A Symbolic Motion Planning Approach for the Reach-avoid Problem

Laya Shamgah, Ali Karimoddini, Abdollah Homaifar

***Abstract*—This paper addresses the motion planning problem for two autonomous robots (the defender and the attacker) with competitive objectives, which are involved in a reach-avoid scenario. This adversarial aspect of the game makes the problem complex with high computational cost. To address this problem, we propose a novel symbolic approach for the robot motion planning and control of the robots, which can effectively manage the complexity of the problem. The basic idea is to partition the environment into convex regions, and then, capture the desired objectives of the defender and the adversarial behavior of the attacker with temporal logic formulas. We also use finite two-player zero-sum games as a tool for the robot decision-making over the partitioned space. An illustrative examples has been provided to detail the steps of the proposed algorithm and the simulation results are presented to verify the effectiveness of the proposed algorithm.**

1. INTRODUCTION

In robotics, traditional approaches can well address the robot motion planning in which the robots are only required to optimally transit from an initial region to the desired location while avoiding (static) obstacles. This problem has been largely investigated in literature (See, e.g., [1], [2], and the references therein). However, there are many applications that are beyond this problem formulations, e.g., consider a flight collision avoidance [3] or an air combat [4] that requires the robots to move in an adversarial environment to achieve a goal.

A common approach to address these problems is to formulate them as a pursuit-evasion game in which the players aim to either capture or avoid of capture. Some novel methods are introduced to solve these pursuit-evasion games [5]–[8].

A pursuit-evasion game is naturally an avoidance problem. Extending to a more general case by combining the path planning problem with dynamic obstacle avoidance, the problem becomes more complicated. Such complex problems can be addressed by reach-avoid games, in which an attacker aims to reach a specific target, while evading from its opponent, and a defender wants to avoid the attacker to reach a target region by capturing the attacker.

The authors are with the Department of Electrical and Computer Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC 27411 USA.
Corresponding author: A. Karimoddini. Address: 1601 East Market Street, Department of Electrical and Computer Engineering North Carolina A&T State University Greensboro, NC, US 27411. Email: akarimod@ncat.edu *(Tel: +13362853847).*

In [9] and [10], optimal paths are obtained while static obstacles are avoided. In [11] and [12], a method is proposed for robot path planning for avoiding dynamic obstacles. One new approach to address reach-avoid path planning is to formulate this problem as differential zero-sum games [13], [14]. In these works using Hamilton-Jacobi reachability analysis, the winning strategies for the players are obtained as the solution of these games. In all of these techniques, the high computational cost remains as a challenge, due to the complex nature of the reach-avoid problem.

To address this problem, we propose a novel symbolic motion planning framework which can effectively handle reach-avoid problems within the context of hybrid supervisory control theory [15]. For this purpose, we formulate the reach-avoid problem as a discrete game over a partitioned environment. We then consider the decision making process for the defender as a finite two-player zero-sum game over the partitioned environment. This formulation allows the defender to make high-level decisions about transiting to the next adjacent region, based on the current positions of the players, without the use of explicit prediction models or imposing any limit on the players actions. Then, the desired objectives of the defender and the adversarial behavior of the attacker are captured by linear temporal logic (LTL) [16], [17]. Reactive synthesis techniques [18]–[20] are then applied to achieve winning strategies for the defender, satisfying the desired objectives. Finally, a hybrid controller is designed to calculate continues control signals for driving the defender over the partitioned game space. A similar method can be applied for driving the attacker robot. An illustrative example and simulation results are provided to verify the proposed algorithm. To the best of our knowledge, this is the first work on symbolic motion planning for reach-avoid problem.

The rest of paper is organized as follows. In Section 2, the reach-avoid problem is formulated. In section 3, the behavior of the defender and attacker are captured by LTL formulas followed by synthesizing an equivalent automaton, which satisfies the requirements of the formulated reach-avoid problem. Section 4 describes the hybrid control synthesis approach, and applies the proposed technique to an illustrative example. In Section 5, we provide some concluding remarks and directions for future work.

2. PROBLEM FORMULATION

In this paper the reach-avoid game is modeled with two players: the defender and the attacker. The objective of the attacker is to reach the target located in a known region, while avoiding the defender. On the other hand, the objective of the defender is to capture the attacker to prevent it from reaching the target. Here, the problem is to derive control strategies, as continuous trajectories, for the defender to win the game; however, with small changes, the method can be analogously modified for the path planning of the attacker.

Consider the attacker and defender with following motion dynamics:

$$\dot{x}(t) = u(t) \in U \subset \mathbb{R}^2 \tag{1}$$

where $x(t) \in P \subset \mathbb{R}^2$ is the position of the robot within the game domain and $u(t)$ is the control input.

Also, the game region $P \subset \mathbb{R}^2$ is assumed to be a convex polygon partitioned into disjoint cells $P_{i,j}$ such that:

$$P = \bigcup_{i\in\{1,\cdots,n\} j\in\{1,\cdots,m\}} P_{i,j}$$
$$[P_{i,j} \cap P_{l,k} = \varnothing \Leftrightarrow (i,j) \neq (l,k)] \tag{2}$$

Initially, the robots are in either a specific region or a group of possible initial regions.

What is needed further to describe the problem of robot motion planning in reach-avoid game, is information about the status of the game. It is assumed that robots have full observability of the position of each other. Similar to [18], we define vectors $\mathcal{A}$ and $\mathcal{D}$ that contain the information about the position of the attacker and the defender, respectively:

$$\mathcal{A} = \{a_{i,j}\} \quad , \quad i \in 1,\ldots,n;\ j \in 1,\ldots,m$$
$$\mathcal{D} = \{d_{i,j}\} \quad , \quad i \in 1,\ldots,n;\ j \in 1,\ldots,m \tag{3}$$

where at each step of the game, only one of the elements of these vectors is true and the rest are false. The index of the true element is equivalent to the index of the cell where the robots are located.

A snapshot of a reach-avoid game over an $n \times m$ partitioned environment is shown in Figure 1, in which the defender is initially in region $P_{n,1}$, the attacker is in region $P_{2,1}$, and the target is located at region $P_{1,m-1}$.

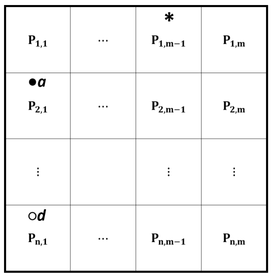


Figure 1. The configuration of a reach-avoid game.

Therefore, the reach-avoid problem, which we are trying to solve in this paper, can be defined as follows:

Problem 1. *Consider the defender and attacker robots with the given motion dynamics in (1). For a reach-avoid game defined over the game region described in (2) and portrayed in Figure 1, design a controller to obtain trajectory* $x(t) \in \mathbb{R}^2$ *satisfying the objective for the defender, which is to prevent attacker from reaching the target cell.*

3. Players' Behaviors

To capture the desired behavior of the defender and the adversarial behavior of the attacker, we use linear temporal logic (LTL) formulas [17]. The LTL formulas (φ) are constructed over a finite set of atomic propositions (Σ) using the standard boolean operators (negation ($\neg$), disjunction ($\bigvee$), conjunction ($\bigwedge$)), and the temporal operators (next ($\bigcirc$), until ($\mathcal{U}$), eventually ($\Diamond$) and always ($\Box$)).

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi\, \mathcal{U} \varphi \tag{4}$$

Because of the dynamic nature of reach-avoid problem, we use a reactive formula in the form of Generalized Reactivity(1) (GR(1)) specifications [19] to specify the reactive dynamics of the Reach-avoid game, as follows:

$$\varphi = \varphi_a \rightarrow \varphi_d \tag{5}$$

where φ_a contains all the assumptions about the attacker and its adversarial behavior, while φ_d represents the assumptions on the defender and its desired behavior.

Now, similar to [18], we can define LTL formulas φ_a and φ_d for the attacker and defender with the proposition vectors $\mathcal{A}$ and $\mathcal{D}$ in the following structure:

$$\varphi_{\alpha\in\{a,d\}} = \varphi_i^\alpha \bigwedge \varphi_t^\alpha \bigwedge \varphi_r^\alpha \bigwedge \varphi_g^\alpha \tag{6}$$

where:

- φ_i^α, $\alpha \in \{a,d\}$, are boolean formulas characterizing the *initial positions* of the attacker and the defender.
- $\varphi_t^\alpha = \varphi_{t,p}^\alpha \wedge \varphi_{t,win}^\alpha$, $\alpha \in \{a,d\}$, represent the *transition rules* of robots over the partitioned space based on the information of the game at each step, where
  - $\varphi_{t,p}^\alpha$, $\alpha \in \{a,d\}$, describe possible transitions of robots during the game. Since the attacker is not under our control, $\varphi_{t,p}^a$ describes its all possible transitions. However, for the defender, $\varphi_{t,p}^d$ will include only those transitions that help the defender to compete with the attacker and if the solution is feasible, win the game.
  - $\varphi_{t,win}^\alpha$, $\alpha \in \{a,d\}$, describe those terminating transitions for the players in case the game is finished (either the defender successfully captures the attacker, or the attacker reaches the target).
- φ_r^α, $\alpha \in \{a,d\}$, describe the *requirements* that at each time only one robot proposition can be true as each robot cannot be in two regions at the same time.
- $\varphi_g^\alpha = \bigwedge \Box\Diamond C_i^\alpha$, $\alpha \in \{a,d\}$, represent the overall objective for the robots, where C_i^α are boolean formulas.

We now need to define all sub-formulas of φ_a and φ_d in (5).

3.1. Players' Decision-making

We propose to utilize the game theory to help the defender to optimally make its decisions during the game. This can be done by constructing a finite two-player zero-sum game to obtain the best decision at each step according to the status of the game at that time. The objective function of this game is defined as:

$$L^k(u_a^{k+1}, u_d^{k+1}) = \begin{cases} \infty & \text{if } x_a^{k+1} = x_d^{k+1} \\ 0 & \text{if } x_a^{k+1} = x_t \\ \frac{\|x_a^{k+1} - x_t^{k+1}\|}{\|x_a^{k+1} - x_d^{k+1}\|} & \text{otherwise} \end{cases} \tag{7}$$

where x_t is the position of the target, and $\{u_a^{k+1}, u_d^{k+1}\}$ is the strategy of the attacker and the defender based on their current positions x_a^k and x_d^k at k^{th} step. In this zero-sum game, the players share the same objective function (7). However, the attacker wants to minimize the objective function, while the defender wants to maximize it. By using this game configuration, the defender will try to find its best decision at each step to avoid the attacker from reaching the target region. If the attacker and the defender are in the same region (the attacker is captured by the defender), or the attacker avoids the defender and reaches to the target, the game is over.

According to [21], for any finite two-player zero-sum game in matrix form, there is at least one security decisions for the players to make, which assures that at each step, the decision of robots cannot be empty:

Theorem 1. *[21] In every matrix game $M = \{m_{ij}\}$, there exists at least one security strategy for each player.*

Example 1. *Consider the reach-avoid game configuration shown in Figure 2. The game environment is partitioned into 9 disjoint convex regions. Assume that the target is located in P_{23}, and the initial positions of the defender and the attacker are P_{31} and P_{11}, respectively. Requiring to stay within the game domain and assuming that the players can only go to their vertical or horizontal neighbors, and not to diagonal ones, both players have two options: the defender can go either right (P_{32}) or up (P_{21}), and the attacker has to go either right (P_{12}) or down (P_{21}). Using the objective function in (7), this game is modeled in a matrix form as the following:*

		attacker	
		Right	Down
defender	Right	0.707	1.414
	Up	1	$+\infty$

The Nash equilibrium decision for this game is $(Right, Up)$ which is obtained by the following computations:

$$\begin{aligned} \min\max\{0.707, 1.414, 1, +\infty\} = 1 \\ \max\min\{0.707, 1.414, 1, +\infty\} = 1 \end{aligned} \tag{8}$$

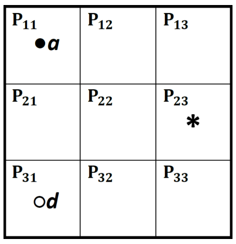


Figure 2. Reach-avoid game configuration of Example 1

For a game environment with $n \times m$ regions there are totally $(nm(nm-2))$ zero-sum games that have to be solved off-line, that will be used later for the on-line implementation of the algorithm. In fact, each of these zero-sum games is representing a particular stage of the original reach-avoid game, in which the defender has to make the best possible decision about its next action based on the current information of the game. Once these zero-sum games were solved, the results can be used for specifying the possible transitions of the defender in φ_t^α as shown in (17). For these zero-sum games, which are played at a particular time during the reach-avoid scenario, we assume both players have full information about the position of each other. By considering the fact that robots can transit just to the adjacent regions, they also are aware of the alternatives of each other. Although when we solve these zero-sum games, we assume both players make their decisions simultaneously, when we implement the proposed symbolic reactive control method, the attacker does not necessary follow these decisions, but the defender is forced to look at the attacker position at each time, and follow these decisions asynchronously with the attacker. This is in accordance with the logic of reach-avoid problem, where the defender decisions depend on its opponent behaviors.

3.2. LTL Task Specification

We now need to precisely define all LTL formulas needed to construct (6). To illustrate the proposed idea, we define the formulas for the problem in Example 1.

3.2.1. Attacker's Behavior. The attacker's behavior can be described by $\varphi_a = \varphi_i^a \wedge \varphi_t^a \wedge \varphi_r^a \wedge \varphi_g^a$. To determine φ_i^a for Example 1, we know that the attacker is initially in cell P_{11}, and hence, its variable a_{11} in vector $\mathcal{A}$ is true and the other variables are false. This can be shown in the formula φ_i^a as following:

$$\varphi_i^a = (a_{11} \wedge \neg a_{12} \wedge \cdots \wedge \neg a_{33}) \tag{9}$$

In general, the initial position of the robots can be expressed either as a specific initial region or a set of possible initial regions.

φ_t^a represents the transition rules of the attacker over the partitioned space based on the information of the game at each step.

$$\varphi_t^a = \varphi_{t,p}^a \wedge \varphi_{t,win}^a \tag{10}$$

To specify $\varphi^a_{t,p}$, since the attacker is not under our control, the possible transitions for the attacker at each step could be any of the adjacent cells of its current cell. For the sake of simplicity, we assume that players can only go to their vertical or horizontal neighbors and not to the diagonal ones. Therefore, for Example 1, $\varphi^a_{t,p}$ will be as follows:

$$\begin{aligned}\varphi^a_{t,p} =&\Box[(a_{11} \rightarrow (\bigcirc a_{11} \vee \bigcirc a_{12} \vee \bigcirc a_{21}))\\ &\bigwedge(a_{12} \rightarrow (\bigcirc a_{12} \vee \bigcirc a_{11} \vee \bigcirc a_{22} \vee \bigcirc a_{13}))\\ &\bigwedge(a_{13} \rightarrow (\bigcirc a_{13} \vee \bigcirc a_{23} \vee \bigcirc a_{12}))\\ &\bigwedge(a_{21} \rightarrow (\bigcirc a_{21} \vee \bigcirc a_{11} \vee \bigcirc a_{22} \vee \bigcirc a_{31}))\\ &\bigwedge(a_{22} \rightarrow (\bigcirc a_{22} \vee \bigcirc a_{21} \vee \bigcirc a_{12} \vee \bigcirc a_{32} \vee \bigcirc a_{23}))\\ &\bigwedge(a_{23} \rightarrow (\bigcirc a_{23} \vee \bigcirc a_{13} \vee \bigcirc a_{22} \vee \bigcirc a_{33}))\\ &\bigwedge(a_{31} \rightarrow (\bigcirc a_{31} \vee \bigcirc a_{21} \vee \bigcirc a_{32}))\\ &\bigwedge(a_{32} \rightarrow (\bigcirc a_{32} \vee \bigcirc a_{31} \vee \bigcirc a_{22} \vee \bigcirc a_{33}))\\ &\bigwedge(a_{33} \rightarrow (\bigcirc a_{33} \vee \bigcirc a_{23} \vee \bigcirc a_{33}))] \qquad (11)\end{aligned}$$

To determine $\varphi^a_{t,win}$, it is assumed that if the attacker wins the game, or in other words, if it reaches the target cell, P_{23}, it must stay there. This is specified as below:

$$\varphi^a_{t,win} = \Box(a_{23} \rightarrow \bigcirc a_{23}) \qquad (12)$$

φ^a_r describes that at each step, only one of the propositions of attacker can be true. This assumption can be achieved by disjunction of 9 temporal formulas as follows:

$$\begin{aligned}\varphi^a_r = &\Box[(\bigcirc a_{11} \wedge \bigcirc\neg a_{12} \wedge \cdots \wedge \bigcirc\neg a_{33})\\ &\bigvee\ (\bigcirc\neg a_{11} \wedge \bigcirc a_{12} \wedge \bigcirc\neg a_{13} \wedge \cdots \wedge \bigcirc\neg a_{33})\\ &\bigvee\ \vdots\\ &\bigvee\ (\bigcirc\neg a_{11} \wedge \bigcirc\neg a_{12} \wedge \cdots \wedge \bigcirc a_{33})] \qquad (13)\end{aligned}$$

No particular goal we can consider for the attacker as it is out of our control leading to:

$$\varphi^a_g = \Box\Diamond True \qquad (14)$$

3.2.2. Defender's Behavior. The defender's behavior can be described by $\varphi_d = \varphi^d_i \wedge \varphi^d_t \wedge \varphi^d_r \wedge \varphi^d_g$.

Since in Example 1, the defender is initially in cell P_{31}, φ^d_i for this example will be:

$$\varphi^d_i = (\neg d_{11} \wedge \neg d_{12} \cdots \wedge \neg d_{23} \wedge d_{31} \wedge \neg d_{32} \wedge \neg d_{33}) \qquad (15)$$

φ^d_t captures the transitions of the defender:

$$\varphi^d_t = \varphi^d_{t,p} \wedge \varphi^d_{t,win} \qquad (16)$$

Since the defender is under our control, we design the controller in a way that it plays its best action to win the game if the solution is feasible. The best actions of the defender have to be found through zero-sum games played in Section 3.1 in an off-line way, and then, have to be embedded in (16) for the online implementation. In Example 1, the zero-sum game is solved for the case that the defender and attacker are in regions P_{31} and P_{11}, respectively, for which, the best play of the defender was found to be d_{21}, meaning that we have to have $\Box[(d_{31} \wedge a_{11}) \rightarrow \bigcirc d_{21})$ as a part of $\varphi^d_{t,p}$. We can solve the zero-sum games for all other possible places of the attacker and the defender to find best plays of the defender, which form the following formula:

$$\begin{aligned}&\varphi^d_{t,p} = \Box[\\ &\bigwedge((d_{11} \wedge a_{12}) \rightarrow \bigcirc d_{12})\\ &\bigwedge((d_{11} \wedge a_{31}) \rightarrow \bigcirc d_{21})\\ &\bigwedge((d_{11} \wedge (a_{13} \vee a_{21} \vee a_{22} \vee a_{32} \vee a_{33})) \rightarrow (\bigcirc d_{21} \vee \bigcirc d_{21}))\\ &\bigwedge((d_{12} \wedge a_{11}) \rightarrow \bigcirc d_{13})\\ &\bigwedge((d_{12} \wedge (a_{13} \vee a_{22} \vee a_{33})) \rightarrow \bigcirc(d_{11} \vee d_{22} \vee d_{13}))\\ &\bigwedge((d_{12} \wedge (a_{21} \vee a_{31} \vee a_{32})) \rightarrow \bigcirc d_{22})\\ &\bigwedge(d_{13} \rightarrow \bigcirc d_{12})\\ &\bigwedge((d_{21} \wedge a_{11}) \rightarrow (\bigcirc d_{11} \vee \bigcirc d_{22}))\\ &\bigwedge((d_{21} \wedge (a_{12} \vee a_{31} \vee a_{32} \vee a_{13} \vee a_{22} \vee a_{33})) \rightarrow \bigcirc d_{22})\\ &\bigwedge((d_{22} \wedge (a_{11} \vee a_{12})) \rightarrow \bigcirc d_{12})\\ &\bigwedge((d_{22} \wedge a_{21}) \rightarrow \bigcirc d_{21})\\ &\bigwedge((d_{22} \wedge (a_{13} \vee a_{32})) \rightarrow \bigcirc d_{32})\\ &\bigwedge((d_{22} \wedge (a_{13} \vee a_{22} \vee a_{33})) \rightarrow (\bigcirc d_{12} \vee \bigcirc d_{21} \vee \bigcirc d_{32}))\\ &\bigwedge((d_{31} \wedge a_{11}) \rightarrow \bigcirc d_{21})\\ &\bigwedge((d_{31} \wedge a_{32}) \rightarrow \bigcirc d_{32})\\ &\bigwedge((d_{31} \wedge (a_{12} \vee a_{13} \vee a_{22} \vee a_{21} \vee a_{33})) \rightarrow (\bigcirc d_{21} \vee \bigcirc d_{32}))\\ &\bigwedge((d_{32} \wedge \neg a_{32} \wedge \neg a_{23}) \rightarrow \bigcirc d_{22})\\ &\bigwedge((d_{33} \wedge \neg a_{33} \wedge \neg a_{32}) \rightarrow \bigcirc d_{32})] \qquad (17)\end{aligned}$$

If the defender wins the game, which means that it captures the attacker(both the defender and the attacker are in the same region), defender must stay in its current position. This is specified as below:

$$\begin{aligned}\varphi^d_{t,win} = &\Box[((d_{11} \wedge a_{11}) \rightarrow \bigcirc d_{11})\\ &\bigwedge((d_{12} \wedge a_{12}) \rightarrow \bigcirc d_{12})\\ &\bigwedge((d_{13} \wedge a_{13}) \rightarrow \bigcirc d_{13})\\ &\bigwedge((d_{21} \wedge a_{21}) \rightarrow \bigcirc d_{21})\\ &\bigwedge((d_{22} \wedge a_{22}) \rightarrow \bigcirc d_{22})\\ &\bigwedge((d_{31} \wedge a_{31}) \rightarrow \bigcirc d_{31})\\ &\bigwedge((d_{32} \wedge a_{32}) \rightarrow \bigcirc d_{32})\\ &\bigwedge((d_{33} \wedge a_{33}) \rightarrow \bigcirc d_{12})] \qquad (18)\end{aligned}$$

$\varphi^d_{t,s}$ includes the assumptions on defender's position saying that it can only be in one region at each time. This is realized by letting only one of the variables in vector $\mathcal{D}$ to be true at each time as follows:

$$\begin{aligned}\varphi^d_{t,s} = &\Box[(\bigcirc d_{11} \wedge \bigcirc\neg d_{12} \wedge \cdots \wedge \bigcirc\neg d_{33})\\ &\bigvee(\bigcirc\neg d_{11} \wedge \bigcirc d_{12} \wedge \bigcirc\neg d_{13} \wedge \cdots \wedge \bigcirc\neg d_{33})\\ &\bigvee\ \vdots\\ &\bigvee(\bigcirc\neg d_{11} \wedge \bigcirc\neg d_{12} \wedge \cdots \wedge \bigcirc d_{33})] \qquad (19)\end{aligned}$$

The objective of the defender is to eventually capture the attacker as specified in (20):

$$\begin{aligned}\varphi_g^d = \Box\Diamond[&(a_{11} \wedge d_{11}) \vee (a_{12} \wedge d_{12}) \vee (a_{13} \wedge d_{13}) \\ &\vee (a_{21} \wedge d_{21}) \vee (a_{22} \wedge d_{22}) \vee (a_{31} \wedge d_{31}) \\ &\vee (a_{32} \wedge d_{32}) \vee (a_{33} \wedge d_{33})] \end{aligned} \tag{20}$$

Putting everything together, the overall specification for this reach-avoid problem is $\varphi = (\varphi_a \rightarrow \varphi_d)$, where φ_a and φ_d are defined in Equations 6 and (9-20).

3.3. Synthesize of Automaton for LTL Specification

The formula φ is a reactive formula in the form of Generalized Reactivity(1) (GR(1)) specifications [19]. In [18], an algorithm is introduced to check if the GR(1) formula φ is realizable. If yes, then the algorithm synthesizes an automaton which satisfies this formula. The synthesis process is basically a game played between two players and the winning condition is $\varphi = (\varphi_a \Rightarrow \varphi_d)$. The defender will win the game if φ is true, and the attacker will win the game if φ is falsified. We refer readers to [18] for detail steps. The algorithm generates an input/output automaton [22], which can be modelled by a tuple $G = (R, r_0, \mathcal{A}, \mathcal{D}, \delta, h)$ where:

- R is the set of states,
- r_0 is the initial state,
- $\mathcal{A}$ is the set of input (attacker) propositions,
- $\mathcal{D}$ is the set of output (defender) propositions,
- $\delta : R \times 2^{\mathcal{A}} \rightarrow 2^R$ is the transition relation,
- $h : Q \rightarrow 2^{\mathcal{D}}$ is the output function.

If the automaton is at state r_k, a decision of the attacker a_k, as the input to G, causes the system to transit from r_k to r_{k+1}, denoted by $\delta(r_k, a_k) = r_{k+1}$, and shown by $r_k \xrightarrow{a_k} r_{k+1}$. Starting from r_0, based on the observed sequence of decisions of the attacker, $\{a_0, a_1, \cdots, a_n\}$, the automaton G transits as $r_0 \xrightarrow{a_0} r_1 \xrightarrow{a_1} r_2 \ldots \xrightarrow{a_n} s_{n+1}$. The generated output then will be $h(r_0), h(r_1), ..., h(r_n) = d_0, d_1, ..., d_n$, with $h(r_0) = \varnothing$ and $h(r_k) = d_{k-1}$. The formula φ_a, has considered all possible transitions (decisions) of the attacker. Therefore, all admissible sequences of attacker decisions result in sequences of defender's actions which are guaranteed to win the game.

Followed this procedure, the automaton is synthesized for Example 1, and it is shown in Figure 3. For instance, for a sequence of attacker decisions $a_{11}, a_{12}, a_{22}, a_{32}, a_{33}$, based on the synthesized automaton G, the defender plays $d_{31}, d_{32}, d_{33}, d_{22}, d_{33}$, and wins the game.

4. CONTROLLER SYNTHESIS

As discussed in the previous section, by synthesizing the automaton G, for any sequences of attacker's decisions, the defender takes a set of actions, which can be considered as a discrete path. We now use our developed hybrid controller [23] to convert the generated discrete path to smooth continuous signals driving the defender over the partitioned space.

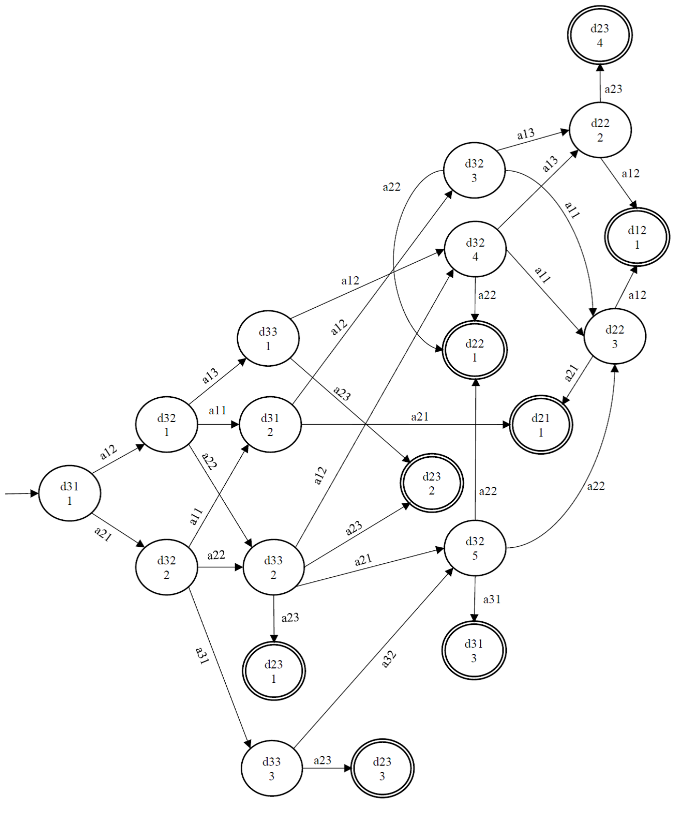
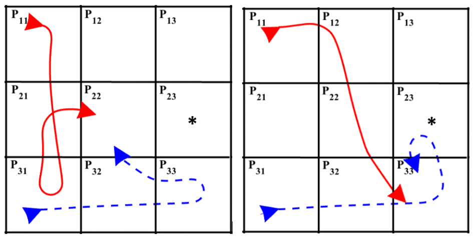


Figure 3. Synthesized automaton for the specification of example 1

Due to the bisimulation relation between the original system and its abstracted partitioned model, proven in [23], it can be guaranteed that the generated continues signals preserves the properties of the discrete paths.

The controller synthesis results for the scenario of Example 1, is demonstrated in Figure 4 and the designed trajectories for defender against different behaviors of attacker has been shown. Based on the specification and the derived automaton G, the control strategy is computed for the initial position of region P_{11} for the attacker, and region P_{31} for the defender. In 4(a) we consider that the attacker behaves as shown with solid line, and visits the regions $\{P_{11}, P_{12}, P_{22}, P_{32}, P_{33}\}$. The defender collects the information about the attacker position during the game which is equivalent to the discrete path $a_{11}, a_{12}, a_{22}, a_{32}, a_{33}$. Using the proposed symbolic motion planning approach to obtain a strategy against the attacker behavior, the synthesized automaton G generates the discrete commands (optimal actions) for the defender in the form of $d_{31}, d_{32}, d_{33}, d_{22}, d_{33}$, which are then translated to continuous signals, driving the defender to go to $\{P_{31}, P_{32}, P_{22}, P_{32}, P_{33}\}$ in order. As it can be seen, the proposed algorithm is capable of creating a winning strategy, and when the defender reaches P_{33}, the defender captures the attacker and wins the game. The discrete path of the defender can be verified in automaton G, shown in Figure 3, and its corresponding continuous



(a) Attacker path: $P_{11}, P_{12}, P_{22}, P_{32}, P_{33}$ (b) Attacker path: $P_{11}, P_{21}, P_{31}, P_{21}, P_{22}$

Figure 4. Demonstration of the results of the proposed algorithm for a reach-avoid scenario presented in Example 1. The results show strategies of defender against different behaviors of attacker. Solid line represents the behavior of attacker and dotted line shows the resulted strategy of defender against the behavior of attacker.

trajectory is depicted in the Figure 4. Figure 4(b) shows the result of same procedure for a different attacker behavior, again confirming that the defender can win the game.

5. CONCLUSION

In this paper, we proposed a novel, computationally-effective symbolic framework to construct hybrid controllers for autonomous robots involved in reach-avoid scenarios. The proposed framework provided a formal algorithm to obtain continuous trajectories for defending robot. In contrast to earlier approaches, this methodology considers less restrictions on the robot motion and assumes no knowledge about the model of the opponent is available. Moreover, due to the abstract nature of the proposed symbolic approach, it has less computational costs. The reactive nature of the proposed framework allows the defender to automatically react to the attacker's actions and make best decisions. Our future work include the extension of the proposed framework to more complex scenarios and environments.

Acknowledgment

The last two authors would like to acknowledge the support from Air Force Research Laboratory and OSD for sponsoring this research under agreement number FA8750-15-2-0116. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory, OSD or the U.S. Government.

References

[1] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 22, no. 2, pp. 224–241, 1992.

[2] C. Torras, "Robot motion planning: A survey," in *Teleoperation: Numerical Simulation and Experimental Validation*. Springer, 1992, pp. 27–39.

[3] K. Margellos and J. Lygeros, "Hamilton-jacobi formulation for reach-avoid problems with an application to air traffic management," in *American Control Conference (ACC), 2010*. IEEE, 2010, pp. 3045–3050.

[4] J. S. McGrew, J. P. How, B. Williams, and N. Roy, "Air-combat strategy using approximate dynamic programming," *Journal of guidance, control, and dynamics*, vol. 33, no. 5, pp. 1641–1654, 2010.

[5] M. Aigner and M. Fromme, "A game of cops and robbers," *Discrete Applied Mathematics*, vol. 8, no. 1, pp. 1–12, 1984.

[6] D. Bhadauria, K. Klein, V. Isler, and S. Suri, "Capturing an evader in polygonal environments with obstacles: The full visibility case," *The International Journal of Robotics Research*, p. 0278364912452894, 2012.

[7] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, "A visibility-based pursuit-evasion problem," *International Journal of Computational Geometry & Applications*, vol. 9, no. 04n05, pp. 471–493, 1999.

[8] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view," *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 299–315, 2006.

[9] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[10] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.

[11] Q. Zhu, "Hidden markov model for dynamic obstacle avoidance of mobile robot navigation," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 390–397, 1991.

[12] A. Ismail, A. Sheta, and M. Al-Weshah, "A mobile robot path planning using genetic algorithm in static environment," *Journal of Computer Science*, vol. 4, no. 4, pp. 341–344, 2008.

[13] H. Huang, J. Ding, W. Zhang, and C. J. Tomlin, "A differential game approach to planning in adversarial scenarios: A case study on capture-the-flag," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1451–1456.

[14] ——, "Automation-assisted capture-the-flag: A differential game approach," *Control Systems Technology, IEEE Transactions on*, vol. 23, no. 3, pp. 1014–1028, 2015.

[15] X. D. Koutsoukos, P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon, "Supervisory control of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1026–1049, 2000.

[16] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science, 18th Annual Symposium on*. IEEE, 1977, pp. 46–57.

[17] E. A. Emerson, "Temporal and modal logic, handbook of theoretical computer science (jan van leeuwen, ed.), vol. b," 1990.

[18] N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive (1) designs," in *Verification, Model Checking, and Abstract Interpretation*. Springer, 2006, pp. 364–380.

[19] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive (1) designs," *Journal of Computer and System Sciences*, vol. 78, no. 3, pp. 911–938, 2012.

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

[21] T. Basar and G. J. Olsder, *Dynamic noncooperative game theory*. Siam, 1999, vol. 23.

[22] N. Lynch and M. Tuttle, "An introduction to input/output automata," *Technical Report MIT/LCS/TM-373, MIT Laboratory for Computer Science*, 1989.

[23] A. Karimoddini and H. Lin, "Hierarchical hybrid symbolic robot motion planning and control," *Asian Journal of Control*, vol. 17, no. 1, pp. 23–33, 2015. [Online]. Available: http://dx.doi.org/10.1002/asjc.1027