2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

Opportunistic Synthesis in Reactive Games under Information Asymmetry

Abhishek N. Kulkarni and Jie Fu

***Abstract*— Reactive synthesis is a class of methods to construct a provably-correct control system, referred to as a robot, with respect to a temporal logic specification in the presence of a dynamic and uncontrollable environment. This is achieved by modeling the interaction between the robot and its environment as a two-player zero-sum game. However, existing reactive synthesis methods assume both players to have complete and symmetrical information, which is not the case in many strategic interactions. In this paper, we use a variant of hypergames to model the interaction between the robot and its environment; where the latter has incomplete information about the specification of the robot. We propose a novel method of *opportunistic synthesis* defined over the hypergame model to identify a subset of hypergame states from where the robot can leverage the asymmetrical information to achieve a better outcome, which is not possible if both players have symmetrical and complete information. By assuming the environment to play a stochastic strategy in its perceived sure-winning and sure-losing regions of the game, we show that by following the opportunistic strategy, the robot is ensured to only improve the outcome of the game—measured by satisfaction of sub-specifications—whenever an opportunity becomes available. We demonstrate the correctness and optimality of this method using a robot motion planning example in the presence of an adversary.**

I. Introduction

Reactive synthesis (RS) is used to synthesize a provably-correct strategy (controller) with respect to a given Linear Temporal Logic (LTL) specification. Pnueli and Rosner [1] showed that an interaction between a controlled agent, called the robot, and its dynamic and uncontrollable environment can be represented as a two-player turn-based zero-sum game. Consequently, synthesizing a strategy satisfying the specification is equivalent to finding a winning strategy for the robot in the zero-sum game. In recent years, RS has found applications in several areas such as autonomous vehicles [2], [3], aircraft mission planning [4], defense [5] etc.

However, the strategies computed using RS are known to be conservative [6], [7]. This conservativeness may be attributed to the *zero-sum* assumption used to model the interaction. This assumption implies that the environment knows the exact specification of the robot and plays a perfect counter-strategy. However, in many of the applications of RS such as autonomous vehicles, the environment, consisting of other vehicles, may not be adversarial. On the other hand, in many strategic interactions, where the environment is indeed adversarial, the enemy may not have complete information regarding the task of the robot.

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00111990015.

Abhishek N. Kulkarni is with Robotics Engineering Program, Worcester Polytechnic Institute, Worcester, MA 01609, USA `ankulkarni@wpi.edu`

Jie Fu is with the faculty of the Department of Electrical and Computer Engineering, with affiliation to Robotics Engineering Program, Worcester Polytechnic Institute, Worcester, MA 01609, USA `jfu2@wpi.edu`

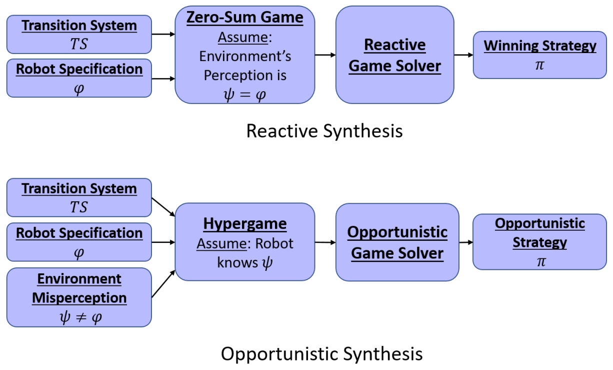

Fig. 1. Comparison between Reactive Synthesis and (Proposed) Opportunistic Synthesis. The task of the robot is the LTL specification φ. The environment misperceives the task of robot as the LTL specification $\psi \neq \varphi$. The (proposed) hypergame formulation assumes that robot is aware of the information asymmetry.

In this paper, we propose a method called *opportunistic synthesis*, which extends the model of hypergames [8] to study reactive synthesis with asymmetrical information, *i.e.* when one of the two players in the game has more or better information than the other [9]. Assuming that the environment *does not know* the complete task specification of the robot, we are interested in addressing the following question – *If the robot is aware of the information asymmetry, then how can it capitalize on the adversary's imperfect counter-strategy to enhance its winning strategy?* The key contributions of this paper are

- **Hypergame Model:** We model the interaction between the robot and its environment as a second-level hypergame [10], in contrast to a zero-sum game as shown in Fig. 1. This model represents the ability of the robot to reason about the environment's behavior when it has incomplete information. Using this model, we show that the robot can synthesize an *opportunistic strategy*, which dominates[1] the one computed using RS given complete and symmetrical information.
- **Characterization:** The solution of RS partitions the game state-space into winning and losing regions for the robot [11]. However, under the assumptions of this paper about information asymmetry, we show that the

[1]A strategy is dominant over another if, regardless of what any other players do, the strategy earns a player a larger payoff than the other strategy.

state-space is partitioned into *five* regions. Assuming the environment to play a stochastic strategy, we present a construction of an Markov Decision Process (MDP), which uses the five regions to synthesize an *opportunistic strategy* for the robot.

A. Literature

To the best of our knowledge, this is the first paper that extends the hypergame model to synthesize provably-correct strategies given temporal logic specifications. Hypergames [10] are used to model the interactions where one or more players are playing different games because of their misperception of other players' capabilities and/or objectives. Hypergame theory allows the agent to improve its strategy by reasoning about multiple games being played by different players [12]. In the literature, this theory has been applied to model strategic interactions such as military conflicts [13], [14], information security and cyber-physical systems security [5]. Similar to the hypergame formulation in this paper, Imamverdiyev [15] modeled the interaction between an attacker and a defender in an information security game using a second-level hypergame. The model is motivated by the information asymmetry that often exists in such a game. He introduced an algorithm to compute equilibrium in a second-level normal form hypergame model. However, the result does not generalize to games in which players receive boolean payoffs for satisfying temporal logic specifications. Kovach [8] introduced the first framework that integrates temporal logic and hypergame theory. He formalized the concepts of trust, mistrust and deception to lay down a mathematical framework for verification. In this work, however, we focus on strategy synthesis instead of verification.

The games with information asymmetry are a subset of games with incomplete information, where at least one player has the privileged information, while other may not [16]. Niu et. al [17] study the problem of security of cyber-physical systems. They note that zero-sum games are a good tool for the worst-case analysis, but the games with information asymmetry better represent the strategic interactions between the attacker and defender. They approach the minimum violation synthesis problem under LTL specifications by modeling the interaction as a concurrent Stakelberg game. In this work, our focus is on turn-based reactive games with asymmetrical information, different from the assumption of symmetrical information in Niu et. al.

In AI literature, *opportunistic planning* is used in a different context. Cashmore et. al [18] treat opportunities as optional goals in the game, but with high payoffs, while making no assumptions on nature of environment. On the contrary, the *opportunistic synthesis*, as proposed in this paper, assumes the environment to be adversarial and with incomplete information about the objective of robot.

II. Reactive Synthesis

Notations: Let Σ be a finite alphabet. A sequence of symbols $w = w_0 w_1 \ldots w_n$ with $w_i \in \Sigma$ for $i \in \mathbb{N}$ is called a *finite word* and Σ^* is the set of finite words that can be generated with alphabet Σ. We denote Σ^ω the set of ω-regular words obtained by concatenating the elements in Σ infinitely many times. Given a set X, let $\mathcal{D}(X)$ be the set of probability distributions over X. The indicator function is defined to be $\mathbf{1}_X(y) = 1$ if $y \in X$ and 0 otherwise.

Let R, E denote the robot and its adversary, respectively. Let the tasks of the robot be specified using a subclass of LTL formulas, called syntactically co-safe LTL formulas. The syntax of LTL formulas are given as follows.

Definition 1 (LTL [19])**.** Let AP be a set of atomic propositions, the Linear Temporal Logic (LTL) has the following syntax

$$\varphi := \top \mid \bot \mid p \mid \varphi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathcal{U} \varphi_2 \mid \Diamond\varphi$$

where $\top, \bot$ are universally true and false, respectively, $p \in AP$ is an atomic proposition, $\bigcirc, \mathcal{U}$ and $\Diamond$ denote the temporal modal operators for *next*, *until* and *eventually*.

A co-safe LTL formula contains only the temporal operator $\bigcirc$, $\mathcal{U}$, and $\Diamond$ and must be written in positive normal form, *i.e.* negation of temporal operator is not allowed [19]. A co-safe LTL formula can be equivalently represented by a Deterministic Finite Automaton (DFA) defined as $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, where Q is the set of states, AP is the set of atomic propositions, $\Sigma = 2^{AP}$ is the set of input symbols, $\delta : Q \times \Sigma \to Q$ is a deterministic transition function, q_0 is the initial state and $F \subseteq Q$ is the set of accepting states.

A run in $\mathcal{A}$ on some word $w = w_0 w_1 \ldots w_n \in \Sigma^*$ is a sequence of states $q_0 q_1 \ldots q_{n+1}$ such that $q_{i+1} = \delta(q_i, w_i)$ for $0 \le i \le n$. A run is accepted by $\mathcal{A}$ if and only if $q_{n+1} \in F$. An infinite word $w \in \Sigma^\omega$ satisfying an LTL formula φ contains a good prefix $w_0 w_1 \ldots w_n$ that is accepted in the DFA corresponding to φ. Given a co-safe LTL formula, the DFA accepting finite good prefixes for φ can be obtained using tools such as Spot [20]. We will assume that all DFAs referred to in this paper are complete, i.e. for every state $q \in Q$ and for every input symbol $a \in \Sigma$ the transition function $\delta(q, a)$ is defined. An incomplete DFA can be made complete by introducing a sink state and directing all undefined transitions into the sink state.

The interaction between the robot and its adversary is captured in a two-player turn-based transition system, defined as a tuple $\mathcal{TS} = \langle S, Act, T, s_0, AP, L \rangle$ where $S = S_R \cup S_E$ is the set of states partitioned on the basis of the turn of R and E, $Act = Act_R \cup Act_E$ is the set of actions for R and E respectively. The function $T : S \times Act \to S$ represents the deterministic transition function. The set AP is a set of atomic propositions and $L : S \to 2^{AP}$ is the labeling function, which maps every state to a subset of atomic propositions that are true in that state.

Given a transition system $\mathcal{TS}$ and a specification automaton $\mathcal{A}$ equivalent to φ, a reactive game between the robot and its adversary can be constructed as $\mathcal{G}(\varphi) = \langle G, Act, \Delta, g_0, F_\varphi \rangle$, where $G = S \times Q$, $g_0 = (s_0, \delta(q_0, L(s_0)))$ and $\Delta : G \times Act \to G$ is the transition function such that given the states $g = (s, q)$ and $g' = (s', q')$, $\Delta(g, a) = g'$ if and only if $T(s, a) = s'$ and $\delta(q, L(s')) = q'$. The set $F_\varphi = S \times F$ is a set of accepting states.

A run in the game $\mathcal{G}(\varphi)$ is an infinite sequence of states $\rho = g_0 g_1 \ldots$. Given a run ρ, the set of states that occur in the run ρ is denoted by $\mathsf{Occ}(\rho) = \{g \in G \mid \exists i \geq 0, g_i = g\}$. A run is said to be winning for R if it satisfies $\mathsf{Occ}(\rho) \cap F_\varphi \neq \emptyset$. If a run is not wining for R, it is winning for E. A state $g \in G$ is said to be winning if the robot can enforce a win from g. Otherwise, g is said to be losing. The exhaustive set of winning states for the robot is called winning region of the robot and is denoted by Win_R. The winning regions for robot and its adversary are mutually exclusive. The winning region for the robot is computed using the Zielonka attractor algorithm [21] as follows: Given a set of final states F_φ,

1) Let $\mathsf{Attr}_0 = F_\varphi$.
2) $\mathsf{Attr}_{k+1} = \mathsf{Attr}_k \cup \{g \in (S_R \times Q) \mid \exists a \in Act_R \cdot \Delta(g,a) \in \mathsf{Attr}_k)\} \cup \{g \in (S_E \times Q) \mid \forall a \in Act_E \cdot \Delta(g,a) \in \mathsf{Attr}_k\}$
3) Repeat (2) until $\mathsf{Attr}_{k+1} = \mathsf{Attr}_k$. Let $\mathsf{Attr}_k = \mathsf{Attr}^*$.
4) Let $\mathsf{Attr}(F) = \mathsf{Attr}^*$.

We denote the set $\mathsf{Attr}(F)$ as the attractor set. The winning region for the robot in the game $\mathcal{G}(\varphi)$ is the set of states in attractor $\mathsf{Attr}(F_\varphi)$.

A specification φ is said to be *realizable* for robot over the transition system $\mathcal{TS}$ if and only if the winning region for the robot, Win_R, contains the initial state $g_0 \in G$. Otherwise, the specification is said to be unrealizable.

Given a game $\mathcal{G}(\varphi)$, a stochastic, memoryless strategy for the robot is a map $\pi : \mathsf{Win}_R \cap (S_R \times Q) \to \mathcal{D}(Act_R)$. A strategy is said to be *almost-sure-winning* if every run ρ, produced as a result of robot using strategy π and adversary using any feasible strategy σ, is a winning run with probability one. A state is called *almost-sure winning* for robot, if there exists an almost-sure winning strategy for robot from that state. The exhaustive set of almost-sure winning states for robot is called her *almost-sure winning region*. In a deterministic and turn-based game, the almost-sure winning region is equal to sure-winning region [22, Thm 3]. Hence, it can be computed using Zielonka's algorithm.

Under the assumption of complete observation and complete information, it follows that there exists no strategy for the robot(or its adversary) to reach a winning state from a losing state. However, when the adversary has incomplete knowledge about the task specification of the robot, we have a reactive game with asymmetrical information. In such games, we show that the robot can synthesize *opportunistic* strategies that exploit the information asymmetry to enforce a win in an otherwise unrealizable game; had there been no information asymmetry.

III. Reactive Game under Information Asymmetry

A. Hypergame

Consider an interaction between the robot and its adversary where the robot has the LTL specification φ while its adversary believes that the robot is trying to satisfy a different specification $\psi \neq \varphi$.

Assumption 1. Let φ_1, φ_2 be two LTL formulas such that $\varphi = \varphi_1 \land \varphi_2$ and $\psi = \varphi_1$.

The above assumption means that the adversary knows partial task specification of the robot. The interaction between two players, where at least one of the player has misperception of the true specification of the opponent, can be represented as a hypergame [10], [23].

Definition 2 (Hypergame)**.** A first-level hypergame between two players is represented as a 2-tuple

$$\mathcal{H}^1 = \langle \mathcal{G}_R, \mathcal{G}_E \rangle = \langle \mathcal{TS}, \{\varphi, \psi\} \rangle,$$

where $\mathcal{G}_R = \mathcal{G}(\varphi)$ is the reactive game from the robot's perspective while $\mathcal{G}_E = \mathcal{G}(\psi)$ is the game from the adversary's perspective. The tuple $\langle \mathcal{TS}, \{\varphi, \psi\} \rangle$ is an equivalent representation that highlights that both the games $\mathcal{G}_R$ and $\mathcal{G}_E$ are defined over same transition system $\mathcal{TS}$ but each player has a different perception of robot's specification.

When the robot is aware of the existing misperception, *i.e.* the adversary's belief about the robot's specification ψ, we have a second-level hypergame.

Definition 3 (Second-Level Hypergame)**.** The second-level hypergame between two players; the robot and its adversary, where only the robot is aware of the misperceived game is represented as

$$\mathcal{H}^2 = \langle \mathcal{H}^1, \mathcal{G}_E \rangle,$$

where the robot computes the strategy by solving the hypergame $\mathcal{H}^1$ and the adversary computes strategy by solving the reactive game $\mathcal{G}_E = \mathcal{G}(\psi)$.

Given the second-level hypergame as defined above, we are interested in the following question

Question 1. *Consider a specification φ that is unrealizable in the reactive game $\mathcal{G}(\varphi)$ with complete information. If there exists information asymmetry as captured by the hypergame $\mathcal{H}^2$, can the robot satisfy the specification $\varphi = \varphi_1 \land \varphi_2$ with a high likelihood? If not, then under what conditions can the robot satisfy at least a part of specification, φ_1 (common knowledge) or φ_2 (private knowledge of the robot)?*

To answer the above question, we first define a transition system that captures the information asymmetry compactly and facilitates game-theoretic analysis and strategic planning.

Definition 4 (Hypergame Transition System)**.** Let $\mathcal{A}_1 = \langle Q_1, \Sigma, \delta_1, q_{10}, F_1 \rangle$ and $\mathcal{A}_2 = \langle Q_2, \Sigma, \delta_2, q_{20}, F_2 \rangle$ be the specification automata for LTL formulas φ_1, φ_2. Then, the hypergame transition system in its explicit form is a 5-tuple,

$$\mathcal{H} = \langle H, Act, \Delta, h_0, \mathcal{F} \rangle,$$

where $H = S \times Q_1 \times Q_2$ is the set of states and $h_0 \in H$ is the initial state. Given a state $h = (s, q_1, q_2)$ and action $a \in Act$, the transition function $\Delta : H \times Act \to H$ is given by $\Delta(h,a) = h' = (s', q'_1, q'_2)$ where $s' = T(s,a)$, $q'_1 = \delta_1(q_1, L(s'))$ and $q'_2 = \delta_2(q_2, L(s'))$. The set $\mathcal{F} = (S \times F_1 \times Q_2) \cup (S \times Q_1 \times F_2)$ is the set of final states.

The choice of $\mathcal{F}$ as the accepting state set is motivated by the fact that it contains the final state sets $\mathcal{F}_1 = S \times F_1 \times$

$Q_2, \mathcal{F}_2 = S \times Q_1 \times F_2$ and $\mathcal{F}_{12} = S \times F_1 \times F_2$ of the games $\mathcal{G}(\varphi_1)$, $\mathcal{G}(\varphi_2)$ and $\mathcal{G}(\varphi_1 \wedge \varphi_2)$. This facilitates the computation and analysis of winning regions of the game $\mathcal{G}(\varphi)$ and the sub-games $\mathcal{G}(\varphi_1), \mathcal{G}(\varphi_2)$ over the same transition system.

The outcome of hypergame in this transition system is the run $\rho = h_0 h_1 h_2 \ldots$. By construction, the run is winning for robot over specification φ (resp. φ_1, φ_2) if and only if $\mathsf{Occ}(\rho) \cap \mathcal{F}_{12} \neq \emptyset$ (resp. $\mathsf{Occ}(\rho) \cap \mathcal{F}_1 \neq \emptyset$, $\mathsf{Occ}(\rho) \cap \mathcal{F}_2 \neq \emptyset$).

B. The Partition of States

1) $\mathsf{Win}_R(\varphi_1) = \mathsf{Attr}(\mathcal{F}_1)$ is the set of winning states in the game $\mathcal{G}(\varphi_1)$.
2) $\mathsf{Win}_R(\varphi_2) = \mathsf{Attr}(\mathcal{F}_2)$ is the set of winning states in the game $\mathcal{G}(\varphi_2)$.
3) $\mathsf{Win}_R(\varphi) = \mathsf{Attr}(\mathcal{F}_{12})$ is the set of winning states in the game $\mathcal{G}(\varphi)$.

We have the following relations between winning regions,

Lemma 1. *Given $\varphi = \varphi_1 \wedge \varphi_2$, it holds that $\mathsf{Win}_R(\varphi) \subseteq \mathsf{Win}_R(\varphi_i)$, for $i = 1, 2$.*

Proof. Let π_φ be the winning strategy for the robot with respect to task φ. For any state $h \in \mathsf{Win}_R(\varphi)$, the robot, by exercising π_φ, can enforce a run to visit $\mathcal{F}_{12}$. Because $\mathcal{F}_{12} \subseteq \mathcal{F}_i$, for $i = 1, 2$, then this run satisfies φ_i, for $i = 1, 2$. By definition of winning region, it holds that $h \in \mathsf{Win}_R(\varphi_i)$, $i = 1, 2$, witnessed by strategy π_φ. Q.E.D.

On the contrary, $\mathsf{Win}_E(\varphi_i) \supseteq \mathsf{Win}_E(\varphi)$. Thus, if the adversary can ensure to win game $\mathcal{G}(\varphi_1)$, then even though it does not know φ_2, it can prevent the robot from satisfying any specification $\varphi_1 \wedge \phi$, where ϕ is an arbitrary LTL formula.

Next, we define a win-labeling function $\mathcal{W} : H \to \{W, L\}^3$ that labels each state $h \in H$ with an ordered 3-tuple denoting whether the state h is winning (W) or losing (L) for the robot in the games $\mathcal{G}(\varphi_1)$, $\mathcal{G}(\varphi_2)$, and $\mathcal{G}(\varphi)$. For example, if a state h is winning for the robot in the game $\mathcal{G}(\varphi_1)$ and the game $\mathcal{G}(\varphi_2)$, but losing in the game $\mathcal{G}(\varphi)$, then its win-label is $\mathcal{W}(h) = \{W, W, L\}$ [2].

Note that the win-labeling function can assign to every state $h \in H$, a unique label from $2^3 = 8$ possible labels. We analyze each possible label separately to understand which of the objectives φ_1, φ_2 or φ should the robot try to satisfy.

a) Case I: $\mathcal{W}(h) = (L, L, L)$: The state h is losing for robot in the games $\mathcal{G}(\varphi_1), \mathcal{G}(\varphi_2)$ and $\mathcal{G}(\varphi)$, *i.e.* the adversary has a winning strategy σ that will ensure that the robot can never satisfy φ_1. Therefore, in this case, the robot can try to satisfy only φ_2, but will never be able to satisfy φ.

b) Case II: $\mathcal{W}(h) = (L, W, L)$: The state is losing for robot in the games $\mathcal{G}(\varphi_1)$ and $\mathcal{G}(\varphi)$, but winning in game over φ_2. That is, the adversary has a winning strategy σ_W that will ensure that robot can never satisfy φ_1. Therefore, in this case, the robot must satisfy only φ_2, and not φ.

[2]If $\varphi = \varphi_1 \wedge \varphi_2$ then for a state $h \in H$ to be winning in the game over φ, it must be winning in both the sub-games over φ_1 and φ_2 [24, Lma 1].

c) Case III: $\mathcal{W}(h) = (W, L, L)$: The state is losing for robot in the games $\mathcal{G}(\varphi_2)$ and $\mathcal{G}(\varphi)$, but winning in game over φ_1. That is, the adversary believes that it has lost the game and can be assumed to play a random strategy, σ_L. In this case, the robot may try to satisfy φ, while staying within the winning region of game $\mathcal{G}(\varphi_1)$.

d) Case IV: $\mathcal{W}(h) = (W, W, L)$: The state is winning for robot in the games $\mathcal{G}(\varphi_1)$ and $\mathcal{G}(\varphi_2)$, but losing in game $\mathcal{G}(\varphi)$. That is, the adversary believes that it has lost the game. This state presents an interesting decision problem where robot must decide whether to *try* satisfying φ or satisfy just one of the specifications, φ_1 or φ_2.

e) Case V: $\mathcal{W}(h) = (W, W, W)$: This is a trivial case, which is the conventional reactive game. The robot can exercise the winning strategy for $\mathcal{G}(\varphi)$ regardless of the strategy and perception of the adversary.

f) Cases VI-VIII: $\mathcal{W}(h) = (L, L, W)$, (L, W, W), or (W, L, W): These cases are not possible, because the robot must be winning in both the specifications, φ_1 and φ_2, to be winning in φ [24, Lemma. 1].

C. Synthesizing opportunistic and reactive strategies

Recall that the winning regions $\mathsf{Win}_R(\cdot)$ we computed in previous subsection ensures a win in the respective reactive games. The winning strategies based on these winning regions do not exploit the information asymmetry. Hence, to identify the opportunities generated due to the information asymmetry, we make certain assumption about the strategy of the adversary.

Assumption 2. For a state $h \in H$ with the win-label $\mathcal{W}(h) = (W, \cdot, \cdot)$ the adversary plays a stochastic strategy $\sigma_L : H \to \mathcal{D}(Act_E)$.

Assumption 3. For a state $h \in H$ with the win-label $\mathcal{W}(h) = (L, \cdot, \cdot)$, the adversary plays an almost-sure winning strategy $\sigma_W : H \to \mathcal{D}(Act_E)$ in the game $\mathcal{G}(\varphi_1)$.

Remark. In this work, we assume that the adversary's losing and winning strategies, σ_L, σ_W, are known to the robot. For example, the adversary selects all actions uniformly at random in a losing state and selects an action in the set of sure-winning action set uniformly at random from a winning state. This assumption can be relaxed if the robot can learn the strategy using model-based reinforcement learning [25] or strategy inference [26]. This extension will be considered in our future work.

To develop opportunistic strategy for the robot, we assume that the robot receives a payoff $r_1 \in \mathbb{R}_{>0}$ if it satisfies φ_1 and payoff $r_2 \in \mathbb{R}_{>0}$ if it satisfies φ_2. Its payoff for satisfying φ is $r \in \mathbb{R}_{>0}$ with the constraint $r \geq r_1 + r_2$ for the decision problem to make sense. The adversary receives the payoff $-r_1$ if the robot satisfies φ_1 and payoff r_1 otherwise. Note that when $r_1 = r_2$, then the robot is considered to be indifferent to satisfying either φ_1 or φ_2. Otherwise, if $r_1 > r_2$, then φ_1 is strictly preferred over φ_2, and vice versa.

Definition 5 (Hypergame for Opportunistic Synthesis)**.** Given the hypergame transition system $\mathcal{H} = \langle H, Act, \Delta, h_0, \mathcal{F} \rangle$

and the strategy $\sigma = (\sigma_W, \sigma_L)$ used by the adversary given its perceived winning and losing states, the opportunistic planning reduces to an MDP, defined by

$$\mathcal{H}^\sigma = \langle H_R, Act_R \cup \{\mathsf{stop}\}, P, h_0, R \rangle,$$

where $H_R = (S_R \times Q_1 \times Q_2) \cup \{\mathsf{sink}, \mathsf{sink}_1\}$ is a set of states where the robot makes a move. The states sink and sink_1 are absorbing states where robot switches to the winning strategy in games $\mathcal{G}(\varphi)$ and $\mathcal{G}(\varphi_1)$ respectively. The probabilistic transition function P and the payoff function R are defined based on the win-label of a state $h \in H_R$ as follows,

- $\mathcal{W}(h) \in (L,L,L)$: All feasible actions of robot are enabled. The special action stop is not enabled.
  - Transition probability function is given by
  $$P(h' \mid h, a_R) = \sum_{a_E \in Act_E} \mathbf{1}_{\{h'\}}(\Delta(h,(a_R,a_E)))\sigma_W(h,a_E),$$
- $\mathcal{W}(h) \in (W,L,L)$: Only actions that have *zero* probability of reaching a state with win-label (L,L,L) are enabled. In other words, the robot is forced to stay within the winning region for $\mathsf{Win}_R(\mathcal{F}_1)$ and at least satisfy φ_1. The special action stop is enabled.
  - Transition probability function is given by
  $$P(h' \mid h, a_R) = \sum_{a_E \in Act_E} \mathbf{1}_{\{h'\}}(\Delta(h,(a_R,a_E)))\sigma_L(h,a_E),$$
  For action stop, the game transitions to an absorbing state sink_1 with $P(\mathsf{sink}_1 \mid h, \mathsf{stop}) = 1$, after which robot must switch to its winning strategy in $\mathcal{G}(\varphi_1)$.
  - Payoff function: The payoff for reaching the absorbing state sink_1 is defined as $R(\mathsf{sink}_1) = r_1$.
- $\mathcal{W}(h) \in (L,W,L)$:
  - Transition probability function is $P(h' \mid h, a_R) = 1$ because all states are labeled as absorbing, because the adversary will play its winning strategy σ_W in the game $\mathcal{G}(\varphi_1)$ and the robot will never satisfy φ_1.
  - Payoff function: The payoff for reaching the state h is defined as $R(h) = r_2$.
- $\mathcal{W}(h) \in (W,W,W)$: All states with this label are absorbing. The robot must switch to its winning strategy in the game $\mathcal{G}(\varphi)$.
  - Payoff function: The payoff for reaching the state h is defined as $R(h) = r$.
- $\mathcal{W}(h) \in (W,W,L)$: Any action that does not lead into partition (L,L,L) is enabled. The special action stop is also enabled.
  - For the action stop, the robot transitions to an absorbing state sink. The payoff for reaching the absorbing state sink is defined as $R(\mathsf{sink}) = \max(r_1, r_2)$. After reaching a absorbing state, robot must switch to the winning strategy of the subgame $i = \arg\max_i\{r_i \mid i \in \{1,2\}\}$.
  - For action $a \in Act_R$, transition probability function is given as
  $$P(h' \mid h, a_R) = \sum_{a_E \in Act_E} \mathbf{1}_{\{h'\}}(\Delta(h,(a_R,a_E)))\sigma_L(h,a_E)$$

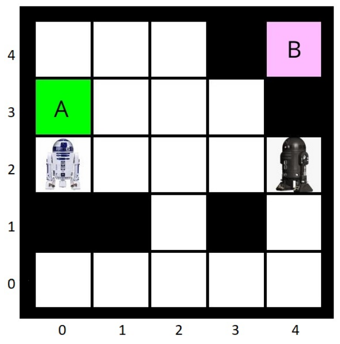

Fig. 2. A 5×5 gridworld with R2D2 at the cell $(0,2)$ and IDroid at $(4,2)$. The objective of R2D2 to visit the cell $(0,3)$ labeled A is known to both players. The objective of R2D2 to visit the cell $(4,4)$ labeled B is not known to IDroid. The black cells represent the obstacles.

The optimal opportunistic strategy π for the robot is the one that solves

$$\max_\pi \mathbb{E}\left[\sum_{t=1}^{T} R(h_t)\right],$$

where T is the first time when an absorbing state is reached. The rationale behind defining absorbing states is to provide the robot with a mechanism to decide whether it wants to explore the state space to find an opportunity or settle for a sub-optimal payoff by satisfying a sub-specification. We define the set of states $\{h \mid \mathcal{W}(h) \in \{(L,W,L),(W,W,W)\}\} \cup \{\mathsf{sink}_1, \mathsf{sink}\}$ as absorbing in the hypergame MDP.

Lemma 2. *By following the optimal strategy in the MDP $\mathcal{H}^\sigma$, the total payoff is finite.*

Proof. We prove this case by case: When the initial state h_0 is labeled (L,L,L), then by following the optimal strategy, the robot can reach a state with labels in $\{(W,W,L),(W,L,L),(L,W,L),(W,W,W)\}$ or stay in (L,L,L). If it stays in (L,L,L), the robot receives a payoff of zero. The total payoff is bounded. When it reaches a state with labels in $\{(W,W,L),(W,L,L),(L,W,L),(W,W,W)\}$, the robot is ensured to at least win one of the games as its feasible actions are restricted to ensure staying in the winning regions it is currently in. Then, in cases when the label is in $\{(W,W,L),(W,L,L)\}$, it will either settle down to win one of the games by taking the action stop or reach a state with label in $\{(L,W,L),(W,W,W)\}$, which are absorbing and have finite payoffs. Thus, the total payoff is bounded and the planning to maximize the total payoff without discounting is well-defined. Q.E.D.

IV. Case Study

We illustrate our approach using a gridworld example as shown in Fig. 2, with two robots - R2D2 and Imperial Droid (IDroid). R2D2 is the controllable robot, whereas IDroid is adversarial. The objective of R2D2 is to visit two regions, A (green) and B (blue), while avoiding obstacles O (black), whereas the objective of IDroid is to prevent R2D2 from completing its task. We consider the case where IDroid *misperceives* that R2D2's task is to visit only region A. Therefore, using the LTL notation, the specification of R2D2 is $\varphi = (\neg O \ \mathcal{U} \ A) \wedge (\neg O \ \mathcal{U} \ B)$, whereas the misperception of IDroid about R2D2's task is $\psi = \neg O \ \mathcal{U} \ A$. This defines the information asymmetry in the interaction. Furthermore, we restrict the actions of R2D2 and IDroid as follows, (we use the symbol R to denote R2D2 and E to denote IDroid)

$$Act_R = \{\text{N, S, E, W, NE, NW, SE, SW}\},$$
$$Act_E = \{\text{N, S, E, W, STAY}\},$$

Given 20 obstacle-free cells of gridworld in Fig. 2 and the action-set, we construct the transition system with $20 \times 20 \times 2 = 800$ states. The automaton equivalent to $\neg O \ \mathcal{U} \ X$ for $X = A, B$ is shown in the Fig. 3. We prune unsafe actions that drive the robot to the obstacle and thus exclude the transitions labeled O and the state 2 in computing the transition system. Therefore, the hypergame transition system has $800 \times 2 \times 2 = 3200$ states, where we keep track of both sub-specification using two automata. Consequently, each sub-game, $\mathcal{G}(\varphi_1)$ and $\mathcal{G}(\varphi_2)$, has $800 \times 2 \times 1 = 1600$ final states and the game $\mathcal{G}(\varphi)$ has 800 final states. The attractor computation for each of the three games generates the winning regions with sizes: $|\mathsf{Win}_R(\varphi_1)| = 2491$, $|\mathsf{Win}_R(\varphi_2)| = 2527$, and $|\mathsf{Win}_R(\varphi)| = 1831$.

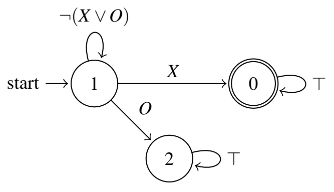
$$\begin{tikzcd}
\text{start} \arrow[r] & 1 \arrow[loop above, "\neg(X \vee O)"] \arrow[r, "X"] \arrow[dr, "O"'] & 0 \arrow[loop right, "\top"] \\
& & 2 \arrow[loop right, "\top"]
\end{tikzcd}$$

Fig. 3. The automaton for $\neg O \ \mathcal{U} \ X$, where $X \in \{A, B\}$.

Given the three winning regions, we first validate that the state-space is indeed partitioned in five regions as discussed in Section III-B. For every state in the $\mathcal{H}$, we assign a win-label to it by determining the winning regions in which the state appears. The result is tabulated in I. We observe that the state-space is partitioned into exactly five regions.

Partition	Number of States
(W, W, W)	1831
(W, W, L)	181
(W, L, L)	479
(L, W, L)	515
(L, L, L)	194
(W, L, W)	0
(L, W, W)	0
(L, L, W)	0

TABLE I
PARTITION OF GAME STATE-SPACE DUE TO INFORMATION ASYMMETRY.

Remark. It is not necessary that the states will always be partitioned into *exactly* five regions. For instance, consider an adversary with only "STAY" action, then the state-space will be partitioned into exactly two regions.

Using the five partitions, we construct the hypergame MDP as defined in Definition 5. We define the stochastic strategies for IDroid as follows: for every state with win-label of $(L, \cdot, \cdot)$, we assume σ_W to be a uniform distribution over all safe actions; *i.e.* the actions that, with probability *one*, lead to a state with a win-label of type $(L, \cdot, \cdot)$. We define σ_L by assigning an arbitrary distribution over all feasible actions from a state within partitions $(W, \cdot, \cdot)$. Given the MDP states and the adversary strategy σ, the transition probabilities are determined based on win-label of the state and the corresponding expression for $P(h' \mid h, a)$ is provided in Definition 5. We compute the value function and opportunistic strategy using the value iteration algorithm [27].

Act	Next State	Partition	Prob	Value
N	**(((0, 3), (4, 2), 0), 0, 1)**	**(W, L, L)**	**0.03**	**288.99**
	(((0, 3), (3, 2), 0), 0, 1)	**(W, L, L)**	**0.36**	**290.20**
	(((0, 3), (4, 1), 0), 0, 1)	**(W, W, W)**	**0.61**	**288.99**
E	(((1, 2), (4, 1), 0), 0, 1)	(W, W, L)	0.25	0
	(((1, 2), (3, 2), 0), 1, 1)	(W, L, L)	0.73	297.41
	(((1, 2), (4, 2), 0), 1, 1)	(W, W, L)	0.02	0
NE	(((1, 3), (3, 2), 0), 1, 1)	(W, L, L)	0.38	259.42
	(((1, 3), (4, 2), 0), 1, 1)	(W, W, L)	0.18	285.03
	(((1, 3), (4, 1), 0), 1, 1)	(W, W, L)	0.44	299.25

TABLE II
A DECISION TABLE FOR STATE $(((0,2),(4,2),0),1,1)$ WITH VALUE 285.03 AND STRATEGY TO CHOOSE ACTION "N".

Next, we illustrate the decision process in the hypergame MDP. Let the initial configuration be such that R2D2 is at the cell $(0,2)$, and IDroid is at $(4,2)$ as shown in Fig. 2. Therefore, the initial state in the hypergame MDP is $h_0 = (((0,2),(4,2),0),1,1)$. We define the payoff for reaching goal A as $r_1 = 200$ and that for reaching goal B as $r_2 = 100$. With this initial configuration we simulate the interaction between R2D2 and IDroid, where R2D2 uses the opportunistic strategy π and IDroid uses the strategy σ. We run the simulation for 100 times. We will use one of the runs obtained from simulation, as given below

1) State: $(((0,2),(4,2),0),1,1)$, win-label: (W,L,L)
2) State: $(((0,3),(3,2),0),0,1)$, win-label: (W,L,L)
3) State: $(((1,2),(2,2),0),0,1)$, win-label: (W,W,W)

To get some insight into the decision process, observe the Table II, which shows the enabled actions, possible next states and their respective partitions, the probability of reaching those states and the value of those states. Based on the value iteration, the value of initial state is 285.03, while the optimal strategy is to select action "N", which has a high likelihood to reach a (W,W,W) state. Note that by choosing action "E", if the robot reaches a state with value 0, then it chooses to settle for sub-optimal payoff of $r_1 = 200$ by satisfying only φ_1. Hence, the action "N" is preferred over "E". A similar argument can be given for the action "NE".

We now point out the key advantage of the opportunistic synthesis over reactive synthesis. Observe that the initial state is losing in the game $\mathcal{G}(\varphi)$ for R2D2. Therefore, if R2D2 uses reactive synthesis approach, it will give up instantaneously and get no payoff. On the contrary, with opportunistic synthesis, R2D2 could leverage the misperception of IDroid to start from a losing state in $\mathcal{G}(\varphi)$ and reach a winning state in the game.

We highlight that the construction of hypergame MDP is such that R2D2 behaves rationally and tries to maximize the payoff. Given the initial state in partition (W,L,L), it could have chosen the stop action and switched to the winning strategy in $\mathcal{G}(\varphi_1)$ to get a payoff of $r_1 = 200$. Instead, it continued exploring for an opportunity to get a payoff of $r = r_1 + r_2 = 300$.

We conclude this section by counting the number of states with opportunities. This is done by counting the number of MDP states with non-zero value. Recall that we label the absorbing states in the MDP as absorbing with a fixed payoff. Therefore, they always have fixed value of *one*. We find that there are a total of 1245 absorbing states and 312 states with opportunities. This implies that out of 1600 total states, there are $1600 - (1245 + 312) = 43$ states with no opportunities. In other words, not all losing states in the reactive game $\mathcal{G}(\varphi)$ have opportunities.

V. Discussion and Conclusion

In this paper, we have introduced a novel strategy synthesis approach—*opportunistic synthesis*—to solve reactive games under information asymmetry. By modeling the misperception in the interaction between the robot and its adversarial environment as a hypergame and the corresponding decision problem as a hypergame MDP, we identify opportunities by maximizing the expected value to reach a winning state in the reactive game from a losing state. This primarily results in a larger number of states from which the robot can satisfy its specification, φ. As a bonus, the approach also allows us to compute the states from which the robot has an opportunity to satisfy a partial specification, φ_1 or φ_2.

From a computational point of view, strategy synthesis in the hypergame has the time and space complexity of $O(H^2)$, which is the same order as Zielonka's attractor computation. The number of computations in our approach is related to that of Zielonka's algorithm by only a constant scaling factor; because we require three attractor computations and a value iteration instead of a single attractor computation. Note that the definition of hypergame transition system in Definition 4 dispenses with constructing different game structures for computing winning regions of the sub-games.

We have presented the preliminary results of our investigation, in what we believe to be the first work, in the use of hypergame model to study a reactive game with information asymmetry. To restrict the hypergame model to second-level, we have introduced two assumptions that, *the adversary has partial information regarding robot's specification* and *the robot knows adversary's losing strategy σ_L and winning strategy σ_W*. The relaxation of these assumptions opens up two directions for future research. The first one investigates opportunistic synthesis when adversary perceives the robot's objective as $\psi \neq \varphi$, *i.e.* the language of ψ may be a subset, superset or disjoint with the language of φ. The second direction investigates the use of policy inference techniques for the robot to learn the adversary's strategies σ_L and σ_W.

References

[1] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," 1989, pp. 179–190.

[2] H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu, "Correct, reactive, high-level robot control," *IEEE Robotics & Automation Magazine*, vol. 18, pp. 65–74, 2011.

[3] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Synthesis of Control Protocols for Autonomous Systems," *Unmanned Systems*, 2013.

[4] L. R. Humphrey, E. M. Wolff, and U. Topcu, "Formal specification and synthesis of mission plans for unmanned aerial vehicles," in *2014 AAAI Spring Symposium Series*, 2014.

[5] M. Wu, H. Zeng, C. Wang, and H. Yu, "Safety Guards: Runtime Enforcement for Safety-Critical Cyber-Physical Systems," vol. 6, 2017, pp. 1–6.

[6] R. Bloem, R. Ehlers, S. Jacobs, and R. Könighofer, "How to handle assumptions in synthesis," in *SYNT*, 2014.

[7] S. Hagihara, A. Ueno, T. Tomita, M. Shimakawa, and N. Yonezaki, "Simple synthesis of reactive systems with tolerance for unexpected environmental behavior," in *Proceedings of the 4th FME Workshop on Formal Methods in Software Engineering*. ACM, 2016, pp. 15–21.

[8] Nicholas Kovach, "A Temporal Framework for Hypergame analysis of Cyber Physical Systems in Contested Environments." Ph.D. Thesis, 2016.

[9] J. Kim and Y. K. Che, "Asymmetric information about rivals' types in standard auctions," *Games and Economic Behavior*, vol. 46, no. 2, pp. 383–397, 2004.

[10] P. Bennett, "Toward a theory of hypergames," *Omega*, vol. 5, no. 6, pp. 749–751, jan 1977.

[11] Z. Manna and A. Pnueli, "A hierarchy of temporal properties (invited paper, 1989)." ACM Press, 1990, pp. 377–410.

[12] P. Bennett and M. Dando, "The arms race as a hypergame: A study of routes towards a safer world," *Futures*, vol. 14, no. 4, pp. 293–306, 1982.

[13] P. G. Bennett and M. R. Dando, "Complex strategic analysis: A hypergame study of the fall of france," *Journal of the Operational Research Society*, vol. 30, no. 1, pp. 23–32, 1979.

[14] L. Thomas, N. M. Fraser, and K. W. Hipel, "Conflict Analysis: Models and Resolutions." *The Journal of the Operational Research Society*, 2006.

[15] Y. Imamverdiyev, "A hypergame model for information security," *International Journal of Information Security Science*, vol. 3, no. 1, pp. 148–155, 2014.

[16] M. Scerala, J. A. Erkoyuncua, and E. Shehaba, "Identifying information asymmetry challenges in the defence sector," *Procedia Manufacturing*, vol. 19, pp. 127–134, 2018.

[17] L. Niu, J. Fu, and A. Clark, "Minimum Violation Control Synthesis on Cyber-Physical Systems under Attacks." IEEE, 2018, pp. 262–269.

[18] M. Cashmore, M. Fox, D. Long, D. Magazzeni, and B. Ridder, "Opportunistic planning in autonomous underwater missions," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 519–530, 2018.

[19] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.

[20] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, and L. Xu, "Spot 2.0 — a framework for LTL and ω-automata manipulation," in *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16)*, ser. Lecture Notes in Computer Science, vol. 9938. Springer, Oct. 2016, pp. 122–129.

[21] W. Zielonka, "Infinite games on finitely coloured graphs with applications to automata on infinite trees," *Theoretical Computer Science*, vol. 200, no. 1-2, pp. 135–183, 1998.

[22] L. de Alfaro, T. A. Henzinger, and O. Kupferman, "Concurrent reachability games," *Theoretical Computer Science*, vol. 386, no. 3, pp. 188 – 217, 2007, expressiveness in Concurrency. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S030439750700504X

[23] B. Gharesifard and J. Cortés, "Evolution of players' misperceptions in hypergames under perfect observations," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1627–1640, 2012.

[24] A. N. Kulkarni and J. Fu, "A compositional approach to reactive games under temporal logic specifications," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 2356–2362.

[25] R. I. Brafman and M. Tennenholtz, "R-max-a general polynomial time algorithm for near-optimal reinforcement learning," *Journal of Machine Learning Research*, vol. 3, no. Oct, pp. 213–231, 2002.

[26] M. K. Hanawal, H. Liu, H. Zhu, and I. C. Paschalidis, "Learning policies for markov decision processes from data," *IEEE Transactions on Automatic Control*, pp. 1–1, 2018.

[27] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons Inc., 2005.