Admissibility Over Winning: A New Approach to Reactive Synthesis in Robotics

Karan Muvvala and Morteza Lahijanian

Abstract— **Reactive synthesis is a framework for modeling and automatically synthesizing strategies in robotics, typically through computing a *winning* strategy in a 2-player game between the robot and the environment. Winning strategies, however, do not always exist, even in some simple cases. In such situations, it is still desirable for the robot to attempt its task rather than "giving up". In this work, we explore the notion of admissibility to define strategies beyond winning, tailored specifically for robotic systems. We introduce an ordering of admissible strategies and define *admissibly rational strategies*, which aim to be winning and cooperative when possible, and non-violating and hopeful when necessary. We present an efficient synthesis algorithm and demonstrate that admissibly rational strategies produce desirable behaviors through case studies.**

I. Introduction

Reactive synthesis in robotics is the problem of automatically generating strategies for robotic systems to interact with a dynamic environment with the purpose of achieving a complex task expressed in a formal language. Its applications span various domains, from human-robot interaction [1]–[3] to autonomous navigation [4], [5], where robots face uncertainty and adversarial conditions. The interaction is modeled as a game between the robot and the environment, and the goal is to find a *winning strategy* for the robot which guarantees task achievement, regardless of the environment's actions. However, in complex scenarios, such strategies may not always exist, leading to robot's inability to take actions strategically. To address this, the concept of *best effort* [6] was introduced but appears too optimistic for robotics. Another approach, *admissibility* [7], [8], relaxes winning strategies, but existing work assumes rationality and known objectives for the environment, which is unrealistic for robotics. In this work, we aim to use the concept of admissibility to define strategies beyond winning, tailored specifically for robotic systems, and develop a reactive synthesis algorithm.

Consider the manipulation example in Fig. 1. The abstraction models a game between a robot (Sys player) and a human (Env player) interacting with objects in a shared workspace. The robot's task is to move a box to a goal location, but it lacks a winning strategy because the human can always intervene. Despite this, we still want the robot to make its best attempt to perform the task rather than *giving up*, which is a limitation of existing reactive synthesis methods.

To approach this problem, we draw inspiration from admissible strategies [7], [8]. These are the strategies that are not dominated by others based on an ordering, such as action costs. A desirable property of admissible strategies

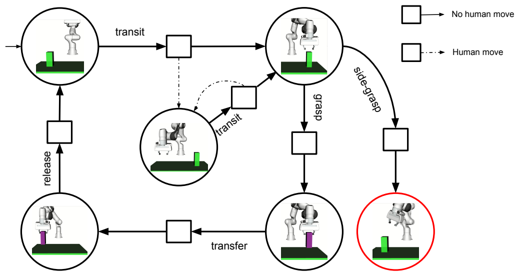

Fig. 1: Manipulation Domain Game [3]: Circle and Square states belong to the Sys and Env player, respectively.

is that they always exist. However, they are too broad, leading to large behavioral variations, not all of which are desirable for a given task. Our key insight is that, given a robotic task specification in *linear temporal logic over finite traces* [9], the objective in the game becomes reachability. This naturally partitions the state space intro three regions: winning, pending, and losing. In the winning region, we define admissible strategies that guarantee task completion while seeking cooperation. In the pending region, we combine safety and admissibility to ensure that the task cannot be violated while trying to reach a goal state. For cases where violation avoidance cannot be guaranteed, we introduce the notion of hopeful-admissible strategies, which assume that the environment player does not always take the worst possible actions. By integrating these concepts, we construct the notion of *admissibly rational strategies* and show that they always exist. Then, we introduce a polynomial-time algorithm for synthesis of these strategies. Our experiments demonstrate that admissibly rational strategies produce desirable behaviors.

In short, our contributions are fourfold. (i) We introduce admissibly rational strategies for robotic reactive synthesis, which like admissible strategies, always exist, and unlike admissible strategies, can be synthesized using fixed-point based approaches. (ii) We provide a synthesis algorithm that is sound and complete. (iii) We illustrate the emergent behavior in two case studies, a robotic manipulation domain and Tic-Tac-Toe, demonstrating that a robot playing admissibly rational exemplifies human-like behavior. Finally, (iv) we open-source our implementation [10].

Related Works: Several works explore alternatives to winning strategies [6]–[8], [11]–[13]. In formal methods, [11] investigates alternative concepts to winning strategies in qualitative games. They use the notion of admissibility to define the notion of *best-effort* (BE). Works [6], [12] focus on worst-case complexity and propose efficient synthesis algorithms. The objective of the Sys player is to reach a

Authors are with the Department of Aerospace Engineering Sciences at the University of Colorado Boulder, CO, USA

goal state and the payoff is binary - reach or not reach. In contrast, our work considers min-cost reachability games, where the robot aims to finish task with minimum total cost.

In [8], the notion of admissibility is considered in quantitative settings for normal-form games. The focus is on n-players, and the objective of each player is known a priori. Under the assumption that every player is playing admissibly, an admissible strategy is synthesized for all players. In [7], [13], [14], the notion of admissibility is explored for graph games with ω-regular objectives. In contrast, in our settings, we focus on objectives that can be accomplished in a finite time. Importantly, we make no assumptions about the Env player's objectives or require them to play admissibly.

Closest to our work is [9]. They extend prior works [6], [12] on BE synthesis to nondeterministic planning domains. The objective for Sys player is boolean - reach the goal state in finite time. We focus on quantitative reachability objectives that can be accomplished in finite time. Our work builds on the theoretical results of recent work on admissibility for quantitative games [15]. They analyze and give necessary and sufficient conditions for admissible strategies to exist. We identify and formalize a class of admissible strategies that are suitable for robotics. We focus on strategies that mitigate optimism and are efficient to synthesize.

II. Problem Formulation

The aim of this work is to enable a robot to do its *best* to achieve a complex task in the face of human interventions using formal reasoning and game theory. In this section, we formulate this problem by first introducing quantitative games as abstractions of such interactions, and a temporal logic specification language that allows expressing tasks formally.

A. Game Abstraction

Definition 1 (2-player Quantitative Game)**.** *A two-player turn-based quantitative game is a tuple $\mathcal{G} = (V = V_s \cup V_e, v_0, A = A_s \cup A_e, \Delta, C, \Pi, L)$, where*

- *V is a finite set of states partitioned into Sys player states V_s and Env player states V_e such that $V_s \cap V_e = \emptyset$,*
- *$v_0 \in V$ is the initial state,*
- *A is the finite set of actions, where $a_s \in A_s$ and $a_e \in A_e$ are Sys and Env player actions, respectively,*
- *$\Delta : V \times A \to V$ is the transition function such that, for every $(v_i, a_i) \in V_i \times A_i$, $\Delta(v_i, a_i) \in V_j$, where $i, j \in \{s, e\}$ and $i \neq j$,*
- *$C : V \times (A_s \cup A_e) \to \mathbb{N}^0$ is the cost function such that $C(v_s, a_s) > 0$ and $C(v_e, a_e) = 0$,*
- *$\Pi = \{\pi_0, \ldots, \pi_n\}$ is the finite set of task-related propositions that are true or false, and*
- *$L : V \to 2^{\Pi}$ is the labeling function that maps each state $v \in V$ to a set of atomic propositions $L(v) \subseteq \Pi$ that are true in v.*

Every state in the game captures the current configuration of the robot and objects in the workspace. In Fig. 1, each state can be uniquely identified by the location of the box and the end effector status. The edges correspond to physically viable actions from Sys player states. We note that such discrete abstractions are common in literature [2], [16]–[19] and can be constructed automatically for dynamical systems and robotic manipulators [20], [21].

Game $\mathcal{G}$, starting from the initial state v_0, is played in turns. We define a *play* to be a sequence of states $P^{v_0} := v_0 v_1 v_2 \ldots$, where, for all $i \geq 0$, $v_i \in V$. Further, we say a sequence is feasible if for every v_i in the play, $\exists a_i \in A$ s.t. $v_{i+1} = \Delta(v_i, a_i)$. A play can be either infinite, denoted by $P^{v_0} = v_0 v_1 \ldots \in V^{\omega}$ or finite $P^{v_0} = v_0 v_1 \ldots v_n \in V^*$. We generalize the notation by using P^v to denote plays that start from state $v \in V$. Finally, we assume game $\mathcal{G}$ is non-blocking, i.e., there exists at least one action from every state.

For $P^{v_0} = v_0 v_1 v_2 \ldots$, we define *trace* ρ to be the sequence of the labels of the states in P^{v_0}, i.e., $\rho := L(v_0)L(v_1)L(v_2)\ldots$ Intuitively, a trace captures the evolution of the game (robot's progress) with respect to a task. As game $\mathcal{G}$ is played between two strategic agents, the choice of action at each state is given by their respective strategies.

Definition 2 (Strategy)**.** *A Sys player strategy $\sigma : V^* V_s \to A_s$ is a function that maps a play that ends in V_s to an action in A_s. Similarly, an Env player strategy is $\tau : V^* V_e \to A_e$. The set of all Sys and Env player strategies are denoted by Σ and T, respectively.*

Strategies that depend only on the last state of plays are called *memoryless*, i.e., for every $P^v \in V^* V_s$, $\sigma(P^v) = \sigma(\mathrm{last}(P^v))$, where $\mathrm{last}(P^v) \in V_s$ is the last element of P^v. Strategies that are not memoryless are called *finite-memory*.

Given σ and τ, a unique play $P^{v_0}(\sigma, \tau) = v_0 v_1 \ldots$ is induced such that, for every $i \geq 0$, $v_{i+1} = \Delta(v_i, a_i)$, where $a_i = \sigma(v_0 v_1 \ldots v_i)$ if $v_i \in V_s$, otherwise $a_i = \tau(v_0 v_1 \ldots v_i)$. We say σ is *compatible* with a play P'^v if $\exists \tau \in \mathrm{T}$ such that $P'^v = P^v(\sigma, \tau)$. For every play on $\mathcal{G}$, there is a cost (aka payoff) associated with that play.

Definition 3 (Total-Payoff [3])**.** *Given strategies σ and τ, the* payoff *of play $P^{v_0}(\sigma, \tau)$ is* $\mathrm{Val}(P^{v_0}(\sigma, \tau)) := \sum_{i=0}^{n-1} C(v_i, a_i)$, *where $n = |P^{v_0}(\sigma, \tau)|$ is the length of $P^{v_0}(\sigma, \tau)$, and $a_i = \sigma(v_0 v_1 \ldots v_i)$ if $v_i \in V_s$, else $a_i = \tau(v_0 v_1 \ldots v_i)$.*

Note that payoff is finite for finite plays; otherwise it is infinite. Also, since the cost of human actions are always zero (see Def. 1), Val captures the cost (energy) spent by the robot.

B. Robotic Tasks in LTL_f

For tasks achievable in finite time, we use *Linear Temporal Logic over finite traces* (LTL_f) [22], [23]. An LTL_f formula φ is defined over atomic propositions in Π with the syntax:

$$\varphi := \pi \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi$$

where $\neg$ and $\vee$ are the boolean *negation* and *disjunction* operators, and $\bigcirc$ and $\mathcal{U}$ are the temporal *next* and *until* operators, respectively. We can derive popular temporal operators *eventually* (F) and *always* (G) as $F\varphi \equiv \mathit{true}\,U\varphi$ and $G\varphi \equiv \neg F \neg \varphi$. The semantics of LTL_f is defined over finite traces in $(2^{\Pi})^*$ (see [24] for more details). We say play

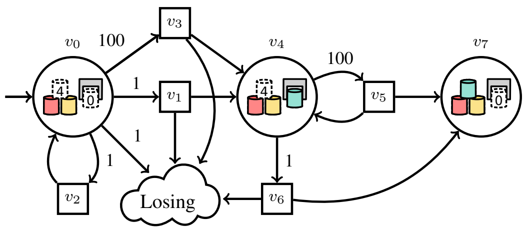

Fig. 2: Manipulation Domain: No path exists from Losing to v_i; $a_{ij} : v_i \to v_j$ and $a_{iL} : v_i \to$ Losing. Actions a_{01} and a_{03}: transit-grasp blue box a_{02}: do-nothing, and a_{0L}: give-up.

$P^{v_0}(\sigma, \tau)$ satisfies formula φ, denoted by $P^{v_0}(\sigma, \tau) \models \varphi$, if the trace induced by $P^{v_0}(\sigma, \tau)$ satisfies φ.

Example 1. *Consider the following example from [25]: "Robot must put pitcher in the dishwasher with lid on, then either place it in drying-rack or dry it with the towel and return it to the original location." It translates to formula: $\varphi = F((p_{dw} \wedge p_{lid}) \wedge F(p_{dr} \vee p_{towel} \wedge \bigcirc F(p_{l0})))$, where the atomic propositions are: p_{dw} (dishwasher), p_{dr} (lid on), p_{towel} (towel-dried), p_{l0} (original location).*

Given task φ, the robot should choose a strategy that results in a φ-satisfying play. Strategy synthesis is the process of synthesizing a *winning* (enforcing) strategy σ such that $P^{v_0}(\sigma, \tau) \models \varphi$ for every $\tau \in \mathrm{T}$. However, such a strategy often does not exist, even in simple cases. Thus, we explore what strategies are apt when winning strategies do not exist.

C. Beyond Winning Strategies: Admissible Strategies

While a winning strategy ensures task completion under every Env player strategy, such strategies do not always exist. Recall the example in Fig. 1, where no winning strategy exists but it is intuitive for the robot to at least *try*. To achieve a desirable behavior, we use the notion of *dominance*, which enables a ranking of strategies according to total-payoff.

Definition 4 (Dominance [8], [26]). *We say strategy σ_1* very weakly dominates *σ_2, denoted $\sigma_1 \succeq \sigma_2$, if σ_1 always does as well as σ_2, i.e.,* $\mathrm{Val}(P^v(\sigma_1, \tau)) \leq \mathrm{Val}(P^v(\sigma_2, \tau))$ *for all $\tau \in \mathrm{T}$. Furthermore, strategy σ_1* weakly dominates *σ_2, denoted $\sigma_1 \succ \sigma_2$, if (i) it very weakly dominates σ_2 ($\sigma_1 \succeq \sigma_2$), and (ii) there exists at least one play under σ_1 that does strictly better, i.e., $\exists \tau \in \mathrm{T}$ s.t. $\mathrm{Val}(P^v(\sigma_1, \tau)) < \mathrm{Val}(P^v(\sigma_2, \tau))$.*

Intuitively, playing a dominated strategy is sub-optimal as Sys player could have done better.

Example 2. *In Fig. 1, say robot task is: grab a box. Let σ_1 be transit from initial state and grasp and σ_2 be same as σ_1 except the robot grasps from side. Here $\sigma_1 \succ \sigma_2$ as σ_1 does as well as σ_2 and sometimes better. Under σ_2,* $\mathrm{Val}(P^{v_0}(\sigma_2, \tau)) = \infty \ \forall \tau \in \mathrm{T}$. *But, $\exists \tau \in \mathrm{T}$ such that* $\mathrm{Val}(P^{v_0}(\sigma_1, \tau)) < \infty$ *for σ_1.*

We say a strategic agent must always play non-dominated (aka admissible) strategies.

Definition 5 (Admissible Strategy [8]). *A Sys player strategy $\sigma_{\mathbf{Adm}} \in \Sigma$ is called* admissible *(**Adm**) if it is not* weakly dominated *by any other Sys player strategy.*

Admissible strategies are attractive as they always exist [14], [15]. Prior approaches that explore admissibility in game theory assume that both players have their respective objectives and are playing admissibly [8], [27]. We relax this strong assumption (i.e., knowledge of human's objective) while still viewing human as a strategic agent. The human can behave adversarially, cooperatively, or somewhere in between. In our setting, we desire the robot to play admissibly and strategically choose actions towards accomplishing its task while still allowing room for cooperation whenever possible.

Example 3 (Admissibly Rational). *In Fig. 2, the robot is tasked with building an arch (v_7). Let $\sigma_1(\ldots v_0) = a_{0L}$ and $\sigma_2(\ldots v_0) = a_{03}$. Under σ_1, robot is* not *doing its best and hence is not admissible. While σ_2 is admissible, it is not a strategic choice. A rational strategy would be $\sigma_3(v_0) = a_{01}$ and $\sigma_3(v_4) = a_{45}$, i.e., commit to a_{01} and a_{45} as they ensure there always exists path to v_7 and open for cooperation.*

We define an admissible strategy as *rational* if it ensures task completion when possible, while actively seeking cooperation. In cases where task completion cannot be guaranteed, an *admissibly rational strategy* should at least ensure no violation to the task and the existence of a satisfying play. If even this is not possible, the robot should adopt a *hopeful* approach, which assumes that the Env player does not consistently take the worst actions. These are called *hopeless* strategies. Thus, playing hopefully means the robot assumes the Env player is not behaving hopelessly, and in response, the robot selects admissible strategies optimal under the worst-case scenario. In this work, we seek such a robot strategy.

Problem 1. *Given robot game abstraction $\mathcal{G}$ and an LTL_f task formula φ, synthesize an admissibly rational strategy σ, which adheres to the following prioritized objectives: (O1) achieves φ while seeking cooperation, (O2) if (O1) is not possible, ensures no violation to φ (safety) with the possibility of achieving φ, and (O3) if neither (O1) nor (O2) are possible, optimize for the worst case payoff under the assumption that the Env player is not acting hopelessly.*

Below, we define two notions of admissibility that formalize the desired characteristics of an admissibly rational strategy. We give synthesis algorithms to solve Problem 1, and show that the algorithms are sound and correct. All the proofs are in provided in the extended version [28].

III. Admissibly-Rational DFA Games

The standard strategy synthesis involves three steps: translation of the specification into a Deterministic Finite Automaton (DFA), composition of the abstraction with the automaton (DFA Game), and computation of a strategy. We first briefly show how to construct the DFA game and then focus on how to compute admissibly rational strategies on this game.

A. Overview of DFA Games

For every LTL_f formula φ, a DFA $\mathcal{A}_\varphi$ can be constructed that accepts precisely the traces that satisfy φ [9]. DFA $\mathcal{A}_\varphi$ is a tuple $\mathcal{A}_\varphi = (q_0, Q, \Gamma, \delta_\varphi, Q_f)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $\Gamma = 2^\Pi$ is a set of symbols, $\delta_\varphi : Q \times \Gamma \to Q$ is the deterministic transition function, and $Q_f \subseteq Q$ is a set of accepting states. A trace $\rho \in \Gamma^*$ is a *finite* sequence of symbols in Γ. Trace $\rho = \rho_1\rho_2 \dots \rho_n$ induces a run $q_0 q_1 \dots q_n$, where $q_{i+1} = \delta_\varphi(q_i, \rho_{i+1})$ for very $0 \le i < n$. Trace ρ is accepted if the last state of its induced run is in Q_f. DFA $\mathcal{A}_\varphi$ captures all possible traces that satisfy φ, called the *language* of φ or $\mathcal{A}_\varphi$ [22], [24].

Composing $\mathcal{A}_\varphi$ with $\mathcal{G}$ results in a DFA Game that models all possible ways a robot can interact with the human while simultaneously capturing the status of the task completion.

DFA Game: A *DFA Game* is defined as $\mathcal{P} = \mathcal{G} \otimes \mathcal{A}_\varphi$ where $\otimes$ is the composition operator. The resulting product is a game $\mathcal{P} = (S = V \times Q, s_0, A = A_s \cup A_e, \delta_\mathcal{P}, C, S_f)$, where A, A_s and A_e are the same as Def. 1. S is the set of states partitioned into Sys $S_s = V_s \times Q$ and Env $S_e = V_e \times Q$ states, and initial state $s_0 = (v_0, \delta_\varphi(q_0, L(v_0)))$. $\delta_\mathcal{P} : S \times A \to S$ is the transition function such that, for $s = (v, q)$ and $s' = (v', q')$, $s' = \delta_\mathcal{P}(s, a)$ if $v' = \Delta(v, a)$ and $q' = \delta_\varphi(q, L(v))$. The cost function C is defined based on $\mathcal{G}$ as $C((v, q), a) = C(v, a)$, and $S_f = V \times Q_f$ is the set of accepting states.

The notions of strategy, play, and payoff extend to $\mathcal{P}$ from $\mathcal{G}$. A play in $\mathcal{P}$ terminates only when it reaches S_f. Since a feasible play on $\mathcal{P}$ is constructed from $\delta_\mathcal{P}$, its projection onto DFA $\mathcal{A}_\varphi$ is a valid run of $\mathcal{A}_\varphi$, and its projection onto $\mathcal{G}$ is a feasible play. Therefore, the problem of computing a winning strategy for task φ on $\mathcal{G}$ reduces to finding a Sys strategy on $\mathcal{P}$ that guarantees every play reaches an accepting state in S_f under every Env strategy [20].

Next, we define two classical notions in game theory for a reachability objective and show how their synthesis is a central element in synthesizing admissibly rational strategies.

B. Cooperative and Adversarial Games

As LTL$_f$ specifications can be satisfied in finite time, all plays induced by a winning strategy σ are finite, and hence their corresponding payoff is finite, i.e., $\mathrm{Val}(P^{s_0}(\sigma, \tau)) < \infty \ \forall \tau \in \mathrm{T}$. We define an optimal winning strategy as a winning strategy that achieves the best worst-case outcome.

Definition 6 (Optimal Winning Strategy)**.** *A Sys player strategy $\sigma^* \in \Sigma$ at state $s \in S$ is optimal winning if it is (i) winning, i.e., $\mathrm{Val}(P^s(\sigma^*, \tau)) < \infty$ for all $\tau \in \mathrm{T}$, and (ii) has the lowest worst-case payoff value, i.e., $\sigma^* = \arg\min_{\sigma \in \Sigma} \max_{\tau \in \mathrm{T}} \mathrm{Val}(P^s(\sigma, \tau))$. The value secured by σ^*, i.e., $\max_{\tau \in \mathrm{T}} \mathrm{Val}(P^s(\sigma^*, \tau))$, is called the* worst-case optimal (**WCO**) *value.*

In a cooperative game, the reachability objective of the Sys player and Env player are the same.

Definition 7 (Optimal Cooperative Strategy)**.** *An optimal cooperative strategy σ^* from state s is an optimal strategy when both players are playing cooperatively, i.e., $\sigma^* = \arg\min_{\sigma \in \Sigma} \min_{\tau \in \mathrm{T}} \mathrm{Val}(P^s(\sigma, \tau))$. The value $\min_{\tau \in \mathrm{T}} \mathrm{Val}(P^s(\sigma^*, \tau))$ is called the cooperative optimal* (**Co-Op**) *value.*

We denote by cVal^s and aVal^s the optimal cooperative and adversarial (worst-case) values for state s, respectively. For strategy σ, the cooperative and adversarial values are defined as $\mathrm{cVal}^s(\sigma) = \min_{\tau \in \mathrm{T}} \mathrm{Val}(P^s(\sigma, \tau))$ and $\mathrm{aVal}^s(\sigma) = \max_{\tau \in \mathrm{T}} \mathrm{Val}(P^s(\sigma, \tau))$. The optimal state values naturally partitions S into three subsets of *winning region S_{win}*, *losing region S_{los}*, and *pending region S_{pen}*, where

$$S_i = \begin{cases} \{s \in S : \mathrm{aVal}^s < \infty\}, & \text{if } i = win \\ \{s \in S : \mathrm{aVal}^s = \mathrm{cVal}^s = \infty\}, & \text{if } i = los \\ \{s \in S : \mathrm{aVal}^s = \infty, \mathrm{cVal}^s < \infty\}, & \text{if } i = pen \end{cases}$$

Note that these regions are disjoint sets such that their union is the set of all states S, and their pairwise intersection is empty. Next, we formally define strategies that capture the desirable characteristics for a strategy to be admissibly rational.

Algorithm 1: Cooperative Game

```
   Input  : DFA Game $\mathcal{P}$
   Output : cVal, $\Sigma_{ra}$, $\Sigma_{\mathbf{Co\text{-}Op}}$
 1 forall $s \in S$ do $W(s), W'(s) \leftarrow \infty$; $\Sigma_{\mathbf{Co\text{-}Op}}(s) \leftarrow \emptyset$;
 2 forall $s \in S_f$ do $W(s) \leftarrow 0$;
 3 while $W' \neq W$ do
 4     $W' = W$ //$s'$ is the successor state
 5     $W(s) = \min(C(s,a) + W'(s'))\quad \forall s \in S \setminus S_f$
 6     $\Sigma_{\mathbf{Co\text{-}Op}}(s) =$
         $\arg\min_a (C(s,a) + W'(s'))\quad \forall s \in S_s \setminus S_f$
 7 forall $s \in S_s \setminus S_f$ do
 8     if $W(s') < W(s)$ then
 9     $\Sigma_{ra}(s) = \Sigma_{ra}(s) \cup \{a : s' = \delta(s,a)\}$;
10 return cVal := $W$, $\Sigma_{ra}$, $\Sigma_{\mathbf{Co\text{-}Op}}$
```

C. Admissibly-Rational: Winning whenever possible

We know that a winning strategy has the desirable property of enforcing a visit to S_f from all states in the winning region. Unfortunately, not every winning strategy is admissible. From [15, Lemma 3], a subset of winning strategies, precisely Worst-case Cooperative Optimal (**WCo-Op**), are always admissible from the states in S_{win}. A strategy σ is **WCo-Op** if it is worst-case optimal and has the lowest cVal among all such strategies, i.e., if the Env player cooperates, this is the most optimal strategy among all winning strategies.

Definition 8 (**WCo-Op**)**.** *Strategy σ is Worst-case Cooperative Optimal if, for all Sys player states $s \in S$, $\mathrm{aVal}^s(\sigma) = \mathrm{aVal}^s < \infty$ and $\mathrm{cVal}^s(\sigma) = \min\{\mathrm{cVal}^s(\sigma) \mid \sigma \in \Sigma_{\mathbf{WCO}}\}$, where $\Sigma_{\mathbf{WCO}}$ is the set of all* **WCO** *strategies with finite adversarial value.*

Next, for the states that do not belong to the winning region, we synthesize an admissible strategy where Sys plays safely, i.e., guarantees no violation of φ.

D. Admissibly-Rational: Safe whenever you can

At every Sys player state in the pending region S_{pen}, every action is admissible as shown by the following lemma.

Lemma 1. *Given DFA game $\mathcal{P}$, every action $a_s \in A_s$ at every state $s \in S_{pen} \cap S_s$ belongs to a* finite-memory *admissible strategy if $\delta_\mathcal{P}(s, a_s) \in S_{pen}$.*

Algorithm 2: Admissibly Rational Synthesis

```
Input  : DFA Game $\mathcal{P}$
Output : $\Sigma_{\textbf{Adm-Rat}}$ strategy
1 $\mathrm{cVal}, \Sigma_{ra}, \Sigma_{\textbf{Co-Op}} \leftarrow$ CooperativeGame($\mathcal{P}$)
2 $\Sigma_{\textbf{WCO}} \leftarrow$ AdversarialGame($\mathcal{P}$)
3 $S_{pen}, S_{win}, S_{los} \leftarrow$ As per Sec. III-B
4 if $s_0 \in S_{win}$ then return $\Sigma_{\textbf{WCo-Op}} \leftarrow$ Def. 8;
5 if $s_0 \in S_{los}$ then return $\Sigma$;
6 $\Sigma_{\textbf{SAdm}} \leftarrow$ SafeAdmGame($\mathcal{P}, \Sigma_{ra}, S_{los}, S_{pen}$)
7 if $\Sigma_{\textbf{SAdm}}(s) \neq \emptyset \ \forall s \in S_{pen}$ then return
    $\Sigma_{\textbf{Adm-Rat}} \leftarrow \Sigma_{\textbf{WCo-Op}} \circ \Sigma_{\textbf{SAdm}}$;
  /* HAdm strategies */
8 $\Sigma_{\textbf{HAdm}} \leftarrow$ AdversarialGame($\mathcal{P}^{\Sigma_{\textbf{Adm}}, \mathrm{T}_{\mathrm{HF}}}$)
9 return $\Sigma_{\textbf{Adm-Rat}} \leftarrow \Sigma_{\textbf{WCo-Op}} \circ \Sigma_{\textbf{SAdm}} \circ \Sigma_{\textbf{HAdm}}$
```

We say a strategy is *safe*, denoted by σ_{Safe}, if every play under σ visits only states in $S_{pen} \cup S_{win}$. Intuitively, such a strategy ensures that φ is not violated. For example, in Fig. 2, σ_{Safe} at v_0 and v_4 chooses a_{02} and a_{45}, respectively. However, these strategies do not guarantee the possibility of reaching S_f. To ensure the robot could progress towards S_f while being safe, we define safe admissible (**SAdm**) strategies.

Definition 9 (**SAdm**)**.** *For all $s \in S$, a strategy is Safe Admissible, denoted by $\sigma_{\textbf{SAdm}}$, if it is safe and admissible, and at least one play under $\sigma_{\textbf{SAdm}}$ reaches S_f, i.e., $\exists \tau \in \mathrm{T}$ and $\exists s'$ in $P^s(\sigma_{\textbf{SAdm}}, \tau)$ s.t. $s' \in S_f$, and $\forall \tau \in \mathrm{T}$ and $\forall s'$ in $P^s(\sigma_{\textbf{SAdm}}, \tau)$, $s' \notin S_{los}$.*

The following lemma shows that, unlike admissible strategies, a **SAdm** strategy does not always exist.

Lemma 2. *Given DFA Game $\mathcal{P}$, for all $s \in S_{pen}$, a safe-admissible strategy does not always exist.*

Consider the example in Fig. 2 again. While a σ_{Safe} exists, a **SAdm** strategy does not since at v_0 the only safe action is a_{02} but it does not allow a path to $S_f = \{v_7\}$. The synthesis of **SAdm** strategies can be done in two stages (see Sec. IV).

E. Admissibly-Rational: Hopeful with Env assumption

In cases that **SAdm** strategies do not exist, we are left with few choices. To ensure the robot still attempts to achieve its task, we synthesize worst-case optimal strategies by assuming that the Env player does *not* choose a hopeless strategy.

Definition 10 (Hopeless strategy)**.** *Given DFA game $\mathcal{P}$, for all $s \in S$, a strategy $\tau_{HL} \in \mathrm{T}$ is* hopeless *if under every $\sigma \in \Sigma$, $\exists s'$ in $P^s(\sigma, \tau_{HL})$ such that $s' \in S_{los}$.*

An Env strategy that is not hopeless is called *hopeful*, denoted by by τ_{HF}. T_{HF} is the set of all such strategies.

Definition 11 (**HAdm**)**.** *A strategy $\sigma_{\textbf{HAdm}} \in \Sigma$ is* hopeful admissible, *if it is (i) admissible and (ii) worst-case optimal under hopeful Env strategies in T_{HF}, i.e, $\sigma_{\textbf{HAdm}} = \arg\min_{\sigma \in \Sigma_{\textbf{Adm}}} \max_{\tau \in \mathrm{T}_{HF}} \mathrm{Val}(P^s(\sigma, \tau))$.*

HAdm strategies are attractive because they always exist.

Lemma 3. *Given DFA Game $\mathcal{P}$, a hopeful-admissible (**HAdm**) strategy always exists.*

Thus, using the notion of **WCo-Op**, **SAdm**, and **HAdm**, we can construct an admissibly rational (**Adm-Rat**) strategy.

Definition 12 (**Adm-Rat**)**.** *For DFA Game $\mathcal{P}$, for all $s \in S$, an admissibly rational strategy **Adm-Rat** is defined as*

$$\sigma_{\textbf{Adm-Rat}}(s) = \begin{cases} \sigma_{\textbf{WCo-Op}}(s) & \text{if } s \in S_{win} \\ \sigma_{\textbf{SAdm}}(s) & \text{if } s \in S_{pen} \wedge \\ & \quad \mathrm{cVal}^s(\sigma_{\textbf{SAdm}}) < \infty \\ \sigma_{\textbf{HAdm}}(s) & \text{otherwise.} \end{cases}$$

By definition, every action at every state in the losing region S_{los} is part of an admissible strategy and is worst-case optimal. From Lemma 3 and [29, Proposition 19], an **Adm-Rat** strategy always exists. Further, using [29, Thm. 1], we can show that optimal winning and cooperative strategies always exist and a witnessing strategy can always be synthesized.

Corollary 1. ***Adm-Rat** always exists, and $\sigma_{\textbf{Adm-Rat}}$ can always be computed.*

IV. Synthesis Framework

Here, we present an algorithm for synthesis of $\sigma_{\textbf{Adm-Rat}}$. The algorithm is based on four fixed-point computations, one for each $\sigma_{\textbf{WCO}}, \sigma_{\textbf{Co-Op}}, \sigma_{\textbf{SAdm}}, \sigma_{\textbf{HAdm}}$. Specifically, Alg. 1 is a Value Iteration based algorithm for Cooperative Games, and Alg. 2 is for the computation of **Adm-Rat** strategies.

Alg. 1 computes optimal cVal and optimal strategy from every state in $\mathcal{P}$ [29]. Lines 3-5 correspond to fixed-point computation when the Env player is cooperative. The While loop terminates when the values of the states have converged, i.e., $W' = W$, where W is a state vector with optimal cVals for every state. For synthesis of (reachable) admissible strategies, under which at least one play can reach S_f, we iterate through every state s and choose actions corresponding to successor states s' such that $\mathrm{cVal}^{s'} < \mathrm{cVal}^s$. As $C(s, a_s) > 0$, at least one successor state always exists. Intuitively, $\mathrm{cVal}^{s'} < \mathrm{cVal}^s$ implies that s' is "closer" to a goal state. Thus, any play induced by reachable admissible strategy (ra) gets closer to S_f at every step [30]. Thus, synthesis of ra strategies can be combined with cVal computation with minimal overhead.

In Alg. 2, we first play Cooperative and Adversarial Game and compute $\sigma_{\textbf{WCO}}$ and $\sigma_{\textbf{Co-Op}}$ along with Σ_{ra}. For the Adversarial Game, $W(s)$ computation can be split into min and max operation at Sys and Env player states, respectively (See [29] for details). If $s_0 \in S_{win}$, we compute $\sigma_{\textbf{WCo-Op}}$ as per Def. 8 and return it. If $s_0 \in S_{los}$, then all actions are admissibly rational and belong to $\sigma_{\textbf{HAdm}}$.

To compute $\sigma_{\textbf{SAdm}}$, we first compute Σ_{Safe} by playing a Safety game. Synthesis of Σ_{Safe} is a greatest fixed-point operation, for which efficient algorithms exist [24]. Note that Σ_{Safe} and Σ_{ra}, respectively, compute the the set of all Safe and reachable admissible actions at state $s \in S_{pen} \cap S_s$. Finally, $\sigma_{\textbf{SAdm}}(s) = \Sigma_{ra}(s) \cap \Sigma_{\mathrm{Safe}}(s)$ for all $s \in S_{pen}$. If $\sigma_{\textbf{SAdm}}(s) = \emptyset$, we compute $\sigma_{\textbf{HAdm}}$. We first compute the minimal set of Env player actions A_{HL} such that restricting the Env to $a \in A_e \setminus A_{\mathrm{HL}}$ gives a maximal T_{HF}. See the extended version for more details [28]. Further, we remove Sys player actions a_s s.t. $\delta_{\mathcal{P}}(s, a_s) \in S_{los}$ (as they are not

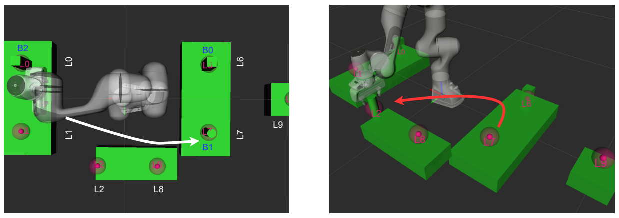

Fig. 3: Initial setup (Left). There exists **HAdm** from s_0 in white and **SAdm** in red after grasping. See suppl. video.

TABLE I: Abs. size & Syn. time (s): averaged over 10 runs.

	$\lvert V\rvert$	$\lvert\Delta\rvert$	$\lvert S\rvert$	$\lvert\delta_\varphi\rvert$	**WCo-Op**	**Co-Op**	**SAdm**	**HAdm**
PP Dom.	11272	47220	22545	94297	13.77	24.48	0.29	13.52
TTT(φ_1)	5478	16167	5479	17038	5.63	6.13	0.07	5.63
TTT(φ_2)	5748	16167	5479	17038	5.94	5.97	–	–

admissible) and play an adversarial game to compute $\sigma_{\mathbf{HAdm}}$. We compose these strategies per Def. 12 to obtain $\sigma_{\mathbf{Adm\text{-}Rat}}$.

Theorem 1 (Sound and Complete)**.** *Alg. 2 returns the set of all* **Adm-Rat** *strategies and has polynomial time complexity.*

V. Experiments

We illustrate the emergent behavior when the robot plays admissibly rational on two case studies. Table I reports the size of $\mathcal{G}$, $\mathcal{P}$ and synthesis times for various strategies. Our synthesis tool is publicly available on Github [10]. See supplementary material for more experiments.

Pick-and-Place Domain [3]**:** Consider the manipulation domain in Fig. 3. There are three objects, B_0, B_1, B_2. The human can intervene and move objects within region $R_1 = \{L_0, L_1, L_2\}$ and $R_2 = \{L_6, L_7, L_8, L_9\}$. Location L_9 is out of robot's reach. Once the human moves an object to L_9, it cannot be returned. The task is to dry B_1 in location L_2 or L_8 then place B_1 at L_6 and B_0 at L_7 or B_1 at L_1 and B_2 at L_0. The LTL$_f$ formula is $\varphi = F((p_{18} \vee p_{12}) \wedge \bigcirc F((p_{20} \wedge p_{11}) \vee (p_{16} \wedge p_{07})))$ where p_{ij} is B_i placed at L_j.

Using our tool, we synthesized a $\sigma_{\mathbf{Adm\text{-}Rat}}$. Note that a winning strategy does not exist as the human can always intervene. The initial state belongs to the pending region and $\sigma_{\mathbf{SAdm}}(s_0) = \emptyset$ as the human has a strategy to move B_1 to L_9. Hence, the robot initially plays hopefully ($\sigma_{\mathbf{Adm\text{-}Rat}}(s_0) = \sigma_{\mathbf{HAdm}}(s_0)$) and transits to B_1 and grasps it. Once the grasp is complete, a safe-admissible strategy exists (i.e., $\sigma_{\mathbf{Adm\text{-}Rat}} = \sigma_{\mathbf{SAdm}}$), and hence the robot can guarantee no violation to φ thereafter. Next, the robot moves B_1 to L_2. This ensures that a path always exists to a goal state. From here on, the robot seeks an opportunity to complete the task. We note that $\sigma_{\mathbf{HAdm}}(s_0)$ also includes the action of going to B_0, which is optimistic that the human player will move B_1 to L_8.

This case study shows that **Adm-Rat** strategies result in reasonable (and arguably human-like) behaviors when winning strategies do not exist. Specifically, **Adm-Rat** is most effective when a **SAdm** strategy exists. However, if **Adm-Rat** must resort to **HAdm** due to the absence of a **SAdm** strategy, overly optimistic behaviors may occur.

Tic-Tac-Toe (TTT) **from [31]:** Unlike the manipulator domain, TTT is naturally competitive. Additionally, TTT's game arena is a tree-like structure. Thus, the Sys player

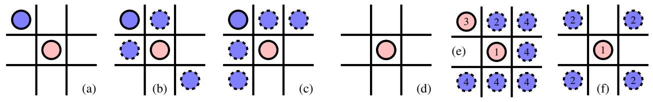

Fig. 4: Blue and Pink are Sys and Env actions. (a-d) current status, (b-e) and (c-f) illustrate $\sigma_{\mathbf{Adm\text{-}Rat}}$ for φ_1, φ_2. (a-c): Sys plays first; (d-f): Env plays first. (e-f): The numbers indicate time step.

cannot exploit cycles to "stay" in the pending region. We consider two specifications: "eventually win" ($\varphi_1 = F\,\text{win}$), and "eventually win or draw" ($\varphi_2 = F(\text{win} \vee \text{draw})$).

$\boldsymbol{\varphi_1 = F}$ **win.** For this task, no winning strategy exists from s_0. Although Safe strategy exists, **SAdm** does not exist from *any* state since a "draw" state is a violation of φ_1. But, $\sigma_{\mathbf{HAdm}}$ from s_0 exists. Fig. 4(a) is the game status when Sys goes first and (b) shows $\sigma_{\mathbf{HAdm}}$ actions. Under $\sigma_{\mathbf{HAdm}}$, Sys is choosing **WCO** while being hopeful - the robot markers are *strategically* placed to win the game. The aVal of all states reachable under $\sigma_{\mathbf{HAdm}}$ in the hopeful game are finite, and thus σ is **WCO** and taking longer paths which eventually end in "draw" states. Thus, under $\sigma_{\mathbf{HAdm}}$, Sys *never* reaches a "lose" state. When Env goes first (Fig. 4(d)), Sys is overly optimistic. The aVal of some states reachable under $\sigma_{\mathbf{HAdm}}$ is ∞. Thus, for states with $\text{aVal}^s = \infty$, $\sigma_{\mathbf{HAdm}}$ choose actions indifferently. In 4(e) Env places a marker in the middle. For Sys, almost all available moves are **HAdm** and cannot enforce reaching "win". Thus, Sys is optimistic and hopes the Env will make a mistake.

$\boldsymbol{\varphi_2 = F}(\text{win} \vee \text{draw})$**.** For this task, a $\sigma_{\mathbf{WCo\text{-}Op}}$ strategy exists from the initial state. The Sys player under **WCo-Op** can always enforce at least a "draw" and particularly chooses actions that could win (**Co-Op**) as quickly as possible (Fig. 4(c)). A Winning strategy ensures always drawing and does *not* strategically choose to reach "win" states as these are *not* **Co-Op**. Thus, in Fig. 4(c), Sys does place its marker on the bottom right. When Env player goes first (Fig. 4(d)), Sys is trying to force Env to make a mistake so as as to enforce drawing while the **Co-Op** attribute will try to win if possible. In Fig. 4(f), Sys places its marker in one of the corners. This allows room for error for Env. Thus, if Env makes mistakes, Sys wins the game else the game draws. The choice of the specification dictates what types of behavior Sys player exhibits and the level of optimism.

Specifications that are too strict (φ_1) may exhibit either strategic or too optimistic behaviors. More flexible specifications (φ_2) can lead to more balanced and realistic strategies. Thus, **Adm-Rat** captures nuanced strategic behaviors, such as forcing opponents into mistakes or playing defensively while looking for opportunities to win.

VI. Conclusion

This paper explores what the robot should do when a winning strategy does not exist. We define admissibly rational strategies and provide algorithms for synthesis. Unlike the prior approach, synthesis can be reduced to classical fixed-point based computation. We discuss the emergent behavior in competitive and non-competitive games. Future work should explore mitigating optimism for the robot in the pending regions and explore symbolic variants for faster computation.

References

[1] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Reactive synthesis for finite tasks under resource constraints," in *Int. Conf. on Intel. Robots and Sys.*, 2017, pp. 5326–5332.

[2] H. Kress-Gazit, M. Lahijanian, and V. Raman, "Synthesis for robots: Guarantees and feedback for robot behavior," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 211–236, 2018. [Online]. Available: https://doi.org/10.1146/annurev-control-060117-104838

[3] K. Muvvala, P. Amorese, and M. Lahijanian, "Let's collaborate: Regret-based reactive synthesis for robotic manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4340–4346.

[4] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE transactions on robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[5] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 55–64, 2011.

[6] B. Aminof, G. De Giacomo, R. Sasha *et al.*, "Best-effort synthesis: Doing your best is not harder than giving up," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021*, 2021.

[7] D. Berwanger, "Admissibility in infinite games," in *Annual Symposium on Theoretical Aspects of Computer Science*. Springer, 2007, pp. 188–199.

[8] A. Brandenburger, A. Friedenberg, and H. J. Keisler, "Admissibility in games 1," *Econometrica*, vol. 76, no. 2, pp. 307–352, 2008.

[9] G. De Giacomo, G. Parretti, and S. Zhu, "Ltl$_f$ best-effort synthesis in nondeterministic planning domains," in *ECAI 2023 - 26th European Conference on Artificial Intelligence*, ser. Frontiers in Artificial Intelligence and Applications, K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein, and R. Radulescu, Eds., vol. 372. IOS Press, 2023, pp. 533–540. [Online]. Available: https://doi.org/10.3233/FAIA230313

[10] K. Muvvala, "Adm tool," https://github.com/aria-systems-group/PDDLtoSim, Sep 2024.

[11] M. Faella, "Admissible strategies in infinite games over graphs," in *International Symposium on Mathematical Foundations of Computer Science*. Springer, 2009, pp. 307–318.

[12] G. De Giacomo, G. Parretti, and S. Zhu, "Symbolic ltl$_f$ best-effort synthesis," in *European Conference on Multi-Agent Systems*. Springer, 2023, pp. 228–243.

[13] R. Brenguier, J.-F. Raskin, and M. Sassolas, "The complexity of admissibility in omega-regular games," in *Proceedings of the 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)*, 2014, pp. 1–10.

[14] R. Brenguier, J.-F. Raskin, and O. Sankur, "Assume-Admissible Synthesis," in *26th International Conference on Concurrency Theory (CONCUR 2015)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 42, 2015, pp. 100–113.

[15] K. Muvvala, Q. H. Ho, and M. Lahijanian, "Beyond winning strategies: Admissible and admissible winning strategies for quantitative reachability games," 2024. [Online]. Available: https://arxiv.org/abs/2408.13369

[16] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.

[17] J. Liu and N. Ozay, "Finite abstractions with robustness margins for temporal logic-based control synthesis," *Nonlinear Analysis: Hybrid Systems*, vol. 22, pp. 1–15, 2016.

[18] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 61–70, 2007.

[19] C. Belta, B. Yordanov, and E. A. Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017, vol. 89.

[20] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Automated abstraction of manipulation domains for cost-based reactive synthesis," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 285–292, 2018.

[21] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[22] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Int. Joint Conf. on Artificial Intelligence*, ser. IJCAI '13. AAAI Press, 2013, p. 854–860.

[23] A. Pnueli, "The temporal logic of programs," in *18th annual symposium on foundations of computer science (sfcs 1977)*. ieee, 1977, pp. 46–57.

[24] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[25] P. Amorese, S. Wakayama, N. Ahmed, and M. Lahijanian, "Online pareto-optimal decision-making for complex tasks using active inference," *arXiv preprint arXiv:2406.11984*, 2024.

[26] K. Leyton-Brown and Y. Shoham, *Further Solution Concepts for Normal-Form Games*. Cham: Springer International Publishing, 2008, pp. 15–30. [Online]. Available: https://doi.org/10.1007/978-3-031-01545-8_3

[27] R. Brenguier, G. A. Pérez, J.-F. Raskin, and O. Sankur, "Admissibility in Quantitative Graph Games," in *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016)*, vol. 65, 2016, pp. 42:1–42:14.

[28] "Extended version," https://mortezalahijanian.com/papers/ICRA2025_adm_str.pdf.

[29] T. Brihaye, G. Geeraerts, A. Haddad, and B. Monmege, "Pseudopolynomial iterative algorithm to solve total-payoff games and min-cost reachability games," *Acta Informatica*, vol. 54, pp. 85–125, 2017.

[30] S. Zhu and G. De Giacomo, "Synthesis of maximally permissive strategies for ltlf specifications." in *IJCAI*, 2022, pp. 2783–2789.

[31] K. Muvvala, A. Wells, M. Lahijanian, L. Kavraki, and M. Vardi, "Stochastic games for interactive manipulation domains," *IEEE International Conference on Robotics and Automation (ICRA)*, 2024 (submitted).