

Non-Zero Sum Games for Reactive Synthesis

Romain Brenguier, Lorenzo Clemente, Paul Hunter, Guillermo A. Pérez, Mickaël Randour, Jean-François Raskin, Ocan Sankur, Mathieu Sassolas

▶ **To cite this version:**

Romain Brenguier, Lorenzo Clemente, Paul Hunter, Guillermo A. Pérez, Mickaël Randour, et al.. Non-Zero Sum Games for Reactive Synthesis . LATA 2016 : 10th International Conference on Language and Automata Theory and Applications, Mar 2016, Prague, Czech Republic. pp.3-23, 10.1007/978-3-319-30000-9_1 . hal-01373546

HAL Id: hal-01373546
https://inria.hal.science/hal-01373546

Submitted on 17 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Non-Zero Sum Games for Reactive Synthesis*

Romain Brenguier[1], Lorenzo Clemente[2], Paul Hunter[3], Guillermo A. Pérez[3], Mickael Randour[3], Jean-François Raskin[3], Ocan Sankur[4], Mathieu Sassolas[5]

[1] University of Oxford, UK
[2] University of Warsaw, Poland
[3] Université Libre de Bruxelles, Belgium
[4] CNRS, Irisa, France
[5] Université Paris-Est – Créteil, LACL, France

Abstract. In this invited contribution, we summarize new solution concepts useful for the synthesis of reactive systems that we have introduced in several recent publications. These solution concepts are developed in the context of non-zero sum games played on graphs. They are part of the contributions obtained in the inVEST project funded by the European Research Council.

1 Introduction

Reactive systems are computer systems that maintain a continuous interaction with the environment in which they operate. They usually exhibit characteristics, like real-time constraints, concurrency, parallelism, etc., that make them difficult to develop correctly. Therefore, formal techniques using mathematical models have been advocated to help to their systematic design.

One well-studied formal technique is *model checking* [20,39,2] which compares a model of a system with its specification. The main objective of this technique is to find design errors early in the development cycle. So model-checking can be considered as a sophisticated *debugging* method. A scientifically more challenging goal, called *synthesis*, is to design algorithms that, given a specification for a reactive system and a model of its environment, directly synthesize a correct system, i.e., a system that enforces the specification *no matter how* the environment behaves.

Synthesis can take different forms: from computing optimal values of parameters to the full-blown automatic synthesis of finite-state machine descriptions for components of the reactive system. The main mathematical models proposed for the synthesis problem are based on *two-player zero-sum games played on graphs* and the main solution concept for those games is the notion of *winning strategy*. This model encompasses the situation where a *monolithic* controller has to be designed to interact with a *monolithic* environment that is supposed to be *fully*

* Work supported by the ERC starting grant inVEST (FP7-279499), G.A. Pérez is supported by F.R.S.-FNRS ASP fellowship, M. Randour is a F.R.S.-FNRS Postdoctoral Researcher.

antagonistic. In the sequel, we call the two players Eve and Adam, Eve plays the role of the system and Adam plays the role of the environment.

A fully antagonistic environment is most often a *bold abstraction* of reality: the environment usually has its own goal which, in general, does not correspond to that of falsifying the specification of the reactive system. Nevertheless, this abstraction is popular because it is simple and sound: a winning strategy against an antagonistic environment is winning against any environment that pursues its own objective. However this approach may fail to find a winning strategy even if solutions exist when the objective of the environment are taken into account, or it may produce sub-optimal solutions because they are overcautious and do not exploit the fact the the environment has its own objective. In several recent works, we have introduced new solution concepts for synthesis of reactive systems that take the objective of the environment into account or relax the fully adversarial assumption.

Assume admissible synthesis In [7], we proposed a novel notion of synthesis where the objective of the environment can be captured using the concept of *admissible* strategies [5,3,8]. For a player with objective ϕ, a strategy σ is *dominated* by σ' if σ' does as well as σ w.r.t. ϕ against all strategies of the other players, and better for some of those strategies. A strategy σ is *admissible* if it is *not* dominated by another strategy. We use this notion to derive a meaningful notion to *synthesize* systems with several players, with the following idea. Only admissible strategies should be played by *rational* players as dominated strategies are clearly *sub-optimal options*. In *assume-admissible synthesis*, we make the assumption that both players play admissible strategies. Then, when synthesizing a controller, we search for an admissible strategy that is *winning* against all admissible strategies of the environment. Assume admissible synthesis is *sound*: if both players choose strategies that are winning against admissible strategies of the other player, the objectives of both players will be satisfied.

Regret minimization: best-responses as yardstick In [32] we studied strategies for Eve which *minimize her regret*. The regret of a strategy σ of Eve corresponds to the difference between the value Eve achieves by playing σ against Adam and the value she could have ensured if she had known the strategy of Adam in advance. Regret is not a novel concept in game theory see, e.g., [30], but it was not explicitly used for games played on graphs before [28]. The complexity of deciding whether a regret-minimizing strategy for Eve exists, and the memory requirements for such strategies change depending on what type of behavior Adam can use. We have focused on three particular cases: arbitrary behaviors, positional behaviors, and time-dependent behaviors (otherwise known as *oblivious* environments). The latter class of regret games was shown in [32] to be related to the problem of determining whether an automaton has a certain form of determinism.

Games with an expected adversary In [12,11,21], we combined the classical formalism of two-player zero-sum games (where the environment is con-

sidered to be completely antagonistic) with *Markov decision processes* (MDPs), a well-known model for decision-making inside a stochastic environment. The motivation is that one has often a good idea of the *expected behavior* (i.e., average-case) of the environment represented as a stochastic model based on statistical data such as the frequency of requests for a computer server, the average traffic in a town, etc. In this case, it makes sense to look for strategies that will maximize the *expected performance* of the system. This is the traditional approach for MDPs, but it gives no guarantee at all if the environment deviates from its expected behavior, which can happen, for example, if events with small probability happen, or if the statistical data upon which probabilities are estimated is noisy or unreliable. On the other hand, two-player zero-sum games lead to strategies guaranteeing a *worst-case performance* no matter how the environment behaves — however such strategies may be far from optimal against the expected behavior of the environment. With our new framework of *beyond worst-case* synthesis, we provide formal grounds to synthesize strategies that *both* guarantee some minimal performance against any adversary *and* provide an higher expected performance against a given expected behavior of the environment — thus essentially combining the two traditional standpoints from games and MDPs.

Structure of the paper Section 2 recalls preliminaries about games played on graphs while Section 3 recalls the classical setting of zero-sum two player games. Section 4 summarizes our recent works on the use of the notion of admissibility for synthesis of reactive systems. Section 5 summarizes our recent results on regret minimization for reactive synthesis. Section 6 summarizes our recent contributions on the synthesis of strategies that ensure good expected performance together with guarantees against their worst-case behaviors.

2 Preliminaries

We consider two-player turn-based games played on finite (weighted) graphs. Such games are played on so-called weighted game arenas.

Definition 1 (Weighted Game Arena). *A (turn-based) two-player* weighted game arena *is a tuple* $\mathcal{A} = \langle S_\exists, S_\forall, \mathsf{E}, s_{\mathsf{init}}, \mathsf{w} \rangle$ *where:*

- $S_\exists$ *is the finite set of states owned by* Eve*,* $S_\forall$ *is the finite set of states owned by* Adam*,* $S_\exists \cap S_\forall = \emptyset$ *and we denote* $S_\exists \cup S_\forall$ *by* S*.*
- $\mathsf{E} \subseteq S \times S$ *is a set of edges, we say that* E *is* total *whenever for all states* $s \in S$*, there exists* $s' \in S$ *such that* $(s, s') \in \mathsf{E}$ *(we often assume this w.l.o.g.).*
- $s_{\mathsf{init}} \in S$ *is the initial state.*
- $\mathsf{w} : \mathsf{E} \to \mathbb{Z}$ *is the weight function that assigns an integer weight to each edge.*

We do not always use the weight function defined on the edges of the weighted game arena and in these cases we simply omit it.

Unless otherwise stated, we consider for the rest of the paper a fixed weighted game arena $\mathcal{A} = \langle S_\exists, S_\forall, \mathsf{E}, s_{\mathsf{init}}, \mathsf{w}\rangle$.

A *play* in the arena $\mathcal{A}$ is an *infinite* sequence of states $\pi = s_0 s_1 \dots s_n \dots$ such that for all $i \geq 0$, $(s_i, s_{i+1}) \in \mathsf{E}$. A play $\pi = s_0 s_1 \dots$ is *initial* when $s_0 = s_{\mathsf{init}}$. We denote by $\mathsf{Plays}(\mathcal{A})$ the set of plays in the arena $\mathcal{A}$, and by $\mathsf{InitPlays}(\mathcal{A})$ its subset of initial plays.

A *history* ρ is a finite sequence of states which is a *prefix* of a play in $\mathcal{A}$. We denote by $\mathsf{Pref}(\mathcal{A})$ the set of histories in $\mathcal{A}$, and the set of prefixes of initial plays is denoted by $\mathsf{InitPref}(\mathcal{A})$. Given an infinite sequence of states π, and two finite sequences of states ρ_1, ρ_2, we write $\rho_1 < \pi$ if ρ_1 is a prefix of π, and $\rho_2 \leq \rho_1$ if ρ_2 is a prefix of ρ_1. For a history $\rho = s_0 s_1 \dots s_n$, we denote by $\mathit{last}(\rho)$ its last state s_n, and for all i, j, $0 \leq i \leq j \leq n$, by $\rho(i..j)$ the infix of ρ between position i and position j, i.e., $\rho(i..j) = s_i s_{i+1} \dots s_j$, and by $\rho(i)$ the position i of ρ, i.e., $\rho(i) = s_i$. The set of histories that belong to Eve, noted $\mathsf{Pref}_\exists(\mathcal{A})$ is the subset of histories $\rho \in \mathsf{Pref}(\mathcal{A})$ such that $\mathit{last}(\rho) \in S_\exists$, and the set of histories that belong to Adam, noted $\mathsf{Pref}_\forall(\mathcal{A})$ is the subset of histories $\rho \in \mathsf{Pref}(\mathcal{A})$ such that $\mathit{last}(\rho) \in S_\forall$.

Definition 2 (Strategy). *A* strategy *for* Eve *in the arena* $\mathcal{A}$ *is a function* $\sigma_\exists : \mathsf{Pref}_\exists(\mathcal{A}) \to S$ *such that for all* $\rho \in \mathsf{Pref}_\exists(\mathcal{A})$, $(\mathit{last}(\rho), \sigma_\exists(\rho)) \in \mathsf{E}$, *i.e., it assigns to each history of* $\mathcal{A}$ *that belongs to* Eve *a state which is a* E*-successor of the last state of the history. Symmetrically, a* strategy *for* Adam *in the arena* $\mathcal{A}$ *is a function* $\sigma_\forall : \mathsf{Pref}_\forall(\mathcal{A}) \to S$ *such that for all* $\rho \in \mathsf{Pref}_\forall(\mathcal{A})$, $(\mathit{last}(\rho), \sigma_\forall(\rho)) \in \mathsf{E}$. *The set of strategies for* Eve *is denoted by* $\Sigma_\exists$ *and the set of strategies of* Adam *by* $\Sigma_\forall$.

When we want to refer to a strategy of Eve or Adam, we write it σ. We denote by $\mathsf{Dom}(\sigma)$ the domain of definition of the strategy σ, i.e., for all strategies σ of Eve (resp. Adam), $\mathsf{Dom}(\sigma) = \mathsf{Pref}_\exists(\mathcal{A})$ (resp. $\mathsf{Dom}(\sigma) = \mathsf{Pref}_\forall(\mathcal{A})$).

A play $\pi = s_0 s_1 \dots s_n \dots$ is *compatible* with a strategy σ if for all $i \geq 0$ such that $\pi(0..i) \in \mathsf{Dom}(\sigma)$, we have that $s_{i+1} = \sigma(\rho(0..i))$. We denote by $\mathsf{Outcome}_s(\sigma)$ the set of plays that start in s and are compatible with the strategy σ. Given a strategy $\sigma_\exists$ for Eve and a strategy $\sigma_\forall$ for Adam, and a state s, we write $\mathsf{Outcome}_s(\sigma_\exists, \sigma_\forall)$ the unique play that starts in s and which is compatible both with $\sigma_\exists$ and $\sigma_\forall$.

A strategy σ is *memoryless* when for all histories $\rho_1, \rho_2 \in \mathsf{Dom}(\sigma)$, if we have that $\mathit{last}(\rho_1) = \mathit{last}(\rho_2)$ then $\sigma(\rho_1) = \sigma(\rho_2)$, i.e., memoryless strategies only depend on the last state of the history and so they can be seen as (partial) functions from S to S. $\Sigma_\exists^{\mathsf{ML}}$ and $\Sigma_\forall^{\mathsf{ML}}$ denotes memoryless strategies of Eve and of Adam, respectively. A strategy σ is *finite-memory* if there exists an equivalence relation $\sim \subseteq \mathsf{Dom}(\sigma) \times \mathsf{Dom}(\sigma)$ of *finite index* such that for all histories ρ_1, ρ_2 such that $\rho_1 \sim \rho_2$, we have that $\sigma(\rho_1) = \sigma(\rho_2)$. If the relation $\sim$ is *regular* (computable by a finite state machine) then the finite memory strategy can be modeled by a finite state transducer (a so-called *Moore* or *Mealy* machine). If a strategy is encoded by a machine with m states, we say that it has *memory size* m.

An *objective* $\mathsf{Win} \subseteq \mathsf{Plays}(\mathcal{A})$ is a subset of plays. A strategy σ is winning from state s if $\mathsf{Outcome}_s(\sigma) \subseteq \mathsf{Win}$. We will consider both *qualitative objectives*, that do not depend on the weight function of the game arena, and *quantitative objectives* that depend on the weight function of the game arena.

Our qualitative objectives are defined with Muller conditions (which are a canonical way to represent all the regular sets of plays). Let $\pi \in S^\omega$, be a play, then $\mathsf{inf}(\pi) = \{s \in S \mid \forall i \cdot \exists j \geq i \geq 0 : \pi(j) = s\}$ is the subset of elements of S that occur infinitely often along π. A Muller objective for a game arena $\mathcal{A}$ is a defined by a set of sets of states $\mathcal{F}$ and contains the plays $\{\pi \in S^\omega \mid \mathsf{inf}(\pi) \in \mathcal{F}\}$. We sometimes take the liberty to define such regular sets using standard LTL syntax. For a formal definition of the syntax and semantics of LTL, we refer the interested reader to [2].

We associate, to each play π, an infinite sequence of weights, denoted $\mathsf{w}(\pi)$, and defined as follows:

$$\mathsf{w}(\pi) = \mathsf{w}(\pi(0), \pi(1))\mathsf{w}(\pi(1), \pi(2)) \dots \mathsf{w}(\pi(i), \pi(i+1)) \cdots \in \mathbb{Z}^\omega.$$

To assign a value $\mathsf{Val}(\pi)$ to a play π, we classically use functions like sup (that returns the supremum of the values along the play), inf (that returns the infimum), limsup (that returns the limit superior), liminf (that returns the limit inferior), MP (that returns the limit of the average of the weights along the play), or dSum (that returns the discounted sum of the weights along the play). We only define the mean-payoff measure formally.

Let $\rho = s_0 s_1 \dots s_n$ be s.t. $(s_i, s_{i+1}) \in \mathsf{E}$ for all i, $0 \leq i < n$, the mean-payoff of this sequence of edges is

$$\mathsf{MP}(\rho) = \frac{1}{n} \cdot \sum_{i=0}^{i=n-1} \mathsf{w}(\rho(i), \rho(i+1)),$$

i.e., the mean-value of the weights of the edges traversed by the finite sequence ρ. The *mean-payoff* of an (infinite) play π, denoted $\mathsf{MP}(\pi)$, is a real number defined from the sequence of weights $\mathsf{w}(\pi)$ as follows:

$$\mathsf{MP}(\pi) = \liminf_{n \to +\infty} \frac{1}{n} \cdot \sum_{i=0}^{i=n-1} \mathsf{w}(\pi(i), \pi(i+1)),$$

i.e., $\mathsf{MP}(\pi)$ is the limit inferior of running averages of weights seen along the play π. Note that we need to use $\liminf$ because the value of the running averages of weights may oscillate along π, and so the limit is not guaranteed to exist.

A game is defined by a (weighted) game arena, and objectives for Eve and Adam.

Definition 3 (Game). *A game* $\mathsf{G} = (\mathcal{A}, \mathsf{Win}_\exists, \mathsf{Win}_\forall)$ *is defined by a game arena* $\mathcal{A}$*, an objective* $\mathsf{Win}_\exists$ *for* Eve*, and an objective* $\mathsf{Win}_\forall$ *for* Adam*.*

3 Classical Zero-Sum Setting

In zero sum games, players have antagonistic objectives.

Definition 4. *A game* $\mathsf{G} = (\mathcal{A}, \mathsf{Win}_\exists, \mathsf{Win}_\forall)$ *is* zero-sum *if* $\mathsf{Win}_\forall = \mathsf{Plays} \setminus \mathsf{Win}_\exists$.

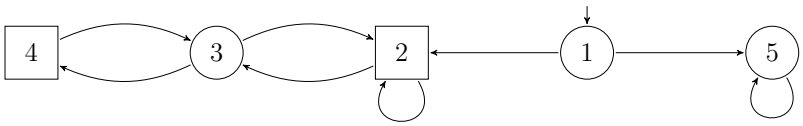


Fig. 1. An example of a two-player game arena. Rounded positions belong to Eve, and squared positions belong to Adam.

Example 1. Let us consider the example of Fig. 1. Assume that the objective of Eve is to visit 4 infinitely often, i.e., $\mathsf{Win}_\exists = \{\pi \in \mathsf{Plays} \mid \pi \models \Box\Diamond 4\}$, and that the objective of Adam is $\mathsf{Win}_\forall = \mathsf{Plays} \setminus \mathsf{Win}_\exists$. Then it should be clear that Eve does not have a strategy that enforces a play in $\mathsf{Win}_\exists$ no matter what Adam plays. Indeed, if Adam always chooses to stay at state 2, there is no way for Eve to visit 4 at all.

As we already said, zero-sum games are usually a bold abstraction of reality. This is because the system to synthesize usually interacts with an environment that has its own objective, and this objective is not necessarily the complement of the objective of the system. A classical way to handle this situation (see e.g., [4]) is to ask the system to win only when the environment meets its own objective.

Definition 5 (Win-Hyp). *Let* $\mathsf{G} = (\mathcal{A}, \mathsf{Win}_\exists, \mathsf{Win}_\forall)$ *be a game,* Eve *achieves* $\mathsf{Win}_\exists$ *from state* s *under hypothesis* $\mathsf{Win}_\forall$ *if there exists* $\sigma_\exists$ *such that*

$$\mathsf{Outcome}_s(\sigma_\exists) \subseteq \mathsf{Win}_\exists \cup \overline{\mathsf{Win}_\forall}.$$

The synthesis rule in the definition above is called *winning under hypothesis*, Win-Hyp for short.

Example 2. Let us consider the example of Fig. 1 again. But now assume that the objective of Adam is to visit 3 infinitely often, i.e., $\mathsf{Win}_\forall = \{\pi \in \mathsf{Plays} \mid \pi \models \Box\Diamond 3\}$. In this case, it should be clear then the strategy $1 \to 2$ and $3 \to 4$ for Eve is winning for the objective

$$\mathsf{Win\text{-}Hyp}_{\Box\Diamond 4 \vee \overline{\Box\Diamond 3}} = \{\pi \in \mathsf{Plays} \mid \pi \models \Box\Diamond 4\} \cup \overline{\{\pi \in \mathsf{Plays} \mid \pi \models \Box\Diamond 3\}}$$

i.e., under the hypothesis that the outcome satisfies the objective of Adam.

Unfortunately, there are strategies of Eve which are winning for the rule Win-Hyp but which are not desirable. As an example, consider the strategy that in 1 chooses to go to 5. In that case, the objective of Adam is unmet and so this strategy of Eve is winning for $\mathsf{Win\text{-}Hyp}_{\Box\Diamond 4 \vee \overline{\Box\Diamond 3}}$, but clearly such a strategy is not interesting as it excludes the possibility to meet the objective of Eve.

4 Assume Admissible Synthesis

To define the notion of admissible strategy, we first need to define when a strategy σ is dominated by a strategy σ'. We will define the notion for Eve, the definition for Adam is symmetric.

Let $\sigma_\exists$ and $\sigma'_\exists$ be two strategies of Eve in the game arena $\mathcal{A}$. We say that $\sigma'_\exists$ *dominates* $\sigma_\exists$ if the following two conditions hold:

1. $\forall \sigma_\forall \in \Sigma_\forall \cdot \mathsf{Outcome}_{s_{\mathsf{init}}}(\sigma_\exists, \sigma_\forall) \in \mathsf{Win}_\exists \rightarrow \mathsf{Outcome}_{s_{\mathsf{init}}}(\sigma'_\exists, \sigma_\forall) \in \mathsf{Win}_\exists$
2. $\exists \sigma_\forall \in \Sigma_\forall \cdot \mathsf{Outcome}_{s_{\mathsf{init}}}(\sigma_\exists, \sigma_\forall) \notin \mathsf{Win}_\exists \wedge \mathsf{Outcome}_{s_{\mathsf{init}}}(\sigma'_\exists, \sigma_\forall) \in \mathsf{Win}_\exists$

So a strategy $\sigma_\exists$ is dominated by $\sigma'_\exists$ if $\sigma'_\exists$ does as well as $\sigma_\exists$ against any strategy of Adam (condition 1), and there exists a strategy of Adam against which $\sigma'_\exists$ does better than $\sigma_\exists$ (condition 2).

Definition 6 (Admissible Strategy). *A strategy is admissible if there does not exist a strategy that dominates it.*

Let $\mathsf{G} = (\mathcal{A}, \mathsf{Win}_\exists, \mathsf{Win}_\forall)$ be a game, the set of admissible strategies for Eve is noted $\mathsf{Adm}_\exists$, and the set of admissible strategies for Adam is denoted $\mathsf{Adm}_\forall$.

Clearly, a rational player should not play a dominated strategy as there always exists some strategy that behaves strictly better than the dominated strategy. So, a rational player only plays admissible strategies.

Example 3. Let us consider again the example of Fig. 1 with $\mathsf{Win}_\exists = \{\pi \in \mathsf{Plays} \mid \pi \models \Box\Diamond 4\}$ and $\mathsf{Win}_\forall = \{\pi \in \mathsf{Plays} \mid \pi \models \Box\Diamond 3\}$. We claim that the strategy $\sigma_\exists$ that plays $1 \rightarrow 5$ is not admissible in $\mathcal{A}$ from state 1. This is because the strategy $\sigma'_\exists$ that plays $1 \rightarrow 2$ and $4 \rightarrow 3$ dominates this strategy. Indeed, while $\sigma_\exists$ is always losing for the objective of Eve, the strategy $\sigma'_\exists$ wins for this objective whenever Adam eventually plays $2 \rightarrow 3$.

Definition 7 (AA). *Let* $\mathsf{G} = (\mathcal{A}, \mathsf{Win}_\exists, \mathsf{Win}_\forall)$ *be a game,* Eve *achieves* $\mathsf{Win}_\exists$ *from* s *under the hypothesis that* Adam *plays admissible strategies if*

$$\exists \sigma_\exists \in \mathsf{Adm}_\exists \cdot \forall \sigma_\forall \in \mathsf{Adm}_\forall \cdot \mathsf{Outcome}_s(\sigma_\exists, \sigma_\forall) \in \mathsf{Win}_\exists.$$

Example 4. Let us consider again the example of Fig. 1 with $\mathsf{Win}_\exists = \{\pi \in \mathsf{Plays} \mid \pi \models \Box\Diamond 4\}$ and $\mathsf{Win}_\forall = \{\pi \in \mathsf{Plays} \mid \pi \models \Box\Diamond 3\}$. We claim that the strategy $\sigma_\exists$ of Eve that plays $1 \rightarrow 2$ and $4 \rightarrow 3$ is admissible (see previous example) and winning against all the admissible strategies of Adam. This is a consequence of the fact that the strategy of Adam that always plays $2 \rightarrow 2$, and which is the only counter strategy of Adam against $\sigma_\exists$, is *not* admissible. Indeed, this strategy falsifies $\mathsf{Win}_\forall$ while a strategy that always chooses $2 \rightarrow 3$ enforces the objective of Adam.

Theorem 1 ([3,8,7]). *For all games* $\mathsf{G} = (\mathcal{A}, \mathsf{Win}_\exists, \mathsf{Win}_\forall)$*, if* $\mathsf{Win}_\exists$ *and* $\mathsf{Win}_\forall$ *are omega-regular sets of plays, then* $\mathsf{Adm}_\exists$ *and* $\mathsf{Adm}_\forall$ *are both non empty sets.*

The problem of deciding if a game $\mathsf{G} = (\mathcal{A}, \mathsf{Win}_\exists, \mathsf{Win}_\forall)$*, where* $\mathsf{Win}_\exists$ *and* $\mathsf{Win}_\forall$ *are omega-regular sets of plays expressed as Muller objectives, satisfies*

$$\exists \sigma_\exists \in \mathsf{Adm}_\exists \cdot \forall \sigma_\forall \in \mathsf{Adm}_\forall \cdot \mathsf{Outcome}_s(\sigma_\exists, \sigma_\forall) \in \mathsf{Win}_\exists$$

is PSPACE-COMPLETE.

Additional Results. The assume-admissible setting we present here relies on procedures for *iterative elimination of dominated strategies* for multiple players which was studied in [3] on games played on graphs. In this context, dominated strategies are repeatedly eliminated for each player. Thus, with respect to the new set of strategies of its opponent, new strategies may become dominated, and will therefore be eliminated, and so on until the process stabilizes. In [8], we studied the algorithmic complexity of this problem and proved that for games with Muller objectives, deciding whether all outcomes compatible with iteratively admissible strategy profiles satisfy an omega-regular objective defined by a Muller condition is PSPACE-COMPLETE and in UP$\cap$coUP for the special case of Büchi objectives.

The assume-admissible rule introduced in [7] is also defined for multiple players and corresponds, roughly, to the first iteration of the elimination procedure. We additionally prove that if players have Büchi objectives, then the rule can be decided in polynomial-time. One advantage of the assume-admissible rule is the *rectangularity* of the solution set: the set of strategy profiles that witness the rule can be written as a product of sets of strategies for each player. In particular, this means that a strategy witnessing the rule can be chosen separately for each player. Thus, the rule is *robust* in the sense that the players do not need to agree on a strategy profile, but only on the admissibility assumption on each other. In addition, we show in [7] that the rule is amenable to abstraction techniques: we show how state-space abstractions can be used to check a sufficient condition for assume-admissible, only doing computations on the abstract state space.

Related Works. The rule "winning under hypothesis" (Win-Hyp) and its weaknesses are discussed in [4]. We have illustrated the limitations of this rule in Example 2.

There are related works in the literature which propose concepts to model systems composed of several parts, each having their own objectives. The solutions that are proposed are based on n-players *non*-zero sum games. This is the case both for *assume-guarantee synthesis* [17] (AG), and for *rational synthesis* [29] (RS).

For the case of two player games, AG is based on the concept of *secure equilibria* [18] (SE), a refinement of Nash equilibria [37] (NE). In SE, objectives of the players are lexicographic: each player first tries to force his own objective, and then tries to falsify the objectives of the other players. It was shown in [18] that SE are the NE that form enforceable contracts between the two players. When the AG rule is extended to several players, as in [17], it no longer corresponds to secure equilibria. We gave a direct algorithm for multiple players in [7]. The difference between AG and SE is that AG strategies have to be resilient to deviations of all the other players, while SE profiles have to be resilient to deviations by only one player. A variant of the rule AG, called *Doomsday equilibria*, has been proposed in [14]. We have also studied quantitative extensions of the notion of secure equilibria in [13].

In the context of infinite games played on graphs, one well known limitation of NE is the existence of *non-credible threats*. Refinements of the notion of

NE, like *sub-game perfect equilibria* (SPE), have been proposed to overcome this limitation. SPE for games played on graphs have been studied in e.g., [42,9]. Admissibility does not suffer from this limitation.

In RS, the system is assumed to be monolithic and the environment is made of several components that are only *partially controllable*. In RS, we search for a profile of strategies in which the system forces its objective and the players that model the environment are given an "*acceptable*" strategy profile, from which it is assumed that they will not deviate. "Acceptable" can be formalized by any *solution concept*, e.g., by NE, *dominant* strategies, or *sub-game perfect equilibria*. This is the existential flavor of RS. More recently, Kupferman et al. have proposed in [34] a *universal* variant of this rule. In this variant, we search for a strategy of the system such that in all strategy profiles that extend this strategy for the system and that are NE, the outcome of the game satisfies the specification of the system.

In [25], Faella studies several alternatives to the notion of winning strategy including the notion of admissible strategy. His work is for two-players but only the objective of one player is taken into account, the objective of the other player is left unspecified. In that work, the notion of admissibility is used to define a notion of *best-effort* in synthesis.

The notion of admissible strategy is definable in strategy logics [19,36] and decision problems related to the assume-admissible rule can be reduced to satisfiability queries in such logics. This reduction does not lead to worst-case optimal algorithms; we presented worst-case optimal algorithms in [7] based on our previous work [8].

5 Regret Minimization

In the previous section, we have shown how the notion of admissible strategy can be used to relax the classical worst-case hypothesis made on the environment. In this section, we review another way to relax this worst-case hypothesis.

The idea is simple and intuitive. When looking for a strategy, instead of trying to find a strategy which is worst-case optimal, we search for a strategy that takes *best-responses* (against the behavior of the environment) as a yardstick. That is, we would like to find a strategy that behaves "not far" from an optimal response to the strategy of the environment — when the latter is fixed. The notion of regret minimization is naturally defined in a quantitative setting (although it also makes sense in a Boolean setting).

Let us now formally define the notion of regret associated to a strategy of Eve. This definition is parameterized by a set of strategies for Adam.

Definition 8 (Relative Regret). *Let $\mathcal{A} = \langle S_\exists, S_\forall, \mathsf{E}, s_{\mathsf{init}}, \mathsf{w}\rangle$ be a weighted game arena, let $\sigma_\exists$ be a strategy of* Eve*, the regret of this strategy relative to a set of strategies $\mathsf{Str}_\forall \subseteq \Sigma_\forall$ is defined as follows:*

$$\mathsf{Reg}(\sigma_\exists, \mathsf{Str}_\forall) = \sup_{\sigma_\forall \in \mathsf{Str}_\forall} \sup_{\sigma'_\exists \in \Sigma_\exists} \mathsf{Val}(\sigma_\forall, \sigma'_\exists) - \mathsf{Val}(\sigma_\forall, \sigma_\exists).$$

We interpret the sub-expression $\sup_{\sigma'_\exists \in \Sigma_\exists} \mathsf{Val}(\sigma_\forall, \sigma'_\exists)$ as the *best-response* of Eve against $\sigma_\forall$. Then, the relative regret of a strategy of Eve can be seen as the supremum of the differences between the value achieved by $\sigma_\exists$ against a strategy of Adam and the value achieved by the corresponding best-response.

We are now equipped to formally define the problem under study, which is parameterized by payoff function $\mathsf{Val}(\cdot)$ and a set $\mathsf{Str}_\forall$ of strategies of Adam.

Definition 9 (Regret Minimization). *Given a weighted game arena $\mathcal{A}$ and a rational threshold r, decide if there exists a strategy $\sigma_\exists$ for* Eve *such that*

$$\mathsf{Reg}(\sigma_\exists, \mathsf{Str}_\forall) \leq r$$

and synthesize such a strategy if one exists.

In [32], we have considered several types of strategies for Adam: the set $\Sigma_\forall$, i.e., any strategy, the set $\Sigma_\forall^{\mathsf{ML}}$, i.e., memoryless strategies for Adam, and the set $\Sigma_\forall^{\mathsf{W}}$, i.e., word strategies for Adam.[6] We will illustrate each of these cases on examples below.

Example 5. Let us consider the weighted game arena of Fig. 2, and let us assume that we want to synthesize a strategy for Eve that minimizes her mean-payoff regret against Adam playing a memoryless strategy. The memoryless restriction is useful when designing a system that needs to perform well in an environment which is only partially known. In practice, a controller may discover the environment with which it is interacting during run-time. Such a situation can be modeled by an arena in which choices in nodes of the environment model an entire family of environments and each memoryless strategy models a specific environment of the family. In such cases, if we want to design a controller that performs reasonably well against all the possible environments, we can consider each best-response of Eve for each environment and then try to choose one unique strategy for Eve that minimizes the difference in performance w.r.t. those best-responses: a regret-minimizing strategy.

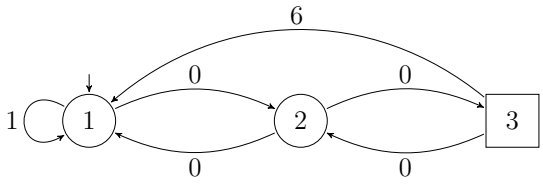


Fig. 2. An example of a two-player game arena with MP objective for Eve. Rounded positions belong to Eve, and squared positions belong to Adam.

[6] To define word strategies, it is convenient to consider game arenas where edges have labels called letters. In that case, when playing a word strategy, Adam commits to a sequence of letters (i.e., a word) and plays that word regardless of the exact state of the game. Word strategies are formally defined in [32] and below.

In our example, prior to a first visit to state 3, we do not know if the edge $3 \to 2$ or the edge $3 \to 1$ will be activated by Adam. But as Adam is bound to play a memoryless strategy, once he has chosen one of the two edges, we know that he will stick to this choice.

A regret-minimizing strategy in this example is as follows: play $1 \to 2$, then $2 \to 3$, if Adam plays $3 \to 2$, then play $2 \to 1$ and then $1 \to 1$ forever, otherwise Adam plays $3 \to 1$ and then Eve should continue to play $1 \to 2$ and $2 \to 3$ forever. This strategy has regret 0. Note that this strategy uses memory and that there is no memoryless strategy of Eve with regret 0 in this game.

Let us now illustrate the interest of the notion of regret minimization when Adam plays *word* strategies. When considering this restriction, it is convenient to consider letters that label the edges of the graph (Fig. 3). A word strategy for Adam is a function $w : \mathbb{N} \to \{a, b\}$. In this setting Adam plays a sequence of letters and this sequence is independent of the current state of the game. We have shown in [32] that the notion of regret minimization relative to word strategies is a generalization of the notion of *good-for-games automata* introduced by Henzinger and Piterman in [31].

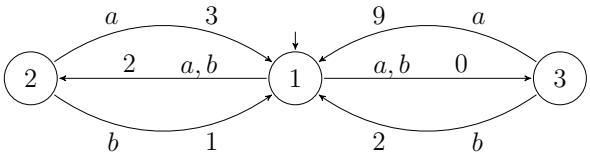


Fig. 3. An example of a two-player game arena with MP objective for Eve. Edges are annotated by letters: Adam chooses a word w and Eve resolves the non-determinism on edges.

Example 6. In this example, a strategy of Eve determines how to resolve non-determinism in state 1. The best strategy of Eve for mean-payoff regret minimization is to always take the edge $1 \to 3$. Indeed, let us consider all the sequences of two letters that Adam can choose and compute the regret of choosing $1 \to 2$ (left) and the regret of choosing $1 \to 3$ (right):

- $*a$ with $* \in \{a, b\}$, the regret of left is equal to 0, and the regret of right is $\frac{5-3}{2} = 1$.
- $*b$ with $* \in \{a, b\}$, the regret of left is equal to $\frac{9-3}{2} = 3$, and the regret of right is 0.

So the strategy that minimizes the regret of Eve is to always take the arrow $1 \to 3$ (right), the regret is then equal to 1.

In [32], we have studied the complexity of deciding the existence of strategies for Eve that have less than a given regret threshold. The results that we have obtained are summarized in the theorem below.

Theorem 2 ([32]). *Let $\mathcal{A} = \langle S_\exists, S_\forall, \mathsf{E}, s_{\mathsf{init}}, \mathsf{w}\rangle$ be a weighted game arena, the complexity of deciding if* Eve *has a strategy with regret less than or equal to a threshold $r \in \mathbb{Q}$ against* Adam *playing:*

- *a strategy in $\Sigma_\forall$, is* PTime-Complete *for payoff functions* inf, sup, liminf, limsup, *and in* NP∩coNP *for* MP.
- *a strategy in $\Sigma_\forall^{\mathsf{ML}}$, is in* PSpace *for payoff functions* inf, sup, liminf, limsup, *and* MP, *and is* coNP-Hard *for* inf, sup, limsup, *and* PSpace-Hard *for* liminf, *and* MP.
- *a strategy in $\Sigma_\forall^{\mathsf{W}}$, is* ExpTime-Complete *for payoff functions* inf, sup, liminf, limsup, *and undecidable for* MP.

The above results are obtained by reducing the synthesis of regret-minimizing strategies to finding winning strategies in classical games. For instance, a strategy for Eve that minimizes regret against $\Sigma_\forall^{\mathsf{ML}}$ for the mean-payoff measure corresponds to finding a winning strategy in a mean-payoff game played on a larger game arena which encodes the witnessed choices of Adam and forces him to play positionally. When minimizing regret against word strategies, for the decidable cases the reduction is done to parity games and is based on the *quantitative simulation* games defined in [15].

Additional Results. Since synthesis of regret-minimizing strategies against word strategies of Adam is undecidable with measure MP, we have considered the sub-case which limits the amount of memory the desired controller can use (as in [1]). That is, we ask whether there exists a strategy of Eve which uses at most memory m and ensures regret at most r. In [32] we showed that this problem is in $\text{NTime}(m^2|\mathcal{A}|^2)$ for MP.

Theorem 3 ([32]). *Let $\mathcal{A} = \langle S_\exists, S_\forall, \mathsf{E}, s_{\mathsf{init}}, \mathsf{w}\rangle$ be a weighted game arena, the complexity of deciding if* Eve *has a strategy using memory of at most m with regret less than or equal to a threshold $\lambda \in \mathbb{Q}$ against* Adam *playing a strategy in $\Sigma_\forall^{\mathsf{W}}$, is in non-deterministic polynomial time w.r.t. m and $|\mathcal{A}|$ for* inf, sup, liminf, limsup, *and* MP.

Finally, we have established the equivalence of a quantitative extension of the notion of good-for-games automata [31] with determinization-by-pruning of the refinement of an automaton [1] and our regret games against word strategies of Adam. Before we can formally state these results, some definitions are needed.

Definition 10 (Weighted Automata). *A finite weighted automaton is a tuple $\langle Q, q_{\mathsf{init}}, A, \Delta, \mathsf{w}\rangle$ where: Q is a finite set of states, $q_{\mathsf{init}} \in Q$ is the initial state, A is a finite alphabet of actions or symbols, $\Delta \subseteq Q \times A \times Q$ is the transition relation, and $\mathsf{w} : \Delta \to \mathbb{Z}$ is the weight function.*

A *run* of an automaton on a word $a \in A^\omega$ is an infinite sequence of transitions $\rho = (q_0, a_0, q_1)(q_1, a_1, q_2)\cdots \in \Delta^\omega$ such that $q_0 = q_{\mathsf{init}}$ and $a_i = a(i)$ for all $i \geq 0$. As with plays in a game, each run is assigned a *value* with a payoff function $\mathsf{Val}(\cdot)$. A weighted automaton $\mathcal{M}$ defines a function $A^\omega \to \mathbb{R}$ by assigning to

$a \in A^\omega$ the supremum over all the values of its runs on a. The automaton is said to be *deterministic* if for all $q \in Q$ and $x \in A^\omega$ the set $\{q' \in Q \mid (q, x, q') \in \Delta\}$ is a singleton.

In [31], Henzinger and Piterman introduced the notion of *good-for-games automata*. A non-deterministic automaton is good for solving games if it fairly simulates the equivalent deterministic automaton.

Definition 11 (α-good-for-games). *A finite weighted automaton $\mathcal{M}$ is* α-good-for-games *if a player (Simulator), against any word $x \in A^\omega$ spelled by Spoiler, can resolve non-determinism in $\mathcal{M}$ so that the resulting run has value v and $\mathcal{M}(x) - v \leq \alpha$.*

The above definition is a quantitative generalization of the notion proposed in [31]. We link their class of automata with our regret games in the sequel.

Proposition 1 ([32]). *A weighted automaton $\mathcal{M} = \langle Q, q_{\mathsf{init}}, A, \Delta, \mathsf{w}\rangle$ is α-good-for-games if and only if there exists a strategy $\sigma_\exists$ for* Eve *with regret α against strategies $\Sigma^{\mathsf{W}}_\forall$ of* Adam *(i.e.,* $\mathsf{Reg}(\sigma_\exists, \Sigma^{\mathsf{W}}_\forall)$*).*

Our definitions also suggest a natural notion of approximate determinization for weighted automata on infinite words. This is related to recent work by Aminof et al.: in [1], they introduce the notion of *approximate-determinization-by-pruning* for weighted sum automata over finite words. For $\alpha \in (0, 1]$, a weighted sum automaton is *α-determinizable-by-pruning* if there exists a finite state strategy to resolve non-determinism and that constructs a run whose value is at least α times the value of the maximal run of the given word. So, they consider a notion of approximation which is a *ratio*. Let us introduce some additional definitions required to formalize the notion of determinizable-by-pruning.

Consider two weighted automata $\mathcal{M} = \langle Q, q_{\mathsf{init}}, A, \Delta, \mathsf{w}\rangle$ and $\mathcal{M}' = \langle Q', q'_{\mathsf{init}}, A, \Delta', \mathsf{w}'\rangle$. We say that $\mathcal{M}'$ *α-approximates* $\mathcal{M}$ if $|\mathcal{M}(x) - \mathcal{M}'(x)| \leq \alpha$, for all $x \in A^\omega$. We say that $\mathcal{M}$ *embodies* $\mathcal{M}'$ if $Q' \subseteq Q$, $\Delta' \subseteq \Delta$, and w' agrees with w on Δ'. For an integer $k \geq 0$, the k-refinement of $\mathcal{M}$ is the automaton obtained by refining the state-space of $\mathcal{M}$ using k boolean variables.

Definition 12 ((α, k)-determinizable-by-pruning). *A finite weighted automaton $\mathcal{M}$ is* (α, k)-determinizable-by-pruning *if the k-refinement of $\mathcal{M}$ embodies a deterministic automaton which α-approximates $\mathcal{M}$.*

We show in [32] that when Adam plays word strategies only, our notion of regret defines a notion of approximation with respect to the *difference* metric for weighted automata (as defined above).

Proposition 2 ([32]). *A weighted automaton $\mathcal{M} = \langle Q, q_{\mathsf{init}}, A, \Delta, \mathsf{w}\rangle$ is α-determinizable-by-pruning if and only if there exists a strategy $\sigma_\exists$ for* Eve *using memory at most 2^m with regret α against strategies $\Sigma^{\mathsf{W}}_\forall$ of* Adam *(i.e.,* $\mathsf{Reg}(\sigma_\exists, \Sigma^{\mathsf{W}}_\forall)$*).*

Related Works The notion of regret minimization is important in game and decision theory, see e.g., [45] and additional bibliographical pointers there. The concept of *iterated* regret minimization has been recently proposed by Halpern et al. for *non-zero* sum games [30]. In [28], the concept is applied to games played on weighted graphs with shortest path objectives. Variants on the different sets of strategies considered for Adam were not considered there.

In [23], Damm and Finkbeiner introduce the notion of *remorse-free strategies*. The notion is introduced in order to define a notion of *best-effort* strategy when winning strategies do not exist. Remorse-free strategies are exactly the strategies which minimize regret in games with ω-regular objectives in which the environment (Adam) is playing word strategies only. The authors of [23] do not establish lower bounds on the complexity of the realizability and synthesis problems for remorse-free strategies.

A concept equivalent to good-for-games automata is that of history-determinism [22]. Proposition 1 thus allows us to generalize history-determinism to a quantitative setting via this relationship with good-for-games automata.

Finally, we would like to highlight some differences between our work and the study of Aminof et al. in [1] on determinization-by-pruning. First, we consider infinite words while they consider finite words. Second, we study a general notion of regret minimization problem in which Eve can use any strategy while they restrict their study to fixed memory strategies only and leave the problem open when the memory is not fixed a priori.

6 Game Arenas with Expected Adversary

In the two previous sections we have relaxed the worst-case hypothesis on the environment (modeled by the behavior of Adam) by either considering an explicit objective for the environment or by considering as yardsticks the best-responses to the strategies of Adam. Here, we introduce another model where the environment is modeled as a stochastic process (i.e., Adam is expected to play according to some known *randomized* strategy) and we are looking for strategies for Eve that ensure good expectation against this stochastic process while guaranteeing acceptable worst-case performance even if Adam deviates from his expected behavior.

To define formally this new framework, we need game arenas in which an expected behavior for Adam is given as a *memoryless randomized strategy*.[7] We first introduce some notation. Given a set A, let $\mathcal{D}(A)$ denote the set of rational probability distributions over A, and, for $d \in \mathcal{D}(A)$, we denote its support by $\mathsf{Supp}(d) = \{a \in A \mid d(a) > 0\} \subseteq A$.

[7] It should be noted that we can easily consider finite-memory randomized strategies for Adam, instead of memoryless randomized strategies. This is because we can always take the synchronized product of a finite-memory randomized strategy with the game arena to obtain a new game arena in which the finite-memory strategy on the original game arena is now equivalent to a memoryless strategy.

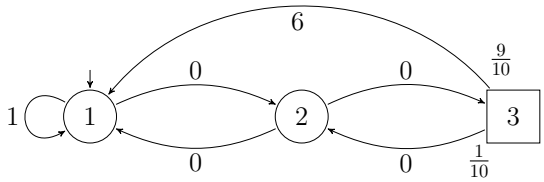


Fig. 4. A game arena associated with a memoryless randomized strategy for Adam can be seen as an MDP: the fractions represent the respective probability to take each outgoing edge when leaving state 3.

Definition 13. *Fix a weighted game arena* $\mathcal{A} = \langle S_\exists, S_\forall, \mathsf{E}, s_{\mathsf{init}}, \mathsf{w}\rangle$. *A memoryless randomized strategy for* Adam *is a function*

$$\sigma_\forall^{\mathsf{rnd}} : S_\forall \to \mathcal{D}(S)$$

such that for all $s \in S_\forall$, $\mathsf{Supp}(\sigma_\forall^{\mathsf{rnd}}(s)) \subseteq \{s' \in S \mid (s, s') \in \mathsf{E}\}$.

For the rest of this section, we model the expected behavior of Adam with a strategy $\sigma_\forall^{\mathsf{rnd}}$, given as part of the input for the problem we will consider. Given a weighted game arena $\mathcal{A}$ and a memoryless randomized strategy $\sigma_\forall^{\mathsf{rnd}}$ for Adam, we are left with a model with both non-deterministic choices (for Eve) and stochastic transitions (due to the randomized strategy of Adam). This is essentially what is known in the literature as a $1\frac{1}{2}$-player game or more commonly, a *Markov Decision Process* (MDP), see for example [38,26]. One can talk about plays, strategies and other notions in MDPs as introduced for games.

Consider the game in Fig. 4. We can see it as a classical two-player game if we forget about the fractions around state 3. Now assume that we fix the memoryless randomized strategy $\sigma_\forall^{\mathsf{rnd}}$ for Adam to be the one that, from 3, goes to 1 with probability $\frac{9}{10}$ and to 2 with the remaining probability, $\frac{1}{10}$. This is represented by the fractions on the corresponding outgoing edges. In the remaining model, only Eve still has to pick a strategy: it is an MDP. We denote this MDP by $\mathcal{A}[\sigma_\forall^{\mathsf{rnd}}]$.

Let us go one step further. Assume now that Eve also picks a strategy $\sigma_\exists$ in this MDP. Now we obtain a fully stochastic process called a *Markov Chain* (MC). We denote it by $\mathcal{A}[\sigma_\exists, \sigma_\forall^{\mathsf{rnd}}]$. In an MC, an *event* is a measurable set of plays. It is well-known from the literature [43] that every event has a uniquely defined probability (Carathéodory's extension theorem induces a unique probability measure on the Borel σ-algebra over plays in the MC). Given $\mathcal{E}$ a set of plays in $M = \mathcal{A}[\sigma_\exists, \sigma_\forall^{\mathsf{rnd}}]$, we denote by $\mathbb{P}_M(\mathcal{E})$ the probability that a play belongs to $\mathcal{E}$ when M is executed for an infinite number of steps. Given a measurable value function Val, we denote by $\mathbb{E}_M(\mathsf{Val})$ the *expected value* or *expectation* of Val over plays in M. In this paper, we focus on the mean-payoff function MP.

We are now finally equipped to formally define the problem under study.

Definition 14 (Beyond Worst-Case Synthesis). *Given a weighted game arena $\mathcal{A}$, a stochastic model of* Adam *given as a memoryless randomized strategy $\sigma_\forall^{\mathsf{rnd}}$, and two rational thresholds $\lambda_{\mathsf{wc}}, \lambda_{\mathsf{exp}}$, decide if there exists a strategy $\sigma_\exists$ for* Eve *such that*

$$\begin{cases} \forall \pi \in \mathsf{Outcome}^{\mathcal{A}}_{s_{\mathsf{init}}}(\sigma_\exists) \cdot \mathsf{Val}(\pi) > \lambda_{\mathsf{wc}} \\ \mathbb{E}_{\mathcal{A}[\sigma_\exists, \sigma_\forall^{\mathsf{rnd}}]}(\mathsf{Val}) > \lambda_{\mathsf{exp}} \end{cases}$$

and synthesize such a strategy if one exists.

Intuitively, we are looking for strategies that can *simultaneously* guarantee a worst-case performance higher than λ_{wc}, i.e., against any behavior of Adam in the game $\mathcal{A}$, *and* guarantee an expectation higher than λ_{exp} when faced to the expected behavior of Adam, i.e., when played in the MDP $\mathcal{A}[\sigma_\forall^{\mathsf{rnd}}]$. We can of course assume w.l.o.g. that $\lambda_{\mathsf{wc}} < \lambda_{\mathsf{exp}}$, otherwise the problem reduces trivially to just a worst-case requirement: any lower bound on the worst-case value is also a lower bound on the expected value.

Example 7. Consider the arena depicted in Fig. 4. As mentioned before, the probability distribution models the expected behavior of Adam. Assume that we want now to synthesize a strategy for Eve which ensures that (C_1) the mean-payoff will be at least $\frac{1}{3}$ no matter how Adam behaves (worst-case guarantee), and (C_2) at least $\frac{3}{2}$ if Adam plays according to his expected behavior (good expectation).

First, let us study whether this can be achieved through the two classical solution concepts used in games and MDPs respectively. We start by considering the arena as a traditional two-player zero-sum game: in this case, it is known that an optimal memoryless strategy exists [24]. Let $\sigma_\exists^{\mathsf{wc}}$ be the strategy of Eve that always plays $1 \to 1$ and $2 \to 1$. That strategy maximizes the worst-case mean-payoff, as it enforces a mean-payoff of 1 no matter how Adam behaves. Thus, (C_1) is satisfied. Observe that if we consider the arena as an MDP (i.e., taking the probabilities into account), this strategy yields an expected value of 1 as the unique possible play from state 1 is to take the self-loop forever. Hence this strategy does not satisfy (C_2).

Now, consider the arena as an MDP. Again, it is known that the expected value can be maximized by a memoryless strategy [38,26]. Let $\sigma_\exists^{\mathsf{exp}}$ be the strategy of Eve that always chooses the following edges: $1 \to 2$ and $2 \to 3$. Its expected mean-payoff can be calculated in two steps: first computing the probability vector that represents the limiting stationary distribution of the irreducible MC induced by this strategy, second multiplying it by the vector containing the expected weights over outgoing edges for each state. In this case, it can be shown that the expected value is equal to $\frac{54}{29}$, hence the strategy does satisfy (C_2). Unfortunately, it is clearly not acceptable for (C_1) as, if Adam does not behave according to the stochastic model and always chooses to play $3 \to 2$, the mean-payoff will be equal to zero.

Hence this shows that the classical solution concepts do not suffice if one wants to go beyond the worst-case and mix guarantees on the worst-case and

the expected performance of strategies. In contrast, with the framework developed in [12,11], it is indeed possible for the considered arena (Fig. 4) to build a strategy for Eve that ensures the worst-case constraint (C_1) and at the same time, yields an expected value arbitrarily close to the optimal expectation achieved by strategy $\sigma_\exists^{\mathsf{exp}}$. In particular, one can build a finite-memory strategy that guarantees *both* (C_1) and (C_2). The general form of such strategies is a combination of $\sigma_\exists^{\mathsf{exp}}$ and $\sigma_\exists^{\mathsf{wc}}$ in a well-chosen pattern. Let $\sigma_\exists^{\mathsf{cmb(K,L)}}$ be a *combined strategy* parameterized by two integers $K, L \in \mathbb{N}$. The strategy is as follows.

1. Play according to $\sigma_\exists^{\mathsf{exp}}$ for K steps.
2. If the mean-payoff over the last K steps is larger than the worst-case threshold λ_{wc} (here $\frac{1}{3}$), then go to phase 1.
3. Otherwise, play according to $\sigma_\exists^{\mathsf{wc}}$ for L steps, and then go to phase 1.

Intuitively, the strategy starts by mimicking $\sigma_\exists^{\mathsf{exp}}$ for a long time, and the witnessed mean-payoff over the K steps will be close to the optimal expectation with high probability. Thus, with high probability it will be higher than λ_{exp}, and therefore higher than λ_{wc} — recall that we assumed $\lambda_{\mathsf{wc}} < \lambda_{\mathsf{exp}}$. If this is not the case, then Eve has to switch to $\sigma_\exists^{\mathsf{wc}}$ for sufficiently many steps L in order to make sure that the worst-case constraint (C_1) is satisfied before switching back to $\sigma_\exists^{\mathsf{exp}}$.

One of the key results of [12] is to show that for any $\lambda_{\mathsf{wc}} < \mu$, where μ denotes the optimal worst-case value guaranteed by $\sigma_\exists^{\mathsf{wc}}$, and for any expected value threshold $\lambda_{\mathsf{exp}} < \nu$, where ν denotes the optimal expected value guaranteed by $\sigma_\exists^{\mathsf{exp}}$, it is possible to compute values for K and L such that $\sigma_\exists^{\mathsf{cmb(K,L)}}$ satisfies the beyond worst-case constraint for thresholds λ_{wc} and λ_{exp}. For instance, in the example, where $\lambda_{\mathsf{wc}} = \frac{1}{3} < 1$ and $\lambda_{\mathsf{exp}} = \frac{3}{2} < \frac{54}{29}$, one can compute appropriate values of the parameters following the technique presented in [12, Theorem 5]. The crux is proving that, for large enough values of K and L, the contribution to the expectation of the phases when $\sigma_\exists^{\mathsf{cmb(K,L)}}$ mimics $\sigma_\exists^{\mathsf{wc}}$ are negligible, and thus the expected value yield by $\sigma_\exists^{\mathsf{cmb(K,L)}}$ tends to the optimal one given by $\sigma_\exists^{\mathsf{exp}}$, while at the same time the strategy ensures that the worst-case constraint is met.

In the next theorem, we sum up some of the main results that we have obtained for the beyond worst-case synthesis problem applied to the mean-payoff value function.

Theorem 4 ([12,11,21]). *The beyond worst-case synthesis problem for the mean-payoff is in* NP $\cap$ coNP*, and at least as hard as deciding the winner in two-player zero-sum mean-payoff games, both when looking for finite-memory or infinite-memory strategies of* Eve*. When restricted to finite-memory strategies, pseudo-polynomial memory is both sufficient and necessary.*

The NP $\cap$ coNP-membership is good news as it matches the long-standing complexity barrier for two-player zero-sum mean-payoff games [24,46,10,16]: the beyond worst-case framework offers *additional modeling power for free* in terms

of decision complexity. It is also interesting to note that in general, infinite-memory strategies are more powerful than finite-memory ones in the beyond worst-case setting, which is not the case for the classical problems in games and MDPs.

Looking carefully at the techniques from [12,11], it can be seen that the main bottleneck in complexity is solving mean-payoff games in order to check whether the worst-case constraint can be met. Therefore, a natural relaxation of the problem is to consider the *beyond almost-sure threshold problem* where the worst-case constraint is softened by only asking that a threshold is satisfied with probability one against the stochastic model given as the strategy $\sigma_{\forall}^{\mathsf{rnd}}$ of Adam. In this case, the complexity is reduced.

Theorem 5 ([21]). *The beyond almost-sure threshold problem for the mean-payoff is in* PTime *and finite-memory strategies are sufficient.*

Related Works We originally introduced the beyond worst-case framework in [12] where we studied both mean-payoff and shortest path objectives. This framework generalizes classical problems for two-player zero-sum games and MDPs. In mean-payoff games, optimal memoryless strategies exist and deciding the winner lies in NP $\cap$ coNP while no polynomial algorithm is known [24,46,10,16]. For shortest path games, where we consider game graphs with strictly positive weights and try to minimize the accumulated cost to target, it can be shown that memoryless strategies also suffice, and the problem is in PTime [33]. In MDPs, optimal strategies for the expectation are studied in [38,26] for the mean-payoff and the shortest path: in both cases, memoryless strategies suffice and they can be computed in PTime. While we saw that the beyond worst-case synthesis problem does not cost more than solving games for the mean-payoff, it is not the case anymore for the shortest path: we jump from PTime to a pseudo-polynomial-time algorithm. We proved in [12, Theorem 11] that the problem is inherently harder as it is NP-hard.

The beyond worst-case framework was extended to the multi-dimensional setting — where edges are fitted with vectors of integer weights — in [21]. The general case is proved to be coNP-complete.

Our strategies can be considered as *strongly risk averse*: they avoid at all cost outcomes that are below a given threshold (no matter what is their probability), and inside the set of those *safe* strategies, we maximize the expectation. Other different notions of risk have been studied for MDPs: in [44], the authors want to find policies which minimize the probability (risk) that the total discounted rewards do not exceed a specified value (target); in [27] the authors want policies that achieve a specified value of the long-run limiting average reward at a specified probability level (percentile). The latter problem was recently extended significantly in the framework of *percentile queries*, which provide elaborate guarantees on the performance profile of strategies in multi-dimensional MDPs [40]. While all those strategies limit risk, they only ensure *low probability* for bad behaviors but they do not ensure their absence, furthermore, they do not ensure good expectation either.

Another body of work is the study of strategies in MDPs that achieve a trade-off between the expectation and the variance over the outcomes (e.g., [6] for the mean-payoff, [35] for the cumulative reward), giving a statistical measure of the stability of the performance. In our setting, we strengthen this requirement by asking for *strict guarantees on individual outcomes*, while maintaining an appropriate expected payoff.

A survey of rich behavioral models extending the classical approaches for MDPs — including the beyond worst-case framework presented here — was published in [41], with a focus on the shortest path problem.

References

1. B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 2010.
2. C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press, 2008.
3. D. Berwanger. Admissibility in infinite games. In *Proc. of STACS*, LNCS 4393, pages 188–199. Springer, 2007.
4. R. Bloem, R. Ehlers, S. Jacobs, and R. Könighofer. How to handle assumptions in synthesis. In *Proc. of SYNT*, EPTCS 157, pages 34–50, 2014.
5. A. Brandenburger, A. Friedenberg, and H. J. Keisler. Admissibility in games. *Econometrica*, 76(2), 2008.
6. T. Brázdil, K. Chatterjee, V. Forejt, and A. Kucera. Trading performance for stability in Markov decision processes. In *Proc. of LICS*, pages 331–340. IEEE, 2013.
7. R. Brenguier, J.-F. Raskin, and O. Sankur. Assume-admissible synthesis. In *Proc. of CONCUR*, LIPIcs 42, pages 100–113. Schloss Dagstuhl–LZI, 2015.
8. R. Brenguier, J.-F. Raskin, and M. Sassolas. The complexity of admissibility in omega-regular games. In *Proc. of CSL-LICS*, pages 23:1–23:10. ACM, 2014.
9. T. Brihaye, V. Bruyère, N. Meunier, and J.-F. Raskin. Weak subgame perfect equilibria and their application to quantitative reachability. In *Proc. of CSL*, LIPIcs 41, pages 504–518. Schloss Dagstuhl - LZI, 2015.
10. L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, and J.-F. Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2011.
11. V. Bruyère, E. Filiot, M. Randour, and J.-F. Raskin. Expectations or guarantees? I want it all! A crossroad between games and MDPs. In *Proc. of SR*, EPTCS 146, pages 1–8, 2014.
12. V. Bruyère, E. Filiot, M. Randour, and J.-F. Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. In *Proc. of STACS*, LIPIcs 25, pages 199–213. Schloss Dagstuhl - LZI, 2014.
13. V. Bruyère, N. Meunier, and J.-F. Raskin. Secure equilibria in weighted games. In *Proc. of CSL-LICS*, pages 26:1–26:26. ACM, 2014.
14. K. Chatterjee, L. Doyen, E. Filiot, and J.-F. Raskin. Doomsday equilibria for omega-regular games. In *Proc. of VMCAI*, LNCS 8318, pages 78–97. Springer, 2014.
15. K. Chatterjee, L. Doyen, and T. A. Henzinger. Quantitative languages. *ACM Transactions on Computational Logic*, 11(4), 2010.
16. K. Chatterjee, L. Doyen, M. Randour, and J.-F. Raskin. Looking at mean-payoff and total-payoff through windows. *Information and Computation*, 242:25–52, 2015.

17. K. Chatterjee and T. A. Henzinger. Assume-guarantee synthesis. In *Proc. of TACAS*, LNCS 4424, pages 261–275. Springer, 2007.
18. K. Chatterjee, T. A. Henzinger, and M. Jurdziński. Games with secure equilibria. *Theoretical Computer Science*, 365(1):67–82, 2006.
19. K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. *Information and Computation*, 208(6):677–693, 2010.
20. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proc. of Logics of Programs*, LNCS 131, pages 52–71. Springer, 1981.
21. L. Clemente and J.-F. Raskin. Multidimensional beyond worst-case and almost-sure problems for mean-payoff objectives. In *Proc. of LICS*, pages 257–268. IEEE, 2015.
22. T. Colcombet. Forms of determinism for automata. In *Proc. of STACS*, LIPIcs 14, pages 1–23. Schloss Dagstuhl - LZI, 2012.
23. W. Damm and B. Finkbeiner. Does it pay to extend the perimeter of a world model? In *Proc. of FM*, LNCS 6664, pages 12–26. Springer, 2011.
24. A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.
25. M. Faella. Admissible strategies in infinite games over graphs. In *Proc. of MFCS*, LNCS 5734, pages 307–318. Springer, 2009.
26. J. Filar and K. Vrieze. *Competitive Markov decision processes*. Springer, 1997.
27. J. A. Filar, D. Krass, and K. W. Ross. Percentile performance criteria for limiting average Markov decision processes. *Transactions on Automatic Control*, 40:2–10, 1995.
28. E. Filiot, T. Le Gall, and J.-F. Raskin. Iterated regret minimization in game graphs. In *Proc. of MFCS*, LNCS 6281, pages 342–354. Springer, 2010.
29. D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *Proc. of TACAS*, LNCS 6015, pages 190–204. Springer, 2010.
30. J. Y. Halpern and R. Pass. Iterated regret minimization: A new solution concept. *Games and Economic Behavior*, 74(1):184–207, 2012.
31. T. A. Henzinger and N. Piterman. Solving games without determinization. In *Proc. of CSL*, LNCS 4207, pages 395–410. Springer, 2006.
32. P. Hunter, G. A. Pérez, and J.-F. Raskin. Reactive synthesis without regret. In *Proc. of CONCUR*, LIPIcs 42, pages 114–127. Schloss Dagstuhl - LZI, 2015.
33. L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43:204–233, 2008.
34. O. Kupferman, G. Perelli, and M. Y. Vardi. Synthesis with rational environments. In *Proc. of EUMAS*, LNCS 8953, pages 219–235. Springer, 2014.
35. S. Mannor and J. Tsitsiklis. Mean-variance optimization in Markov decision processes. In *Proc. of ICML*, pages 177–184. Omnipress, 2011.
36. F. Mogavero, A. Murano, and M. Y. Vardi. Reasoning about strategies. In *Proc. of FSTTCS*, LIPIcs 8, pages 133–144. Schloss Dagstuhl - LZI, 2010.
37. J. Nash. Equilibrium points in n-person games. *PNAS*, 36:48–49, 1950.
38. M. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. Wiley, 1st edition, 1994.
39. J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proc. of International Symposium on Programming*, LNCS 137, pages 337–351. Springer, 1982.

40. M. Randour, J.-F. Raskin, and O. Sankur. Percentile queries in multi-dimensional Markov decision processes. In *Proc. of CAV*, LNCS 9206, pages 123–139. Springer, 2015.
41. M. Randour, J.-F. Raskin, and O. Sankur. Variations on the stochastic shortest path problem. In *Proc. of VMCAI*, LNCS 8931, pages 1–18. Springer, 2015.
42. M. Ummels. Rational behaviour and strategy construction in infinite multiplayer games. In *Proc. of FSTTCS*, LNCS 4337, pages 212–223. Springer, 2006.
43. M. Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Proc. of FOCS*, pages 327–338. IEEE, 1985.
44. C. Wu and Y. Lin. Minimizing risk models in Markov decision processes with policies depending on target values. *Journal of Mathematical Analysis and Applications*, 231(1):47–67, 1999.
45. M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Proc. of NIPS*, pages 905–912, 2008.
46. U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1):343–359, 1996.