IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 December 2022, accepted 24 January 2023, date of publication 3 February 2023, date of current version 8 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3241946

RESEARCH ARTICLE

Finite-Horizon Shield for Path Planning Ensuring Safety/Co-Safety Specifications and Security Policies

KOKI KANASHIMA AND TOSHIMITSU USHIO, (Member, IEEE)
Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560-8531, Japan
Corresponding author: Toshimitsu Ushio (ushio@sys.es.osaka-u.ac.jp)

This work was supported in part by Japan Science and Technology Agency (JST) CREST under Grant JPMJCR2012.

ABSTRACT With the development of network technology, security in path planning problems has attracted widespread attention. We consider a path planning problem in which a planner computes a finite path that satisfies a specification. We assume that the specification includes mandatory safety/co-safety specifications. Moreover, we consider a security policy for this path. However, we assume that the information leaked to an intruder is not known beforehand. Then, we propose an enforcement mechanism referred to as a finite-horizon shield. This mechanism modifies the path computed by the planner as small as possible to satisfy the safety/co-safety specifications and security policy under the leaked information. We assume that the safety/co-safety specifications are described by LTL_f formulas and the security policy by a $\mathrm{hyperLTL}_f$ formula. Subsequently, we convert the formulas into quantified formulas and compute the modified path using a satisfiability modulo theories solver. As an example, we consider an opacity problem where there is another path whose leaked information is the same as that of the modified path. By simulations, it confirms that the output of shield depends on the leaked information and the modified path may have additional movements to ensure opacity. We also compare the computation time of the shield with that of a security-aware planning by simulation.

INDEX TERMS HyperLTL, opacity, path planning, safety/co-safety specification, security policy, shield synthesis.

I. INTRODUCTION

Safety-critical automated systems have numerous practical applications. The specifications of these systems are complex, and they consist of mandatory and optional specifications. For example, it is mandatory for a synthesized system to satisfy safety properties, which ensure that system behaviors remain in a safe state set. Safety properties are characterized by the bad prefixes of infinite state sequences [1]. It is important to ensure that system behaviors always satisfy safety specifications under different environments. In recent years, shield synthesis [2], [3] has attracted widespread attention for the design of complex engineering systems because it is an efficient method for enforcing safety specifications at runtime when the environment changes. A shield is attached to a system, as illustrated in FIGURE 1. It monitors the inputs and outputs of the system and modifies incorrect outputs to correct outputs. An advantage of shield synthesis is that the modification is minimized. If the system output satisfies a given safety specification, the shield does not make a modification, and its output is the same as the system output.

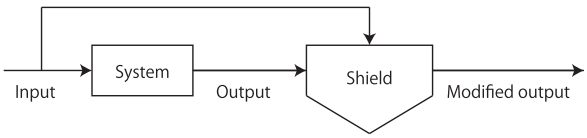

FIGURE 1. Shield that modifies system output to enforce safety properties.

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

Otherwise, the shield modifies its output to satisfy the safety specification such that the modified output is as close to the system output as possible. Thus, the effect of the modification on optional specifications is minimized. Shield synthesis has been applied to several engineering problems. In [4], a shield was attached to a multiagent system to prevent congestion and collisions. In [5] and [6], shields were used to ensure the safety of human-interactive robotics. A shield that adapted to a changing environment was proposed and applied to traffic light controllers to maintain the correct traffic flow [7]. Safe reinforcement learning using shields was proposed in [8] and [9].

Specifications have become complex owing to recent advances in automated systems, and temporal logics are useful tools for their detailed descriptions [10], [11]. Linear temporal logic (LTL) formulas are typically used for logical specifications in synthesis problems [12], and numerous reactive synthesis problems for LTL specifications have been studied [13], [14], [15], [16], [17], [18], [19]. Signal temporal logic (STL) formulas can describe the dynamic properties of real-valued continuous signals [20], [21], and they have been applied to controller synthesis problems [22], [23], [24]. Furthermore, LTL and STL formulas are used to describe the safety specifications for shield synthesis [2], [25]. Shield synthesis with specifications described by quantified discrete-duration calculus logic formulas has been recently proposed [26]. In reactive synthesis, temporal logic formulas are evaluated over infinite sequences of states or transitions. In the path planning problem of a mobile robot to find a finite sequence of states (referred to as a path) from the initial state to a goal state while meeting specifications, the LTL formulas that describe specifications are evaluated over finite sequences. This logic is referred to as LTL_f [27], [28], [29], [30], [31]. Moreover, safety and co-safety properties are important [32], [33]. A typical example of co-safety properties is that a robot reaches a goal state in a finite horizon. Co-safety properties are characterized by good prefixes. Thus, shield synthesis that satisfies safety and co-safety specifications is important in path planning problems.

Security in path planning problems has attracted considerable attention with the development of network technology [34], [35]. Malicious intruders observe partial information about a robot, which is referred to as leaked information, and estimate the secret about the robot. The secret about the robot is its location, action and so on. This is because it can lead to the identification of the entire path, which may reveal tasks or attributes of the robot. As example, for a mobile robot that transports a very valuable item, its current location is a secret to avoid being robbed of it by the intruder. Secure path planning, where an intruder cannot identify the secret, is an important issue that depends on leaked information. A security property is not a property of individual state sequences but a hyperproperty that is a set of the subsets of sequences. For example, opacity is a useful notion in the analysis of cryptographic protocols [36], and it is applied to multiple classes of systems such as discrete event systems [37], [38], [39] and cyber-physical systems [40], [41], [42]. Intuitively, opacity is defined such that, for each state sequence with a secret, there exists a different sequence without a secret whose leaked information is the same as that of the secret sequence. HyperLTL, which is an extension of LTL with trace variables, was proposed in [43], [44], and [45] to formally describe hyperproperties. It has been shown that several specifications of the path planning problems of mobile robots are described by hyperLTL formulas [46]. A unified framework that uses hyperLTL for specifying observational properties, including opacity, diagnosability, predictability, and detectability, has been proposed [47].

There is increasing demand for ensuring path security in robot path planning problems [34], [48], and this depends on the leaked information observed by an intruder. For example, when sensors send some (partial) information to an operator using a network and the vulnerability of the network is discovered, the operator modifies a path to guarantee a security policy under the assumption that the information may be leaked to a potential intruder. As another example, in the case where eavesdropping devices are set up and are detected while the robot works, it may be a good strategy against the intruder that, if the current plan is unsecure, the operator modifies the plan to be secure under the existence of the devices because the intruder does not notice the detection of the devices by the operator. To achieve the modification, the concept of shielding is applicable. Thus, in our scenario, a planner may generally not know about leaked information beforehand. Hence, it must compute a path that satisfies given specifications, including a safety and/or co-safety property, without considering a given security policy. When eavesdropping devices are detected, the corresponding leaked information is identified and the path is modified as small as possible such that the modified path satisfies safety/co-safety properties and the security policy. Therefore, it is important to provide a method for the modification.

In this study, we assume that a planner that computes the finite path of a robot is pre-designed. We will call the path a pre-planned path. We apply shield synthesis to an enforcement mechanism for the planner, whose objective is to satisfy safety and/or co-safety specifications and a security policy under the working environment of the robot. A conventional shield considers infinite sequences of inputs and outputs and modifies the outputs that satisfy safety specifications if necessary. However, in the path planning problem, the path is finite and satisfies safety and co-safety specifications and the security policy. Thus, we propose a shield-like enforcement mechanism, which is referred to as a *finite-horizon shield*, to modify the pre-planned path as small as possible while satisfying the safety/co-safety specifications and security policy under the working environment where the intruder exists. FIGURE 2 illustrates the proposed enforcement mechanism. While the robot moves along the pre-planned path, if leaked information is identified, it is sent to the finite-horizon shield.

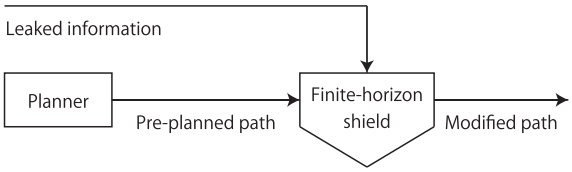

FIGURE 2. Enforcement mechanism with leaked information and a finite-horizon shield that modifies a finite path to satisfy safety and co-safety specifications and a security policy.

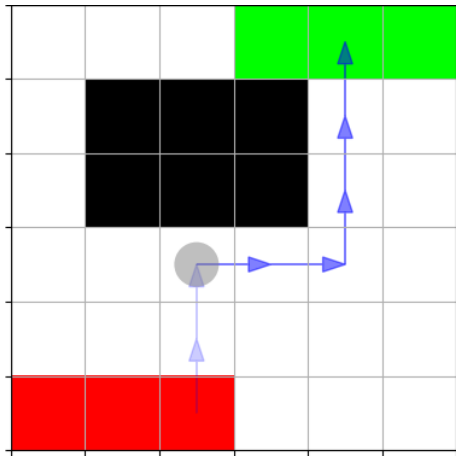

FIGURE 3. Example of a workspace modeled by 6 × 6 grid plane, where obstacles exist in black grids. The initial and goal grids are colored red and green, respectively, and the blue arrows indicate a pre-planned path.

The shield checks whether the pre-planned path satisfies the security policy under the leaked information. If not, it modifies the path to satisfy the safety/co-safety specifications and security policy. Therefore, the path output by the finite-horizon shield is guaranteed to satisfy safety/co-safety specifications and the security policy.

The following example illustrates the problem we consider. The planner determines a path based on safety and/or co-safety specifications without taking into account the security policy because it does not know what information may be leaked. Suppose the planner determines the path that satisfies the specification of avoiding obstacles and eventually reaching the goal in the workstation represented by the grid shown in FIGURE 3. The red, green, and black locations in this figure indicate starts, goals, and obstacles, respectively. The blue arrows are the pre-planned path determined by the planner. Assume that at time 3, that is, when the mobile robot is on the grid with the gray circle, the leaked information is identified and the horizontal position of the robot is leaked. We consider the case where a secret is the initial state of the robot, that is, the initial state cannot be determined only from the leaked information. Such a policy is called initial-state opacity. See Section V for details. The pre-planned path does not satisfy the security policy. Then, we modify it after time 3 to satisfy the security policy. Shown in FIGURE 4 is an example of the modified path, where the red arrows indicate the modified path and the yellow arrows indicate a path that

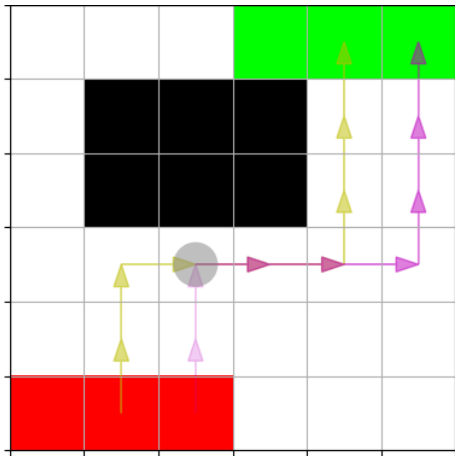

FIGURE 4. Example of a modified path that satisfies initial-state opacity.

the intruder cannot distinguish from the modified path. Note that the initial states of the paths are different and the modified path satisfies the security policy.

We assume that the specification for which the planner computes a pre-planned path is described by an LTL_f formula and that the safety/co-safety specifications are described as its subformulas. Moreover, the leaked information is represented by atomic propositions, and the security policy is represented by a hyperLTL_f formula over the set of the atomic propositions. Then, we construct the finite-horizon shield using SAT-based bounded model-checking approaches [49], [50], [51], [52]. As any hyperLTL_f formula is converted into a quantified formula (QF) [53], the safety/co-safety specifications and security policy are encoded as QFs. Then, a modified path is computed using a satisfiability modulo theories (SMT) solver. While, in a security-aware planning problem, a path that satisfies a specification including a security policy is computed [42], [46], [48], we address a problem of modifying an unsecure path to be secure under the leaked information that is detected while the robot works, i.e., we extend a shield approach to secure path re-planning and the proposed shield modified a finite path if it does not satisfy the security policy under the identified leaked information. Such an approach is also different from supervisory control based approaches proposed in [38], [47], and [54], where the supervisor determines a set of events that satisfy the security policy under the assumption that the leaked information is known. The proposed approach is also different from the enforcement mechanism proposed in [55] because the finite-horizon shield receives a single sequence and modifies it to a single sequence, whereas the mechanism in [55] receives a set of sequences and modifies them.

In this study, as a comparative approach, we also consider a security-aware planning problem to compute a path satisfying a specification including a security policy (see Section VI for details). Since the security policy is encoded as a QF, the planning problem is converted into a quantified satisfiability (QSAT) problem, which is PSPACE-complete [56]. In other

words, the comparative approach is a one-step approach. On the other hand, the proposed method can be regarded as a two-step approach to a security-aware planning problem with known leaked information, that is, the planner computes a pre-planned path by solving a SAT problem, which is NP-complete [57], since the security policy is not included in the specification for the planner and the finite-horizon shield computes a secure path by the modification of the pre-planned path, which is a QSAT problem. Then, in the planner, we do not use Boolean variables that describe the security policy while, in the finite-horizon shield, we do not use those that describe the specification except the security policy. Thus, the numbers of the Boolean variables used in the planner and the finite-horizon shield are less than that in the security-aware planning problem. In other words, the proposed method computes a secure path by solving a SAT problem and a QSAT problem with fewer numbers of Boolean variables than that used in the QSAT problem of the one-step approach. By simulation, we show that the computation time of the proposed approach is less than that of the one-step approach when the number of the variables used in the encoding of the planning problems is large. Thus, the proposed method is practically useful for a security-aware planning problem under known leaked information.

The remainder of this paper is organized as follows: In Section II, we review hyperLTL, which is an extension of LTL. In Section III, we formulate a path planning problem in which a specification and security policy for a desired finite path are described by hyperLTL formulas. In Section IV, we propose a finite-horizon shield synthesis algorithm for the modification of a path planned by a planner to satisfy the hyperLTL$_f$ formulas that represent the safety/co-safety specifications and security policy. We demonstrate the application of the finite-horizon shield to the enforcement of opacity for mobile robot path planning in Section V. We compare a security-aware planner and the finite-horizon shield in Section VI. Section VII concludes the paper.

II. HyperLTL OVER FINITE TRACES

Let AP be a set of atomic propositions and 2^{AP} be the power set of AP. A finite trace over 2^{AP} is a finite sequence of the subsets of AP. Let $t = t_0 t_1 \dots t_{H-1}$ be a finite trace with length H. For $j \in \{0, 1, \dots, H-1\}$, we denote $t[j] = t_j$ and $t[j, H-1] = t_j t_{j+1} \dots t_{H-1}$. $(2^{AP})^H$ is the set of all traces with length H over 2^{AP}. For $T \subseteq (2^{AP})^H$, let $T[j, H-1] = \{t[j, H-1] \mid t \in T\} \subseteq (2^{AP})^{H-j}$.

HyperLTL is an extension of LTL with trace quantifiers ($\exists, \forall$) and trace variables. Numerous hyperproperties related to security policies are described by hyperLTL formulas.

Let Π be a set of trace variables. The hyperLTL formulas are recursively generated as follows:

$$\psi ::= \forall \pi.\, \psi \mid \exists \pi.\, \psi \mid \varphi, \tag{1}$$

$$\varphi ::= ap_\pi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2, \tag{2}$$

where $ap \in AP$ is an atomic proposition and $\pi \in \Pi$ is a trace variable. Subscript π indicates that ap should be checked over trace variable π. Some trace variables in the objectives are quantified by $\exists$ or $\forall$. The other Boolean operators ($\vee$, $\rightarrow$, and $\equiv$) are defined as $\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \rightarrow \varphi_2 := \neg\varphi_1 \vee \varphi_2$, and $\varphi_1 \equiv \varphi_2 := (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$. In addition, two temporal operators, i.e., *eventually* ($\Diamond$) and *always* ($\Box$), are defined as follows:

$$\Diamond\varphi := \top \,\mathcal{U}\, \varphi,$$
$$\Box\varphi := \neg\Diamond\neg\varphi.$$

The hyperLTL formulas are interpreted over infinite and finite traces. The hyperLTL formulas interpreted over finite traces with length H are referred to as hyperLTL$_f^H$ formulas. The syntax of hyperLTL$_f^H$ formulas is the same as that of the hyperLTL formulas defined by (1) and (2), and their semantics is given by the satisfaction relation, $\models_T$, over a set of finite traces $T \subseteq (2^{AP})^H$. Let $V : \Pi \rightarrow (2^{AP})^H$ be an assignment function. The semantics is recursively defined as follows:

$$\begin{aligned}
V \models_T ap_\pi &\Leftrightarrow ap \in V(\pi)[0],\\
V \models_T \neg\varphi &\Leftrightarrow V \not\models_T \varphi,\\
V \models_T \varphi_1 \wedge \varphi_2 &\Leftrightarrow V \models_T \varphi_1 \wedge V \models_T \varphi_2,\\
V \models_T \bigcirc\varphi &\Leftrightarrow H \geq 2 \wedge V[1, H-1] \models_T \varphi,\\
V \models_T \varphi_1 \,\mathcal{U}\, \varphi_2 &\Leftrightarrow 0 \leq \exists j \leq H-1.\, V[j, H-1] \models_T \varphi_2\\
&\quad \wedge 0 \leq \forall j' < j.\, V[j', H-1] \models_T \varphi_1,\\
V \models_T \exists\pi.\, \psi &\Leftrightarrow \exists t \in T.\, V[\pi \rightarrow t] \models_T \psi,\\
V \models_T \forall\pi.\, \psi &\Leftrightarrow \forall t \in T.\, V[\pi \rightarrow t] \models_T \psi,
\end{aligned}$$

where $V[j, H-1] : \Pi \rightarrow T[j, H-1]$ is an assignment function such that for each $\pi \in \Pi$, $V[j, H-1](\pi) = V(\pi)[j, H-1]$. $V[\pi \rightarrow t]$ is an assignment function such that $V[\pi \rightarrow t](\pi) = t$ and $V[\pi \rightarrow t](\pi') = V(\pi')$ for each $\pi' \in \Pi$ with $\pi' \neq \pi$. Intuitively, $\exists\pi.\psi$ holds if and only if there exists a trace in T such that ψ satisfies. $\forall\pi.\psi$ holds if and only if all traces in T satisfy ψ.

III. PROBLEM FORMULATION

We consider a path planning problem of a mobile robot in a workspace that is partitioned into a finite number of regions. We assume that the transitions between the regions are given. Additionally, a specification for the path, which is referred to as a path specification, is given by an LTL formula. We assume that the path specification includes safety and/or co-safety subspecifications as mandatory requirements. Let AP^p be the finite set of atomic propositions that are used to describe the path specification. We assume the existence of a planner that computes a path that satisfies the path specification described by an LTL$_f$ formula. Then, we consider a case where there is a security policy that specifies a secret of the path and an intruder who observes information about the behavior of the robot and attempts to reveal the secret. We assume that the leaked information is unknown when the planner computes the path. Let AP^{ob} be a finite set of the atomic propositions that are used to describe the observation

of the leaked information. Let $AP := AP^p \cup AP^{ob}$. Then, the behavior of the mobile robot is modeled by a transition system $\mathbb{P} = (P, P^0, I, \delta, L^p, L^{ob})$, where

- P is the set of states that represent regions in the workspace,
- $P^0 \subseteq P$ is the set of initial states,
- I is the set of inputs,
- $\delta : P \times I \to P$ is a partial transition function, where $p' = \delta(p, i)$ indicates that the robot moves from the state p to the state p' by the input i,
- $L^p : \Delta \to 2^{AP^p}$ is the labeling function with respect to set AP^p, where $\Delta := \{(p, i, q) \mid q = \delta(p, i)\} \subseteq P \times I \times P$ is a set of transitions, that is, $L^p((p, i, q))$ is the set of atomic propositions in AP^p that hold at transition (p, i, q),
- $L^{ob} : \Delta \to 2^{AP^{ob}}$ is the labeling function with respect to set AP^{ob}. Intuitively, $L^{ob}(\ell) \subseteq AP^{ob}$ represents the information leaked by the occurrence of transition $\ell \in \Delta$.

We introduce the following labeling function, $L : \Delta \to 2^{AP}$. For each $\ell \in \Delta$,

$$L(\ell) := L^p(\ell) \cup L^{ob}(\ell).$$

Let $P' := \{p' \mid p \in P\}$. Then, we assume that $P \cup P' \subseteq AP$, that is, each state and its primed symbol are atomic propositions, and for transition $(p, i, q) \in \Delta$, $L((p, i, q)) \cap (P \cup P') = \{p, q'\}$. Intuitively, for transition $\ell \in \Delta$, $\{p, q'\} \subseteq L(\ell)$ implies that it is a transition from state p to state q. For transition system $\mathbb{P}$, a finite sequence of transitions ρ defined by (3) is referred to as a path.

$$\rho := (p[0], i[0], p[1])(p[1], i[1], p[2]) \ldots \ldots (p[H-1], i[H-1], p[H]) \in \Delta^H, \quad (3)$$

where H is the length of the path, $p[0] \in P^0$, and $\delta(p[h], i[h]) = p[h+1]$ for $h = 0, \ldots, H-1$. Let $\mathcal{L}(H, \mathbb{P}) \subseteq \Delta^H$ be a set of paths with length H for $\mathbb{P}$. Note that $\mathcal{L}(1, \mathbb{P}) = \{(p, i, q) \mid p \in P^0 \wedge (p, i, q) \in \Delta\}$. We extend the two labeling functions, L^p and L, to $\mathcal{L}(H, \mathbb{P})$ for any positive integer $H \geq 2$ as follows: For path $\rho\ell \in \mathcal{L}(H, \mathbb{P})$ with $\rho \in \mathcal{L}(H-1, \mathbb{P})$ and $\ell \in \Delta$,

$$L^p(\rho\ell) = L^p(\rho)L^p(\ell), \quad (4)$$

$$L(\rho\ell) = L(\rho)L(\ell). \quad (5)$$

Let $T^p(H, \mathbb{P}) \subseteq (2^{AP^p})^H$ (*resp.* $T(H, \mathbb{P}) \subseteq (2^{AP})^H$) be a set of traces with length H for $\mathbb{P}$ and L^p (*resp.* L), that is,

$$T^p(H, \mathbb{P}) := \{L^p(\rho) \mid \rho \in \mathcal{L}(H, \mathbb{P})\}, \quad (6)$$

$$T(H, \mathbb{P}) := \{L(\rho) \mid \rho \in \mathcal{L}(H, \mathbb{P})\}. \quad (7)$$

Π^p and Π denote the sets of trace variables used for the path specification and the security policy, respectively. Without loss of generality, we set $\Pi^p = \{\pi\}$ because the path specification is described by an LTL formula that is equivalent to a quantifier-free hyperLTL formula with the trace variable π. Let $V^p : \Pi^p \to T^p(H, \mathbb{P})$ and $V : \Pi \to T(H, \mathbb{P})$ be assignment functions.

The path specification is described by an LTL formula that is a quantifier-free hyperLTL$_f^H$ formula φ^p over AP^p with trace variable π. The planner computes path $\rho_p \in \mathcal{L}(H_p, \mathbb{P})$, which is referred to as a pre-planned path, such that

$$V^p \models_{T^p(H_p, \mathbb{P})} \varphi^p, \quad (8)$$

where H_p is the length of ρ_p and $V^p(\pi) = L^p(\rho_p)$. We assume that path specification φ^p is partitioned into a mandatory and an optional specification, that is, $\varphi^p := \varphi^s \wedge \varphi^o$, where φ^s and φ^o represent the mandatory and the optional specification, respectively: a similar partition of the specification was proposed in [58] and [19]. The mandatory specification is related to safety or co-safety properties and it should be satisfied even after modification. In other words, it is considered to be the specification that does not want to be effected by the modification. For example, a safety property is that the mobile robot never enters a dangerous region such as a river, and a co-safety property is that the mobile robot reaches a target region in a finite horizon. The optional specification aims to increase the quality of service for the path such as passing a specified location if possible.

We consider the case where an intruder attempts to reveal the secret of the pre-planned path using the leaked information that is partially observed by the intruder's sensors. Then, the leaked information is identified, which is represented by a set of atomic propositions AP^{ob}. A security policy is described by a hyperLTL formula ψ^{sp} over AP as follows:

$$\psi^{sp} = \mathbb{Q}_2\pi_2 \ldots \mathbb{Q}_n\pi_n.\ \varphi^{sp}, \quad (9)$$

where $\mathbb{Q}_i \in \{\exists, \forall\} (i = 2, \ldots, n)$ is a trace quantifier and φ^{sp} is a quantifier-free hyperLTL formula with trace variables π_1, $\pi_2, \ldots, \pi_n$. Path $\rho \in \mathcal{L}(H, \mathbb{P})$ satisfies security policy ψ^{sp} if assignment function V with $V(\pi_1) = L(\rho)$ satisfies

$$V \models_{T(H, \mathbb{P})} \psi^{sp}. \quad (10)$$

Opacity is an example of a security policy of the mobile robot; this will be discussed in Section V.

If the pre-planned path does not satisfy the security policy, we modify the path such that it satisfies φ^s and φ^{sp}. The next section describes the method proposed for the modification.

IV. FINITE-HORIZON SHIELD

We consider a mobile robot whose workspace is modeled by transition system $\mathbb{P} = (P, P^0, I, \delta, L^p, L^{ob})$. For a given path specification $\varphi^p = \varphi^s \wedge \varphi^o$, a planner computes a pre-planned path $\rho_p \in \mathcal{L}(H_p, \mathbb{P})$ such that (8) holds for $V^p(\pi) = L^p(\rho_p)$. Note that the planner does not consider the security policy. Moreover, labeling function L^{ob} is not determined beforehand because it depends on the leaked information (e.g., partial observation of the position of the mobile robot). While the mobile robot operates, if the leaked information is identified at time $h \in [0, H_p]$, it is represented by set AP^{ob} and labeling function $L^{ob} : \Delta \to 2^{AP^{ob}}$. The leaked information at each transition $\ell \in \Delta$ is described by subset $L^{ob}(\ell) \subseteq AP^{ob}$, and the security policy is described

by a hyperLTL formula φ^{sp} over $AP = AP^p \cup AP^{ob}$. Then, we modify the pre-planned path as small as possible while satisfying mandatory specification φ^s and security policy φ^{sp}.

On the basis of shield synthesis [2], [3], we propose an enforcement mechanism referred to as a *finite-horizon shield* to achieve the modification. The finite-horizon shield is illustrated in FIGURE 2. Security policy ψ^{sp} is given by (9). When the leaked information is identified, it informs the finite-horizon shield about labeling function $L^{ob} : \Delta \to 2^{AP^{ob}}$. We require the following constraints:

- The finite-horizon shield does nothing until the leaked information is detected,
- When the leaked information is identified at time $h \in [0,\ H_p - 1]$, the remaining path in time interval $[h + 1, H_p]$ is modified in such a way that the overall path in time interval $[0,\ H_p]$ satisfies the security policy.

The first constraint is based on the design policy of the shield. The second one is necessary to satisfy the security policy even if the leaked information is identified while the robot is moving. Then, the finite-horizon shield checks whether the pre-planned path ρ_p satisfies ψ^{sp}. If ρ_p satisfies ψ^{sp}, its output is ρ_p; that is, it does not modify the pre-planned path. Otherwise, it computes a path with the minimum modification while satisfying mandatory specification φ^s and security policy ψ^{sp}. The overall procedure for the computation of output $\rho_s \in \mathcal{L}(H_s, \mathbb{P})$ of the finite-horizon shield is illustrated in FIGURE 5, where $H_s \geq H_p$ is the length of ρ_s. The method is described bellow.

A. FIRST STEP

The finite-horizon shield receives pre-planned path ρ_p as its input (Process1 in FIGURE 5). Let $V : \Pi \to T(H_p, \mathbb{P})$ be an assignment function such that $V(\pi_1) = L(\rho_p)$. Then, it checks whether ρ_p satisfies ψ^{sp} (Process2), that is,

$$\begin{aligned} &\text{Check} V \models_{T(H_p,\mathbb{P})} \mathbb{Q}_2\pi_2\mathbb{Q}_3\pi_3 \ldots \mathbb{Q}_n\pi_n.\varphi^{sp} \\ &\text{subject to } V(\pi_1) = L(\rho_p). \end{aligned} \tag{11}$$

If (11) is satisfied, the finite-horizon shield does not modify the path and outputs ρ_p, that is, $\rho_s = \rho_p$ (Process3). However, when (11) does not hold, we proceed to the next step.

B. SECOND STEP

The finite-horizon shield modifies ρ_p such that the modified path is closest to ρ_p among the paths that satisfy φ^s and ψ^{sp}.

It should be noted that, in general, if ρ_p does not satisfy the security policy, there may not exist a secure path with length H_p that satisfies the mandatory specification. Then, the finite-horizon shield outputs secure path ρ_s with length H_s larger than H_p. In this case, we evaluate the closeness of ρ_s to ρ_p by extending ρ_p to path $\hat{\rho}_p = \hat{\rho}_p[0]\hat{\rho}_p[1] \ldots \hat{\rho}_p[H_s - 1] \in \mathcal{L}(H_s, \mathbb{P})$ such that $\hat{\rho}_p[h] = \rho_p[h]$ for each $h \in \{0, 1, \ldots, H_p - 1\}$. The extension is an important issue that depends on the path planning problem. Therefore, it is beyond the scope of this study. An example of the extension is provided in the next section.

Let $cls : \mathcal{L}(H_s, \mathbb{P}) \times \mathcal{L}(H_s, \mathbb{P}) \to \mathbb{N}$ be a function that evaluates the effect of the modification. For extended pre-planned path $\hat{\rho}_p$ and its modified path ρ_s, the effect of the modification decreases with $cls(\hat{\rho}_p, \rho_s)$. Note that $cls(\rho, \rho) = 0$ for any $\rho \in \mathcal{L}(H_s, \mathbb{P})$. Examples of cls can be similarity between input strings (described in Section V for detail) and similarity between states.

Let $V^p : \Pi^p \to T^p(H_s, \mathbb{P})$ and $V : \Pi \to T(H_s, \mathbb{P})$ be assignment functions. For $k \in \mathbb{N}$ and $h \in [0, \ldots, H_p]$, modified path ρ_s starting from the same state as ρ_p under the constraint that the effect of the modification is less than or equal to k is computed as follows:

$$\begin{aligned} &\text{Determine } \rho_s \\ &\text{subject to } \rho_s \in \mathcal{L}(H_s, \mathbb{P}), \\ &\qquad (h > 0) \to \bigwedge_{\tau=0}^{h-1} (\rho_s[\tau] = \hat{\rho}_p[\tau]), \\ &\qquad V^p(\pi) = L^p(\rho_s), \\ &\qquad V^p \models_{T^p(H_s,\mathbb{P})} p^0_\pi \wedge \varphi^s, \\ &\qquad V(\pi_1) = L(\rho_s), \\ &\qquad V \models_{T(H_s,\mathbb{P})} \mathbb{Q}_2\pi_2\mathbb{Q}_3\pi_3 \ldots \mathbb{Q}_n\pi_n. \\ &\qquad \varphi^{sp} \wedge \left(cls(\hat{\rho}_p, \rho_s) \leq k\right), \end{aligned} \tag{12}$$

where $p^0 \in P^0$ is the initial state of ρ_p. The second condition in (12) indicates that the past path from the initial state to the current state cannot be modified. Intuitively, k represents the tolerance of the modification, and it should be as small as possible. However, the existence of the modified path depends on k. First, k is set to be sufficiently small to practically minimize the effect of the modification. If a modified path does not exist, then k is increased and the modified path is recomputed. This procedure is repeated until a modified path is obtained. For simplicity, in FIGURE 5, we initially set $k = 1$ (Process4). For each k, (12) is solved to find path ρ_s (Process7). If it exists, the finite-horizon shield outputs ρ_s as the modified path (Process8). Otherwise, k is increased (Process9). If $k > K_{max}$, the finite-horizon shield outputs an error (Process6), where K_{max} is a hyperparameter that ensures the termination of the computation.

In this study, to solve (11) and (12), we convert the constraints described by hyperLTL$_f$ formulas into QFs and solve the satisfiability problem of the QFs. The conversion is done in such a way that the hyperLTL$_f$ formulas are satisfiable if and only if the corresponding QFs are satisfiable. See Appendix A for the conversion of the hyperLTL$_f$ formulas into the QFs. Thus, (11) is satisfied if and only if the QF corresponding to (11) is satisfiable. For (12), the path ρ_s exists if and only if the QF corresponding to the constraints of (12) is satisfiable. Then, the path is obtained by an instance satisfying the QF. The next section describes an example of this conversion. Algorithm 1 shows the entire procedure for the modification of ρ_p. We encode (11) (Algorithm 1 line 1) and check whether (11) is *true* or *false* (line 2). If it is *true*, then ρ_p satisfies ψ^{sp}. Therefore, the finite-horizon

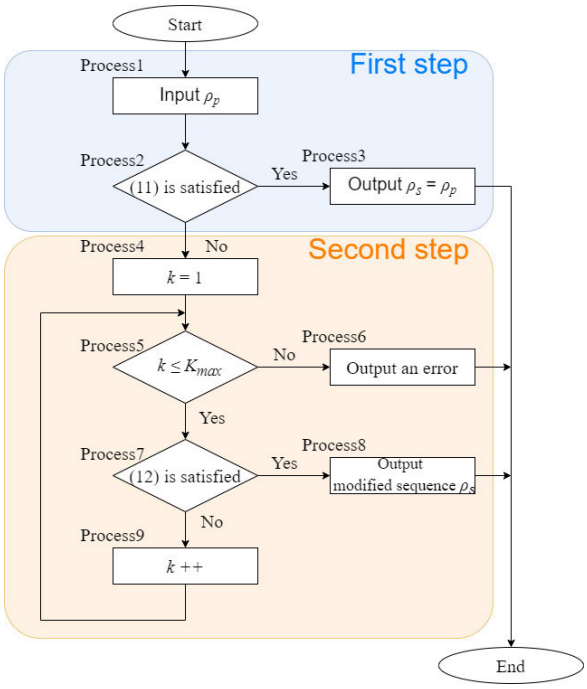

FIGURE 5. Flowchart of the proposed finite-horizon shield.

shield outputs $\rho_s = \rho_p$ and $cls(\hat{\rho}_p, \rho_s) = 0$ (line 5). If (11) is *false* (line 6), the **for** loop (line 7) iterates over parameter $k \in \{1, \ldots, K_{max}\}$. For each k, we encode (12) (line 8) and check the satisfiability of (12) (line 9). If it is satisfiable, the finite-horizon shield outputs ρ_s and $cls(\hat{\rho}_p, \rho_s) = k$. If (12) has no solution until $k = K_{max}$, then the finite-horizon shield outputs an error (line 15). Then, a possible counterplan is to complete the pre-planed path with giving up the security policy or to return to the initial state. However, it is beyond the scope of this paper because it depends on the system or the priority of specifications.

Algorithm 1 Modify ρ_p to ρ_s via Finite-Horizon Shield

Input: ρ_p, K_{max}
Output: ρ_s, cls
1: Encode (11)
2: **if** (11) is *true* **then**
3: $\rho_s \leftarrow \rho_p$
4: $cls \leftarrow 0$
5: **return** ρ_s, cls
6: **else**
7: **for** $k = 1$ to K_{max} **do**
8: Encode (12)
9: **if** (12) is satisfied **then**
10: $cls \leftarrow k$
11: **return** ρ_s, cls
12: **end if**
13: **end for**
14: **end if**
15: **return** -1

The computational complexity of Algorithm 1 is PSPACE complete since it uses the satisfiability of quantified formulas [56]. We consider the number of variables used in Algorithm 1, which depends mainly on the number of subformulas in φ^{sp} and the number of variables that encode paths. Let $N_{f,i}$ be the number of different subformulas related to trace variable π_i in formula f. For H_s transitions and all subformulas in φ^{sp}, $\sum_{i=1}^{n} N_{\varphi^{sp},i} H_s$ variables are needed, where n is the number of trace variables in Π. Variables that represent paths are also needed. For each transition, $(|P| + |I| + |O|)$ variables are needed to store the states, inputs, outputs, where O is the set of partial observations. For H_s transitions, we have H_s times as many variables. Algorithm 1 requires n paths. In addition, we prepare variables that encode cls and let N_{cls} be the number of them. Therefore, the total number N_{Alg1} of variables required for Algorithm 1 is as follows:

$$N_{Alg1} = \sum_{i=1}^{n} N_{\varphi^{sp},i} H_s + n(|P| + |I| + |O|)H_s + N_{cls}. \quad (13)$$

In practice, we also have to consider the number of variables N_{pre} needed for the SAT problem to find the pre-planned path and N_{pre} is expressed as follows:

$$N_{pre} = N_{\varphi^p} H_s + (|P| + |I|)H_s, \quad (14)$$

where N_{φ^p} is the number of different subformulas in formula φ^p.

V. APPLICATION TO OPAQUE PATH PLANNING PROBLEMS

In this section, we described the application of the finite-horizon shield to opaque path planning problems for a mobile robot.

A. PROBLEM SETTINGS

Opacity is an information flow security property, and several definitions of opacity have been proposed. We consider the case where an intruder can partially observe the behaviors of a mobile robot, and there is a secret for the path planning of the robot. Intuitively, a path is opaque if the intruder cannot expose the secret under the partial observation. We consider that the mobile robot moves in a 2-dimensional workspace, which is partitioned into $N_x \times N_y$ grid regions. Subsequently, the behavior of the robot is represented by a finite-state transition system, $\mathbb{P} = (P, P^0, I, \delta, L^p, L^{ob})$, where

- $P = \{(0, 0), \ldots, (N_x - 1, N_y - 1)\}$ is the set of states that indicates the current locations of the robot,
- P^0 is the set of initial states,
- $I = \{up, down, right, left, stay\}$ is the set of inputs,
- δ is the partial transition function, and for each $(x, y) \in P$,

$$\delta((x, y), \text{ up}) := \begin{cases} (x, y+1) & \text{if } (x, y+1) \in P, \\ \text{undefined} & \text{otherwise,} \end{cases}$$

$$\delta((x, y), \text{ down}) := \begin{cases} (x, y-1) & \text{if } (x, y-1) \in P, \\ \text{undefined} & \text{otherwise,} \end{cases}$$

$$\delta((x,y),\ \text{right}) := \begin{cases} (x+1,y) & \text{if } (x+1,y)\in P,\\ \text{undefined} & \text{otherwise,}\end{cases}$$
$$\delta((x,y),\ \text{left}) := \begin{cases} (x-1,y) & \text{if } (x-1,y)\in P,\\ \text{undefined} & \text{otherwise,}\end{cases}$$
$$\delta((x,y),\ \text{stay}) := (x,y),$$

- $L^p : \Delta \to 2^{AP^p}$ is the labeling function for the safety and co-safety specifications and $L^{ob} : \Delta \to 2^{AP^{ob}}$ is that for opacity, where the set Δ of transitions is given by

$$\Delta = \{((x,y), i, (\hat{x},\hat{y})) \in P \times I \times P \mid (\hat{x},\hat{y}) = \delta((x,y), i)\}. \quad (15)$$

Let $G \subset P$ be a set of goal locations. We assume that there are obstacles in the workspace. Let $D \subset P$ be a set of locations with obstacles. For simplicity, We assume that $D \cap P^0 = G \cap P^0 = D \cap G = \emptyset$. Let $\rho_p = \rho_p[0]\rho_p[1]\dots\rho_p[H_p-1] \in \mathcal{L}(H_p, \mathbb{P})$ be a pre-planned path computed by the planner, where H_p is its length and $\rho_p[h] = ((x_0^h, y_0^h), i_0^h, (x_0^{h+1}, y_0^{h+1})) \in \Delta$ for each $h \in \{0,1,\dots,H_p-1\}$. Then, the path satisfies a mandatory specification such as safety/co-safety specifications.

Let $AP^p = (X \times Y) \cup I \cup (X' \times Y')$ be the set of atomic propositions, where $X = \{0,1,\dots,N_x-1\}$ and $Y = \{0,1,\dots,N_y-1\}$ are the sets of atomic propositions related to the horizontal and vertical positions of the mobile robot, respectively; $X' = \{x' \mid x \in X\}$ and $Y' = \{y' \mid y \in Y\}$. Labeling function L^p is defined as follows:

$$L^p(((x,y), i, (\hat{x},\hat{y}))) = \{(x,y), i, (\hat{x}',\hat{y}')\}. \quad (16)$$

Let $\Pi^p = \{\pi\}$ be the set of trace variable π and let V^p be an assignment function that assigns π to a trace over AP^p. We consider a mandatory specification that describes a safety/co-safety property such that the mobile robot never enters locations where obstacles exist and eventually reaches a goal. This is described by φ_π^s over AP^p, as follow:

$$\varphi_\pi^s := \Box \bigvee_{(x,y)\in P\setminus D} (x',y')_\pi \wedge \Diamond \bigvee_{(x,y)\in G} (x',y')_\pi. \quad (17)$$

Then, for pre-planned path ρ_p, we have

$$V^p \models_{T^p(H_p,\mathbb{P})} \varphi_\pi^s, \quad (18)$$

where $V^p(\pi) = L^p(\rho_p)$.

FIGURE 3 shows an example of the workspace with $N_x = N_y = 6$ and a pre-planned path, where

$$P^0 = \{(0,0),(1,0),(2,0)\}, \quad (19)$$
$$G = \{(3,5),(4,5),(5,5)\}, \quad (20)$$
$$D = \{(1,3),(1,4),(2,3),(2,4),(3,3),(3,4)\}. \quad (21)$$

If there is no opaque path with length H_p that satisfies the mandatory specification, then the finite-horizon shield outputs an opaque path whose length is larger than H_p. Let H_s be the length of modified path ρ_s. The determination of H_s is an important issue in practice. There are several decision rules for obtaining H_s. For example, there is a brute force approach in which the initial value of H_s is set to be equal to H_p and increased by one until an opaque path is obtained. However, this is out of the scope of this work, and we assume that H_s is given. We extend ρ_p to path $\hat{\rho}_p = \hat{\rho}_p[0]\hat{\rho}_p[1]\dots\hat{\rho}_p[H_s-1]$ with length H_s as follows.

- $\hat{\rho}_p[h] = \rho_p[h]$ for each $h \in \{0,1,\dots,H_p-1\}$.
- $\hat{\rho}_p[h] = ((x_{H_p}, y_{H_p}), \text{stay}, (x_{H_p}, y_{H_p}))$ for each $h \in \{H_p, H_p+1, \dots, H_s-1\}$.

Intuitively, $\hat{\rho}_p$ is constructed by making the robot stay at the last reached location.

For $\hat{\rho}_p$, the finite-horizon shield computes modified path ρ_s denoted by

$$\rho_s = \rho_s[0]\rho_s[1]\dots\rho_s[H_s-1]. \quad (22)$$

Recall that set AP^{ob} represents the leaked information identified by the intruder detector. For labeling function $L^{ob} : \Delta \to 2^{AP^{ob}}$, $L^{ob}(\ell)$ denotes the set of atomic propositions observed by the intruder when transition ℓ occurs.

As an example, we consider a case in which the leaked information is the horizontal position of the robot. Then, we have $AP^{ob} = X \cup X'$, and $L^{ob} : \Delta \to 2^{X \cup X'}$ is given as follows: for each transition $\ell = ((x,y), i, (\hat{x},\hat{y})) \in \Delta$,

$$L^{ob}(\ell) = \{x, \hat{x}'\}. \quad (23)$$

According to (16), labeling function $L : \Delta \to 2^{AP}$, where $AP = AP^p \cup AP^{ob}$ is given as follows: for each transition $\ell = ((x,y), i, (\hat{x},\hat{y})) \in \Delta$,

$$L(\ell) = L^p(\ell) \cup L^{ob}(\ell) = \{(x,y), i, (\hat{x}',\hat{y}'), x, \hat{x}'\}.$$

We consider two types of opacity: initial-state opacity and current-state opacity. Intuitively, a path is initial-state (*resp.* current-state) opaque if there is another path with a different initial state (*resp.* current state) and the same leaked information. Opacity can be described by a hyperLTL$_f^H$ formula. For simplicity, we introduce the following notation ($S_{\pi_1} \equiv S_{\pi_2}$) for set S of atomic propositions and trace variables π_1 and π_2:

$$S_{\pi_1} \equiv S_{\pi_2} := \bigwedge_{s\in S} s_{\pi_1} \equiv s_{\pi_2}. \quad (24)$$

Let $\Pi = \{\pi_1, \pi_2\}$ be the set of trace variables and $V : \Pi \to T(H, \mathbb{P})$ be an assignment function. Path $\rho \in \mathcal{L}(H, \mathbb{P})$ with $\rho[0] = (p[0], i[0], p[1])$ is initial-state opaque if, for assignment function V with $V(\pi_1) = L(\rho)$,

$$V \models_{T(H,\mathbb{P})} \exists\pi_2.\ opac^{init}_{\pi_1,\pi_2}(p[0]), \quad (25)$$

where

$$opac^{init}_{\pi_1,\pi_2}(p) := \neg p_{\pi_2} \wedge \Box(I_{\pi_1} \equiv I_{\pi_2}) \wedge \Box(AP^{ob}_{\pi_1} \equiv AP^{ob}_{\pi_2}). \quad (26)$$

In contrast, path ρ is current-state opaque if, for assignment function V with $V(\pi_1) = L(\rho)$,

$$V \models_{T(H,\mathbb{P})} \exists\pi_2.\ opac^{curr}_{\pi_1,\pi_2}(p[0]), \quad (27)$$

where

$$opac^{curr}_{\pi_1,\pi_2}(p) := p_{\pi_2} \wedge \neg\Box(I_{\pi_1} \equiv I_{\pi_2}) \wedge \Box(AP^{ob}_{\pi_1} \equiv AP^{ob}_{\pi_2}). \quad (28)$$

We define function $cls : \mathcal{L}(H,\mathbb{P}) \times \mathcal{L}(H,\mathbb{P}) \to \mathbb{N}$ as follows: For each $\rho = \rho[0]\rho[1]\dots\rho[H-1]$ and $\rho' = \rho'[0]\rho'[1]\dots\rho'[H-1] \in \mathcal{L}(H,\mathbb{P})$, where $\rho[h] = (p[h], i[h], p[h+1])$ and $\rho'[h] = (p'[h], i'[h], p'[h+1])$ for each $h \in \{0,1,\dots H-1\}$,

$$cls(\rho,\rho') = Ham(i[0]\dots i[H-1], i'[0]\dots i'[H-1]), \quad (29)$$

where $Ham(a,b)$ is the Hamming distance between sequences a and b.

Recall that the finite-horizon shield modifies the pre-planned path such that the modified path satisfies the mandatory specification and security policy. Note that trace variable π_2 represents a path that is different from the modified path but also satisfies the mandatory specification. Thus, the formula for the security policy is obtained as given bellow. For state $p \in P$ that represents the initial state of the pre-planned path, let

$$\varphi^{sp}_{\pi_1,\pi_2}(p) := \begin{cases} opac^{init}_{\pi_1,\pi_2}(p) \wedge \varphi^{s}_{\pi_2} & \text{for the initial-state opacity,} \\ opac^{curr}_{\pi_1,\pi_2}(p) \wedge \varphi^{s}_{\pi_2} & \text{for the current-state opacity.} \end{cases} \quad (30)$$

Then, for pre-planned path ρ_p with $\rho_p[0] = (p_0[0], i_0[0], p_0[1])$, (11) can be rewritten as follows:

$$\begin{aligned} &\text{Check } V \models_{T(H_p,\mathbb{P})} \exists\pi_2.\ \varphi^{sp}_{\pi_1,\pi_2}(p_0[0]) \\ &\text{subject to } V(\pi_1) = L(\rho_p). \end{aligned} \quad (31)$$

(12) can be rewritten as follows:

$$\begin{aligned} &\text{Determine } \rho_s \\ &\text{subject to } \rho_s \in \mathcal{L}(H_s,\mathbb{P}) \\ &\qquad (h>0) \to \bigwedge_{\tau=0}^{h-1}(\rho_s[\tau] = \hat{\rho}_p[\tau]), \\ &\qquad V^p(\pi) = L^p(\rho_s) \\ &\qquad V^p \models_{T^p(H_s,\mathbb{P})} p_0[0]_\pi \wedge \varphi^s_\pi \\ &\qquad V(\pi_1) = L(\rho_s) \\ &\qquad V \models_{T(H_s,\mathbb{P})} \exists\pi_2.\ \varphi^{sp}_{\pi_1,\pi_2}(p_0[0]) \\ &\qquad \wedge\ (cls(\hat{\rho}_p,\rho_s) \le k). \end{aligned} \quad (32)$$

Thus, the finite-horizon shield checks (31) to determine whether ρ_p satisfies opacity. If this is *true*, then the finite-horizon shield outputs ρ_p. Otherwise, $k = 1$ and (32) is repeatedly solved for $k \in \{1,\dots,K_{max}\}$ until there is a solution that provides modified path ρ_s that has length $H_s \ge H_p$, satisfies the mandatory specification and opacity, and is the closest to extended pre-planned path ρ_p in terms of the Hamming distance. If there is no solution for $k \in \{1,\dots,K_{max}\}$, the finite-horizon shield concludes that there is no opaque path whose Hamming distance from the extended pre-planned path is less than or equal to K_{max}.

B. SMT-BASED APPROACH

We encode hyperLTL^H_f formulas using QFs and solve (31) and (32) as the satisfiable problems of the QFs using an SMT solver [53], [59].

We consider the initial-state opacity as a security policy. Let $\rho_p = (p_p[0], i_p[0], p_p[1])(p_p[1], i_p[1], p_p[2])\dots(p_p[H_p-1], i_p[H_p-1], p_p[H_p])$ be a pre-planned path. To check (31), let $p^H_k = \{p_k[0], p_k[1], \dots, p_k[H], i_k[0], i_k[1], \dots, i_k[H-1]\}$, where $k \in \{1,2\}$ is a set of variables that represents a path with length H, $p_k[j] \in P$, and $i_k[j] \in I$. We define the following (quantifier-free) SMT formulas:

$$z^{path}(p^H_1) := \bigwedge_{h=0}^{H-1}(p_1[h+1] = \delta(p_1[h], i_1[h])) \wedge \Big(\bigvee_{p^0\in P^0}(p_1[0] = p^0)\Big), \quad (33)$$

$$z^{s}(p^H_1) := \Big(\bigwedge_{h=1}^{H}\bigvee_{p\in P\setminus D}(p_1[h] = p)\Big) \wedge \Big(\bigvee_{h=1}^{H}\bigvee_{p\in G}(p_1[h] = p)\Big), \quad (34)$$

$$\begin{aligned} z^{sp}(p^H_1, p^H_2) := {} & \neg(p_1[0] = p_2[0]) \\ & \wedge \bigwedge_{h=0}^{H-1}(i_1[h] = i_2[h]) \\ & \wedge \bigwedge_{h=0}^{H}\Big(L^{ob}((p_1[h], i_1[h], p_1[h+1])) \\ & \quad = L^{ob}((p_2[h], i_2[h], p_2[h+1]))\Big) \\ & \wedge z^s(p^H_2) \wedge z^{path}(p^H_2). \end{aligned} \quad (35)$$

(33) indicates that p^H_1 is the path of transition system $\mathbb{P}$. (34) and (35) correspond to the mandatory specification and security policy, respectively.

Then, recall that $h \in [0,\dots,H_p]$ denotes the time when the leaked information observed by the intruder is identified and we have

$$z^{past}(p^H_1, p^H_2, h) := \bigwedge_{\tau=0}^{h}(p_1[\tau] = p_2[\tau]). \quad (36)$$

(36) indicates that the past motion of the robot up to the current state cannot be modified. Moreover, we express function cls as follows:

$$cls(p^H_1, p^H_2) := \sum_{h=0}^{H-1} d(i_1[h], i_2[h]), \quad (37)$$

where

$$d(i_1,i_2) = \begin{cases} 0 & \text{if } i_1 = i_2, \\ 1 & \text{if } i_1 \ne i_2. \end{cases}$$

For simplicity, we define $p_1^{H_p} \leftarrow \rho_p$ as follows.

$$p_1^{H_p} \leftarrow \rho_p := \bigwedge_{h=0}^{H_p} (p_1[h] = p_p[h]) \wedge \bigwedge_{h=0}^{H_p-1} (i_1[h] = i_p[h]).$$

Then, (31) is converted into a satisfiability problem of the SMT formula given by z^{check} as follows:

$$z^{check} := \exists p_2^{H_p}. (p_1^{H_p} \leftarrow \rho_p) \wedge z^{sp}(p_1^{H_p}, p_2^{H_p}). \quad (38)$$

If (38) is satisfiable, ρ_p satisfies the security policy and the finite-horizon shield outputs ρ_p. Otherwise, the finite-horizon shield computes a modified path. Let H_s be the length of the modified path. The extended pre-planned path is given by $\hat{\rho}_p = \rho_p(p_p[H_p], \text{stay}, p_p[H_p+1]) \ldots (p_p[H_s-1], \text{stay}, p_p[H_s])$, where $p_p[h] = p_p[H_p]$ for each $h \in \{H_p+1, \ldots, H_s\}$. Then, (32) is converted into a satisfiability problem of the SMT formula given by z^{mod} as follows:

$$\begin{aligned} z^{mod} := \exists p_1^{H_s} \exists p_2^{H_s}. z^{path}(p_1^{H_s}) \wedge (p_1[0] = p_p[0]) \\ \wedge z^{past}(\hat{\rho}_p, p_1^{H_s}, h) \wedge z^{s}(p_1^{H_s}) \\ \wedge z^{sp}(p_1^{H_s}, p_2^{H_s}) \wedge (cls(\hat{\rho}_p, p_1^{H_s}) \leq k). \end{aligned} \quad (39)$$

If (39) is satisfiable, we have examples $\rho_1^{H_s}$ and $\rho_2^{H_s} \in \mathcal{L}(H_s, \mathbb{P})$ of variables $p_1^{H_s}$ and $p_2^{H_s}$, respectively, and the finite-horizon shield outputs $\rho_1^{H_s}$ as the modified path of ρ_p.

We consider the current-state opacity as a security policy. Then, formula $z^{sp}(p_1^H, p_2^H)$ is expressed as follows:

$$\begin{aligned} &z^{sp}(p_1^H, p_2^H) \\ &:= (p_1[0] = p_2[0]) \\ &\quad \wedge \neg \bigwedge_{h=0}^{H-1} (i_1[h] = i_2[h]) \\ &\quad \wedge \bigwedge_{h=0}^{H} \Big(L^{ob}((p_1[h], i_1[h], p_1[h+1])) \\ &\quad = L^{ob}((p_2[h], i_2[h], p_2[h+1])) \Big) \\ &\quad \wedge z^{s}(p_2^H) \wedge z^{path}(p_2^H). \end{aligned} \quad (40)$$

The finite-horizon shield checks the satisfiability of (38) and (39) to determine output ρ_s.

C. SIMULATIONS

Algorithm 1 is implemented in Python. We consider the following two cases of the leaked information: The intruder can observe the (i) horizontal and the (ii) vertical coordinate of the states. For each case, set AP^{ob} is given by

$$AP^{ob} := \begin{cases} X \cup X' & \text{for case (i)}, \\ Y \cup Y' & \text{for case (ii)}. \end{cases} \quad (41)$$

Labeling function $L^{ob} : \Delta \to 2^{AP^{ob}}$ is expressed as follows: For each $((x, y), i, (\hat{x}, \hat{y})) \in \Delta$,

$$L^{ob}(((x, y), i, (\hat{x}, \hat{y}))) := \begin{cases} \{x, \hat{x}'\} & \text{for case (i)}, \\ \{y, \hat{y}'\} & \text{for case (ii)}. \end{cases} \quad (42)$$

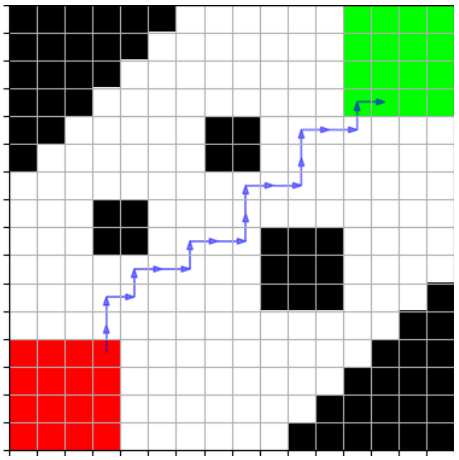

FIGURE 6. Pre-planned path.

We use Z3 [60] as the SMT solver. The results shown in this section are obtained on a system with a 1.8 GHz quad-core processor with 16 GB RAM. $N_x = N_y = 15$, $H = 25$, and $K_{max} = 3$.

1) SIMULATIONS FOR THE INITIAL-STATE OPACITY

The blue arrows in FIGURE 6 indicate the pre-planned path computed by the planner. We simulate the case where leaked information is turned out at time 3. The pre-planned path is the input of the finite-horizon shield. The output ρ_s of the shield that ensures the initial-state opacity for the horizontal (*resp.* vertical) coordinates of the states is indicated by the red arrows in FIGURE 7 (*resp.* FIGURE 8). ρ_s satisfies $cls(\hat{\rho}_p, \rho_s) = 1$ (*resp.* $cls(\hat{\rho}_p, \rho_s) = 0$). The yellow arrows in FIGURES 7 and 8 indicate that the path represented by the red arrows satisfies the initial-state opacity. Gray circles indicate the robot's position at time 3. The light red colored arrows indicate the past movements of the robot, which cannot be modified from the pre-planned path. The pre-planned path shown in FIGURE 6 does not satisfy the initial-state opacity if the intruder can observe the horizontal coordinates of the states. Therefore, the shield modifies the path. However, the initial-state opacity is satisfied if the intruder can observe the vertical coordinates of the states, and the shield does not modify the path. We confirm that the output of the finite-horizon shield depends on the information observed the intruder.

2) SIMULATIONS FOR THE CURRENT-STATE OPACITY

We consider the current-state opacity in case (ii) and the case where the leaked information is identified at time 6. A pre-planned path is shown by the blue arrows in FIGURE 9. The output of the finite-horizon shield is shown in FIGURE 10. As the pre-planned path does not satisfy the current-state opacity, the output is modified by adding the input that the robot moves to the right at the end of path ρ_p. Output ρ_s satisfies $cls(\hat{\rho}_p, \rho_s) = 1$. In this case, we can not obtain a path that satisfies the current-state opacity for $H_s = H_p$.

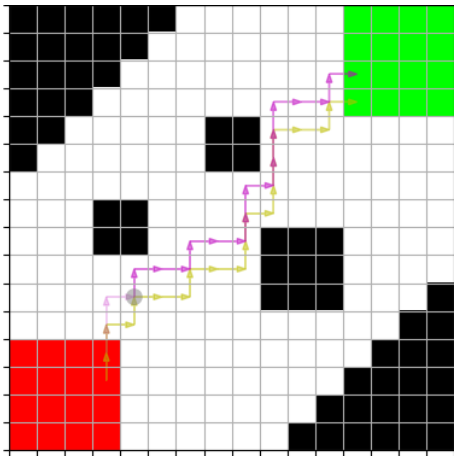

FIGURE 7. Output for the horizontal coordinates of the states (modified).

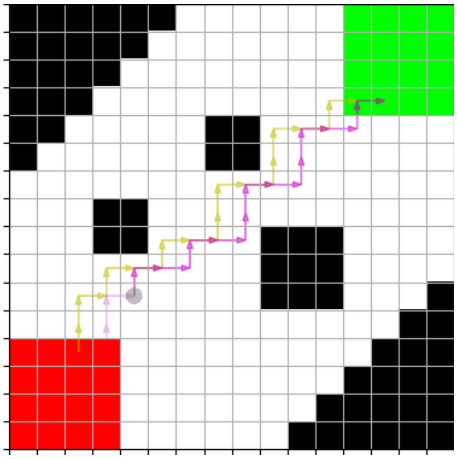

FIGURE 8. Output for the vertical coordinates of the states (not modified).

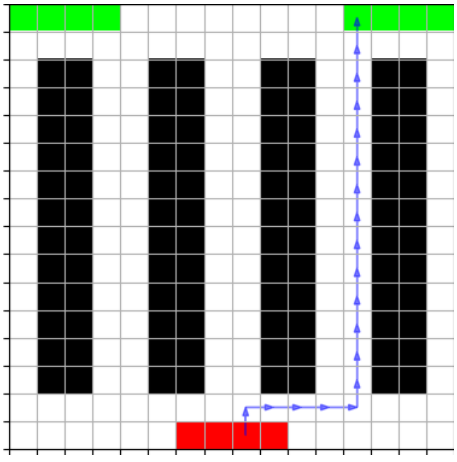

FIGURE 9. Pre-planned path.

We confirm that the finite-horizon shield outputs a modified path that has additional movement to ensure opacity.

VI. COMPARISON WITH SECURITY-AWARE PLANNING

If the leaked information is known when a pre-planned path is computed, the finite-horizon shield is applied with $h = 0$. But, in this case, we can compute a security-aware path directly without using the finite-horizon shield. Thus, in this section, we consider the case where the leaked information is identified beforehand and compare the finite-horizon shield with secure-aware planning.

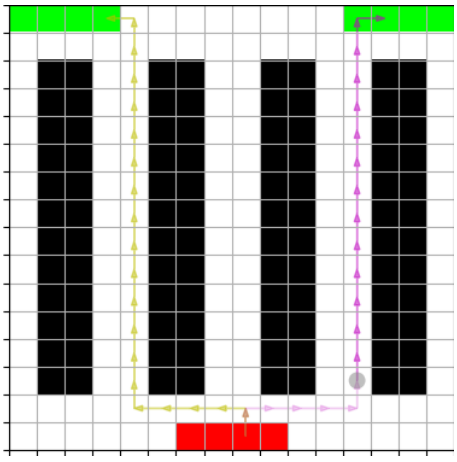

FIGURE 10. Output (modified).

A. SECURITY-AWARE PLANNING

We consider a security-aware planner that computes path ρ_s that satisfy optional specification φ^o as much as possible while satisfying safety/co-safety specification φ^s and security policy ψ^{sp} under the known leaked information. A security-aware planning problem has been considered in [34], but no optional specifications have been considered, which is different from our problem setting. The security-aware planner has the role of both the (non-secure) pre-planner to determine the pre-planed path and finite-horizon shield to guarantee the security policy. In other words, it finds path ρ_p that satisfies path specification $\varphi^p = \varphi^s \wedge \varphi^o$, and simultaneously determine path ρ^s that is as close to ρ_p as possible among paths that satisfy φ^s and ψ^{sp}. For $k \in \{0, \ldots, K_{max} - 1\}$, the security-aware planner determine if there exists path ρ_s whose closeness to ρ_p is less than or equal to k. If not, it recomputes for $k + 1$. Let $\Pi^p = \{\pi, \pi'\}$ and $\Pi = \{\pi_1\}$ be sets of trace variables. Let $V^p : \Pi^p \to T^p(H_s, \mathbb{P})$ and $V : \Pi \to T(H_s, \mathbb{P})$ be assignment functions. Then the security-aware planner computes the following problem.

$$
\begin{aligned}
&\text{Determine} \quad \rho_s \\
&\text{subject to } \rho_p,\, \rho_s \in \mathcal{L}(H_s, \mathbb{P}), \\
&\qquad V^p(\pi) = L^p(\rho_s), \\
&\qquad V^p(\pi') = L^p(\rho_p) \\
&\qquad V^p \models_{T^p(H_s,\mathbb{P})} \varphi^s_\pi \wedge \varphi^s_{\pi'} \wedge \varphi^o_{\pi'}, \\
&\qquad V(\pi_1) = L(\rho_s), \\
&\qquad V \models_{T(H_s,\mathbb{P})} \mathbb{Q}_2\pi_2\mathbb{Q}_3\pi_3 \ldots \mathbb{Q}_n\pi_n. \\
&\qquad \varphi^{sp} \wedge \left(cls(\rho_p, \rho_s) \leq k\right),
\end{aligned}
\tag{43}
$$

where φ_π indicates that the subscript of all atomic propositions in φ is π. Algorithm 2 shows the procedure of computing path ρ_s in the security-aware planner.

Algorithm 2 Compute ρ_s via the Security-Aware Planner

Input: K_{max}
Output: ρ_s, cls

```
for k = 0 to K_max do
    Encode (43)
    if (43) is satisfied then
        cls ← k
        return ρ_s, cls
    end if
end for
return -1
```

B. DISCUSSION

We discuss the computation times of the finite-horizon shield (Algorithm 1) and the security-aware planning (Algorithm 2). Both algorithms are based on the satisfiability of quantified formulas and are PSAPCE-complete [56].

We discuss the numbers of variables used in both algorithms. While the number of variables used for the pre-planner and Algorithm 1 are N_{pre} and N_{Alg1}, shown in (14), (13), that used for Algorithm 2 is $N_{pre} + N_{Alg1}$. Algorithm 1 divides the problem into two parts and computes them sequentially. Since QSAT problems are PSPACE-complete, it is expected that the computation time of Algorithm 2 will be longer than that of Algorithm 1 as the size of the problem increases.

On the other hands, both algorithms are sound, that is, paths that they return satisfy the mandatory specifications (the safety/co-safety specification and the security policy) and whose closeness to a path satisfying the optional specification is less than or equal to the parameter K_{max}. Moreover, Algorithm 2 returns such a path whenever it exists, that is, Algorithm 2 is complete. However, Algorithm 1 computes a pre-planned path ρ_p that satisfies the mandatory specifications and outputs a path whose closeness to ρ_s is less than or equal to K_{max}. Thus, there is no guarantee that the finite-horizon shield always outputs a desired path even if it exists, that is, Algorithm 1 is not complete.

From the above discussions, to reduce the computation time, Algorithm 1 is useful but may fail to find a desired path even if it exists.

C. SIMULATION

We investigate computation times for Algorithms 1 and 2 by simulation. Let $task \subset P$ be a set of states and $\varphi^o = \Diamond task$ be an optional specification. TABLE 1 shows the times required to compute the paths that guarantee initial-state opacity when we use Algorithms 1 and 2, respectively. Simulation is performed with $H_p = H_s$ for the same grid size. If the grid size is 6×6, 7×7, or 8×8, Algorithm 2 takes less computation times than Algorithm 1. However, If it is equal or larger than 9×9, Algorithm 1 takes less computation times. As the size of the problem increases, Algorithm 1 is more efficient in terms of computation time. TABLE 2 shows the satisfaction number of the optional specification. The notation x/y in TABLE 2 indicates that the optional specification is satisfied x times in y experiments. If there exists a path that satisfies safety/co-safety and optional specifications and security policy, Algorithm 2 always return it. Therefore, the probability of satisfying the optional specification is larger than that in Algorithm 1. TABLE 2 also shows the ratio defined by n_1/n_2, where n_1 (*resp.* n_2) is the number of experiments for which Algorithm 1 (*resp.* Algorithm 2) returns a path satisfying φ^o. We confirm that the ratio is around 0.8 regardless of the grid size. The result shown in TABLES 1 and 2 are obtained on a computer with a 3.4 GHz 16-core processor with 128 GB RAM.

VII. CONCLUSION AND FUTURE WORK

We develop a finite-horizon shield that ensures that a finite trace satisfies hyperLTL$_f^H$ specifications. This shield checks and modifies the finite trace. We express the requirement of the finite-horizon shield for hyperLTL$_f^H$ formulas and propose an algorithm to compute the output of the shield using an SMT solver. Then, we consider an opaque path planning problem in which a pre-planned path that satisfies specifications, including safety/co-safety properties as mandatory specifications, is computed by a planner. This path is modified as small as possible while satisfying mandatory specifications and opacity. Simulations confirm that the modified path is suitable when an intruder exists.

Moreover, the finite-horizon shield is applicable for the case where the leaked information is known beforehand. Then, its computation time is less than that of the security-aware planning when the planning problems are sufficiently large.

In future work, we will extend the proposed finite-horizon shield for hyperproperties to other settings such as multi-agent systems or model predictive control [61]. Moreover, extensions to an infinite-horizon planning problem such as a surveillance problem are also interesting future work and one research direction is a usage of a lasso-type infinite path. Another research direction is the construction of the shield using an automaton-game-based approach: by describing ω-regular specifications, it is possible to construct a shield that guarantees security for non-terminating systems such as web servers.

We modified a pre-planned path in such a way that a distance between the modified path and pre-planned path is less than or equal to a given constant. But, there is a different modification such as the revision of the optional specification and it is a future work how to revise the specification such that an effect of the modification is as small as possible.

APPENDIX A
ENCODING OF HyperLTL FORMULAS

In the bounded model checking for hyperLTL, a problem is converted into a satisfiability problem of the QFs that are the encodings of hyperLTL formulas [53]. We review the encodings of the hyperLTL formulas using QFs. Consider a set of trace variables, Π, and an assignment function, V :

TABLE 1. Computation time for Algorithm 1 and 2.

grid size	H_p or H_s	Algorithm 1			Algorithm 2
		pre-planner [s]	finite-horizon shield [s]	total [s]	security-aware planner [s]
6×6	11	0.371	1.52	1.89	1.42
7×7	12	0.565	2.72	3.28	2.09
8×8	13	0.843	3.30	4.15	3.38
9×9	14	1.20	5.71	6.92	8.71
10×10	15	1.72	7.96	9.68	10.8
11×11	16	2.41	9.65	12.1	17.9
12×12	17	3.49	17.1	20.6	31.3
13×13	18	5.09	18.9	24.0	75.2
14×14	20	7.26	24.2	31.5	201
15×15	22	17.5	23.0	40.5	303

TABLE 2. Satisfaction number of optional specification for Algorithm 1 and 2.

grid size	Algorithm 1	Algorithm 2	ratio
6×6	79/100	98/100	0.81
7×7	75/100	100/100	0.75
8×8	85/100	100/100	0.85
9×9	78/100	98/100	0.80
10×10	80/100	99/100	0.81
11×11	83/100	100/100	0.83
12×12	79/100	100/100	0.79
13×13	87/100	100/100	0.87
14×14	88/100	100/100	0.88
15×15	80/100	100/100	0.80

$\Pi \to T(H, \mathbb{P})$. Let $\pi \in \Pi$ and $\rho \in \mathcal{L}(H, \mathbb{P})$ denote a trace variable and path, respectively. For atomic proposition $ap \in AP$ and time $h \in \{0, 1, \ldots, H-1\}$, we introduce Boolean variable $z_{ap_\pi, \rho}(h) \in \{true, false\}$ that is *true* if and only if $V(\pi) = L(\rho)$ and $ap \in V(\pi)[h]$. We define Boolean variable $z_\varphi(h) \in \{true, false\}$ that indicates the satisfaction of quantifier-free hyperLTL$_f^H$ formula φ at time $h \in [0, H-1]$. $z_\varphi(h)$ is *true* if and only if $V[h, H-1] \models_{T(H,\mathbb{P})} \varphi$. The temporal operators used in the hyperLTL$_f^H$ formulas are encoded as follows:

- eventually operator : $\varphi' = \Diamond\varphi$

$$z_{\varphi'}(h) = \bigvee_{\tau=h}^{H-1} z_\varphi(\tau). \tag{44}$$

- always operator : $\varphi' = \Box\varphi$

$$z_{\varphi'}(h) = \bigwedge_{\tau=h}^{H-1} z_\varphi(\tau). \tag{45}$$

- until operator : $\varphi' = \varphi_1 \mathcal{U} \varphi_2$

For $h = 0, \ldots, H-2$,

$$z_{\varphi'}(h) = z_{\varphi_2}(h) \vee \left(z_{\varphi_1}(h) \wedge z_{\varphi'}(h+1)\right). \tag{46}$$

For $h = H-1$,

$$z_{\varphi'}(H-1) = z_{\varphi_2}(H-1). \tag{47}$$

We describe the encoding of path ρ given by (3). $\rho = \rho[0]\rho[1]\ldots\rho[H-1]$, where $\rho[h] = (p[h], i[h], p[h+1]) \in \Delta$ for each $h \in [0, H-1]$. For any transition $\ell = (p, i, p')$ and $\hat{\ell} = (\hat{p}, \hat{i}, \hat{p}') \in \Delta$, we introduce predicate $C(\ell, \hat{\ell})$ which is *true* if and only if $p' = \hat{p}$. Then, we have

$$\rho \in \mathcal{L}(H, \mathbb{P}) \Leftrightarrow \bigwedge_{h=0}^{H-2} C(\rho[h], \rho[h+1]). \tag{48}$$

For assignment $V(\pi) = L(\rho)$ and hyperLTL$_f^H$ formula ψ, we introduce Boolean variable $z_\psi(0) \in \{true, false\}$. $z_\psi(0)$ is *true* if and only if $V \models_{T(H,\mathbb{P})} \psi$. The trace quantifiers used in the hyperLTL$_f^H$ formulas are encoded as follows:

- exists: $\psi' = \exists\pi.\ \psi$

$$z_{\psi'}(0) = \exists\rho[0]\exists\rho[1]\ldots\exists\rho[H-1] \cdot \bigwedge_{h=0}^{H-2} C(\rho[h], \rho[h+1]) \wedge z_\psi(0). \tag{49}$$

- forall: $\psi' = \forall\pi.\ \psi$

$$z_{\psi'}(0) = \forall\rho[0]\forall\rho[1]\ldots\forall\rho[H-1] \cdot \left(\bigwedge_{h=0}^{H-2} C(\rho[h], \rho[h+1])\right) \to z_\psi(0). \tag{50}$$

REFERENCES

[1] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods Syst. Des.*, vol. 19, no. 3, pp. 291–314, Oct. 2001.
[2] B. Könighofer, M. Alshiekh, R. Bloem, L. Humphrey, R. Könighofer, U. Topcu, and C. Wang, "Shield synthesis," *Form. Methods Syst. Des.*, vol. 51, no. 2, pp. 332–361, 2017.
[3] M. Wu, H. Zeng, and C. Wang, "Synthesizing runtime enforcer of safety properties under burst error," in *Proc. 8th Int. Symp. NASA Form. Methods*. Cham, Switzerland: Springer, 2016, pp. 65–81.
[4] S. Bharadwaj, R. Bloem, R. Dimitrova, B. Konighofer, and U. Topcu, "Synthesis of minimum-cost shields for multi-agent systems," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 1048–1055.
[5] J. P. Inala, Y. J. Ma, O. Bastani, X. Zhang, and A. Solar-Lezama, "Safe human-interactive control via shielding," 2021, *arXiv:2110.05440*.
[6] H. Hu, K. Nakamura, and J. F. Fisac, "SHARP: Shielding-aware robust planning for safe and efficient human-robot interaction," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 5591–5598, Apr. 2022.
[7] S. Pranger, B. Konighofer, M. Tappler, M. Deixelberger, N. Jansen, and R. Bloem, "Adaptive shielding under uncertainty," in *Proc. Amer. Control Conf. (ACC)*, May 2021, pp. 3467–3474.

[8] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2669–2678.

[9] I. Elsayed-Aly, S. Bharadwaj, C. Amato, R. Ehlers, U. Topcu, and L. Feng, "Safe multi-agent reinforcement learning via shielding," in *Proc. 20th Int. Conf. Auton. Agents Multiagent Syst.*, 2021, pp. 483–491.

[10] S. Demri, V. Goranko, and M. Lange, *Temporal Logics in Computer Science Finite-State Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[11] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.

[12] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Cham, Switzerland: Springer, 2017.

[13] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," in *Proc. 16th ACM SIGPLAN-SIGACT Symp. Princ. Program. Lang. (POPL)*, 1989, pp. 179–190.

[14] M. Y. Vardi, "An automata-theoretic approach to linear temporal logic," *Logics for Concurrency: Structure Versus Automata* (Lecture Notes in Computer Science), vol. 1043. Cham, Switzerland: Springer, 1996, pp. 238–266.

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

[16] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive(1) designs," *J. Comput. Syst. Sci.*, vol. 78, no. 3, pp. 911–938, May 2012.

[17] E. Filiot, N. Jin, and J.-F. Raskin, "Antichains and compositional algorithms for LTL synthesis," *Formal Methods Syst. Des.*, vol. 39, no. 3, pp. 261–296, Dec. 2011.

[18] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Trans. Autom. Control*, vol. 57, no. 11, pp. 2817–2830, Nov. 2012.

[19] R. Dimitrova, M. Ghasemi, and U. Topcu, "Reactive synthesis with maximum realizability of linear temporal logic specifications," *Acta Inf.*, vol. 57, nos. 1–2, pp. 107–135, Apr. 2020.

[20] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech and S. Yovine, Eds. Cham, Switzerland: Springer, 2004, pp. 152–166.

[21] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Proc. 8th Int. Conf. Formal Model. Anal. Timed Syst.*, 2010, pp. 92–106.

[22] V. Raman, A. Donze, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 81–87.

[23] L. Lindemann and D. V. Dimarogonas, "Robust control for signal temporal logic specifications using discrete average space robustness," *Automatica*, vol. 101, pp. 377–387, Mar. 2019.

[24] M. Charitidou and D. V. Dimarogonas, "Barrier function-based model predictive control under signal temporal logic specifications," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2021, pp. 734–739.

[25] M. Wu, J. Wang, J. V. Deshmukh, and C. Wang, "Shield synthesis for real: Enforcing safety in cyber-physical systems," in *Proc. Form. Methods Comput. Aided Des.*, 2019, pp. 129–137.

[26] P. K. Pandya and A. Wakankar, "Specification and optimal reactive synthesis of run-time enforcement shields," *Inf. Comput.*, vol. 285, May 2022, Art. no. 104865.

[27] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Proc. Int. Joint Conf. Artif. Intell.*, 2013, pp. 854–860.

[28] J. Li, G. Pu, Y. Zhang, M. Y. Vardi, and K. Y. Rozier, "SAT-based explicit LTL_f satisfiability checking," *Artif. Intell.*, vol. 289, Dec. 2020, Art. no. 103369.

[29] S. Zhu, L. M. Tabajara, J. Li, G. Pu, and M. Y. Vardi, "Symbolic LTL_f synthesis," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 1362–1369.

[30] S. Zhu, G. De Giacomo, G. Pu, and M. Y. Vardi, "LTL_f synthesis with fairness and stability assumptions," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 3088–3095.

[31] K. He, A. M. Wells, L. E. Kavraki, and M. Y. Vardi, "Efficient symbolic reactive synthesis for finite-horizon tasks," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8993–8999.

[32] K. Cho, J. Suh, C. J. Tomlin, and S. Oh, "Cost-aware path planning under co-safe temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2308–2315, Oct. 2017.

[33] K. Hashimoto, N. Tsumagari, and T. Ushio, "Collaborative rover-copter path planning and exploration with temporal logic specifications based on Bayesian update under uncertain environments," *ACM Trans. Cyber-Phys. Syst.*, vol. 6, no. 2, pp. 1–24, Apr. 2022.

[34] S. Yang, X. Yin, S. Li, and M. Zamani, "Secure-by-construction optimal path planning for linear temporal logic tasks," in *Proc. 59th IEEE Conf. Decis. Control (CDC)*, Dec. 2020, pp. 4460–4466.

[35] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Comput. Commun.*, vol. 149, pp. 270–299, Jan. 2020.

[36] L. Mazaré, "Using unification for opacity properties," in *Proc. Workshop Issues Theory Secur.*, 2004, pp. 165–176.

[37] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau, "Concurrent secrets," *Discrete Event Dyn. Syst.*, vol. 17, no. 4, pp. 425–446, Nov. 2007.

[38] J. Dubreil, P. Darondeau, and H. Marchand, "Supervisory control for opacity," *IEEE Trans. Autom. Control*, vol. 55, no. 5, pp. 1089–1100, May 2010.

[39] J. C. Basilio, C. N. Hadjicostis, and R. Su, "Analysis and control for resilience of discrete event systems: Fault diagnosis, opacity and cyber security," *Found. Trends Syst. Control*, vol. 8, no. 4, pp. 285–443, 2021.

[40] L. An and G.-H. Yang, "Opacity enforcement for confidential robust control in linear cyber-physical systems," *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 1234–1241, Mar. 2020.

[41] M. Mizoguchi and T. Ushio, "Abstraction-based control under quantized observation with approximate opacity using symbolic control barrier functions," *IEEE Control Syst. Lett.*, vol. 6, pp. 2222–2227, 2022.

[42] S. Liu, A. Trivedi, X. Yin, and M. Zamani, "Secure-by-construction synthesis of cyber-physical systems," *Annu. Rev. Control*, vol. 53, pp. 30–50, Apr. 2022.

[43] B. Finkbeiner, M. N. Rabe, and C. Sánchez, "Algorithms for model checking HyperLTL and HyperCTL," in *Proc. Int. Conf. Comput. Aided Verif.*, 2015, pp. 30–48.

[44] J. Baumeister, N. Coenen, B. Bonakdarpour, B. Finkbeiner, and C. Sanchez, "A temporal logic for asynchronous hyperproperties," in *Proc. Int. Conf. Comput. Aided Verif.*, 2021, pp. 694–717.

[45] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. and Sánchez, "Temporal logics for hyperproperties," in *Principles of Security and Trust*, M. Abadi and S. Kremer, Eds. Cham, Switzerland: Springer, 2014, pp. 265–284.

[46] Y. Wang, S. Nalluri, and M. Pajic, "Hyperproperties for robotics: Planning via HyperLTL," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 8462–8468.

[47] J. Zhao, X. Yin, and S. Li, "A unified framework for verification of observational properties for partially-observed discrete-event systems," 2022, *arXiv:2205.01392*.

[48] C. N. Hadjicostis, "Trajectory planning under current-state opacity constraints," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 337–342, 2018.

[49] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," in *Proc. Int. Conf. Tools Algorithms Constr. Anal. Syst.* Cham, Switzerland: Springer, 1999, pp. 193–207.

[50] A. Biere and D. Kröning, "SAT-based model checking," in *Handbook of Model Checking*, E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, Eds. Cham, Switzerland: Springer, 2018, pp. 277–303.

[51] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu, "Bounded model checking," in *Handbook of Satisfiability*, A. Biere, M. Heule, H. Van Maaren, and T. Walsh, Eds. Amsterdam, The Netherlands: IOS Press, 2009, pp. 457–481.

[52] A. Biere, K. Heljanko, T. Junttila, T. Latvala, and V. Schuppan, "Linear encodings of bounded LTL model checking," *Log. Methods Comput. Sci.*, vol. 2, no. 5, pp. 1–64, Nov. 2006.

[53] T.-H. Hsu, C. Sánchez, and B. Bonakdarpour, "Bounded model checking for hyperproperties," in *Proc. Int. Conf. Tools Algorithms Constr. Anal. Syst.* Cham, Switzerland: Springer, 2021, pp. 94–112.

[54] A. Saboori and C. N. Hadjicostis, "Opacity-enforcing supervisory strategies via state estimator constructions," *IEEE Trans. Autom. Control*, vol. 57, no. 5, pp. 1155–1165, May 2012.

[55] N. Coenen, B. Finkbeiner, C. Hahn, J. Hofmann, and Y. Schillo, "Runtime enforcement of hyperproperties," in *Proc. Int. Symp. Autom. Tech. Verif. Anal.* Cham, Switzerland: Springer, 2021, pp. 283–299.
[56] C. H. Papadimitriou, *Computational Complexity*. Reading, MA, USA: Addison-Wesley, 1994.
[57] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. 3rd Annu. ACM Symp. Theory comput.*, 1971, pp. 151–158.
[58] T. Tomita, A. Ueno, M. Shimakawa, S. Hagihara, and N. Yonezaki, "Safraless LTL synthesis considering maximal realizability," *Acta Inf.*, vol. 54, no. 7, pp. 655–692, Nov. 2017.
[59] D. Kroening and O. Strichman, *Decision Procedures: An Algorithmic Point of View*. Cham, Switzerland: Springer, 2008.
[60] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Proc. Int. Tools Algorithms Constr. Anal. Syst.* Cham, Switzerland: Springer, 2008, pp. 337–340.
[61] B. Kouvaritakis and M. Cannon, *Model Predictive Control: Classical, Robust and Stochastic*. Cham, Switzerland: Springer, 2016.

KOKI KANASHIMA received the B.S. degree from Osaka University, Japan, in 2021, where he is currently pursuing the graduate degree. His research interest includes formal approaches to path planning.

TOSHIMITSU USHIO (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Kobe University, Japan, in 1980, 1982, and 1985, respectively. He was a Research Assistant at UC Berkeley, in 1985. He was a Research Associate at Kobe University, from 1986 to 1990. In 1994, he joined Osaka University, where he is currently a Professor. His research interests include the control of discrete event and hybrid systems and analysis of nonlinear systems.