Hyperproperties for Robotics: Motion Planning via HyperLTL

Yu Wang, Siddhartha Nalluri, and Miroslav Pajic

***Abstract*— There is a growing interest on formal methods-based robotic motion 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 motion 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 robotic path/motion planning problems from temporal logic objectives (e.g., [1], [2]). Using temporal logics, such as the linear temporal logic (LTL) [3]–[5], a wide class of objectives beyond reachability can be defined, such as infinite recurrence, complex dependency of many tasks [6], and time-dependent formations of multiple robotics [7]–[9].

However, temporal logics commonly used in robotics (e.g., for motion planning) can only specify properties for individual executions (i.e., paths). This effectively prevents them from capturing important motion planning objectives, such as optimality, robustness, and privacy/opacity, that involve interrelations between multiple paths. For example, for a derived plan to be optimal from the mission-time perspective, the synthesis objective should ask for an existence of a path π such that all other paths π' make no better than π.

Such objectives specifying the interrelations of multiple paths are called *hyperproperties* [10]. To formally reason about hyperproperties over time, hyper-temporal logics, such as HyperLTL [11], are needed. Specifically, HyperLTL adds to 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 "exists" $\exists$ and "for all" $\forall$ quantifications of the path variables, which enables specifying relevant planning objectives such as the described motion planning optimality requirement that employs $\exists\pi\forall\pi'$ quantifiers.

In this work, we show the effectiveness of using hyper temporal logics for synthesizing motion planning strategies with strong performance (e.g., optimality), robustness, and privacy guarantees. Specifically, we study the motion planning from HyperLTL objectives on a commonly used modeling formalism – discrete transition system (DTS), where the states are discrete and the transitions are driven by actions. This model be viewed as the high-level discrete abstraction of the full workspace [7], [8], that is obtained by the low-level explorations, like RRT or probabilistic roadmaps [12], 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], [13]–[15]). Specifically, the objective is first converted to an automaton, and then the strategy-search is done on the intersection 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 the 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 formally introduce in Eq. (2)), the derived automata is to be intersected with the n-fold self-product of the model. As a result, in the formal methods community, automata-theoretic model checking of HyperLTL is mainly confined to quantifier-alternation-free objectives [16].

Consequently, to mitigate the state explosion for HyperLTL objectives in robotic motion planning, in this work, we adopt a symbolic approach for synthesizing strategies via SMT solvers [17], [18]. Specifically, the dynamics of the DTS model is converted into a set of logic formulas, and feasible strategies should satisfy the conjunction the HyperLTL objectives and the 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 [19], Yices [20] or CVC4 [21]. As with previous work on symbolic synthesis from regular LTL (e.g., [6], [18], [22], [23]), 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 regular motion planning models, reducing synthesis complexity by avoiding constructing the n-fold self product [6], [22]; it can even handle motion planning on DTS with infinite states [23], [24]. We show by case studies that our symbolic synthesis approach effectively handles motion planning problems with hyper temporal logic objectives, deriving strategies with strong optimality, robustness, and privacy/opacity guarantees.

This paper is organized as follows. After preliminaries in

This work is sponsored in part by the ONR under agreements N00014-17-1-2012 and N00014-17-1-2504, AFOSR under the award number FA9550-19-1-0169, as well as the NSF CNS-1652544 grant.

Yu Wang and Miroslav Pajic are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA, {yu.wang094, miroslav.pajic}@duke.edu. Siddhartha Nalluri is with the Computer Science Department, Duke University, Durham, NC 27708, USA, siddhartha.nalluri@duke.edu.

Section II, we show the need for the use of hyperproperties in motion planning (Section III). We then formulate our motion planning problem on discrete transition systems (Section IV), before showing how HyperLTL objectives should be employed to ensure optimality, robustness, and privacy/opacity of derived motion plans (Section V). We propose a symbolic synthesis method for the HyperLTL objectives on the discrete transition systems in Section VI, and evaluate the proposed method on several case studies in Section VII. Finally, we conclude this work in Section VIII.

II. Preliminaries

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 $[n] = \{1, \dots, n\}$. 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)

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

where $T \in \mathbb{N}_\infty$ and $\mathsf{a} \in \mathsf{AP}$. For a planned path $\pi : \mathbb{N} \to 2^{\mathsf{AP}}$, 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 using (the semantics)

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

where $\pi^{(t)}(\cdot) = \pi(\cdot + t)$ is the t-time shift. Roughly, $\bigcirc\mathsf{a}$ means a holds *next*, and $\mathsf{a}_1 \,\mathcal{U}_T\, \mathsf{a}_2$ means a_1 holds *until* a_1 holds before T. Other common logic operators are derived by

$$\text{Or: } \varphi \vee \varphi' \equiv \neg(\neg\varphi \wedge \neg\varphi'), \quad \text{Implies: } \varphi \Rightarrow \varphi' \equiv \neg\varphi \vee \varphi'$$
$$\text{Finally: } \Diamond_T\varphi \equiv \mathtt{T}\ \mathcal{U}_T\ \varphi, \quad \text{Always: } \Box_T\varphi \equiv \neg\Diamond_T\neg\varphi.$$

We also denote $\mathcal{U}_\infty$, $\Diamond_\infty$, $\Box_\infty$ by $\mathcal{U}$, $\Diamond$, $\Box$, respectively.

Using LTL, we can express a large class of planning objectives. For example, reaching point a_1 via reaching waypoint a_2 can be expressed in LTL by $\Diamond(\mathsf{a}_1 \wedge \Diamond\mathsf{a}_2)$. The navigation task of always returning to the point a_1 within time T after leaving it, can be expressed as $\Box(\mathsf{a}_1 \wedge \Diamond_T\mathsf{a}_1)$.

III. HyperLogics for Robotics Motion Planing

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

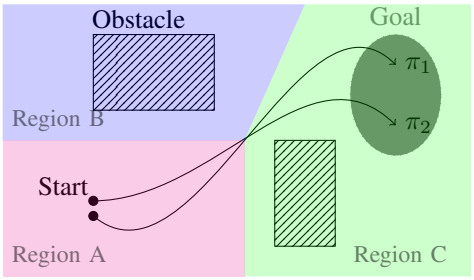

Fig. 1. The use of hyperproperties in motion planning.

Optimality of Synthesized Plans: A well-known shortcoming of LTL-based motion 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 reasoning about 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

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

Robustness of Synthesized Plans: An major concern for open-loop motion planning is robustness of the derived strategy. Specifically, (i) the assumed initial position may be inaccurate; (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 is specified by

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

Motion Planning with Privacy/Opacity: A problem that has recently attracted significant attentions (e.g., [25]–[27]) 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 motion planning strategy to reach the goal implies finding a path π (or equivalently π') such that

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

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

In this work, we consider the motion 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 [28] or random exploration [12]. On the domain, the robot motion is modeled by a discrete transition system (DTS), whose transitions are labeled by actions.

Definition 1 (DTS)**.** *Given a set of atomic propositions* AP *a DTS is a tuple* $\mathcal{M} = (\mathtt{S}_\mathcal{M}, \mathtt{A}_\mathcal{M}, \mathtt{T}_\mathcal{M}, \mathtt{L}_\mathcal{M})$ *where*

- $\mathtt{S}_\mathcal{M}$ *is a set of* states*;*
- $\mathtt{A}_\mathcal{M}$ *is a set of* actions*;*
- $\mathtt{T}_\mathcal{M} : \mathtt{S}_\mathcal{M} \times \mathtt{A}_\mathcal{M} \to \mathtt{S}_\mathcal{M}$ *is a partial* transition function*;*
- $\mathtt{L}_\mathcal{M} : \mathtt{S}_\mathcal{M} \to 2^{\mathsf{AP}}$ *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** $\mathrm{str} : \mathbb{N} \to \mathtt{A}$ on the DTS $\mathtt{s}$ 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 $\mathtt{s}_0 \in \mathtt{S}$ of the DTS, under the strategy str, a **path** $\pi : \mathbb{N} \to \mathtt{S}$ can be generated, if $\pi(t+1) = \mathtt{T}(\pi(t), \mathrm{str}(t))$ 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 $\mathcal{M}$ introduced in Definition 1 does not directly allow for reasoning over actions $\mathtt{A}_\mathcal{M}$ using the atomic propositions, which are only associated to states by $\mathtt{L}_\mathcal{M}$. To formalize our discussion (especially in Section V-C), we introduce a mapping from $\mathcal{M}$ to an *augmented DTS* $\mathcal{A}$ by encoding into states the actions taken previously.[1] The procedure is similar to the conversion from Moore machines to finite state automata.

Definition 2 (Augmented DTS)**.** *The DTS* $\mathcal{A} = (\mathtt{S}_\mathcal{A}, \mathtt{A}_\mathcal{A}, \mathtt{T}_\mathcal{A}, \mathtt{L}_\mathcal{A})$ *is an augmentation to the DTS* $\mathcal{M} = (\mathtt{S}_\mathcal{M}, \mathtt{A}_\mathcal{M}, \mathtt{T}_\mathcal{M}, \mathtt{L}_\mathcal{M})$*, if*

- $\mathtt{A}_\mathcal{M} = \mathtt{A}_\mathcal{A}$ *and* $(\varepsilon, \mathtt{s}_\mathcal{M}) \subseteq \mathtt{S}_\mathcal{A} \subseteq (\mathtt{A}_\mathcal{A} \cup \{\varepsilon\}) \times \mathtt{S}_\mathcal{A}$*, where* ε *stands for the empty sequence;*
- $(\mathtt{a}_\mathcal{M}, \mathtt{s}_\mathcal{M}) \in \mathtt{S}_\mathcal{A}$ *iff there exists* $\mathtt{s}'_\mathcal{M} \in \mathtt{S}_\mathcal{M}$ *such that* $\mathtt{T}_\mathcal{M}(\mathtt{s}'_\mathcal{M}, \mathtt{a}_\mathcal{M}) = \mathtt{s}_\mathcal{M}$*;*
- *for any* $\mathtt{a}_\mathcal{M} \in \mathtt{A}_\mathcal{M}$*,* $\big((\cdot, \mathtt{s}'_\mathcal{M}), \mathtt{a}_\mathcal{M}, (\mathtt{a}_\mathcal{M}, \mathtt{s}_\mathcal{M})\big) \in \mathtt{T}_\mathcal{A}$ *iff* $\mathtt{T}_\mathcal{M}(\mathtt{s}'_\mathcal{M}, \mathtt{a}_\mathcal{M}) = \mathtt{s}_\mathcal{M}$*;*
- *for any* $\mathtt{s}_\mathcal{A} = (\cdot, \mathtt{s}_\mathcal{M}) \in \mathtt{S}_\mathcal{A}$*,* $\mathtt{L}_\mathcal{A}(\mathtt{s}_\mathcal{A}) = \mathtt{L}_\mathcal{M}(\mathtt{s}_\mathcal{M})$*;*

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* $\pi_\mathcal{M} = \mathtt{s}_\mathcal{M}(0)\mathtt{s}_\mathcal{M}(1)\ldots \subseteq \mathtt{S}_\mathcal{M}$ *be a path of a DTS* $\mathcal{M}$ *under the strategy* $\mathrm{str} = \mathtt{a}(0)\mathtt{a}(1)\ldots \subseteq \mathtt{A}_\mathcal{M}$*. Then* $\pi_\mathcal{A} = (\varepsilon, \mathtt{s}_\mathcal{M}(0))(\mathtt{a}_\mathcal{M}(0), \mathtt{s}_\mathcal{M}(1))\ldots \subseteq \mathtt{S}_\mathcal{A}$ *is a path of the equivalent augmented DTS* $\mathcal{A}$ *of* $\mathcal{M}$*; and* vice versa*.*

To simplify our presentation, we include the states and actions of the initial DTS $\mathcal{M}$ as labels of the augmented DTS $\mathcal{A}$. That is, we assume that the set atomic propositions $\mathsf{AP} \supseteq \mathtt{S}_\mathcal{M} \cup \mathtt{A}_\mathcal{M}$, and for any $\mathtt{s}_\mathcal{M} \in \mathtt{S}_\mathcal{M}$ and $\mathtt{a}_\mathcal{M} \in \mathtt{A}_\mathcal{M}$, we have $\mathtt{s}_\mathcal{M} \in \mathtt{L}_\mathcal{A}(\mathtt{s}_\mathcal{A})$ or $\mathtt{a}_\mathcal{M} \in \mathtt{L}_\mathcal{A}(\mathtt{s}_\mathcal{A})$ iff $\mathtt{s}_\mathcal{A} = (\mathtt{a}_\mathcal{M}, \mathtt{s}_\mathcal{M})$. Thus, motion planning on a DTS $\mathcal{M}$ with objectives specified over both its states and actions, can be mapped into planning on the augmented DTS $\mathcal{A}$ with objectives specified only over states, which can then be formally defined through the labels.

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

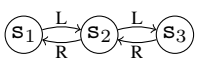

Fig. 2. Example DTS

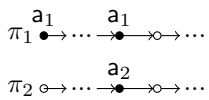

Fig. 3. Illustration for semantics of $\mathtt{a}_1^{\pi_1} \mathcal{U} \mathtt{a}_2^{\pi_2}$.

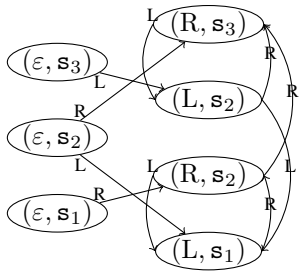

Fig. 4. Augmented DTS

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 [29], or PRISM [30]) with only a moderate modification (see [11] for details). For hyper objectives with alternating quantifiers, automata-based model checking is computational challenging; consequently, in the rest of the paper, we will adopt a symbolic model checking approach for control synthesis.

V. Motion Planning from HyperLTL Specifications

As shown in Section III, many important objectives in motion planning are *hyperproperties* that involve the interrelation of multiple paths. To formally express such objectives, in this section, we describe the logic HyperLTL [11], which can be viewed as an extension of LTL to multiple paths. We then show how HyperLTL can be used for motion planning with hyper-objectives, such as optimality, robustness and privacy.

A. HyperLTL Syntax

HyperLTL allows for reasoning the interrelation of multiple paths by introducing a set of path variables $\Pi = \{\pi_1, \pi_2, \ldots\}$, in which each path variable represents an individual paths. The atomic propositions are of the form $\mathtt{a}^\pi$, in which the meaning of $\mathtt{a} \in \mathsf{AP}$ is similar to LTL, and the superscript π indicates that $\mathtt{a}$ should be checked on the path π. These atomic propositions $\mathtt{a}^\pi$ 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:

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

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

where $T \in \mathbb{N}_\infty$, $\mathtt{a} \in \mathsf{AP}$, 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 paths to all these path variables. Therefore, we define $V : \Pi \to (2^{\mathsf{AP}})^\omega$ as

an *assignment* for all possible path variables. The satisfaction relation for the HyperLTL path formulas is defined for V by:

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

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 $\vee$, $\Rightarrow$, $\Box_T$, $\Diamond_T$, $\mathcal{U}$, $\Diamond$ and $\Box$ are defined as for the LTL in Section II.

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.

HyperLTLf: We derive the finite-time fragment of HyperLTL, referred to as HyperLTLf, by prohibiting the "unbounded until" $\mathcal{U}_{\infty}$ from the syntax in Section V-A. From the semantics in Section V-B, the satisfaction of HyperLTLf objectives can always be determined by a finite prefix of a path. In Section VI, we introduce a symbolic synthesis method for handling the HyperLTLf objectives.

C. Applications of HyperLTL for Motion Planning

We now show how HyperLTL (and HyperLTLf) can be used to formally express motion 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_1 is the *start*, and the *goal* is to reach either s_5 or s_6; 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 they implicitly involve comparison between multiple paths. For example,

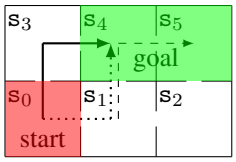

Fig. 5. Example workspace

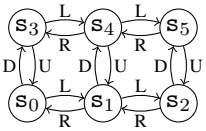

Fig. 6. DTS model

a path π_1 reaches a goal set g with the *shortest time*, if it reaches g before any other path π_2 HyperLTL can specify this property by $\Diamond_T(g^{\pi_2} \Rightarrow \Diamond_T g^{\pi_1})$. Thus, the objective of synthesizing a strategy for reaching g from an initial state s_0 **with the least steps** for a time horizon $T \in \mathbb{N}_{\infty}$ is given by

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

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

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

In Figure 5, it is easy to check that the strategy shown by the solid line satisfies the HyperLTL objective (6) for any π_2, and therefore, it is the shortest path.

2) Robustness: HyperLTL enables capturing requirements for synthesizing a motion planning strategy, 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 $\mathrm{cls}_{\mathtt{A}}(\pi_1, \pi_2)$ be notions of "closeness" of the initial states and actions. Robust motion 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:

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

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

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

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

• **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

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

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 exists 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** [31]: 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

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

• **Current-state opacity** [32]: 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 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 capture in HyperLTL as

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

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 from HyperLTL objectives. Similarly to existing work on symbolic motion planning, such as [6], [18], [22] and references therein, we focus on HyperLTL objectives with bounded time horizons – i.e., HyperLTLf objectives.

Specifically, our framework for strategy synthesis from HyperLTLf objectives consists of 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',

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

In the above rules, we follow the convention that $x+(-\infty)=x$ 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, $H(\varphi,\pi)=-\infty$.

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 the DTS $\mathcal{M}$ introduced in Definition 1, it should satisfy that

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

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 motion planning strategy should satisfy the following formula

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

where P_i is introduced in (14) and $[\mathsf{Q}_i\pi_i]$ for $i\in[n]$ is defined in (15). Now, the formulas from (16) can be directly used as SMT queries, and solved by SMT solvers, such as Z3, Yices [20] or CVC4 [21].

VII. Implementation and Case Studies

We implemented the descried 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 the SMT solver Z3 [19]. The source code is available at [33].

The implemented method is evaluated on several motion planning problems of a mobile robot on grid worlds with obstacles, as illustrated in Figure 7 to 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-C for a finite horizon $T \in \mathbb{N}$.

The feasible paths on 10×10 grids for several objectives are illustrated in Figure 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 time 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 motion planning for specifying objectives involving the interrelation of multiple paths, such as optimality, robustness and privacy/opacity, which cannot be expressed by usual 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, and evaluated the proposed method on several motion planning case studies.

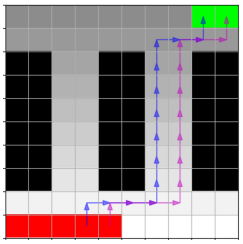

Fig. 7. Strategy for initial-state opacity (12) for time horizon $T = 20$; the synthesis time is < 0.14s.

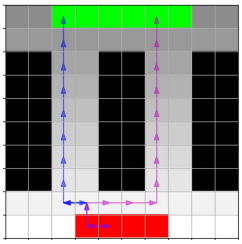

Fig. 8. Strategy for current-state opacity (13) for time horizon $T = 20$; the synthesis time is < 0.12s.

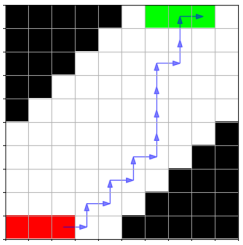

Fig. 9. Strategy for initial-state robustness (9) for time horizon $T = 20$; the synthesis time is < 0.24s.

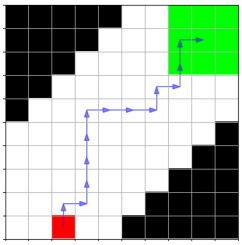

Fig. 10. Strategy for action robustness (10) for time horizon $T = 20$; the synthesis time is < 0.15s.

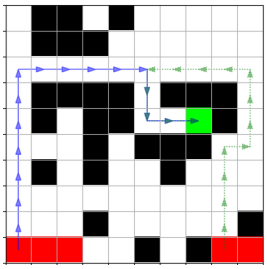

Fig. 11. Synthesized strategy (in blue) for shortest path (6) for time horizon $T = 20$; the synthesis time is < 0.15s.

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

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

[19] 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.

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

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

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

[30] 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.

[31] 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.

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

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