A Symbolic Approach for Multi-target Dynamic Reach-avoid Problem

Laya Shamgah, Tadewos G. Tadewos, Ali Karimoddini, and Abdollah Homaifar

***Abstract*— This paper develops a symbolic technique for the control and path planning of autonomous vehicles in adversarial environments. We consider the objective of the autonomous vehicle as visiting a set of targets sequentially, while avoiding a dynamic adversarial (defending) vehicle. To address this multi-target dynamic reach-avoid path planning problem, this paper proposes a novel efficient hybrid, correct-by-design, symbolic solution. For this purpose, the dynamic interactions between the vehicles and their logical behaviors are captured in the form of General Reactivity(1), and then a hybrid control system is constructed to generate winning trajectories, which satisfy the objective of the attacking autonomous vehicle. The proposed technique is verified through several simulations and the results show the effectiveness of the developed algorithm.**

Keywords: reach-avoid, symbolic planning, motion planning, hybrid control, reactive synthesis, multi-target

I. INTRODUCTION

During the last decays, many types of autonomous vehicles have been developed such as autonomous cars [1], [2] or unmanned aerial, sea surface, or underwater vehicles [3]–[6]. The first step toward constructing the autonomy architecture of these vehicles is to develop low-level controllers that stabilizes them and take care of their attitude control. Then, an effective motion planning mechanism is needed to enable these vehicles to be autonomously driven in a given environment. Motion planning problem in the most basic form is defined as the path planning for vehicles to transit from an initial position to the desired position, while avoiding obstacles, all in a static setup. In this sense, motion planning in a static environment has been widely addressed in the literature [7]–[11]. Recent technological advances in autonomous vehicles, however, have enabled the development of advanced motion planning techniques to accomplish more sophisticated tasks in more complex and dynamic environments. Moving toward motion planning in dynamic environments, a common approach is to formulate these problems as a pursuit-evasion game [12]–[15], and use different techniques such as probabilistic formulations [16]–[22] or differential games [23], [24]. These techniques commonly suffer from high computation cost and are not flexible enough to be extended to more complex setups. For example, consider the planning and control for an attacking vehicle in a complex scenario, requiring the vehicle to visit multiple target regions in a single attempt, while avoiding an adversarial competitive defending vehicle, which is beyond these existing techniques.

Therefore, in this paper, we address the Multi-target Dynamic Reach-avoid problem (MTDRA), in which the attacker aims at attacking a sequence of targets in particular order, while avoiding the defender as a dynamic adversarial obstacle. To address this problem, we employ symbolic planning techniques over a partitioned environment within the context of hybrid supervisory control theory [25]–[30]. To capture the requirements of the attacker and the defender as well as their conflicting objectives, we use linear temporal logic (LTL) [31], [32] which is a semi-natural language, capable of describing complex high-level tasks. Reactive synthesis techniques [33]–[35] are then applied to capture the competing interactions between the attacker and the defender and to achieve winning strategies for the attacker, satisfying the desired specifications and requirements. Finally, a hybrid controller is designed to calculate continuous control signals for driving the attacker vehicle over the partitioned region. The paper provides descriptive examples to illustrate the implementation of the proposed algorithm. Simulation results are provided to verify the efficacy of the proposed method.

The rest of the paper is organized as follows: In Section II, the MTDRA problem is formulated. Section III describes the procedure to capture vehicles' requirements and specifications. Section IV explains synthesizing discrete and continuous controllers for the attacking vehicle involved in MTDRA scenario. Finally, section V concludes the paper.

II. PROBLEM FORMULATION

The vehicle's dynamic in multi-target dynamic reach-avoid (MTDRA) scenario, is considered as follows:

$$\dot{x}(t) = u(t) \tag{1}$$

where $x(t) \in P \subset \mathbb{R}^2$ is the position of the vehicle within a bounded 2D operation region P, and $u(t) \in U \subset \mathbb{R}^2$ is the control input. We assume that both vehicles move with maximum velocity of V_m.

The operation takes place in a region represented as a bounded set $P \subset \mathbb{R}^2$. To manage the complexity of the problem and to employ symbolic motion planning techniques, we partition the environment P into finite disjoint rectangular regions P_{ij} such that:

$$P = \bigcup_{i\in\{1,\cdots,n\},j\in\{1,\cdots,m\}} P_{ij} \tag{2}$$

where $P_{ij} \cap P_{lk} = \emptyset$ for all $(i,j) \neq (l,k)$.

To capture real-time discrete position of both vehicles over these partitions, we define attacker Boolean sensors a_{ij} and defender Boolean sensors d_{ij}, over partitions, P_{ij},

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: +13362853313).

$i \in \{1, \cdots, n\}$, $j \in \{1, \cdots, m\}$. The truth evaluation of these Boolean propositions changes whenever the vehicles move to a different partition. Since they can only be in one partition at a time, the sensor variable with the same index of that partition, where they are located in, is $True$ and the rest are $False$. We define sets $\mathcal{A}$ and $\mathcal{D}$ containing these variables as follows:

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

We assume that the number of targets is N_t and they should be visited in a pre-determined order. For each target $k \in \{1, \cdots, N_t\}$ located in P_{i_k, j_k}, we define a Boolean proposition t_{i_k, j_k}, and the set $\mathcal{T}$, which contains all these propositions. Finally, we define a proposition "$Accomplished$" which is $True$ when the attacker visits all the targets. Having this information, we can then capture all the assumptions, requirements and objectives of the vehicles as LTL formulas in the form of General Reactivity(1).

The multi-target dynamic reach-avoid (MTDRA) problem can then be described as follows:

Problem 1: Consider an attacking vehicle with the continuous dynamics in (1) driven over the region P described in (2). Also, consider the attacker objective is to visit targets with positions $x_{t_1}, \cdots x_{t_N}$, $x_{t_k} \in \mathbb{R}^2$, in order, while avoiding the defender which tries to capture the attacker before it accomplishes its objective. Design a controller, $u(t)$, for the attacker so that the attacker's path, $x(t) \in P \subseteq \mathbb{R}^2$, satisfies its desired objective.

III. Specification of a Multi-target Reach-avoid Scenario

A. Linear Temporal Logic

The requirements and high-level constraints of the vehicles can be captured by Linear temporal logic (LTL) formulas [32]. An LTL formula, φ, is constructed over a finite set of atomic propositions, Σ, using the standard boolean operators (negation, $\neg$, disjunction, $\bigvee$), and the temporal operators (next $\bigcirc$, until $\mathcal{U}$).

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

Other Boolean operators (such as conjunction, $\bigwedge$, and implication, $\rightarrow$) and other temporal operators (such as eventually, $\Diamond$, and always, $\Box$) can be constructed based on the aforementioned list of operators in (4).

Consider Σ as the set of all propositions and $\sigma = \sigma_0, \sigma_1, \ldots$ as a sequence of truth assignments to propositions in Σ, where σ_i is the set of propositions at position i. We say $\sigma, i \vDash \varphi$ if φ is true at position $i \geq 0$ of σ.

A special class of LTL formulas is General Reactivity(1) (or simply GR(1)), which provides an appropriate format to describe the specifications in a dynamic environment, where there are interactions between the system and its environment [33], [34]. A GR(1) formula can be described as:

$$\varphi = \varphi_e \rightarrow \varphi_s \tag{5}$$

where φ_e contains all the assumptions about the environment, and φ_s represents the assumptions on the system and its desired behavior. Formulas φ_e and φ_s are the conjunction of some sub-formulas in all three forms of B, $\Box B$ and $\Box\Diamond B$, where B could be a Boolean or temporal formula.

B. Assumptions and Requirements of the Vehicles in a MTDRA Scenario

Considering the defender vehicle as part of the environment of the attacker, we use GR(1) over the proposition set $\Sigma = \mathcal{A} \cup \mathcal{D} \cup \mathcal{T} \cup \{Accomplished\}$ to describe the MTDRA problem in the form of:

$$\varphi = \varphi_d \rightarrow \varphi_a \tag{6}$$

where φ_d and φ_a capture the assumptions and requirements of the defender and the attacker, respectively. Formulas φ_d and φ_a can be represented as the conjunction of five subformulas:

$$\varphi_d = \varphi^d_{init} \wedge \varphi^d_{sing} \wedge \varphi^d_{term} \wedge \varphi^d_{rul} \wedge \varphi^d_{obj} \tag{7}$$

$$\varphi_a = \varphi^a_{init} \wedge \varphi^a_{sing} \wedge \varphi^a_{term} \wedge \varphi^a_{rul} \wedge \varphi^a_{obj} \wedge \varphi^a_{opt} \tag{8}$$

where for $r \in \{a, d\}$ we have,

- φ^r_{init} describes the initial value of all propositions in $\mathcal{A} \cup \mathcal{D} \cup \mathcal{T} \cup \{Accomplished\}$,
- φ^r_{sing} describes the *singularity requirement*, which requires each vehicle to be only in one of the partitions,
- φ^r_{term} describes the *termination condition*, which requires that no change occurs after the game is over (one of the vehicles reaches its objective),
- φ^r_{rul} describes the *transition rules*,
- φ^r_{obj} describes the *objectives* of the vehicles,
- φ^a_{opt} describes the *optimal discrete transition rules* for the attacker.

The following example explains how to capture these formulas for the vehicles in a multi-target dynamic reach-avoid scenario described in Problem 1.

Example 1: We consider the environment that is partitioned into 68 disjoint cells with targets T_1, T_2, and T_3 located in regions P_{23}, P_{35} and P_{57}, respectively. Assume that the initial positions of the defender and the attacker are P_{41} and P_{11}, respectively. This configuration has been shown in Fig. 1.

The defender is initially in partition P_{41}. Therefore φ^d_{init} will be:

$$\varphi^d_{init} = d_{41} \wedge (\neg d_{11} \wedge \cdots \neg d_{35} \wedge \neg d_{42} \wedge \neg d_{68}) \tag{9}$$

The attacker is initially in P_{11}, and the first target to be visited is T_1. Therefore φ^a_{init} will be:

$$\begin{aligned} \varphi^a_{init} = a_{11} \wedge (\neg a_{12} \wedge \cdots \wedge \neg a_{68}) \wedge t_1 \wedge \neg t_2 \wedge \neg t_3 \\ \wedge \neg Accomplished \end{aligned} \tag{10}$$

According to the *singularity requirement*, each vehicle can physically be in only one partition of P at a time, and hence, only one of the vehicles vector propositions can be $True$. The singularity requirement for the defender can be captured

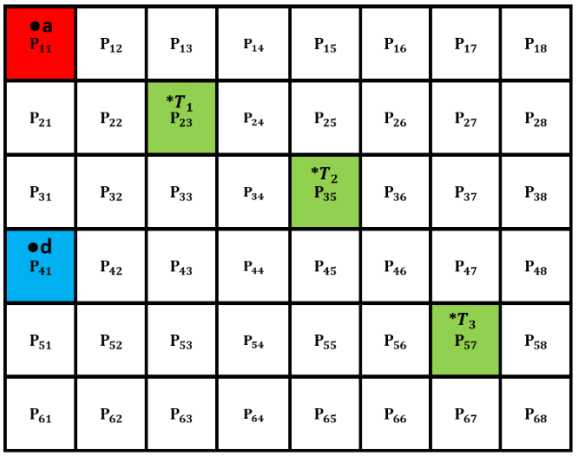


Fig. 1. Configuration of an example of multi-target dynamic reach-avoid scenario. The targets are in regions P_{23}, P_{35} and P_{57}, and the attacker and the defender are initially in regions P_{11} and P_{41}, respectively.

by the following temporal formula:

$$\begin{aligned}\varphi^d_{sing} =&\Box[(d_{11} \wedge (\neg d_{12} \wedge \neg d_{13} \wedge \cdots \wedge \neg d_{68})) \\ &\vee (d_{12} \wedge (\neg d_{11} \wedge \neg d_{13} \wedge \cdots \wedge \neg d_{68})) \\ &\quad\vdots \\ &\vee (d_{67} \wedge (\neg d_{11} \wedge \neg d_{12} \wedge \cdots \wedge \neg d_{66} \wedge \neg d_{68})) \\ &\vee (d_{68} \wedge (\neg d_{11} \wedge \neg d_{12} \wedge \cdots \wedge \neg d_{67}))]\end{aligned} \tag{11}$$

Similarly, for the attacker, we have:

$$\begin{aligned}\varphi^a_{sing} =&\Box[(a_{11} \wedge (\neg a_{12} \wedge \neg a_{13} \wedge \cdots \wedge \neg a_{68})) \\ &\vee (a_{12} \wedge (\neg a_{11} \wedge \neg a_{13} \wedge \cdots \wedge \neg a_{68})) \\ &\quad\vdots \\ &\vee (a_{67} \wedge (\neg a_{11} \wedge \neg a_{12} \wedge \cdots \wedge \neg a_{66} \wedge \neg a_{68})) \\ &\vee (a_{68} \wedge (\neg a_{11} \wedge \neg a_{12} \wedge \cdots \wedge \neg a_{67}))]\end{aligned} \tag{12}$$

Once either of the vehicles achieve its goal, the game is over and both vehicles stay at the place they are and do not make any new decision. This occurs if the attacker visits all the targets, which means $Accomplished$ is true, or if the defender captures the attacker, which happens when both the attacker and the defender are in the same region, i.e., $(a_{ij} \wedge d_{ij})$ is true for some $i \in \{1, \cdots m\}$ and $j \in \{1, \cdots n\}$. The *termination requirement* for the defender, therefore, can be described as:

$$\begin{aligned}\varphi^d_{term} = \Box[&((Accomplished \vee a_{11}) \wedge d_{11} \rightarrow \bigcirc d_{11}) \\ &\wedge ((Accomplished \vee a_{12}) \wedge d_{12} \rightarrow \bigcirc d_{12}) \\ &\quad\vdots \\ &\wedge ((Accomplished \vee a_{67}) \wedge d_{67} \rightarrow \bigcirc d_{67}) \\ &\wedge ((Accomplished \vee a_{68}) \wedge d_{68} \rightarrow \bigcirc d_{68})]\end{aligned} \tag{13}$$

and the *termination requirement* for the attacker is:

$$\begin{aligned}\varphi^a_{term} = \Box[&((Accomplished \vee (d_{11} \wedge a_{11})) \rightarrow \bigcirc a_{11}) \\ &\wedge ((Accomplished \vee (d_{12} \wedge a_{12})) \rightarrow \bigcirc a_{12}) \\ &\quad\vdots \\ &\wedge ((Accomplished \vee (d_{67} \wedge a_{67})) \rightarrow \bigcirc a_{67}) \\ &\wedge ((Accomplished \vee (d_{68} \wedge a_{68})) \rightarrow \bigcirc a_{68})]\end{aligned} \tag{14}$$

We assume that the vehicles can transit to the adjacent regions only through the edges not the vertices of the cells. So, for the transition rules of the defender we have:

$$\begin{aligned}\varphi^d_{rul} = \Box[&(d_{11} \rightarrow (\bigcirc d_{12} \vee \bigcirc d_{21})) \\ &\wedge (d_{12} \rightarrow (\bigcirc d_{11} \vee \bigcirc d_{22} \vee \bigcirc d_{13})) \\ &\wedge (d_{13} \rightarrow (\bigcirc d_{12} \vee \bigcirc d_{14} \vee \bigcirc d_{23})) \\ &\quad\vdots \\ &\wedge (d_{67} \rightarrow (\bigcirc d_{66} \vee \bigcirc d_{57} \vee \bigcirc d_{68})) \\ &\wedge (d_{68} \rightarrow (\bigcirc d_{67} \vee \bigcirc d_{58}))]\end{aligned} \tag{15}$$

Similar rules hold for the attacker. In addition, we assume that initially the attacker aims at visiting the target t_1. Once it reaches t_1, then it targets for t_2, then t_3, and finally, when it visits t_3, the proposition $Accomplished$ becomes true. Considering all these requirements for the attacker, φ^a_{rul} will be:

$$\begin{aligned}\varphi^a_{rul} = \Box[&(a_{11} \rightarrow (\bigcirc a_{12} \vee \bigcirc a_{21})) \\ &\wedge (a_{12} \rightarrow (\bigcirc a_{11} \vee \bigcirc a_{22} \vee \bigcirc a_{13})) \\ &\wedge (a_{13} \rightarrow (\bigcirc a_{12} \vee \bigcirc a_{14} \vee \bigcirc a_{23})) \\ &\quad\vdots \\ &\wedge (a_{67} \rightarrow (\bigcirc a_{66} \vee \bigcirc a_{57} \vee \bigcirc a_{68})) \\ &\wedge (a_{68} \rightarrow (\bigcirc a_{67} \vee \bigcirc a_{58}))] \\ \bigwedge \Box[&((a_{23} \wedge t_1 \wedge \neg t_2 \wedge \neg t_3) \rightarrow \bigcirc t_2) \\ &\wedge ((a_{35} \wedge t_1 \wedge t_2 \wedge \neg t_3) \rightarrow \bigcirc t_3)] \\ &\wedge ((a_{57} \wedge t_1 \wedge t_2 \wedge t_3) \rightarrow \bigcirc Accomplished)]\end{aligned} \tag{16}$$

To address problem 1, the objective of the attacker is to capture the targets in a defined order. Using LTL, this complex mission can be easily expressed as:

$$\varphi^a_{obj} = \Box\Diamond(a_{23} \wedge (\Box\Diamond(a_{35} \wedge \Box\Diamond a_{57}))) \tag{17}$$

The opponent (defending) vehicle, however, is not under our control, and hence, its objective can be trivially written as $\varphi^d_{obj} = \Box\Diamond True$.

C. Optimal Discrete Decision Making for the Attacker

The formula φ^a_{rul} in (8) describes the feasible transitions of the attacker which are based on this assumption that the players should transit to the next region through the edges of their current region. φ^a_{rul} then, is conjuncted with optimal transition rules, φ^a_{opt}. To find these optimal strategies, we propose to utilize a finite two-player zero-sum game in a

matrix form to make a decision at each step for the attacker, according to the current status of the vehicles. For this purpose, we define the game's objective function as:

$$L(x'_a, x'_d) = \begin{cases} \infty, \text{if } x'_a = x'_d \\ 0, \text{if } x'_a = x_t, x'_d \neq x_t \\ \alpha\|x'_a - x_t\| + \beta/\|x'_a - x'_d\|, \text{otherwise} \end{cases} \tag{18}$$

where, x'_a x'_d are possible next regions of the attacker and the defender, respectively; x_t is the position of the next target; $\|.\|$ is norm 2, and α and β are tuning coefficients factors.

In this game, the vehicles share the same objective function (18). However, the attacker wants to minimize this objective function, while the defender wants to maximize it. By using this game configuration, the attacker will try to make the best decision by minimizing the cost function in (18), which requires the attacker to avoid the defender by maximizing the distance between the two vehicles, $\|x'_a - x'_d\|$, and to reach the target by reducing its distance from the target, $\|x'_a - x_t\|$. If the attacker and the defender are in the same region (the attacker is captured by the defender, $x'_a = x'_d$), or the attacker reaches the target ($x'_a = x_t$), the game is over. As the first player, the attacker tries to independently and conservatively minimize its loss to make an optimal decision. To illustrate this procedure, consider the following example:

Example 2: In Example 1, consider an arbitrary step of the game in which the defender and the attacker are at P_{61} and P_{11}, respectively, and the attacker aims to visit target T_1 located in P_{23} (See Fig. 1). Let $\alpha = 1$ and $\beta = 5$. Requiring to stay within P, and assuming that the vehicles can only go to their immediate vertical or horizontal neighbors, both vehicles have two options: the defender can go to either P_{62} or P_{51}, and the attacker can go to P_{12} or P_{21}. Using the objective function in (18), this game is represented in the following matrix form:

		attacker	
		P_{12}	P_{21}
defender	P_{52}	2.25	2.58
	P_{41}	2.58	3.5

in which the optimal solution for the attacker, as the first player, will be transiting to the region P_{12} which minimizes the maximum loss as:

$$\min\max\{\{2.25, 2.58\}, \{2.58, 3.5\}\} = 2.58$$

Therefore, when the target is in P_{23}, and the current position of the attacker and the defender are P_{11} and P_{61}, the optimal decision for the attacker is to choose P_{12} as its next destination. The temporal formula which describes this optimal strategy will be:

$$\square[t_1 \wedge (a_{11} \wedge d_{61})) \rightarrow \bigcirc a_{12}] \tag{19}$$

In general, the complete formula for the attacking agent will be:

$$\varphi^a_{opt} = \bigwedge_{i,j,l,k,s} \square[t_s \wedge (a_{ij} \wedge d_{lk})) \rightarrow \bigvee_{(f,g)\in Q_{ij}} \bigcirc a_{fg}] \tag{20}$$
$$[(i,j) \in M, (l,k) \in M - (i,j), s \in \{1, \cdots k\}]$$

where t_s is the current target to be reached; s is the index of the current target; Q_{ij} contains the set of index of all desired regions in the neighborhood of a_{ij} (which are actually the solutions of the finite zero-sum game with the objective function in 18), and $M = \{(i,j)|i \in \{1, \dots m\}, j \in \{1, \dots n\}\}$ is the set of all indices for the regions in P.

Remark 1: For a game environment with $n \times m$ regions and k targets, $k(nm(nm-2))$ zero-sum games need to be solved, to extract the transition rules for the attacker.

IV. Hybrid Controller Synthesis

We provide the solution of Problem 1 in two steps. First, we design a discrete controller to derive discrete strategies satisfying the specification φ in (6). Then, we design a hybrid controller to generate continuous winning paths satisfying this specification.

A. Discrete Controller Design

Here, we synthesize the discrete controller as a finite state machine which satisfies the GR(1) formulas (if they are realizable). For reactive formula of type GR(1), [33] introduced an algorithm which has two steps. It first checks the realizability of the specification by evaluating a fixed-point equation, and then for the realizable formula, the algorithm synthesizes the winning strategy satisfying that formula. For detailed information about this process, we refer readers to [34].

Applying this algorithm to Problem 1 with the specification $\varphi = (\varphi_d \Rightarrow \varphi_s)$, the attacker always wins if φ is realizable, resulting is an automaton $G = (Q, q_0, \mathcal{D}, \delta, \mathcal{L}, \mathcal{X})$ where:

- $Q = \mathbb{N} \times \mathcal{A}$ is the set of discrete states,
- $q_0 = (1, a_0)$ is the initial state,
- $\mathcal{D}$ is the set of input propositions (defender's moves),
- $\delta : Q \times \mathcal{D} \rightarrow Q$ is the transition relation,
- $\mathcal{L} : Q \rightarrow \mathcal{A}$ is the labeling function,
- $\mathcal{X}$ is the set of final states.

If the automaton G is at state q_k, the current position of the defender $d_{k+1} \in \mathcal{D}$ is considered as input to G, and causes a transition from q_k to q_{k+1}, denoted by $\delta(q_k, d_{k+1}) = q_{k+1}$, and shown by $q_k \xrightarrow{d_{k+1}} q_{k+1}$. The newly generated output label at state q_{k+1} will be $a_{k+1} = \mathcal{L}(q_{k+1}) \in \mathcal{A}$. Based on the moves of the defender, a set of input labels will be received by G in the form of $d_0, d_1, \dots$. The automaton G, then, executes a run in the form of $r = q_0, q_1, \dots$, which is a sequence of states starting from the initial state q_0, where $q_0 \xrightarrow{d_0} q_1 \xrightarrow{d_1} q_2 \dots$. Correspondingly, a sequence of labels $L = \mathcal{L}(q_0), \mathcal{L}(q_1), \mathcal{L}(q_2), \dots = a_0, a_1, a_2, \dots$ will be generated as the discrete path for the defender. Since the formula φ_d has considered all possible transitions (decisions) of the defender, all admissible sequences of decisions of the

defender can be reacted by a sequence of actions of the attacker, leading the attacker to win.

B. Hybrid Controller Design

In order to convert the generated discrete path to smooth continuous signals driving the attacker over the partitioned space, we use the method we proposed in [27] to construct a hybrid controller. Due to the bisimulation relation between the original system with a multi-affine dynamics and its abstract model, shown in [27], it can be guaranteed that the generated continuous signals preserve the properties of the discrete path.

C. Results

For Example 1, the automaton for the attacker is synthesizable and it is constructed using the described method as shown in Fig. 2. In this figure, the nodes represent the activated (true) attacker sensors, which means the position of the attacker. The defender sensors are not shown on the edges of the automaton graph.

A hybrid controller is designed where the simulation results are shown for two different actions of the defender in Fig. 3. Based on the derived automaton G, the control strategy is computed for the initial position of region P_{11} for the attacker, and region P_{41} for the defender.

In Fig. 3(a), we show a behavior of the defender as a red curve, which is interpreted for the attacker as the discrete input sequence $d_{41}, d_{31}, d_{21}, d_{22}, d_{32}, d_{33}, d_{34}, d_{44}, d_{45}, d_{46}, d_{47}$. Observing these discrete decisions of the defender at each position, the attacker makes an optimal decision, about its next transition, according to the automaton G, which generates the discrete path $a_{11}, a_{12}, a_{13}, a_{23}, a_{24}, a_{25}, a_{35}, a_{36}, a_{37}, a_{47}, a_{57}$.

This discrete path is translated to a continuous signal, shown in blue, driving the attacker within P. As it can be seen, the proposed algorithm is capable of creating a winning strategy, so that the attacker can visit all three targets before being captured by the defender and win the game. Figure 3(b) shows the result of the same procedure for a different behavior of the defender $d_{41}, d_{31}, d_{21}, d_{22}, d_{23}, d_{24}, d_{25}, d_{35}, d_{36}, d_{46}, d_{47}$, which is reacted by the attacker's decision sequence $a_{11}, a_{12}, a_{13}, a_{23}, a_{24}, a_{25}, a_{35}, a_{45}, a_{46}, a_{56}, a_{57}$, and again confirms victory for the attacker.

V. CONCLUSION

In this paper, we developed a novel effective symbolic framework for motion path planning of an autonomous attacking vehicle involved in a multi-target dynamic reach-avoid (MTDRA) scenario. This problem was formulated within reactive synthesis framework GR(1) to describe all the assumptions and requirements of the vehicles as well as their interactions and conflicting objectives. Illustrative examples were provided to describe the implementation of the proposed approach.

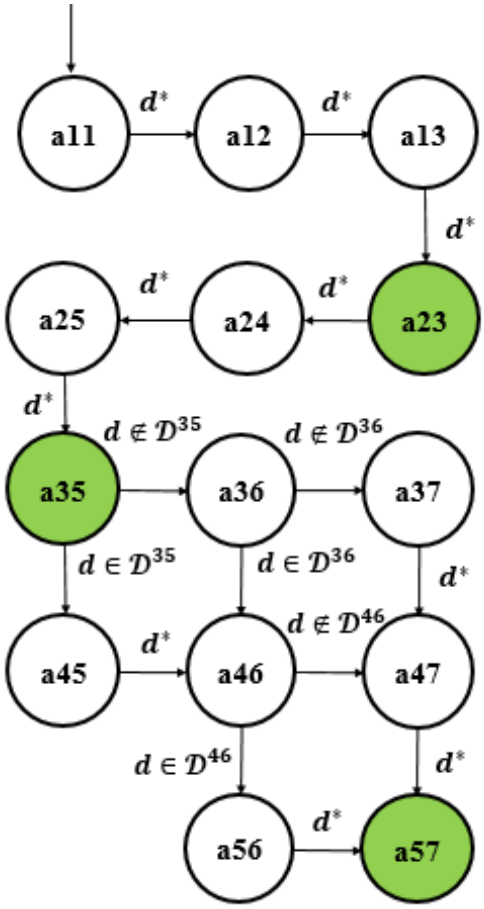


Fig. 2. Some parts of the Synthesized automaton G for the attacker in Example 1. The nodes show the true attacker sensor, which means the position of the attacker. The edges are defender sensors, where $D^{35} = \{d_{24}, d_{25}, d_{26}, d_{65}\}$, $D^{3}6 = \{d_{25}, d_{26}, d_{35}, d_{37}\}$, $D^{46} = \{d_{25}, d_{26}, d_{35}, d_{36}, d_{37}, d_{38}, d_{45}, d_{47}\}$, and d^* represents all possible defender's position information.

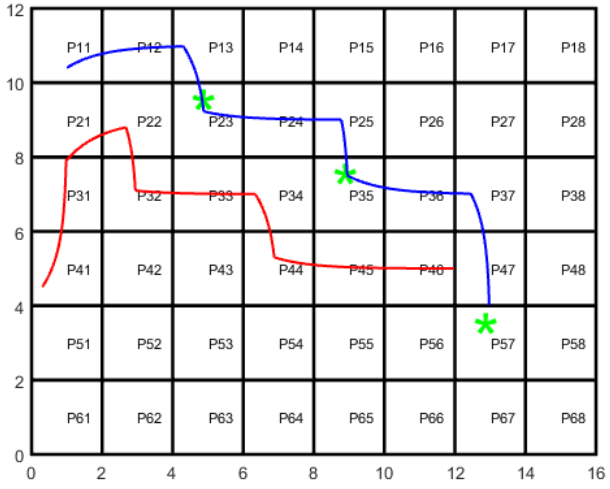
ACKNOWLEDGMENT

The authors would like to acknowledge the support from Air Force Research Laboratory and Office of the Secretary of Defense under agreement number FA8750-15-2-0116 as well as US ARMY Research Office under agreement number W911NF-16-1-0489. 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, ARMY Research Office, OSD, or the U.S. Government.

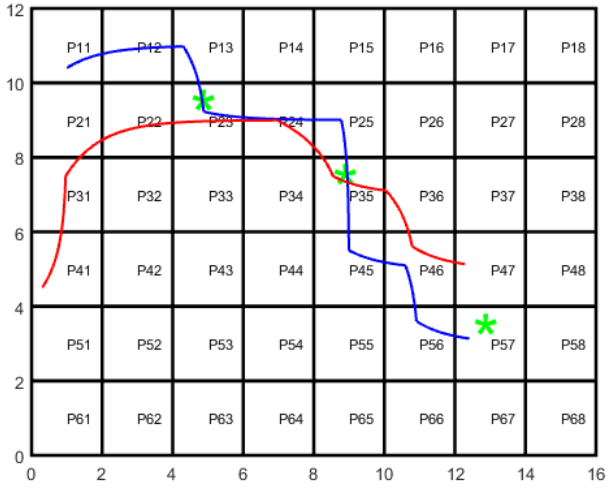
REFERENCES

[1] K. Jo, J. Kim, D. Kim, C. Jang, M. Sunwoo, Development of autonomous carpart i: Distributed system architecture and development process, IEEE Transactions on Industrial Electronics 61 (12) (2014) 7131–7140.

[2] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, et al., Towards fully autonomous driving: Systems and algorithms, in: Intelligent Vehicles Symposium (IV), 2011 IEEE, IEEE, 2011, pp. 163–168.



(a) Attacker discrete path: a_{11}, a_{12}, a_{13}, a_{23}, a_{24}, a_{25}, a_{35}, a_{36}, a_{37}, a_{47}, a_{57}, Defender discrete path: d_{41}, d_{31}, d_{21}, d_{22}, d_{32}, d_{33}, d_{34}, d_{44}, d_{45}, d_{46}, d_{47}



(b) Attacker discrete path: a_{11}, a_{12}, a_{13}, a_{23}, a_{24}, a_{25}, a_{35}, a_{45}, a_{46}, a_{56}, a_{57}; Defender discrete path: d_{41}, d_{31}, d_{21}, d_{22}, d_{23}, d_{24}, d_{25}, d_{35}, d_{36}, d_{46}, d_{47}.

Fig. 3. Simulation results for Example 1, which show the attacker behavior in response to two arbitrary behaviors of the defender.

[3] A. Karimoddini, H. Lin, B. M. Chen, T. H. Lee, Hierarchical hybrid modelling and control of an unmanned helicopter, International Journal of Control 87 (9) (2014) 1779–1793.

[4] J. E. Manley, Unmanned surface vehicles, 15 years of development, in: OCEANS 2008, IEEE, 2008, pp. 1–4.

[5] M. Caccia, G. Indiveri, G. Veruggio, Modeling and identification of open-frame variable configuration unmanned underwater vehicles, IEEE journal of Oceanic Engineering 25 (2) (2000) 227–240.

[6] J. Yuh, Design and control of autonomous underwater robots: A survey, Autonomous Robots 8 (1) (2000) 7–24.

[7] I. Mitchell, A. M. Bayen, C. J. Tomlin, Validating a hamilton-jacobi approximation to hybrid system reachable sets, in: HSCC, Springer, 2001, pp. 418–432.

[8] S. Summers, J. Lygeros, Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem, Automatica 46 (12) (2010) 1951–1961.

[9] M. Kamgarpour, J. Ding, S. Summers, A. Abate, J. Lygeros, C. Tomlin, Discrete time stochastic hybrid dynamical games: Verification & controller synthesis, in: Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on, IEEE, 2011, pp. 6122–6127.

[10] J. Barraquand, B. Langlois, J.-C. Latombe, Numerical potential field techniques for robot path planning, Systems, Man and Cybernetics, IEEE Transactions on 22 (2) (1992) 224–241.

[11] C. Torras, From geometric motion planning to neural motor control in robotics, AI Communications 6 (1) (1993) 3–17.

[12] M. Aigner, M. Fromme, A game of cops and robbers, Discrete Applied Mathematics 8 (1) (1984) 1–12.

[13] D. Bhadauria, K. Klein, V. Isler, S. Suri, Capturing an evader in polygonal environments with obstacles: The full visibility case, The International Journal of Robotics Research 31 (10) (2012) 1176–1189.

[14] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, R. Motwani, A visibility-based pursuit-evasion problem, International Journal of Computational Geometry & Applications 9 (04n05) (1999) 471–493.

[15] B. P. Gerkey, S. Thrun, G. Gordon, Visibility-based pursuit-evasion with limited field of view, The International Journal of Robotics Research 25 (4) (2006) 299–315.

[16] S. Russell, P. Norvig, A. Intelligence, A modern approach, Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs 25 (1995) 27.

[17] S. Thrun, W. Burgard, D. Fox, Probabilistic robotics, MIT press, 2005.

[18] M. P. Vitus, C. J. Tomlin, Closed-loop belief space planning for linear, gaussian systems, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 2152–2159.

[19] T. Lozano-Pérez, M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Communications of the ACM 22 (10) (1979) 560–570.

[20] K. Kant, S. W. Zucker, Toward efficient trajectory planning: The path-velocity decomposition, The International Journal of Robotics Research 5 (3) (1986) 72–89.

[21] Q. Zhu, Hidden markov model for dynamic obstacle avoidance of mobile robot navigation, Robotics and Automation, IEEE Transactions on 7 (3) (1991) 390–397.

[22] A. Ismail, A. Sheta, M. Al-Weshah, A mobile robot path planning using genetic algorithm in static environment, Journal of Computer Science 4 (4) (2008) 341–344.

[23] H. Huang, J. Ding, W. Zhang, 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.

[24] H. Huang, J. Ding, W. Zhang, C. J. Tomlin, Automation-assisted capture-the-flag: A differential game approach, Control Systems Technology, IEEE Transactions on 23 (3) (2015) 1014–1028.

[25] L. Shamgah, A. Karimoddini, A. Homaifar, A symbolic motion planning approach for the reach-avoid problem, in: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2016, pp. 3955–3960.

[26] X. D. Koutsoukos, P. J. Antsaklis, J. A. Stiver, M. D. Lemmon, Supervisory control of hybrid systems, Proceedings of the IEEE 88 (7) (2000) 1026–1049.

[27] A. Karimoddini, H. Lin, Hierarchical hybrid symbolic robot motion planning and control, Asian Journal of Control 17 (1) (2015) 23–33.

[28] A. Karimoddini, H. Lin, B. M. Chen, T. H. Lee, A smooth hybrid symbolic control for the formation of uavs over a partitioned space, in: 2013 American Control Conference, 2013, pp. 982–987.

[29] A. Karimoddini, H. Lin, Hybrid symbolic control for robot motion planning, in: 2013 10th IEEE International Conference on Control and Automation (ICCA), 2013, pp. 1650–1655.

[30] A. Karimoddini, H. Lin, B. M. Chen, T. H. Lee, A bumpless hybrid supervisory control algorithm for the formation of unmanned helicopters, Mechatronics 23 (6) (2013) 677 – 688.

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

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

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

[34] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, Y. Saar, Synthesis of reactive (1) designs, Journal of Computer and System Sciences 78 (3) (2012) 911–938.

[35] H. Kress-Gazit, G. E. Fainekos, G. J. Pappas, Temporal-logic-based reactive mission and motion planning, Robotics, IEEE Transactions on 25 (6) (2009) 1370–1381.