Hyperproperties for Robotics: Planning via HyperLTL

Yu Wang, Siddhartha Nalluri, and Miroslav Pajic

Abstract— There is a growing interest on formal methods-based robotic planning for temporal logic objectives. In this work, we extend the scope of existing synthesis methods to hyper-temporal logics. We are motivated by the fact that important planning objectives, such as optimality, robustness, and privacy, (maybe implicitly) involve the interrelation between multiple paths. Such objectives are thus hyperproperties, and cannot be expressed with usual temporal logics like the linear temporal logic (LTL). We show that such hyperproperties can be expressed by HyperLTL, an extension of LTL to multiple paths. To handle the complexity of planning with HyperLTL specifications, we introduce a symbolic approach for synthesizing planning strategies on discrete transition systems. Our planning method is evaluated on several case studies.

I. INTRODUCTION

The past decade has seen an increasing interest on planning problems from temporal logic objectives (e.g., [1]–[3]). Using temporal logics, such as the linear temporal logic (LTL) [4]–[6], a wide class of objectives beyond reachability can be defined; these include infinite recurrence, complex dependency of many tasks [7], and time-dependent formations of multiple robotics [8]–[10].

However, temporal logics commonly used in robotics (e.g., for planning) can only specify properties for individual executions (i.e., paths). This effectively prevents them from capturing important planning objectives, such as optimality, robustness, and privacy/opacity, which involve inter-relations between multiple paths. For example, to (directly) specify an optimal strategy for a temporal logic objective, we should ask for the existence of a path π such that all other paths π' are no better than π. Previous works on optimal or robust planning with temporal logic objectives rely on converting these objectives into cost functions [11]–[13], or into games [14]. Such conversions are specific to individual problems.

In this paper, we show that such objectives, which directly specify the relations of multiple paths (or computation trees) and are commonly referred to as *hyperproperties* [15], can be *generally* and *formally* specified by hyper-temporal logics, such as HyperLTL [16].[1] HyperLTL extends LTL with a set of path variables to denote individual paths, and associates each atomic proposition with a path variable to indicate on which path it should hold. HyperLTL also allows for the

use of "exists" $\exists$ and "for all" $\forall$ quantifications of the path variables; this enables specifying relevant planning objectives such as the described planning optimality requirement that employs $\exists\pi\forall\pi'$ quantifiers.

We also propose a symbolic method to synthesize planning strategies with performance (e.g., optimality, robustness, and privacy) guarantees specified by HyperLTL. Specifically, we study the planning from HyperLTL objectives on a commonly used modeling formalism – discrete transition systems (DTSs), where the states are discrete and the transitions are driven by actions. This model can be viewed as a high-level discrete abstraction of the full workspace [8], [9], that is obtained by the low-level explorations, like RRT or probabilistic roadmaps [17], or from the abstraction and simulation [1].

For finite-state discrete models like finite-state DTS, feasible strategies for temporal logic objectives, such as LTL, can be synthesized using automata-theoretic model checking (e.g., [1], [18]–[20]). Specifically, the objective is first converted to an automaton, and then the strategy-search is performed on the product of the automaton and the discrete system model. However, this approach is extremely computationally intensive for HyperLTL objectives, since the possible quantifier alternation, e.g., $\exists\pi\forall\pi'$ for optimality objectives, dramatically increases the state of corresponding automata. In addition, to keep track of the n paths involved in φ (e.g., $n=2$ for the optimality objectives since they employ two path variables π and π', as we introduce in (1)), the n-fold self-product of the discrete system model is needed. As a result, in the formal methods community, automata-theoretic model checking of HyperLTL is mainly confined to quantifier-alternation-free objectives [21].

Consequently, to mitigate the state explosion for HyperLTL objectives in robotic planning, we adopt a symbolic approach for synthesizing strategies via SMT solvers [22], [23]. Specifically, the dynamics of the DTS model is converted into a set of logic formulas, and feasible strategies should satisfy the conjunction of the HyperLTL objectives and model dynamics. This conjunction is a first-order logic formula whose solution can be obtained by using off-the-shelf SMT solvers such as Z3 [24], Yices [25] or CVC4 [26]. As with previous work on symbolic synthesis from regular LTL (e.g., [7], [23], [27], [28]), we focus on HyperLTL objectives with a bounded time horizon T, which we referred to as HyperLTLf.

Compared to the automata-theoretic approach, the symbolic synthesis method yields a more compact representation of the regular planning models; thus, it reduces the synthesis complexity by avoiding constructing the n-fold self product [7], [27], and can even handle planning on DTSs with infinite states [28], [29]. We show in case studies that our symbolic synthesis approach effectively handles planning problems

[1]It is worth noting that non-hyper temporal logics, such as the Computation Tree Logic (CTL), cannot express such relations between multiple non-nested computation trees; HyperLTL subsumes CTL [15].

with hyper temporal logic objectives, deriving strategies with strong robustness, privacy/opacity, and optimality guarantees.

This paper is organized as follows. After preliminaries (Section II), we show the need for the use of hyperproperties in path planning (Section III). We formulate our planning problem on DTSs in Section IV, and show how HyperLTL objectives should be used to ensure robustness, privacy/opacity, and optimality of derived plans (Section V). We introduce a symbolic synthesis method for the HyperLTL objectives on the DTSs in Section VI, evaluate it on case studies in Section VII, and finally conclude this work in Section VIII.

II. PRELIMINARIES

Notation: The sets of integers and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$. The phrase "if and only if" is abbreviated as "iff". For $n \in \mathbb{N}$, let $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$ and $[n] = \{1, \ldots, n\}$. The cardinality and the power set of a set are denoted by $|\cdot|$ and $2^{\cdot}$, and $\emptyset$ denotes the empty set.

Linear Temporal Logic (LTL): Let AP be a set of properties (atomic propositions) related to the planning objective. Formally, an LTL objective (i.e., specification) is constructed inductively by (the syntax)

$$\varphi ::= a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \bigcirc \varphi \mid \varphi \, \mathcal{U}_T \, \varphi,$$

where $T \in \mathbb{N}_\infty$ and $a \in AP$; note that using "bounded until" $\mathcal{U}_T$ instead of the common "until" $\mathcal{U}$ is equivalent in defining LTL [30]. For a path $\pi : \mathbb{N} \to 2^{AP}$, where $\pi(t)$ is the set of properties satisfied at time t by π, satisfaction of an LTL formula on path π is checked recursively via (the semantics)

$$
\begin{aligned}
\pi &\models a & &\Leftrightarrow & a &\in \pi(0) \\
\pi &\models \neg\varphi & &\Leftrightarrow & \pi &\not\models \varphi \\
\pi &\models \varphi_1 \wedge \varphi_2 & &\Leftrightarrow & \pi &\models \varphi_1 \text{ and } \pi \models \varphi_2 \\
\pi &\models \bigcirc\varphi & &\Leftrightarrow & \pi^{(1)} &\models \varphi \\
\pi &\models \varphi_1 \, \mathcal{U}_T \, \varphi_2 & &\Leftrightarrow & \exists t &\leq T. \, \big(\pi^{(t)} \models \varphi_2 \text{ and} \\
& & & & & (\forall t' < t. \, \pi^{(t')} \models \varphi_1)\big)
\end{aligned}
$$

where $\pi^{(t)}(\cdot) = \pi(\cdot + t)$ is the t-time shift. Roughly, $\bigcirc a$ means a holds *next*, and $a_1 \, \mathcal{U}_T \, a_2$ means a_1 holds *until* a_2 holds before T. Other common logic operators are derived by

Or: $\varphi \vee \varphi' \equiv \neg(\neg\varphi \wedge \neg\varphi')$, Implies: $\varphi \Rightarrow \varphi' \equiv \neg\varphi \vee \varphi'$

Finally: $\diamondsuit_T \varphi \equiv \mathsf{T} \, \mathcal{U}_T \, \varphi$, Always: $\square_T \varphi \equiv \neg\diamondsuit_T \neg\varphi$.

We also denote $\mathcal{U}_\infty, \diamondsuit_\infty, \square_\infty$ by $\mathcal{U}, \diamondsuit, \square$, respectively.

A large class of planning objectives can be expressed with LTL. For example, "visiting a_1 and then a_2" can be expressed by $\diamondsuit(a_1 \wedge \diamondsuit a_2)$; and "always returning to the point a_1 within time T after leaving it" can be captured as $\square(a_1 \Rightarrow \diamondsuit_T a_1)$.

III. HYPERLOGICS FOR ROBOTICS PLANING

Although non-hyper temporal logics, such as LTL, are very expressive in temporal relations, they can only do so for individual paths; for example, whether a path π_1 in Figure 1 reaches the goal before hitting an obstacle. However, many important planning objectives involve the interrelation between multiple paths, and thus cannot be expressed by these non-hyper temporal logics. On the other hand, these objectives can be expressed in hyper-temporal logics, where explicit quantifications over different paths are allowed. In this section, we show the need for the use of hyperproperties in robotic planning on several motivating examples.
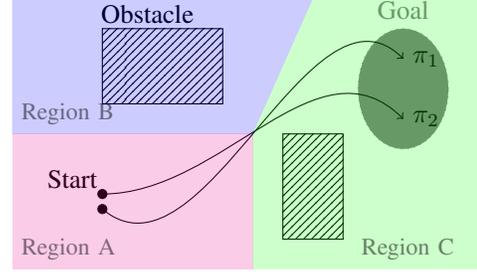

Fig. 1. The use of hyperproperties in planning.

Optimality of Synthesized Plans: A well-known shortcoming of LTL-based planning is the lack of support for optimality. For example, the objectives such as "reaching the goal with the shortest time" cannot be expressed in LTL, since formally defining such objectives implicitly involves comparison of the optimal path and other paths. On the other hand, with explicit path quantifications, the objective is achieved by finding a path π such that

$$
\begin{aligned}
\exists \pi. \big(&(\pi \text{ reaches goal}) \wedge \\
&\big(\forall \pi'. ((\pi' \text{ reaches goal}) \Rightarrow (\pi \text{ reaches goal}))\big)\big).
\end{aligned}
\tag{1}
$$

Robustness of Synthesized Plans: A major concern for open-loop planning is robustness of the derived strategy. Specifically, (i) the assumed initial position may be inaccurate; or (ii) an action may not be executed correctly due to faults or attacks. Yet, in many cases, the knowledge of the possible forms of inaccuracy, faults or attacks are available. Hyper-temporal logics (e.g., HyperLTL) allow for incorporating this knowledge into the design objectives to preemptively synthesize strategies that are immune to those adversarial/environmental factors. For example, a robust strategy under disturbance may be specified by

$$
\begin{aligned}
\exists \pi \forall \pi'. \big(&\pi' \text{ is derived by disturbing } \pi\big) \\
&\wedge \big(\pi \text{ and } \pi' \text{ reach goal}\big).
\end{aligned}
\tag{2}
$$

Planning with Privacy/Opacity: A problem that has recently attracted significant attentions (e.g., [31]–[33]) is ensuring location privacy in mobile navigation – i.e., keeping the individual locations private, even when they are partially shared to achieve coordinated planning (e.g., coverage). Opacity ensures location privacy by requiring that for a planned path, there exists (at least) another different path, such that the shared partial location information is identical for the two paths; hence, they are anonymized. An example is illustrated in Figure 1, where a path is partially observed by whether the robot is in Region A, B, or C or it reaches the goal. Synthesizing an opaque planning strategy to reach the goal implies finding a path π (or equivalently π') such that

$$
\begin{aligned}
\exists \pi \exists \pi'. \big(&\pi \text{ and } \pi' \text{ are different paths}\big) \\
&\wedge \big(\pi \text{ and } \pi' \text{ give identical observation}\big) \\
&\wedge \big(\pi \text{ and } \pi' \text{ reach goal}\big).
\end{aligned}
\tag{3}
$$

The paths π and π' in Figure 1 are examples of privacy-preserving paths, as they go through different regions, finally reaching the goal in the same pace; thus are indistinguishable.

IV. Planning on Discrete Transition Systems

We consider the planning on a discrete domain, which can be either the full model of a complex workspace or its high-level abstraction derived from either simulation relation [34] or random exploration [17]. On the domain, the robot motion is modeled by a DTS whose transitions are labeled by actions.

Definition 1 (DTS). *Given a set of atomic propositions* AP *a DTS is a tuple* $\mathcal{M} = (S_{\mathcal{M}}, A_{\mathcal{M}}, T_{\mathcal{M}}, L_{\mathcal{M}})$ *where*

- $S_{\mathcal{M}}$ *is a set of* states*;*
- $A_{\mathcal{M}}$ *is a set of* actions*;*
- $T_{\mathcal{M}} : S_{\mathcal{M}} \times A_{\mathcal{M}} \to S_{\mathcal{M}}$ *is a partial transition function;*
- $L_{\mathcal{M}} : S_{\mathcal{M}} \to 2^{AP}$ *is a labeling function determining the truth value of the atomic propositions on the states.*

The subscript $\cdot_{\mathcal{M}}$ *is omitted when it is clear from the context.*

A (open-loop) planning **strategy** $\text{str} : \mathbb{N} \to A$ on a DTS $\mathcal{M}$ is given by an infinite sequence of action; clearly, for a finite time horizon planning problem, only a finite prefix of str takes effect. Given an initial state $s_0 \in S$ of the DTS, under the strategy str, a **path** $\pi : \mathbb{N} \to S$ can be generated, if $\pi(t+1) = T(\pi(t), \text{str}(t))$ for all $t \in \mathbb{N}$. The planning task is then finding a path π and corresponding strategy str such that objective φ is satisfied.

DTS Augmentation: The DTS $\mathcal{M}$ from Definition 1 does not allow direct reference to the actions $A_{\mathcal{M}}$ using the atomic propositions, which are only associated to the states by $L_{\mathcal{M}}$. To formalize our discussion, we introduce a mapping from $\mathcal{M}$ to an *augmented DTS* $\mathcal{A}$ by encoding into states the actions taken previously.[2] The procedure is similar to the conversion from Moore machines into finite state automata.

Definition 2 (Augmented DTS). *DTS* $\mathcal{A} = (S_{\mathcal{A}}, A_{\mathcal{A}}, T_{\mathcal{A}}, L_{\mathcal{A}})$ *is an augmentation to a DTS* $\mathcal{M} = (S_{\mathcal{M}}, A_{\mathcal{M}}, T_{\mathcal{M}}, L_{\mathcal{M}})$, *if*

- $A_{\mathcal{A}} = A_{\mathcal{M}}$ *and* $\{\varepsilon\} \times S_{\mathcal{M}} \subseteq S_{\mathcal{A}} \subseteq (A_{\mathcal{M}} \cup \{\varepsilon\}) \times S_{\mathcal{M}}$, *where ε is the empty sequence;*
- $(a_{\mathcal{M}}, s_{\mathcal{M}}) \in S_{\mathcal{A}}$ *iff there exists* $s'_{\mathcal{M}} \in S_{\mathcal{M}}$ *such that* $T_{\mathcal{M}}(s'_{\mathcal{M}}, a_{\mathcal{M}}) = s_{\mathcal{M}}$;
- *for any* $a_{\mathcal{M}} \in A_{\mathcal{M}}$, $((\cdot, s'_{\mathcal{M}}), a_{\mathcal{M}}, (a_{\mathcal{M}}, s_{\mathcal{M}})) \in T_{\mathcal{A}}$ *iff* $T_{\mathcal{M}}(s'_{\mathcal{M}}, a_{\mathcal{M}}) = s_{\mathcal{M}}$;
- *for any* $s_{\mathcal{A}} = (a_{\mathcal{M}}, s_{\mathcal{M}}) \in S_{\mathcal{A}}$, $L_{\mathcal{A}}(s_{\mathcal{A}}) = L_{\mathcal{M}}(s_{\mathcal{M}}) \cup \{a_{\mathcal{M}}, s_{\mathcal{M}}\}$;

For example, the DTS $\mathcal{M}$ in Figure 2 is augmented to the DTS $\mathcal{A}$ in Figure 4, where the actions L and R represent moving left and right. Following Definition 2, there is a correspondence between the paths of a DTS $\mathcal{M}$ and its augmented DTS $\mathcal{A}$, as formalized below.

Lemma 1. *Let* $\pi_{\mathcal{M}} = s_{\mathcal{M}}(0)s_{\mathcal{M}}(1)\ldots$ *be a path of a DTS* $\mathcal{M}$ *under the strategy* $\text{str} = a_{\mathcal{M}}(0)a_{\mathcal{M}}(1)\ldots$. *Then* $\pi_{\mathcal{A}} = (\varepsilon, s_{\mathcal{M}}(0))(a_{\mathcal{M}}(0), s_{\mathcal{M}}(1))\ldots \subseteq S_{\mathcal{A}}$ *is a path of the equivalent augmented DTS* $\mathcal{A}$ *of* $\mathcal{M}$; *and* vice versa.

From Definition 2, the states and actions of the initial DTS $\mathcal{M}$ are included in the labels of the augmented DTS $\mathcal{A}$. Thus, planning on a DTS $\mathcal{M}$ with objectives specified over both its states and actions, can be mapped into planning on the

[2]Encoding the next action could incur unnecessary nondeterminism.

augmented DTS $\mathcal{A}$ with objectives specified only over states, which can then be formally defined through the labels.

For quantifier-alternation-free hyper temporal objectives, a strategy can be synthesized by feeding the augmented DTS and the objective solely over states to an automata-based model checker (e.g., SPIN [35], PRISM [36]) with only a moderate modification (see [16] for details). For hyper objectives with alternating quantifiers, automata-based model checking is computational challenging. Hence, in this work, we use a symbolic model checking approach for the synthesis.

V. Planning from HyperLTL Specifications

To formally express hyperproperties-based planning objectives, in this section, we describe HyperLTL [16], which can be viewed as an extension of LTL to multiple paths. We then show how HyperLTL can be used for planning with hyper-objectives, such as optimality, robustness and privacy.

A. HyperLTL Syntax

HyperLTL enables reasoning about the interrelation of multiple paths by introducing a set of path variables $\Pi = \{\pi_1, \pi_2, \ldots\}$, where each path variable represents an individual path. The atomic propositions are of the form a^{π}, where the meaning of $a \in AP$ is similar to LTL, and the superscript π indicates that a should be checked on the path π. These atomic propositions a^{π} are concatenated by logic operators (e.g., $\neg$, $\wedge$, $\bigcirc$ and $\mathcal{U}_T$) as in LTL. Finally, all the path variables in the objectives are quantified by $\exists$ or $\forall$. Formally, HyperLTL objectives are defined inductively by the syntax:

$$\psi ::= \forall \pi.\ \psi \mid \exists \pi.\ \psi \mid \varphi \tag{4}$$

$$\varphi ::= a^{\pi} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc \varphi \mid \varphi\, \mathcal{U}_T\, \varphi \tag{5}$$

where $T \in \mathbb{N}_\infty$, $a \in AP$, and $\pi \in \Pi$. Other common logic operators are derived in the same way as in Section II.

B. HyperLTL Semantics

As a HyperLTL objective may contain multiple path variables, its satisfaction involves assigning concrete (infinite) paths to all these path variables. Therefore, we define $V : \Pi \to (2^{AP})^\omega$ as an *assignment* for all possible path variables. The satisfaction relation for the HyperLTL path formulas is then defined for V by:

$$
\begin{aligned}
V \models a^{\pi} &\Leftrightarrow a \in V(\pi)(0) \\
V \models \neg\varphi &\Leftrightarrow V \not\models \varphi \\
V \models \varphi_1 \wedge \varphi_2 &\Leftrightarrow V \models \varphi_1 \text{ and } V \models \varphi_2 \\
V \models \bigcirc\varphi &\Leftrightarrow V^{(1)} \models \varphi \\
V \models \varphi_1\, \mathcal{U}_T\, \varphi_2 &\Leftrightarrow \exists t \le T.\ \big(V^{(t)} \models \varphi_2 \text{ and} \\
&\qquad (\forall t' < t.\ V^{(t')} \models \varphi_1)\big) \\
V \models \exists \pi.\ \psi &\Leftrightarrow \text{there exists } \sigma \in (2^{AP})^\omega, \\
&\qquad \text{such that } V[\pi \mapsto \sigma] \models \psi \\
V \models \forall \pi.\ \psi &\Leftrightarrow \text{for all } \sigma \in (2^{AP})^\omega, \\
&\qquad V[\pi \mapsto \sigma] \models \psi \text{ holds}
\end{aligned}
$$

where $T \in \mathbb{N}_\infty$ is a time horizon, and $V^{(t)}$ is the t-shift of the assignment V, defined by $\big(V^{(t)}(\pi)\big) = (V(\pi))^{(t)}$ for all path variables $\pi \in \Pi$. Other logic operators, like $\vee$, $\Rightarrow$, $\square_T$, $\diamondsuit_T$, $\mathcal{U}$, $\diamondsuit$ and $\square$ are defined as for the LTL in Section II.
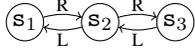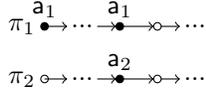
Fig. 2. Example DTS

Fig. 3. Illustration for semantics of $\mathsf{a}_1^{\pi_1} \mathcal{U} \mathsf{a}_2^{\pi_2}$.
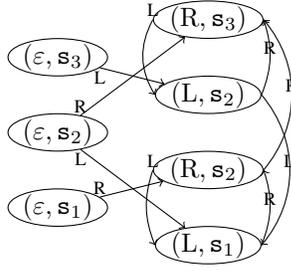

Fig. 4. Augmented DTS
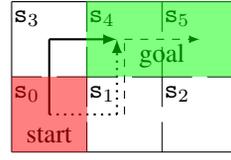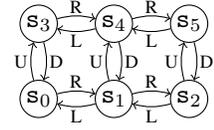

Fig. 5. Example workspace

Fig. 6. DTS model

HyperLTL subsumes LTL: Any LTL objective can be expressed in HyperLTL. For example, $(\mathcal{A}, \mathsf{s}_0) \models \varphi$ for an LTL objective φ on a augmented DTS $\mathcal{A}$ with the initial state s_0, is expressed in HyperLTL by $V \models \varphi^\pi$, where $V(\pi)$ gives the path starting from s_0 of $\mathcal{A}$, and φ^π means adding superscript π to all atomic propositions in π.

HyperLTL is strictly more expressive than LTL: Although the meaning of the logic operators in HyperLTL are similar to those in LTL, the "until" $\mathcal{U}$ (and the "bounded until" $\mathcal{U}_T$) in HyperLTL can be used between different paths. For example, HyperLTL allows $\mathsf{a}_1^{\pi_1} \mathcal{U} \mathsf{a}_2^{\pi_2}$, meaning a_1 should hold on π_1 until a_2 should hold on π_2. The satisfaction of $\mathsf{a}_1^{\pi_1} \mathcal{U} \mathsf{a}_2^{\pi_2}$ for the two paths π_1 and π_2 is illustrated in Figure 3.

Also, HyperLTL allows alternating path quantifiers, like $\exists \pi_1 \forall \pi_2. \ \mathsf{a}_1^{\pi_1} \ \mathcal{U} \ \mathsf{a}_2^{\pi_2}$, which means that we look for a path π_1 such that for any path π_2 (possibly different from π_1), the objective $\mathsf{a}_1^{\pi_1} \ \mathcal{U} \ \mathsf{a}_2^{\pi_2}$ should be satisfied. These "until among multiple paths" and "exists such that for all" cannot be expressed by LTL; thus, HyperLTL strictly subsumes LTL.

C. HyperLTLf

Most planning problems focus on finding a finite path, while the semantics of HyperLTL is defined for infinite paths. We note that the semantics of HyperLTL can be adapted to finite paths to derive *HyperLTLf* in the same way as adapting LTL to LTLf [7]. Generally, HyperLTLf can be viewed as the finite-time fragment of HyperLTL. Specifically, each finite path π can be converted into an infinite path π' by repeating the last entry. We note that the semantics of HyperLTLf on the finite paths π agrees with the semantics of HyperLTL on the infinite paths π'. In Section VI, we introduce a symbolic synthesis method for handling the HyperLTLf objectives.

D. Applications of HyperLTL for Planning

We now show how HyperLTL (and HyperLTLf) can be used to formally express planning objectives with various types of robustness, optimality and privacy properties discussed in Section III. As a running example, consider the inner navigation on a map of 3×2 rooms (Figure 5) where any two adjacent rooms are connected, s_0 is the *start*, and the *goal* is to reach either s_4 or s_5; a feasible path is shown with a thick solid line. The problem can be modeled by the DTS in Figure 6, where each state represents a room and the actions L, R, U and D denote moving left, right, up and down.

1) Optimality: LTL objectives cannot specify optimal strategies, such as shortest or longest paths, as these implicitly involve comparison between multiple paths. For example, a path π_2 reaches a goal set g with the *shortest time*, if it

reaches g before any other path π_1 HyperLTL can specify this property by $\Diamond(g^{\pi_2} \Rightarrow \Diamond g^{\pi_1})$. Thus, the objective of synthesizing a strategy for reaching g from an initial state s_0 **the shortest path** is given by

$$\exists \pi_2 \forall \pi_1. \ \big(\mathsf{s}_0^{\pi_1} \wedge \mathsf{s}_0^{\pi_2}\big) \wedge \big(\Diamond g^{\pi_2}\big) \wedge \big(\Diamond(g^{\pi_2} \Rightarrow \Diamond g^{\pi_1})\big); \quad (6)$$

and the objective for **the longest path** is captured by

$$\exists \pi_2 \forall \pi_1. \ \big(\mathsf{s}_0^{\pi_1} \wedge \mathsf{s}_0^{\pi_2}\big) \wedge \big(\Diamond(g^{\pi_1} \Rightarrow \Diamond g^{\pi_2})\big). \quad (7)$$

Note that assuming the paths are no longer than some $T \in \mathbb{N}_\infty$, specifications (6) and (7) can be equivalently expressed by replacing the "unbounded" $\Diamond$ with the "bounded" $\Diamond_T$. In Figure 5, the path π_2 in solid line satisfies the HyperLTL objective (6) hence, is the shortest path.

2) Robustness: HyperLTL enables capturing requirements for synthesizing a planning strategy that, with its initial objectives achieved, is also robust to various types of uncertainties, and even faults and adversarial factors. Generally, let φ be an LTL objective to be *robustly* satisfied, and $\mathrm{cls}_{\mathsf{s}_0}(\pi_1, \pi_2)$ and $\mathrm{cls}_{\mathcal{A}}(\pi_1, \pi_2)$ be notions of "closeness" of the initial states and actions. Robust planning is defined by an objective that there exists a path π_1 such that, for any other path π_2 close to π_1, the objective φ should still be satisfied:

$$\exists \pi_1 \forall \pi_2. \ \big(\mathrm{cls}_{\mathsf{s}_0}(\pi_1, \pi_2) \wedge \mathrm{cls}_{\mathcal{A}}(\pi_1, \pi_2)\big) \Rightarrow \big(\varphi^{\pi_1} \wedge \varphi^{\pi_2}\big), \quad (8)$$

where φ^π is derived by replacing all the atomic propositions a in φ by a^π. Depending on the different sources of uncertainty, we highlight the following notions of robustness.

- **Initial-state robustness** when the uncertainty comes not fully knowing the initial state – i.e, from replacing a predefined initial state s_0 to an arbitrary state from a set S_0. In this case, we capture the objective of synthesizing an initial-state robust strategy for a time horizon $T \in \mathbb{N}_\infty$ for an LTL objective φ, as the HyperLTL formula

$$\exists \pi_1 \forall \pi_2. \big(\mathsf{s}_0^{\pi_1} \wedge S_0^{\pi_2}\big) \wedge \big(\varphi^{\pi_1} \wedge \varphi^{\pi_2}\big) \wedge \big(\Box_T(\mathsf{a}^{\pi_1} = \mathsf{a}^{\pi_2})\big) \quad (9)$$

Note that in (9) and the formulas below, "=" is not an arithmetic relation, but a notation simplification: $\mathsf{a}^{\pi_1} = \mathsf{a}^{\pi_2}$ stands for $\bigwedge_{\mathsf{a} \in \mathsf{A}}(\mathsf{a}^{\pi_1} \wedge \mathsf{a}^{\pi_2})$.

- **Action robustness** when the uncertainty comes from control faults – e.g., from replacing at most one action with another arbitrary action. Then, in HyperLTL, we capture the objective of synthesizing an action robust strategy for a time horizon $T \in \mathbb{N}_\infty$ for an LTL objective φ as the objective

$$\exists \pi_1 \forall \pi_2. \ \big(\mathsf{s}_0^{\pi_1} \wedge \mathsf{s}_0^{\pi_2}\big) \wedge \big(\varphi^{\pi_1} \wedge \varphi^{\pi_2}\big)$$
$$\wedge \Box_T\big((\mathsf{a}^{\pi_1} \neq \mathsf{a}^{\pi_2}) \Rightarrow \big(\bigcirc \Box_T(\mathsf{a}^{\pi_1} = \mathsf{a}^{\pi_2})\big)\big) \quad (10)$$

Robustness to other types of action uncertainties (disturbances), such as at most N or no N successive replacements, can be similarly expressed in HyperLTL.

In Figure 5, the strategy (first up then right) shown by the solid line is initial-state robust if the initial state is uncertain between $\mathtt{s}_0$ and $\mathtt{s}_1$, because the same strategy generates another feasible path shown in the dashed line. However, the strategy is not action robust as the robot will not reach the goal, if either the first or second action is replaced.

3) Privacy/Opacity: Let $\mathrm{sec}(\cdot)$ be a *secret* and $\mathrm{obs}(\cdot)$ be a (partial) *observation* on a path. A opaque strategy satisfies that there exist at least two paths with the same observation but bearing different secrets, such that the secret of each path cannot be identified exactly only from the observation – i.e.,

$$\exists \pi_1 \exists \pi_2. \big(\mathrm{sec}(\pi_1) \neq \mathrm{sec}(\pi_2)\big) \wedge \big(\mathrm{obs}(\pi_1) = \mathrm{obs}(\pi_2)\big). \quad (11)$$

Depending on the specific forms of the secrets and observations, we consider the following notions of privacy/opacity.

• **Initial-state opacity for fixed strategy** [37]: Let the secret be the initial state of the path and observe whether the robot finally reaches a goal set g. Then, the objective of synthesizing an initial-state opaque strategy from the initial state $\mathtt{s}_0$ for a time horizon $T \in \mathbb{N}_\infty$ can be captured by

$$\begin{aligned}
\exists \pi_1 \exists \pi_2. & \big(\mathtt{s}_0^{\pi_1} \wedge (\neg \mathtt{s}_0^{\pi_2})\big) \\
& \wedge \big(\Box_T(\mathtt{a}^{\pi_1} = \mathtt{a}^{\pi_2})\big) \wedge \big((\Diamond_T g^{\pi_1}) \wedge (\Diamond_T g^{\pi_2})\big).
\end{aligned} \quad (12)$$

• **Current-state opacity** [38]: Let the secret be the synthesized strategy, and the observation be the initial state and whether the path is *currently* in a set o. Then, the HyperLTL objective of synthesizing a current-state opaque strategy from the initial state $\mathtt{s}_0$ for a time horizon $T \in \mathbb{N}_\infty$ is captured as

$$\begin{aligned}
\exists \pi_1 \exists \pi_2. & \big(\mathtt{s}_0^{\pi_1} \wedge \mathtt{s}_0^{\pi_2}\big) \wedge \\
& \wedge \big(\neg \Box_T(\mathtt{a}^{\pi_1} = \mathtt{a}^{\pi_2})\big) \wedge \big(\Box_T(o^{\pi_1} = o^{\pi_2})\big).
\end{aligned} \quad (13)$$

The above formula requires that there are two different paths π_1 and π_2, generated by two different strategies, such that they give the same observations; and any one of these two is a current-state opaque strategy.

In Figure 5, it is easy to check that the strategy shown by the solid line is not initial-state opaque. However, the strategy is current state private because of the existence of the strategy shown by the dotted line.

VI. STRATEGY SYNTHESIS

In this section, we introduce a symbolic approach for synthesizing strategies for HyperLTL objectives. Similarly to existing work on symbolic planning, such as [7], [23], [27] and references therein, we consider HyperLTL objectives with bounded time horizons (effectively, HyperLTLf from Section V-C) via the following three steps. First, we identify a required time horizon for synthesizing a strategy for a considered objective, as introduced in Section VI-A. Then, as presented in Section VI-B, by replacing the $\exists$ and $\forall$ quantifications over paths to that over a finite sequence of states and actions within the required horizon time, we convert the HyperLTL objective and the constraint of the DTS model on its path, into first-order logic formulas. Finally, the conjunction of the two formulas representing the system model and the synthesis objective, is solved using an off-the-shelf SMT solver.

A. Computing Required Time Horizon

A HyperLTL objective contains multiple paths. Therefore, unlike with LTL formulas, the required time horizon may be different among the utilized path variables. Specifically, let $H(\varphi, \pi)$ be the required time horizon for a path variable π in a HyperLTL objective φ. For an atomic proposition $\mathtt{a}^\pi$, the required time horizon is 0, if the path variable π appears in it, and $-\infty$ otherwise – i.e., formally we define

$$H(\mathtt{a}^\pi, \pi') = \begin{cases} 0 & \text{if } \pi' = \pi \\ -\infty & \text{otherwise.} \end{cases}$$

Furthermore, every "next" and "until" temporal operator employed in formula φ changes the required time horizon as captured by the following recursive rules

$$\begin{aligned}
H(\bigcirc \varphi, \pi) &= H(\varphi, \pi) + 1, \\
H(\varphi_1 \, \mathcal{U}_T \, \varphi_2, \pi) &= \max\{H(\varphi_1, \pi), H(\varphi_2, \pi)\} + T.
\end{aligned}$$

Finally, the negation and quantification for path variables do not change the required time horizon – i.e., for any π,

$$\begin{aligned}
H(\neg \varphi, \pi) &= H(\varphi, \pi), \\
H(\varphi_1 \wedge \varphi_2, \pi) &= \max\{H(\varphi_1, \pi), H(\varphi_2, \pi)\}, \\
H(\exists \pi'. \, \psi, \pi) &= H(\psi, \pi), \quad H(\forall \pi'. \, \psi, \pi) = H(\psi, \pi).
\end{aligned}$$

In the above rules, we follow the convention that $x + (-\infty) = -\infty$ and $\max\{x, -\infty\} = x$ for any $x \in \mathbb{N}$. Also, for HyperLTLf objective φ, if a path variable π appears in φ, then its time horizon $H(\varphi, \pi)$ is finite; otherwise, $H(\varphi, \pi) = -\infty$. Finally, as an example, $H(\exists \pi_2 \forall \pi_1. \, (\mathtt{s}_0^{\pi_1} \wedge \mathtt{s}_0^{\pi_2}) \wedge (\Diamond_T(g^{\pi_2} \Rightarrow \Diamond_T g^{\pi_1})), \pi_1) = H(\Diamond_T(g^{\pi_2} \Rightarrow \Diamond_T g^{\pi_1}), \pi_1) = H(\Diamond_T g^{\pi_1}, \pi_1) + T = 2T$.

B. Model Conversion for SMT-based Synthesis

Consider a HyperLTLf objective of the general form $\mathsf{Q}_1 \pi_1 \ldots \mathsf{Q}_n \pi_n. \, \varphi$, where $\mathsf{Q}_i \in \{\exists, \forall\}$ for $i \in [n]$. For each path variable π_i, let its required time horizon in the objective be $H_i = H(\varphi, \pi_i)$, where $H(\cdot, \cdot)$ is computed as described in Section VI-A. Then, the path quantifications over π_i is equivalently represented by its initial state $\mathtt{s}_i(0)$ and its actions $\mathtt{a}_i(0) \ldots \mathtt{a}_i(H_i - 1)$. Since the path should be generated from a DTS $\mathcal{M}$ as introduced in Definition 1, it should satisfy that

$$P_i = \bigwedge_{t \in [H_i]} \big(\mathtt{s}_i(t) = \mathtt{T}_\mathcal{M}(\mathtt{s}_i(t-1), \mathtt{a}_i(t-1))\big), \quad i \in [n], \quad (14)$$

where $\mathtt{s}_i(0) \in \mathtt{S}_\mathcal{M}$ and $\mathtt{a}_i(0), \ldots, \mathtt{a}_i(H_i - 1) \in \mathtt{A}_\mathcal{M}$ are viewed as variables. Equivalently, this constraint (14) can be generated from the augmented DTS $\mathcal{A}$. Finally, for each $i \in [n]$, the path quantification $\mathsf{Q}_i \pi_i$ is equivalently represented by

$$[\mathsf{Q}_i \pi_i] = \mathsf{Q}_i \mathtt{s}_i(0) \mathsf{Q}_i \mathtt{a}_i(0) \ldots \mathsf{Q}_i \mathtt{a}_i(H_1 - 1). \quad (15)$$

Consequently, the planning strategy should satisfy the following first-order logic formula

$$[\mathsf{Q}_1 \pi_1] \ldots [\mathsf{Q}_n \pi_n]. \big(\bigwedge_{i \in [n]} P_i\big) \wedge \varphi, \quad (16)$$

where P_i is introduced in (14) and $[\mathsf{Q}_i \pi_i]$ for $i \in [n]$ is defined in (15). This formula (16) can be directly solved by SMT solvers, such as Z3, Yices [25] or CVC4 [26].

VII. Implementation and Case Studies

We implemented the described symbolic synthesis method. Specifically, the conversion from the DTS and HyperLTL objectives to first-order logic expressions is implemented in Python. Then, the expressions of the form (16) are solved by Z3 SMT solver [24]. The source code is available at [39].

The implemented method is evaluated on several planning problems of a mobile robot on grid worlds with obstacles, as illustrated in Figures 7-11, where the black, white, red, and green colors stand for obstacles, allowable states, start states and goal states, respectively. At each step, the robot can move up, down, left or right; upon hitting an obstacle, the objective immediately fails. We focused on the HyperLTL objectives discussed in Section V-D for a finite horizon $T \in \mathbb{N}$.

The feasible paths on 10×10 grids for several objectives are illustrated in Figures 7 to 11. For the **privacy/opacity objectives** from (12) and (13), let the partial observation be the row number of the current state of the robot, depicted by the color gradient from bottom to top. In Figure 7, the synthesized blue path is initial-state opaque, as there exists the red path that follows the same strategy, yields the same observation along the path, and reaches the goal, but starts from a different state. In Figure 8, the synthesized blue path is current-state opaque, as there exists the red path that starts from the same state, yields the same observation along the path, and reaches the goal, but follows a different path. It is worth noting that to achieve current-state opacity, the synthesized blue path actually waits at the bottom row for two steps to ensure another indistinguishable path can catch up.

For the **robustness objectives** from (9) and (10), the synthesized blue path in Figure 9 is initial-state robust, meaning the corresponding strategy is feasible for any initial state in the red region. The synthesized blue path in Figure 10 is action robust, as defined in (10) – i.e., the corresponding strategy is feasible for any single action replacement. In both cases, the blue path avoids getting close to the obstacles, in case for initial state inaccuracy or action errors. Finally, for the **optimality objective** from (6), the synthesized blue path in Figure 11 shows the shortest path from a red state to the green state (another shortest path is the dotted green path).

Table I presents the times for synthesizing strategies for the above cases, as well as problems on larger grids; all computations were done on an Intel i7-7820X CPU @3.60GHz and RAM 32GB (only one core was used). The time horizon T was chosen such that the goal is reachable to prevent easy fails. As shown in the table, the strategy synthesis of the HyperLTL objectives can be performed in a reasonable amount of time even on nontrivial problems. As expected, there is an increase in synthesis times as the grid size and time horizon increase, since in the worst case, the size of the first-order logic formula (16) to be evaluated (i.e., solved) by Z3, can grow exponentially with the time horizon and the number of states.

VIII. Conclusion

In this work, we proposed the use of HyperLTL in planning for specifying objectives involving the interrelation of multiple paths, such as optimality, robustness and privacy/opacity,

Fig. 7. Strategy for initial-state opacity (12) for time horizon $T = 20$; the synthesis time is < 0.14s.

Fig. 8. Strategy for current-state opacity (13) for time horizon $T = 20$; the synthesis time is < 0.12s.

Fig. 9. Strategy for initial-state robustness (9) for time horizon $T = 20$; the synthesis time is < 0.24s.

Fig. 10. Strategy for action robustness (10) for time horizon $T = 20$; the synthesis time is < 0.15s.

Fig. 11. Synthesized strategy (in blue) for shortest path (6) for time horizon $T = 20$; the synthesis time is < 0.15s.

TABLE I

Symbolic synthesis times for HyperLTL objectives on grid worlds with obstacles. ISO, CSO, ISR, AR and S stands for initial-state opacity, current-state opacity, initial-state robustness, action robustness and shortest path, respectively.

Grid	Obj.	T	Time (s)	Grid	Obj.	T	Time (s)
10^2	ISO	20	0.14	20^2	ISO	40	5.2
10^2	CSO	20	0.12	20^2	CSO	40	2.9
10^2	ISR	20	0.24	20^2	ISR	40	4.7
10^2	AR	20	0.15	20^2	AR	40	5.5
10^2	SP	20	0.15	20^2	SP	40	5.0
40^2	ISO	80	30	60^2	ISO	120	382
40^2	CSO	80	24	60^2	CSO	120	191
40^2	ISR	80	49	60^2	ISR	120	320
40^2	AR	80	38	60^2	AR	120	306
40^2	SP	80	172	60^2	SP	120	244

which cannot be expressed by the widely used temporal logics, such as linear temporal logic (LTL). We showed how those hyperproperties can be expressed by HyperLTL, which is an extension of LTL to multiple paths. Then, we introduced a method for symbolic synthesis of high-level planning strategies from such HyperLTL objectives, using off-the-shelf tools. Finally, we evaluated the proposed method on several planning case studies.

REFERENCES

[1] G. Fainekos, H. Kress-Gazit, and G. Pappas, "Temporal Logic Motion Planning for Mobile Robots," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 2020–2025.

[2] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI Communications*, vol. 29, no. 1, pp. 151–162, 2016.

[3] M. Elfar, Y. Wang, and M. Pajic, "Security-aware synthesis using delayed-action games," in *Computer Aided Verification (CAV)*. Springer International Publishing, 2019, pp. 180–199.

[4] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-Logic-Based Reactive Mission and Motion Planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[5] S. Moarref and H. Kress-Gazit, "Reactive Synthesis for Robotic Swarms," in *Formal Modeling and Analysis of Timed Systems*. Springer International Publishing, 2018, vol. 11022, pp. 71–87.

[6] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Where's Waldo? Sensor-Based Temporal Logic Motion Planning," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3116–3121.

[7] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Reactive synthesis for finite tasks under resource constraints," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5326–5332.

[8] Y. Kantaros and M. M. Zavlanos, "Global Planning for Multi-Robot Communication Networks in Complex Environments," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1045–1061, 2016.

[9] Y. Kantaros, M. Guo, and M. M. Zavlanos, "Temporal Logic Task Planning and Intermittent Connectivity Control of Mobile Robot Networks," *IEEE Transactions on Automatic Control*, pp. 1–1, 2019.

[10] Y. Kantaros and M. M. Zavlanos, "STyLuS: A Temporal Logic Optimal Control Synthesis Algorithm for Large-Scale Multi-Robot Systems," *arXiv:1809.08345 [cs]*, 2018.

[11] S. S. Farahani, V. Raman, and R. M. Murray, "Robust Model Predictive Control for Signal Temporal Logic Synthesis," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 323–328, Jan. 2015.

[12] L. Lindemann and D. V. Dimarogonas, "Robust motion planning employing signal temporal logic," in *2017 American Control Conference (ACC)*, May 2017, pp. 2950–2955.

[13] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimal control of non-deterministic systems for a computationally efficient fragment of temporal logic," in *52nd IEEE Conference on Decision and Control*. Firenze: IEEE, Dec. 2013, pp. 3197–3204.

[14] G. Jing, R. Ehlers, and H. Kress-Gazit, "Shortcut through an evil door: Optimality of correct-by-construction controllers in adversarial environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 4796–4802.

[15] M. R. Clarkson and F. B. Schneider, "Hyperproperties," in *2008 21st IEEE Computer Security Foundations Symposium*. IEEE, 2008, pp. 51–65.

[16] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez, "Temporal Logics for Hyperproperties," in *Principles of Security and Trust*. Springer Berlin Heidelberg, 2014, vol. 8414, pp. 265–284.

[17] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[18] Y. Lu, Y. Guan, X. Li, R. Wang, and J. Zhang, "A framework of model checking guided test vector generation for the 6DOF manipulator," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4262–4267.

[19] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Multi-objective search for optimal multi-robot planning with finite LTL specifications and resource constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 768–774.

[20] O. Kupferman, "Automata Theory and Model Checking," in *Handbook of Model Checking*. Springer International Publishing, 2018, pp. 107–151.

[21] B. Finkbeiner, M. N. Rabe, and C. Sánchez, "Algorithms for Model Checking HyperLTL and HyperCTL*," in *Computer Aided Verification*. Springer International Publishing, 2015, pp. 30–48.

[22] A. Biere and D. Kröning, "SAT-Based Model Checking," in *Handbook of Model Checking*. Springer International Publishing, 2018, pp. 277–303.

[23] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu, "Bounded Model Checking," in *Advances in Computers*. Elsevier, 2003, vol. 58, pp. 117–148.

[24] L. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer Berlin Heidelberg, 2008, pp. 337–340.

[25] B. Dutertre and L. M. de Moura, "The YICES SMT Solver," 2006.

[26] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli, "CVC4," in *Computer Aided Verification*. Springer Berlin Heidelberg, 2011, pp. 171–177.

[27] K. He, A. M. Wells, L. E. Kavraki, and M. Y. Vardi, "Efficient Symbolic Reactive Synthesis for Finite-Horizon Tasks," p. 7, 2019.

[28] Z. Huang, Y. Wang, S. Mitra, and G. E. Dullerud, "Controller Synthesis for Linear Dynamical Systems with Adversaries," in *3rd ACM Symposium and Bootcamp on the Science of Security (HoTSoS)*. ACM, 2016, pp. 53–62.

[29] Z. Huang, Y. Wang, S. Mitra, G. E. Dullerud, and S. Chaudhuri, "Controller Synthesis with Inductive Proofs for Piecewise Linear Systems: An SMT-Based Algorithm," in *54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 7434–7439.

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

[31] B. Gedik and Ling Liu, "Location Privacy in Mobile Systems: A Personalized Anonymization Model," in *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, 2005, pp. 620–629.

[32] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, "Distributed trajectory estimation with privacy and communication constraints: A two-stage distributed Gauss-Seidel approach," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5261–5268.

[33] L. Li, A. Bayuelo, L. Bobadilla, T. Alam, and D. A. Shell, "Coordinated multi-robot planning while preserving individual privacy," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2188–2194.

[34] D. Dams and O. Grumberg, "Abstraction and Abstraction Refinement," in *Handbook of Model Checking*. Springer International Publishing, 2018, pp. 385–419.

[35] G. J. Holzmann, *The Spin Model Checker: Primer and Reference Manual*, 4th ed. Addison-Wesley, 2008.

[36] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of Probabilistic Real-Time Systems," in *Computer Aided Verification*. Springer Berlin Heidelberg, 2011, vol. 6806, pp. 585–591.

[37] A. Saboori and C. N. Hadjicostis, "Verification of initial-state opacity in security applications of discrete event systems," *Information Sciences*, vol. 246, pp. 115–132, 2013.

[38] X. Yin and S. Lafortune, "A new approach for synthesizing opacity-enforcing supervisors for partially-observed discrete-event systems." IEEE, 2015, pp. 377–383.

[39] CPSL@Duke, "Motion Planning using HyperProperties," 2019. [Online]. Available: https://gitlab.oit.duke.edu/cpsl/mp_hyper