega \mid \mathsf{proj}_1(\pi) \models \mathsf{Obj}_1\} \ .$$

We argue that his new objective can be encoded as a EPP game. Indeed, notice that Equation (4.12) is equivalent to

$$\left(\widehat{\mathsf{Energy}} \cap \underbrace{\widehat{\mathsf{Obj}_1} \cap \widehat{\mathsf{Obj}_\mathsf{W}}}_{A}\right) \cup \left(\underbrace{\widehat{\mathsf{Obj}_\mathsf{D}} \cap \widehat{\mathsf{Obj}_\mathsf{W}}}_{B}\right)$$

finally we obtain the following set:

$$\left(\widehat{\mathsf{Energy}} \cup A\right) \cap C \ ,$$

where

$$A = \widehat{\mathsf{Obj}_1} \cap \widehat{\mathsf{Obj}_\mathsf{W}} \ , \qquad B = \widehat{\mathsf{Obj}_\mathsf{D}} \cap \widehat{\mathsf{Obj}_\mathsf{W}} \ , \qquad C = A \cup B \ .$$

We conclude by applying the result from Proposition 4.24 and noticing that A, B, and C can be written as LTL formulas whose size is polynomial in the size of the input game.

4.8 Discussion

Wrap-up

In this chapter we presented a series of results regarding the synthesis in the context of multi-player games. We extend the notion of synthesis problem where one is interested in designing a control policy to the case where one suggests a policy for multiple entities, i.e. the rational synthesis. In general one relies on the fact that this policy is compliant with some sort of rationality to incentivize the agents' cooperation. We introduced a quantitative notion of the rational synthesis where the players share a common resource, that we choose to call energy, beside their own qualitative objective.

Following the qualitative setting, we extended the cooperative and non-cooperative version of rational synthesis problem.

The cooperative rational synthesis was first studied in work by Ummels [Umm08] where he considered the following setting. Each agent is assigned an individual specification together with a list of the specification that "must-hold". In this case, one aims at constructing a global controller that ensures a rational behavior for the global system such that its outcome ensures the "must-hold" specifications. Later, Fisman et. al. [FKL10] gave a logical characterization of this problem, stating it in terms of model checking with Strategy Logic. This is from where we borrow the name cooperative rational synthesis, as it assumes that the different part of the system will agree on a global behavior as long as some rationality is guaranteed, viz., no agent has an incentive to unilaterally defect from it. Finally Condurache et. al. [CFGR16a] presented a complete picture of complexity bounds for a variety of ω-regular specifications.

In a different line of work Bouyer et. al. [UW11] studied the problem of existence of a Nash equilibrium in systems with qualitative specifications but the obtained results were of negative nature. The decidability was hard to obtain and quite strict restriction on the behavior of the agents had to be made in order to handle this case.

While mixing temporal and quantitative specifications is a rather natural idea, it has been mostly studied in the setting of zero-sum games [CD12b, CHJ05, CRR12, CDGO14, MSTW17]. The model checking of resource-bounded logics of strategies has also been rather extensively investigated [Ves15, NALR18, ABDL18, ADL20].

Two important extensions of LTL were useful in the results presented in this chapter to obtain optimal algorithms for parity objectives. They played the role that plays plain LTL in [CFGR16a]. Energy LTL [BBFR13] was used for the careless case, and Constrained LTL [DG09] was used for the careful case. In fact Energy LTL is a particular case of Constrained LTL. We used both to highlight the core differences between the careless and the careful cases. Energy LTL allowed us to verify that an LTL property holds over a run whose energy does not drop below zero, which was just enough for the careless case. In the careful case, one also needed to check as much, but needed to check that some states are not traversed with a level of energy too high to allow a profitable deviation from the agent controlling it. The extended language of Constrained LTL, allowing the comparison of the level of energy with any constant is providing the necessary formal machinery. Keeping the energy not only above zero, but below certain bounds in some states is thus crucial for the careful case. This was reflected by our use of bounded one-counter automata [FJ15] to obtain optimal lower-bounds for our problems.

We also investigated the problem of non-cooperative rational synthesis in systems with one common-pool resource, where the specification of the system is a qualitative objective to be realized in a run that never depletes the common-pool resource (the controller is thus careful), and the system's components (apart from the controller) are careless. We do not investigate the case of careful players in our case. The carefulness of a player, as far as we were concerned, defies the concept of non cooperation as players (with the exception of the controller) are putatively uncontrollable, and requiring carefulness seemed therefore contrived.

Perspectives

We first hint at some results that can be derived in the cooperative setting. In the careful case, the proof of Proposition 4.17 already establishes that the problem is PSPACE-hard for reachability objectives, and it can be straightforwardly extended to show that the problem is PSPACE-hard for co-Büchi objectives.

With LTL objectives, a corollary of our results hints at a 2EXPTIME membership for both settings. This should be obtained by relying on the same techniques used in the original LTL synthesis problem [PR89].

Another direction that one could look at is to introduce a hybrid setting where a subset of players is careless and the other is careful. Our results hint at the fact that it should not be more difficult than the careful case since the logic used in that case subsumes energy LTL, and thus a formula to describe this new setting for the players could be easily described.

Regarding the non-cooperative setting, we first plan to study the more general case of having multiple resources in the system, as in [CDJ$^+$22] where we developed an approach for multiple resources in the cooperative case. Another research direction is investigating the interesting and more tractable fragments of LTL such as the $GR(1)$ fragment [BJP$^+$12, GNPW22].

Conclusion

In this manuscript, we presented some results that fall in the context of controller synthesis. In particular, we focused on the problem of implementability. We studied this problem for different models and with respect to different aspects.

In Chapter 1 we tackled the problem of implementability through memory, i.e., we presented a series of results that help understand the memory requirements of optimal controllers. In Chapter 2, we shifted the point of view and focused more on the behavior of an optimal controller, i.e., we aimed at designing optimal controllers with robust behavior with respect to long term specifications. In Chapter 3, studied real timed systems. In this formalism, the main challenge is to design controllers that do not require infinite precision. Finally in Chapter 4, we studied multi-player games and developed rationality notions taking into account the constraints imposed by a common resource. Some perspectives have already been proposed in the end of each chapter. We describe now additional possible developments, more ambitious and long term, that have not yet been investigated.

Imprecise abstractions A challenge that one usually faces when modeling complex systems is the size of the state space. To the best of our knowledge, state of the art algorithms for the controller synthesis assume that the full system is modeled, even though in the case of imperfect information formalism, the entire system is modeled and only the controller has a partial knowledge of current state of the model. When the formalism is decidable, almost-all the techniques rely on abstraction-based techniques. For these abstractions to perform correctly, one has to prove that they are faithful to the behavior of the initial system. This way one can transform a correct controller of the abstracted system into a correct controller in the concrete system. Here we propose to design imprecise abstractions displaying a close enough behavior with respect to the concrete system. However, one has to still ensure that a correct controller for the abstraction is going to be good enough for the concrete system (once transformed). We plan to introduce distance-based techniques to formalize the notion of close enough, and continuity notion to formalize the notion of good enough.

Timed bisimulation in hybrid systems In this manuscript we presented techniques for handling robustness issues for timed games. In this direction, we aim at extending these techniques to hybrid systems. Unfortunately, hybrid systems suffer from poor tractability properties. Our goal in this part is to design simulation-based technique with the aim of identifying tractable classes of hybrid games. Intuitively, we want to identify classes of hybrid games that can be simulated by timed automata (and vice-versa). We believe that through such simulations, one can transfer the techniques developed for timed games to hybrid games. Preliminary work is progress for this part in the PhD of Mariem Hammami jointly supervised by Catalin Dima, Régine Laleau, and myself.

A second goal, is to develop distance-based techniques to tackle an even larger class of hybrid systems. For instance, in the case where one cannot find a bisimulation between the hybrid system and a timed system. We aim at developing notion of abstraction that approximates the behavior of the initial system. The notion distance, should be used here to bring guarantees on the quality of the abstraction.

Bibliography

[ABDL18] N. Alechina, N. Bulling, S. Demri, and B. Logan. On the complexity of resource-bounded logics. Theoretical Computer Science, 750:69 – 100, 2018. Reachability Problems: Special Issue.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. Theoretical Computer Science, 126(2):183–235, 1994.

[ADD99] Robert B. Ash and Catherine A. Doleans-Dade. Probability and Measure Theory. Harcourt/Academic Press, second edition, 1999.

[ADL20] Natasha Alechina, Stéphane Demri, and Brian Logan. Parameterised resource-bounded ATL. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 7040–7046. AAAI Press, 2020.

[AGH$^+$21] Alessandro Abate, Julian Gutierrez, Lewis Hammond, Paul Harrenstein, Marta Kwiatkowska, Muhammad Najib, Giuseppe Perelli, Thomas Steeples, and Michael J. Wooldridge. Rational verification: game-theoretic verification of multi-agent systems. Applied Intelligence, 51(9):6569–6584, 2021.

[AR17] Benjamin Aminof and Sasha Rubin. First-cycle games. Inf. Comput., 254:195–216, 2017.

[BA11] Nicolas Basset and Eugene Asarin. Thin and thick timed regular languages. In Uli Fahrenberg and Stavros Tripakis, editors, Formal Modeling and Analysis of Timed Systems, volume 6919 of Lecture Notes in Computer Science, pages 113–128. Springer, 2011.

[Bai15] Christel Baier. Reasoning about cost-utility constraints in probabilistic models. In Mikolaj Bojanczyk, Slawomir Lasota, and Igor Potapov, editors, Reachability Problems - 9th International Workshop, RP 2015, Warsaw, Poland, September 21-23, 2015, Proceedings, volume 9328 of Lecture Notes in Computer Science, pages 1–6. Springer, 2015.

[BBB$^+$08a] Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus Größer. Almost-sure model checking of infinite paths in one-clock timed automata. In Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science (LICS'08), pages 217–226, Pittsburgh, Pennsylvania, USA, June 2008. IEEE Computer Society Press.

[BBB$^+$08b] Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus Größer. A probabilistic semantics for timed automata. Research Report LSV-08-13, Laboratoire Spécification et Vérification, ENS Cachan, France, April 2008. 42 pages.

[BBFR13] Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, and Jean-François Raskin. Synthesis from LTL specifications with mean-payoff objectives. In Nir Piterman and Scott A. Smolka, editors, Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings, volume 7795 of Lecture Notes in Computer Science, pages 169–184. Springer, 2013.

[BBGDO24] Benoît Barbot, Damien Busatto-Gaston, Catalin Dima, and Youssouf Oualhadj. Controller synthesis in timed büchi automata: Robustness and punctual guards, 2024.

[BBMU15] Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Pure nash equilibria in concurrent deterministic games. Logical Methods in Computer Science, 11(2), 2015.

[BCHK14] Udi Boker, Krishnendu Chatterjee, Thomas A. Henzinger, and Orna Kupferman. Temporal specifications with accumulative values. ACM Trans. Comput. Log., 15(4):27:1–27:25, 2014.

[BDOR19] Thomas Brihaye, Florent Delgrange, Youssouf Oualhadj, and Mickael Randour. Life is random, time is not: Markov decision processes with window objectives. In CONCUR, volume 140 of LIPIcs, pages 8:1–8:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[BDOR20] Thomas Brihaye, Florent Delgrange, Youssouf Oualhadj, and Mickael Randour. Life is random, time is not: Markov decision processes with window objectives. Log. Methods Comput. Sci., 16(4), 2020.

[Bea03] Danièle Beauquier. On probabilistic timed automata. Theor. Comput. Sci., 292(1):65–84, January 2003.

[BFKN16] Tomás Brázdil, Vojtech Forejt, Antonín Kucera, and Petr Novotný. Stability in graphs and games. In Desharnais and Jagadeesan [DJ16], pages 10:1–10:14.

[BGMR18] Patricia Bouyer, Mauricio González, Nicolas Markey, and Mickael Randour. Multi-weighted Markov decision processes with reachability objectives. In Andrea Orlandini and Martin Zimmermann, editors, Proceedings Ninth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2018, Saarbrücken, Germany, 26-28th September 2018., volume 277 of EPTCS, pages 250–264, 2018.

[BHR16a] Véronique Bruyère, Quentin Hautem, and Mickael Randour. Window parity games: an alternative approach toward parity games with time bounds. In Domenico Cantone and Giorgio Delzanno, editors, Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016, volume 226 of EPTCS, pages 135–148, 2016.

[BHR16b] Véronique Bruyère, Quentin Hautem, and Jean-François Raskin. On the complexity of heterogeneous multidimensional games. In Desharnais and Jagadeesan [DJ16], pages 11:1–11:15.

[BJP$^+$12] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. Journal of Computer and System Sciences, 78(3):911–938, 2012.

[BK08] Christel Baier and Joost-Pieter Katoen. Principles of model checking. MIT Press, 2008.

[BKKW14] Christel Baier, Joachim Klein, Sascha Klüppelholz, and Sascha Wunderlich. Weight monitoring with linear temporal logic: complexity and decidability. In Thomas A. Henzinger and Dale Miller, editors, Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014, pages 11:1–11:10. ACM, 2014.

[BMR08] Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust analysis of timed automata via channel machines. In Roberto Amadio, editor, Proceedings of the 11th International Conference on Foundations of Software Science and Computation Structure (FoSSaCS'08), volume 4962 of Lecture Notes in Computer Science, pages 157–171. Springer, 2008.

[BMR14] Véronique Bruyère, Noémie Meunier, and Jean-François Raskin. Secure equilibria in weighted games. In Thomas A. Henzinger and Dale Miller, editors, Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014, pages 26:1–26:26. ACM, 2014.

[BMRS19] Damien Busatto-Gaston, Benjamin Monmege, Pierre-Alain Reynier, and Ocan Sankur. Robust controller synthesis in timed büchi automata: A symbolic approach. In Isil Dillig and Serdar Tasiran, editors, Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I, volume 11561 of Lecture Notes in Computer Science, pages 572–590. Springer, 2019.

[BMS12] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust reachability in timed automata: A game-based approach. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP'12) – Part II, volume 7392 of Lecture Notes in Computer Science, pages 128–140. Springer, 2012.

[BMS13] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robustness in timed automata. In Parosh Aziz Abdulla and Igor Potapov, editors, Proceedings of the 7th Workshop on Reachability Problems in Computational Models (RP'13), volume 8169 of Lecture Notes in Computer Science, pages 1–18. Springer, 2013.

[BORV21] Patricia Bouyer, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Arena-independent finite-memory determinacy in stochastic games. In Serge Haddad and Daniele Varacca, editors, 32nd International Conference on Concurrency

Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference, volume 203 of LIPIcs, pages 26:1–26:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[Bra79] Gilles Brassard. A note on the complexity of cryptography (corresp.). IEEE Trans. Information Theory, 25(2):232–233, 1979.

[BRO$^+$20] Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. In Igor Konnov and Laura Kovács, editors, 31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference), volume 171 of LIPIcs, pages 24:1–24:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[BRO$^+$22] Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. Log. Methods Comput. Sci., 18(1), 2022.

[CD12a] Krishnendu Chatterjee and Laurent Doyen. Energy parity games. Theor. Comput. Sci., 458:49–60, 2012.

[CD12b] Krishnendu Chatterjee and Laurent Doyen. Energy parity games. Theoretical Computer Science, 458:49–60, 2012.

[CDFR14] Krishnendu Chatterjee, Laurent Doyen, Emmanuel Filiot, and Jean-François Raskin. Doomsday equilibria for omega-regular games. In Kenneth L. McMillan and Xavier Rival, editors, Verification, Model Checking, and Abstract Interpretation - 15th International Conference, VMCAI 2014, San Diego, CA, USA, January 19-21, 2014, Proceedings, volume 8318 of Lecture Notes in Computer Science, pages 78–97. Springer, 2014.

[CDGO14] Krishnendu Chatterjee, Laurent Doyen, Hugo Gimbert, and Youssouf Oualhadj. Perfect-information stochastic mean-payoff parity games. In FoSSaCS, volume 8412 of Lecture Notes in Computer Science, pages 210–225. Springer, 2014.

[CDJ$^+$22] Rodica Condurache, Catalin Dima, Madalina Jitaru, Youssouf Oualhadj, and Nicolas Troquard. Careful autonomous agents in environments with multiple common resources. In Proceedings of the Second Workshop on Agents and Robots for reliable Engineered Autonomy, volume 362 of EPTCS, pages 3–14, 2022.

[CDOT21] Rodica Condurache, Catalin Dima, Youssouf Oualhadj, and Nicolas Troquard. Rational synthesis in the commons with careless and careful agents. In Frank Dignum, Alessio Lomuscio, Ulle Endriss, and Ann Nowé, editors, AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021, pages 368–376. ACM, 2021.

[CDOT23] Rodica Condurache, Catalin Dima, Youssouf Oualhadj, and Nicolas Troquard. Synthesis of resource-aware controllers against rational agents. In Noa Agmon, Bo An, Alessandro Ricci, and William Yeoh, editors, Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023, pages 775–783. ACM, 2023.

[CDRR15] Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin. Looking at mean-payoff and total-payoff through windows. Inf. Comput., 242:25–52, 2015.

[CFGR16a] Rodica Condurache, Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. The complexity of rational synthesis. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy, volume 55 of LIPIcs, pages 121:1–121:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[CFGR16b] Rodica Condurache, Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. The complexity of rational synthesis. In 43rd International Colloquium on Automata, Languages, and Programming, ICALP'16, volume 55 of LIPIcs, pages 121:1–121:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[CH14] Krishnendu Chatterjee and Monika Henzinger. Efficient and dynamic algorithms for alternating Büchi games and maximal end-component decomposition. J. ACM, 61(3):15:1–15:40, 2014.

[CHJ04] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdzinski. Games with secure equilibria. In 19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings, pages 160–169. IEEE Computer Society, 2004.

[CHJ05] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdzinski. Mean-payoff parity games. In 20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings, pages 178–187. IEEE Computer Society, 2005.

[CHP07] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Generalized parity games. In Helmut Seidl, editor, Foundations of Software Science and Computational Structures, 10th International Conference, FOSSACS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007, Braga, Portugal, March 24-April 1, 2007, Proceedings, volume 4423 of Lecture Notes in Computer Science, pages 153–167. Springer, 2007.

[CHP11] Krishnendu Chatterjee, Thomas A. Henzinger, and Vinayak S. Prabhu. Timed parity games: Complexity and robustness. Logical Methods in Computer Science, 7(4), 2011.

[CJH04] Krishnendu Chatterjee, Marcin Jurdzinski, and Thomas A. Henzinger. Quantitative stochastic parity games. In J. Ian Munro, editor, Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004, pages 121–130. SIAM, 2004.

[COT18] Rodica Condurache, Youssouf Oualhadj, and Nicolas Troquard. The complexity of rational synthesis for concurrent games. In 29th International Conference on Concurrency Theory, CONCUR'18, volume 118 of LIPIcs, pages 38:1–38:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[CRR12] Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. In Maciej Koutny and Irek Ulidowski, editors, CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings, volume 7454 of Lecture Notes in Computer Science, pages 115–131. Springer, 2012.

[CRR14] Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. Acta Inf., 51(3-4):129–163, 2014.

[DDMR08] Martin De Wulf, Laurent Doyen, Nicolas Markey, and Jean-François Raskin. Robust safety of timed automata. Formal Methods in System Design, 33(1-3):45–84, 2008.

[DG09] Stéphane Demri and Régis Gascon. The effects of bounding syntactic resources on presburger LTL. J. Log. Comput., 19(6):1541–1575, 2009.

[DGG23] Laurent Doyen, Pranshu Gaba, and Shibashis Guha. Stochastic window mean-payoff games. CoRR, abs/2304.11563, 2023.

[DJ16] Josée Desharnais and Radha Jagadeesan, editors. 27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada, volume 59 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[FHKM15] Nathanaël Fijalkow, Florian Horn, Denis Kuperberg, and Michal Skrzypczak. Trading bounds for memory in games with counters. In Halldórsson et al. [HIKS15], pages 197–208.

[FJ15] John Fearnley and Marcin Jurdzinski. Reachability in two-clock timed automata is pspace-complete. Inf. Comput., 243:26–36, 2015.

[FKL10] Dana Fisman, Orna Kupferman, and Yoad Lustig. Rational synthesis. In 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'10, pages 190–204, 2010.

[FKNT10] Vojtęch Forejt, Marta Kwiatkowska, Gethin Norman, and Ashutosh Trivedi. Expected reachability-time games. In Krishnendu Chatterjee and ThomasA. Henzinger, editors, Formal Modeling and Analysis of Timed Systems, volume 6246 of Lecture Notes in Computer Science, pages 122–136. Springer Berlin Heidelberg, 2010.

[Gim07] Hugo Gimbert. Pure stationary optimal strategies in Markov decision processes. In Wolfgang Thomas and Pascal Weil, editors, STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science, Aachen, Germany, February 22-24, 2007, Proceedings, volume 4393 of Lecture Notes in Computer Science, pages 200–211. Springer, 2007.

[GK14] Hugo Gimbert and Edon Kelmendi. Two-player perfect-information shift-invariant submixing stochastic games are half-positional. CoRR, abs/1401.6575, 2014.

[GNPW22] Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, and Michael Wooldridge. On the complexity of rational verification. Annals of Mathematics and Artificial Intelligence, 2022.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. Automata, Logics, and Infinite Games: A Guide to Current Research, volume 2500 of Lecture Notes in Computer Science. Springer, 2002.

[GZ05] Hugo Gimbert and Wieslaw Zielonka. Games where you can play optimally without any memory. In Martín Abadi and Luca de Alfaro, editors, CONCUR 2005

- Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings, volume 3653 of Lecture Notes in Computer Science, pages 428–442. Springer, 2005.

[HIKS15] Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors. Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II, volume 9135 of Lecture Notes in Computer Science. Springer, 2015.

[HJKQ18] Arnd Hartmanns, Sebastian Junges, Joost-Pieter Katoen, and Tim Quatmann. Multi-cost bounded reachability in MDP. In Dirk Beyer and Marieke Huisman, editors, Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II, volume 10806 of Lecture Notes in Computer Science, pages 320–339. Springer, 2018.

[HK15] Christoph Haase and Stefan Kiefer. The odds of staying on budget. In Halldórsson et al. [HIKS15], pages 234–246.

[HPR18] Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin. Looking at mean payoff through foggy windows. Acta Inf., 55(8):627–647, 2018.

[HS06] Thomas A. Henzinger and Joseph Sifakis. The embedded systems design challenge. In Formal Methods, 14th International Symposium on Formal Methods, volume 4085 of Lecture Notes in Computer Science, pages 1–15, Hamilton, Canada, 2006. Springer.

[Jen96] Henrik E Jensen. Model checking probabilistic real time systems. In Proc. 7th Nordic Workshop on Programming Theory, pages 247–261. Citeseer, 1996.

[JLS15a] Marcin Jurdzinski, Ranko Lazic, and Sylvain Schmitz. Fixed-dimensional energy games are in pseudo-polynomial time. In Halldórsson et al. [HIKS15], pages 260–272.

[JLS15b] Marcin Jurdzinski, Ranko Lazic, and Sylvain Schmitz. Fixed-dimensional energy games are in pseudo-polynomial time. In Halldórsson et al. [HIKS15], pages 260–272.

[KNSS02] Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. Theor. Comput. Sci., 282(1):101–150, June 2002.

[Kop06] Eryk Kopczyński. Half-positional determinacy of infinite games. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II, volume 4052 of Lecture Notes in Computer Science, pages 336–347. Springer, 2006.

[Koz22] Alexander Kozachinskiy. One-to-two-player lifting for mildly growing memory. In Petra Berenbrink and Benjamin Monmege, editors, 39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference), volume 219 of LIPIcs, pages 43:1–43:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[KPV09] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From liveness to promptness. Formal Methods Syst. Des., 34(2):83–103, 2009.

[KPV14] Orna Kupferman, Giuseppe Perelli, and Moshe Y. Vardi. Synthesis with rational environments. In Nils Bulling, editor, Multi-Agent Systems - 12th European Conference, EUMAS 2014, Prague, Czech Republic, December 18-19, 2014, Revised Selected Papers, volume 8953 of Lecture Notes in Computer Science, pages 219–235. Springer, 2014.

[KS22] Orna Kupferman and Noam Shenwald. The complexity of ltl rational synthesis. In 28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'22, pages 25–45, Cham, 2022. Springer International Publishing.

[MCB$^+$17] Emilia Melville, Ian Christie, Kate Burningham, Celia Way, and Phil Hampshire. The electric commons: A qualitative study of community accountability. Energy Policy, 106:12–21, 2017.

[MSTW17] Richard Mayr, Sven Schewe, Patrick Totzke, and Dominik Wojtczak. Mdps with energy-parity objectives. In LICS, pages 1–12. IEEE Computer Society, 2017.

[NALR18] Hoang Nga Nguyen, Natasha Alechina, Brian Logan, and Abdur Rakib. Alternating-time temporal logic with resource bounds. J. Log. Comput., 28(4):631–663, 2018.

[ORS14] Youssouf Oualhadj, Pierre-Alain Reynier, and Ocan Sankur. Probabilistic robust timed games. In CONCUR, volume 8704 of Lecture Notes in Computer Science, pages 203–217. Springer, 2014.

[Ost90] Elinor Ostrom. Governing the Commons: The evolution of institutions for collective action. Cambridge University Press, 1990.

[PR89] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In 16th Annual ACM Symposium on Principles of Programming Languages, POPL'1989, pages 179–190, 1989.

[Pur00] Anuj Puri. Dynamical properties of timed automata. Discrete Event Dynamic Systems, 10(1-2):87–113, 2000.

[Put14] Martin L Puterman. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

[Rou13] Stéphane Le Roux. Infinite sequential Nash equilibrium. Logical Methods in Computer Science, 9(2), 2013.

[RPR18] Stéphane Le Roux, Arno Pauly, and Mickael Randour. Extending finite-memory determinacy by Boolean combination of winning conditions. In Sumit Ganguly and Paritosh K. Pandya, editors, 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India, volume 122 of LIPIcs, pages 38:1–38:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[RRS17] Mickael Randour, Jean-François Raskin, and Ocan Sankur. Percentile queries in multi-dimensional Markov decision processes. Formal Methods in System Design, 50(2-3):207–248, 2017.

[SBMR13] Ocan Sankur, Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust controller synthesis in timed automata. In Pedro R. D'Argenio and Hernán) Melgratti, editors, Proceedings of the 24th International Conference on Concurrency Theory (CONCUR'13), volume 8052 of Lecture Notes in Computer Science, pages 546–560. Springer, 2013.

[SVV07] Matthias Schmalz, Hagen Völzer, and Daniele Varacca. Model checking almost all paths can be less expensive than checking all paths. In FSTTCS, volume 4855 of Lecture Notes in Computer Science, pages 532–543. Springer, 2007.

[Umm08] Michael Ummels. The complexity of nash equilibria in infinite multiplayer games. In Roberto Amadio, editor, Foundations of Software Science and Computational Structures, pages 20–34, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[UW11] Michael Ummels and Dominik Wojtczak. The complexity of nash equilibria in limit-average games. In Joost-Pieter Katoen and Barbara König, editors, CONCUR 2011 - Concurrency Theory - 22nd International Conference, CONCUR 2011, Aachen, Germany, September 6-9, 2011. Proceedings, volume 6901 of Lecture Notes in Computer Science, pages 482–496. Springer, 2011.

[VCD$^+$15a] Yaron Velner, Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, Alexander Moshe Rabinovich, and Jean-François Raskin. The complexity of multi-mean-payoff and multi-energy games. Inf. Comput., 241:177–196, 2015.

[VCD$^+$15b] Yaron Velner, Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, Alexander Moshe Rabinovich, and Jean-François Raskin. The complexity of multi-mean-payoff and multi-energy games. Inf. Comput., 241:177–196, 2015.

[Ves15] Steen Vester. On the complexity of model-checking branching and alternating-time temporal logics in one-counter systems. In Bernd Finkbeiner, Geguang Pu, and Lijun Zhang, editors, Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings, volume 9364 of Lecture Notes in Computer Science, pages 361–377. Springer, 2015.

[Zie98] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theor. Comput. Sci., 200(1-2):135–183, 1998.