Motion Planning Using Hyperproperties for Time Window Temporal Logic

Ernest Bonnah, Luan Nguyen, and Khaza Anuarul Hoque

Abstract—**Hyperproperties are increasingly popular in verifying security policies and synthesis of control for dynamic systems. Hyperproperties generalize trace properties to enable reasoning about multiple computation traces that traditional trace properties cannot. Recent works show the effectiveness and prospect of Hyperproperties, specifically Hyperproperties for Linear Temporal Logic (HyperLTL), in optimality-, robustness-, and privacy-aware robotic motion planning. However, despite their rich expressiveness, HyperLTL cannot express tasks with time constraints. This letter presents HyperTWTL, which extends the compact semantics of Time Window Temporal Logic (TWTL) with explicit and concurrent quantification over multiple execution traces. We demonstrate that HyperTWTL can be used to formalize complex robotic planning objectives. Given HyperTWTL specifications, we also propose a symbolic approach for synthesizing optimality-, robustness-, and privacy-aware strategies by reducing the planning problem to a first-order logic satisfiability problem. The planning problem was then solved using two industrial-strength SMT solvers. The feasibility of HyperTWTL and the efficiency and scalability of the proposed strategy synthesis approach are demonstrated by formalizing important motion planning objectives of a surveillance mission case study and synthesizing the respective strategies using Z3 and CVC4 SMT solvers.**

Index Terms—**Formal methods in robotics and automation, time window temporal logic, hyperproperties, motion planning.**

I. Introduction

Motion planning and control problems in robotic systems are often formulated as temporal logic specifications over a discrete system representation. Temporal logic such as Linear Temporal Logic (LTL) [1], [2], [3], [4], Metric Temporal Logic (MTL) [5], [6], Signal Temporal Logic (STL) [7], Metric Interval Temporal Logic (MITL) [8], [9], Bounded Linear Temporal Logic (BLTL) [10] and Time Window Temporal Logic (TWTL) [11] have been extensively used to formalize such complex requirements. In recent times, TWTL [12], [13], [14], [15], [16], [17] has gained traction in robotics applications for specifying time-bounded specifications more compactly and comprehensively when compared to other bounded logics such as Metric Temporal Logic (MTL) [18], Signal Temporal Logic (STL) [19], Bounded Linear Temporal Logic (BLTL) [20]. This is because the TWTL uses an explicit concatenation operator, which is very useful in expressing serial tasks in robotic mission specification and planning. For instance, let us consider a specification as "*stay at P for 4-time steps within the time window [0, 6]*". This can be expressed in TWTL as $[\mathbf{H}^4 P]^{0,6}$. The exact same specification can be expressed in STL as $\mathbf{F}_{[0,6-4]}\mathbf{G}_{[0,4]}P$ where the outermost time window needs to be modified with respect to the inner time window.

The majority of the existing works in temporal logic-based motion planning ignore security as a part of the requirements for mission planning. Many security requirements (e.g., information-flows) and serial tasks in robotics and control applications expressed in these logics are complicated and incomprehensible for formal analysis. The TWTL (and other conventional temporal logics) can only express trace properties, i.e., the specified properties involve reasoning about individual executions or traces. This limits their application to many other domains, which requires reasoning about multiple traces. For example, consider a Service Level Agreement (SLA) requirement, which specifies the percentage uptime of a system to accept service requests. Such a property cannot be verified based on individual executions of the system because the satisfiability of such a requirement depends on analyzing the uptime of all system executions. Thus, SLA is not a trace property but a *hyperproperty*. As another example, lets us consider an observational determinism requirement formalized as $\varphi = \forall\pi_1\forall\pi_2 \cdot [\mathbf{H}^5\, I_{\pi_1} = \mathbf{H}^5\, I_{\pi_2}]^{[0,5]} \rightarrow [\mathbf{H}^5\, O_{\pi_1} = \mathbf{H}^5\, I_{\pi_2}]^{[0,5]}$, which specifies that given any pair of traces π_1 and π_2, if the observable inputs I are the same for 5-time units within the bound [0,5], then the observable outputs O should be the same for 5-time units within the same time bound. Similar to SLA, this property cannot be verified based on individual executions of the system because the satisfiability of such a requirement depends on analyzing all system executions. Hence, observational determinism is also not a trace property but a *hyperproperty*.

Hyperproperties [21] generalize trace properties by relating multiple system execution traces to each other. Traditional temporal logic, such as LTL, can express trace properties and reason if they are satisfied by traces. In contrast, *hyperproperties* are satisfied by sets of traces and thus hyperproperties can be more expressive than trace properties. Hyperproperties can be used to specify a wide range of important properties such as information-flow security [22], [23], consistency models in concurrent computing [24], [25], robustness models in cyber-physical systems [26], [27], and also service level agreements (SLA) [21].

Manuscript received 5 January 2023; accepted 9 May 2023. Date of publication 29 May 2023; date of current version 14 June 2023. This letter was recommended for publication by Associate Editor L. Biagiotti and Editor L. Pallottino upon evaluation of the reviewers' comments. *(Corresponding author: Khaza Anuarul Hoque.)*

Ernest Bonnah and Khaza Anuarul Hoque are with the Department of Electrical and Computer Engineering, University of Missouri, Columbia, MO 65211 USA (e-mail: ernestbonnah@gmail.com; hoquek@missouri.edu).

Luan Nguyen is with the Department of Computer Science, University of Dayton, OH 45469 USA (e-mail: lnguyen1@udayton.edu).

Digital Object Identifier 10.1109/LRA.2023.3280830

Motivated by the expressiveness of hyperproperties, several hyper-temporal logics, such as HyperLTL [28], HyperSTL [29], and HyperMTL [30], were recently proposed by extending the conventional temporal logics such as Linear Temporal Logic (LTL) [31], Signal Temporal Logic (STL) [19], and Metric Temporal Logic (MTL) [18], respectively for security/privacy policy verification [29], [32] and control synthesis [24]. Very recently, authors in [33] used HyperLTL for robotic motion planning. However, similar to LTL, HyperLTL also can not express tasks with explicit time constraints, which may limit its application to many applications. For instance, a real-world example can be a robotic inspection mission validating the routines in a chemical plant [34] where robots provide plant operators the information to maximize equipment uptime and improve safety while reducing costs.

This letter introduces HyperTWTL, an extension of TWTL with explicit quantification over multiple bounded execution traces for expressing timed hyperproperties. Leveraging the compact semantics of TWTL and expressibility of hyperproperties to reason about multiple execution trace, HyperTWTL can be used to express bounded hyperproperties more compactly and comprehensively compared to HyperMTL and HyperSTL. For example, consider a hyperproperty that requires that "*for any pair of traces π_1 and π_2, site X should be serviced for 2-time units in trace π_1 within the time bound* $[0, 5]$ *and site Y should also be serviced for 3-time steps in trace π_2 within the time bound* $[0, 7]$ ". This requirement can not be expressed directly using TWTL formalism, but can be expressed using the HyperTWTL formalism as $\varphi = \forall\pi_1\forall\pi_2 \cdot [\mathbf{H}^2\, X_{\pi_1}]^{[0,5]} \wedge [\mathbf{H}^3\, Y_{\pi_2}]^{[0,7]}$. The exact same requirement can be expressed as a HyperSTL formula as $\varphi = \forall\pi_1\forall\pi_2 \cdot (\mathbf{F}_{[0,5-2]}\mathbf{G}_{[0,2]}X_{\pi_1}) \wedge (\mathbf{F}_{[0,7-3]}\mathbf{G}_{[0,3]}Y_{\pi_2})$. Similarly, using the HyperMTL formalism, the exact same requirement can be expressed as $\forall\pi_1\forall\pi_2 \cdot \bigvee_{i=0}^{5-2}\mathbf{G}_{[i,i+2]}X_{\pi_1} \wedge \bigvee_{i=0}^{7-3}\mathbf{G}_{[i,i+3]}Y_{\pi_2}$. The compact semantics of HyperTWTL allows for a more succinct representation of this requirement than HyperMTL and HyperSTL, which require nested operators, shifted time windows, and the disjunction of several sub-formulae. For instance, in the specifications above it can also be observed that the given requirement can be formalized in HyperTWTL formula with total 5 temporal operators (without considering the quantifiers). This same requirement can be formalized as a HyperMTL formula using 17 temporal operators (excluding the quantifiers). This example shows the succinct characteristics of HyperTWTL over HyperMTL. Indeed, with the increasing complexity of requirements, the complexity of HyperMTL formulae will also grow, which makes the formal analysis of HyperMTL formulae very expensive for complicated robotic applications.

With its compact representation of specification, HyperTWTL can be used to specify important planning objectives such as privacy, optimality, and robustness in robotic planning missions with strict time constraints. To synthesize strategies adhering to HyperTWTL specifications, we adopt a symbolic strategy synthesis approach by reducing the planning problem to a first-order logic satisfiability problem which an SMT solver then solves. To demonstrate the effectiveness and scalability of our approach, we formalize the motion planning objectives (related to optimality, information-flow security, and robustness) of a robotic surveillance mission using HyperTWTL. Then, the feasible strategies (paths adhering to the HyperTWTL specifications) are then synthesized using Z3 [35] and CVC4 [36] SMT solvers, both known for their industrial application [37], [38]. The results show that using the proposed approach; we can successfully and efficiently synthesize strategies from complex robustness, opacity, and optimality-related HyperTWTL properties for safety-critical robotic missions.

The rest of the letter is organized as follows: Section II presents the preliminaries. The syntax and semantics of HyperTWTL, as well as its application in robotic planning, are presented in Section III. The problem formulation and strategy synthesis are presented in Sections IV and V, respectively. We evaluate the feasibility and scalability of our proposed logic in Section VI. Related works are discussed in section VII. Finally, Section VIII concludes the letter.

II. Preliminaries

Let AP be a finite set of *atomic propositions* and $\Sigma = 2^{AP}$ be the powerset of AP. We call each element of Σ an event and is of the form e_i, where $i \in \mathbb{Z}_{\geq 0}$. A trace $t \in \Sigma^\omega$ denotes an infinite sequence of events over Σ, and $t \in \Sigma^*$ denotes a finite sequence of events over Σ. For a trace t, we denote $t[i].e$ as the event at time i, i.e. e_i. Let $t[i, j]$ denote the subtrace of trace t starting from time i up to time j.

Time Window Temporal Logic (TWTL): The set of TWTL formulae over a finite set of atomic propositions is inductively generated by:

$$\phi := \top \mid \mathbf{H}^d a \mid \mathbf{H}^d \neg a \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \phi_1 \odot \phi_2 \mid [\phi]^{[x,y]}$$

where $\top$ stands for true, a is an atomic proposition in AP. The operators $\mathbf{H}^d$, $\odot$ and $[\,]^{[\tau,\tau']}$ represent the hold operator with $d \in \mathbb{Z}_{\geq 0}$, concatenation operator and within operator respectively within a discrete-time constant interval $[x, y]$, where $x, y \in \mathbb{Z}_{\geq 0}$ and $y \geq x$, respectively and $\wedge$ and $\neg$ are the conjunction and negation operators respectively. The disjunction operator ($\vee$) can be derived from the negation and conjunction operators. Likewise, the implication operator ($\rightarrow$) can also be derived from the negation and disjunction operators.

TWTL semantics: The satisfaction relation defined by $\models$ defines when subtrace $t[i, j]$ of a timed-trace t from time i up to time j, satisfies the TWTL formula. This is denoted by $t[i, j] \models \phi$. Given a TWTL formula ϕ and a timed-trace $t[i, j]$, the semantics of the operators is defined in Table I. With $\phi = \mathbf{H}^d a$ and $\mathbf{H}^d \neg a$, a is expected to be repeated or not repeated for d time units with the condition that $a \in t[n].e$ and $a \notin t[n].e$ respectively. With $\phi = \phi_1 \wedge \phi_2$, the trace $t[i, j]$ satisfies both formula. The trace $t[i, j]$ is expected to satisfy at least one of the formulae in $\phi = \phi_1 \vee \phi_2$ while in $\neg\phi$, the trace, $t[i, j]$ does not satisfy the given formula. A given formula in the form $\phi_1 \odot \phi_2$ specifies that a given trace should satisfy the first formula first and the second afterwards with one time unit difference between the end of execution of ϕ_1 and the start of execution of ϕ_2. The trace, $t[i, j]$ must satisfy ϕ between the time window $[x, y]$ given $[\phi]^{[x,y]}$.

TABLE I
SEMANTICS OF TWTL

$t[i,j] \models \top$		
$t[i,j] \models \mathbf{H}^d a$	iff	$a \in t[n].e, \forall n \in i, ..., i+d \wedge (j-i) \geq d$
$t[i,j] \models \mathbf{H}^d \neg a$	iff	$a \notin t[n].e, \forall n \in i, ..., i+d \wedge (j-i) \geq d$
$t[i,j] \models \phi_1 \wedge \phi_2$	iff	$(t[i,j] \models \phi_1) \wedge (t[i,j] \models \phi_2)$
$t[i,j] \models \neg\phi$	iff	$\neg(t[i,j] \models \phi)$
$t[i,j] \models \phi_1 \odot \phi_2$	iff	$\exists k = \arg\min_{i \leq k < j}\{t[i,k] \models \phi_1\} \wedge (t[k+1,j] \models \phi_2)$
$t[i,j] \models [\phi]^{[x,y]}$	iff	$\exists k \geq i+x \ s.t., t[k,i+y] \models \phi \wedge (j-i) \geq y$

III. HYPERTWTL

A *hyperproperty* is a set of sets of infinite traces. HyperTWTL is a hyper-temporal logic to specify hyperproperties for TWTL [11] by extending the TWTL with quantification over multiple and concurrent execution traces. In this section, we present the syntax and semantics of HyperTWTL.

A *hyperproperty* is a set of trace properties. HyperTWTL is a temporal logic for specifying hyperproperties by extending TWTL [11] with trace variables and explicit quantifiers over multiple traces. We assume in the semantics that the timestamps of all the quantified traces are synchronous, i.e., all the timestamps of traces match at each point in time. The syntax and semantics of HyperTWTL are now described as follows.

A. Syntax and Semantics of HyperTWTL

The set of formula in HyperTWTL is inductively defined by the following syntax:

$$\varphi := \exists\pi \cdot \varphi \mid \forall\pi \cdot \varphi \mid \phi$$

$$\phi := \mathbf{H}^d a_\pi \mid \mathbf{H}^d \neg a_\pi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \phi_1 \odot \phi_2 \mid [\phi]^{[x,y]}$$

The quantified HyperTWTL formula $\exists\pi$ and $\forall\pi$ are interpreted as "there exists some trace π" and "for all the traces π" respectively. a is an atomic proposition in AP and π is a trace variable in the set of trace variables $\mathcal{V}$. All other operators are interpreted as seen in Section II.

The satisfaction relation gives the semantics of synchronous HyperTWTL $\models$ over time-stamped traces $\mathbb{T}$. We define an assignment $\Pi : \mathcal{V} \to \Sigma^\omega \times \mathbb{Z}_{\geq 0}$ as a partial function mapping trace variables to traces. We then denote the explicit mapping of the trace variable π to a trace $t[i,j] \in \mathbb{T}$ as $\Pi[\pi \to t[i,j]]$. We present the semantics of HyperTWTL in Table II. From Table II, with $\phi = \mathbf{H}^d a_\pi$ and $\mathbf{H}^d \neg a_\pi$, a is expected to be repeated or not repeated for d time units with the condition that $a \in t[p].e$ and $a \notin t[p].e$ respectively on the trace mapped to trace variable π. With $\phi = \phi_1 \wedge \phi_2$, the set $\mathbb{T}$ must satisfy both sub-formulae. The trace set $\mathbb{T}$ is expected to satisfy both formulae in $\phi = \phi_1 \wedge \phi_2$ while in $\neg\phi$, $\mathbb{T}$, does not satisfy the given formula. Given a formula in the form $\phi_1 \odot \phi_2$, every $t[i,j] \in \mathbb{T}$ should satisfy the first formula first and the second afterwards with one time unit difference between the end of execution of ϕ_1 and start of execution of ϕ_2. The set of traces $\mathbb{T}$, must satisfy ϕ between the time window $[x,y]$ given $[\phi]^{[x,y]}$.

B. HyperTWTL Execution Deadline

The satisfaction of a HyperTWTL formula can be verified within bounded time. We denote the maximum time needed to satisfy φ by $\|\varphi\|$, which can be recursively computed as follows:

$$\|\varphi\| = \begin{cases} \|\varphi\| & \text{if} \quad \varphi \in \{\exists\pi \cdot \varphi, \forall\pi \cdot \varphi\} \\ d & \text{if} \quad \varphi \in \{\mathbf{H}^d a_\pi, \mathbf{H}^d \neg a_\pi\} \\ \max(\|\varphi_1\|, \|\varphi_2\|) & \text{if} \quad \varphi \in \{\varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2\} \\ \|\varphi_1\| & \text{if} \quad \varphi = \neg\varphi_1 \\ \|\varphi_1\| + \|\varphi_2\| + 1 & \text{if} \quad \varphi = \varphi_1 \odot \varphi_2 \\ y & \text{if} \quad \varphi = [\varphi_1]^{[x,y]} \end{cases} \tag{1}$$

In the rest of the letter, we refer to this computed deadline $\|\varphi\|$ as the *unrolling bound* $\|\varphi\|$. We describe the details of unrolling the HyperTWTL formula and the Deterministic Transition System (DTS) based on $\|\varphi\|$ in the following sections.

IV. PROBLEM FORMULATION

We assume an agent moves in a 2D environment whose abstraction is given by an $N \times M$ grid. Given the environment, we model the dynamics of each agent as a Deterministic Transition System (DTS) whose transitions are labelled by actions.

Definition 1: (Deterministic Transition System) A deterministic transition system (DTS) is a tuple $\mathcal{D} = (S, s_{init}, A, AP, \Delta, l)$ where S is a finite set of states; $s_{init} \subseteq S$ is a set of initial states; A is a set of actions; $\Delta : S \times A \to S$ is a partial transition function; AP is a finite set of atomic propositions; and $l : S \to 2^{AP}$ is a labelling function.

We assume that for each state $s \in S$, there exists another state $s' \in S$ that can be reached in a finite number of transitions. We represent a transition on an action $a_i \in A$ at time step k from state s_k to s_{k+1} as $s_{k+1} = (a_i, s_k)$.

A planning strategy $pol : Z_{\geq 0} \to A$ is therefore given by an infinite valid sequence of actions, $A = a_0 a_1 \ldots$. However, due to the fact that the semantics of HyperTWTL is already time bounded, only a finite prefix of the pol is considered. Given a strategy pol of $\mathcal{D}$ with initial state $s_0 \in s_{init}$, a path $\pi : Z_{\geq 0} \to S$ of $\mathcal{D}$ can be generated as $\pi(k+1) = \Delta(\pi(k), pol(k)) \, \forall k \in Z_{\geq 0}$. A path can then be defined as a sequence of states $\pi = s_0 s_1 s_2 \cdots \in S^\omega$. This path π generates a trace $t = e_0 e_1 e_2 \cdots \in \Sigma^\omega$ where $e_i = l(s_i) \, \forall i \geq 0$.

Problem statement: Given a DTS $\mathcal{D} = (S, s_{init}, A, AP, \Delta, l)$ and a HyperTWTL formula φ with an unrolling bound $\|\varphi\|$, the planning task is to find a path π over $\mathcal{D}$ with a depth of $\|\varphi\|$ and a related strategy policy pol such that the HyperTWTL objective φ is satisfied.

V. STRATEGY SYNTHESIS

This section describes the details of the symbolic strategy synthesis for HyperTWTL specifications which is a 3-step approach. First, we compute the unrolling bound based on (1) for synthesizing the strategy for the given objective expressed as a HyperTWTL specification. Second, the given HyperTWTL objective and the constraints associated with the given DTS model capturing the dynamics of the agent are converted into a

TABLE II
SEMANTICS OF HYPERTWTL

$(\mathbb{T},\Pi) \models \exists\pi.\varphi$	iff	$\exists t[i,j] \in \mathbb{T} \cdot (\mathbb{T},\Pi[\pi \to t[i,j]]) \models \varphi$
$(\mathbb{T},\Pi) \models \forall\pi.\varphi$	iff	$\forall t[i,j] \in \mathbb{T} \cdot (\mathbb{T},\Pi[\pi \to t[i,j]]) \models \varphi$
$(\mathbb{T},\Pi) \models \mathbf{H}^d a_\pi$	iff	$a \in t[p].e$ for $t[i,j] = \Pi(\pi)$, $\forall p \in \{p,...,p+d\} \wedge (j-i) \geq d$, for some $i > 0$
$(\mathbb{T},\Pi) \models \mathbf{H}^d \neg a_\pi$	iff	$a \notin t[p].e$ for $t[i,j] = \Pi(\pi)$, $\forall p \in \{p,...,p+d\} \wedge (j-i) \geq d$, for some $i > 0$
$(\mathbb{T},\Pi) \models \phi_1 \wedge \phi_2$	iff	$((\mathbb{T},\Pi) \models \phi_1) \wedge ((\mathbb{T},\Pi) \models \phi_2)$
$(\mathbb{T},\Pi) \models \neg\phi$	iff	$\neg((\mathbb{T},\Pi) \models \phi)$
$(\mathbb{T},\Pi) \models \phi_1 \odot \phi_2$	iff	$\exists k = \arg\min_{i \leq k \leq j} ((\mathbb{T},\Pi) \models \phi_1)$ for some $p \in [i,k]$, $((\mathbb{T},\Pi) \models \phi_2)$ for some $p' \in [k+1,j]$
$(\mathbb{T},\Pi) \models [\phi]^{[x,y]}$	iff	$\exists k \geq i+x,\ s.t.$ for some $p \in [k,i+y]$ $(\mathbb{T},\Pi) \models \phi \wedge (j-i) \geq y$

first-order logic formula by ensuring the $\exists$ and $\forall$ quantification comply with the sequence of states and actions within the computed bound in the first step. Lastly, using an off-the-shelf SMT solver, the converted formulae representing the DTS and the HyperTWTL formula are then combined and solved.

A. Encoding the HyperTWTL Formula

Given φ is a HyperTWTL formula of the form $\varphi = Q_1\pi_1 \ldots Q_n\pi_n \cdot \varphi$ where each $Q_i \in \{\forall, \exists\}$ $(i \in [1,n])$ and φ is a TWTL formula.

The unrolling of a TWTL formula on a path π_i, with bound $\|\varphi\|$ for all $i \leq \|\varphi\|$ results in a first-order logic formula which can be inductively defined as follows.

$$\begin{aligned}
[\![\mathbf{H}^d\, a_\pi]\!]_{i,\|\varphi\|} &:= a^i_\pi \text{ if } i \leq d \leq \|\varphi\| \\
[\![\mathbf{H}^d\, \neg a_\pi]\!]_{i,\|\varphi\|} &:= \neg a^i_\pi \text{ if } i \leq d \leq \|\varphi\| \\
[\![\phi_1 \wedge \phi_2]\!]_{(i,\|\varphi\|)} &:= [\![\phi_1]\!]_{(i,\|\varphi\|)} \wedge [\![\phi_2]\!]_{(i,\|\varphi\|)} \\
[\![\neg\phi]\!]_{i,\|\varphi\|} &:= \neg[\![\phi]\!]_{i,\|\varphi\|} \\
[\![\phi_1 \odot \phi_2]\!]_{(i,\|\varphi\|)} &:= \exists k = \arg\min_{i \leq k \leq \|\varphi\|} [\![\phi_1]\!]_{(i,k)} \\
&\quad \wedge [\![\phi_2]\!]_{(k+1,\|\varphi\|)} \\
[\![[\phi]^{[x,y]}]\!]_{i,\|\varphi\|} &:= \exists k \geq i+x, s.t. [\![\phi]\!]_{k,i+x} \wedge \\
&\quad (\|\varphi\| - i \geq y)
\end{aligned}$$

B. Encoding Path Constraints

Given the DTS $\mathcal{D}$, each path is unrolled to the depth of $\|\varphi\|$. Thus, the path generated over the DTS $\mathcal{D}$ should therefore satisfy the constraints

$$P_i = I(s^i_0) \wedge \bigwedge_{k=0}^{\|\varphi\|-1} R(s^i_k, s^i_{k+1}) \tag{2}$$

where $I(s^i_0)$ is the characteristic function for the boolean formula that encodes the initial states and $R(s^i_k, s^i_{k+1})$ is the function for the boolean formula that encodes transition relation for a state s^i_k and its successor s^i_{k+1}.

From Definition 1, the transition on an action a_i at time step k from a predecessor state s^i_k to a successor state s^i_{k+1} can be defined as $s^i_{k+1} = \Delta(s^i_k, a^i_k)$.

We further extend our framework to capture potential obstacles that can be encountered during the path synthesis process. Our framework tracks and updates the collision set $\mathcal{L}$ with all states occupied by obstacles. A successor, state s^i_{k+1} from a predecessor state s^i_k within the constraint P_i as seen in (2) is first checked if it is a member of the collision set $\mathcal{L}$. To avoid collisions into these obstacles, a robot does not transition into state s^i_{k+1} if $s^i_{k+1} \in \mathcal{L}$. This constraint can be expressed as:

$$\mathcal{C}_i = \bigwedge_{k \in [\|\varphi_i\|]} (s^i_{k+1} \in \mathcal{L} \to \neg a_i(k)) \tag{3}$$

Hence, the first-order logic formula for the SMT Solver to solve can be expressed as follows:

$$[Q_1\pi_1] \ldots [Q_n\pi_n] \cdot \left(\bigwedge_{i=1}^{n} P_i \wedge \mathcal{C}_i\right) \wedge \varphi \tag{4}$$

where $[Q_i\pi_i]$ for $i \leq n$, P_i is presented in (2) and $\mathcal{C}_i$ is presented in (3).

Remark 1: Given a DTS $\mathcal{D} = (S, s_{init}, A, AP, \Delta, l)$ and a HyperTWTL formula φ with an unrolling bound $\|\varphi\|$, the proposed strategy synthesis approach returns SAT only when a path that satisfies φ exists, otherwise returns UNSAT.

C. Soundness and Completeness

In this section, we prove the soundness and completeness properties of our strategy synthesis approach.

Theorem 1: The strategy synthesis approach is sound and complete.

Proof sketch: We will first prove the soundness of our path synthesis approach and then prove that the adopted approach is also complete.

Soundness: The soundness of our approach follows from the semantics and execution deadline of HyperTWTL presented in Section III. Hence, a synthesized path is required to satisfy the constraints as presented in the semantics of HyperTWTL in Table II. In addition to satisfying the semantics of HyperTWTL, each generated path over the states of $\mathcal{D}$ satisfies all other constraints generated by the encoding of the HyperTWTL formula φ.

Completeness: SMT solvers integrate a modern Davis–Putnam–Logemann–Loveland (T) (DPLL(T)) solver, a core theory (T) solver that handles equalities and uninterpreted functions, satellite solvers (for arithmetic, arrays, etc.), and an E-matching abstract machine (a technique for handling quantifiers) [35]. With DPLL(T) algorithm being the framework for determining the satisfiability of our SMT problems, we conclude that paths synthesized using our approach are complete since they are based on the sound and complete algorithm of DPLL(T).

Complexity: Our proposed approach converts a given DTS $\mathcal{D}$ and HyperTWTL formula φ into first-order logic formula. An SMT solver then combines and solves the converted first-order logic formula. The construction of DTS $\mathcal{D}$ has a computational complexity $\mathcal{O}(\Sigma_e \omega_e)$, where ω_e is the travel time of edge e [39]. The computational complexity for translating a HyperTWTL formula into a first-order logic formula is $\mathcal{O}(\|\varphi\| * |\varphi| * m)$ where $|\varphi|$ is the length of the formula and m is the number of quantifiers in φ. The computational complexity for solving a first-order logic formula with SMT can be given as $\mathcal{O}(2^m)$ in worst case where m is the number of variables in the formula. This is because given m variables, the final formula to be solved will have $\mathcal{O}(2^m)$ Boolean variables which have to be partially assigned (some variables are assigned values, some are left unassigned), applying the unit propagation and pure literal rules, and then determining if the resulting formula is trivially unsatisfiable. [40] The overall computational complexity is then given as $\mathcal{O}(\Sigma_e \omega_e + 2^{\|\varphi\| * |\varphi| * m})$.

VI. Implementation and Simulations

In this section, we evaluate the feasibility of our proposed path synthesis approach using a surveillance mission. In this case study, paths for robotic planning are synthesized by unfolding the discrete transition relations and HyperTWTL specifications using scripts implemented in Python 3.7. The satisfiability problem of the form (4) is then solved using Z3 and CVC4. If a path exists, the SMT solver returns the synthesized strategy, otherwise, a violation of the formula is returned. We consider 5 different scenarios by varying the planning objective expressed as a HyperTWTL specification. In each scenario, the synthesized path starts from an initial state from the set I within time bound $[0, T_1]$ and then to a service state(s) from the set P within the time bound $[T_2, T_3]$ to perform assigned surveillance mission. Finally, the path must end in any of the landing states from the set L within the time bound $[T_4, T_5]$. In all the transitions, all obstacles in the grid should be avoided. Based on this case study, the five different planning objectives considered are formalized as HyperTWTL specifications as follows:

Requirement 1 (Shortest path): A surveillance robot is required to find the shortest route from a given initial state to a goal state (landing state). Hence, given a set of traces, there exists a trace π_2 that reaches the landing state from the same initial state before any other trace π_1. We consider the following time bounds for the synthesis of this requirement: $T_1 = 2, T_2 = 3, T_3 = 8, T_4 = 9, T_5 = 13$. This requirement can be formalized as a HyperTWTL formula as:

$$\varphi_1 = \exists \pi_2 \forall \pi_1 \cdot [[\mathbf{H}^1 I_{\pi_1} \asymp \mathbf{H}^1 I_{\pi_2}]^{[0,T_1]} \odot [\mathbf{H}^1 P_{\pi_1} \wedge \mathbf{H}^1 P_{\pi_2}]^{[T_2,T_3]} \odot [[\mathbf{H}^1 L_{\pi_2}]^{[T_4,T_5]} \wedge [\mathbf{H}^1 L_{\pi_2}] \rightarrow [\mathbf{H}^1 L_{\pi_1}]^{[T_4,T_5]}]$$

Note that in φ_1 and all subsequent formulae, "$\asymp$" is not an arithmetic operator but a notation of simplification such that $[\mathbf{H}^1 I_{\pi_1} \asymp \mathbf{H}^1 I_{\pi_2}]$ stands for $\bigwedge_{i \in I}([\mathbf{H}^1 i_{\pi_1} \wedge \mathbf{H}^1 i_{\pi_2}])$.

Requirement 2 (Robustness under initial uncertainty): The robustness under initial uncertainty guarantees that a robot reaches the landing state regardless of the initial state it starts from. Thus, given two paths π_1 and π_2 starting from different initial states, both reach the charging state with the same set of actions A within the mission time. We consider the following time bounds for the synthesis of this requirement: $T_1 = 2\,s, T_2 = 3\,s, T_3 = 8\,s, T_4 = 9\,s, T_5 = 13\,s$. This requirement can be formalized as a HyperTWTL formula:

$$\varphi_2 = \exists \pi_1 \forall \pi_2 \cdot [[\mathbf{H}^1 I_{\pi_1} \not\asymp \mathbf{H}^1 I_{\pi_2}]^{[0,T_1]} \odot [\mathbf{H}^1 P_{1\pi_1} \wedge \mathbf{H}^1 P_{1\pi_2}]^{[T_2,T_3]} \odot [\mathbf{H}^1 P_{2\pi_1} \wedge \mathbf{H}^1 P_{2\pi_2}]^{[T_2,T_3]} \odot [\mathbf{H}^1 P_{3\pi_1} \wedge \mathbf{H}^1 P_{3\pi_2}]^{[T_2,T_3]} \odot [\mathbf{H}^1 L_{\pi_1} \wedge \mathbf{H}^1 L_{\pi_2}]^{[T_4,T_5]}] \wedge [\mathbf{H}^{T_5-T_2} A_{\pi_1} \not\asymp \mathbf{H}^{T_5-T_2} A_{\pi_2}]^{[T_2,T_5]}.$$

Requirement 3 (Robustness under action uncertainty): This form of robustness guarantees that a robot always reaches the landing state irrespective of the uncertainties, faults or adversarial factors along the synthesized path. Given a set of paths with the same set of actions, for any path π_2, there exists a feasible strategy for another path π_1 from the initial state to the landing state even if an action from the set A is replaced with another arbitrary action on path π_1. The following time bounds for the synthesis of this requirement: $T_1 = 2\,s, T_2 = 3\,s, T_3 = 8\,s, T_4 = 9\,s, T_5 = 13\,s$. This can be formalized as a HyperTWTL requirement as:

$$\varphi_3 = \exists \pi_1 \forall \pi_2 \cdot ([\mathbf{H}^1 I_{\pi_1} \wedge \mathbf{H}^1 I_{\pi_2}]^{[0,T_1]} \odot [\mathbf{H}^1 P_{\pi_1} \wedge \mathbf{H}^1 P_{\pi_2}]^{[T_2,T_3]} \odot [\mathbf{H}^1 L_{\pi_1} \wedge \mathbf{H}^1 L_{\pi_2}]^{[T_4,T_5]}) \wedge [\mathbf{H}^1 A_{\pi_1} \not\asymp \mathbf{H}^1 A_{\pi_2}] \rightarrow [\mathbf{H}^{T_5-T_2} A_{\pi_1} \asymp \mathbf{H}^{T_5-T_2} A_{\pi_2}]^{[T_1,T_5]}$$

Requirement 4 (Initial-state opacity): Opacity is a major security requirement in path synthesis often used to determine whether the secrets of a path have been leaked to intruders. A system satisfies the opacity requirement if it meets both of the following conditions: (i) there are at least two executions of the system mapped to π_1 and π_2 with the same observations but bearing different secrets; and (ii) the secret of each path cannot be accurately determined by observing(partially) the system alone. For this requirement, let the initial state of the paths be the secret and observe whether both paths reach the landing states with the same set of observations. We consider the following time bounds for the synthesis of this requirement: $T_1 = 2\,s, T_2 = 3\,s, T_3 = 8\,s, T_4 = 9\,s, T_5 = 13\,s$. This objective can be expressed using HyperTWTL as:

$$\varphi_4 = \exists \pi_1 \exists \pi_2 \cdot ([\mathbf{H}^1 I_{\pi_1} \not\asymp \mathbf{H}^1 I_{\pi_2}]^{[0,T_1]} \odot ([\mathbf{H}^1 P_{1\pi_1} \wedge \mathbf{H}^1 P_{1\pi_2}]^{[T_2,T_3]} \odot [\mathbf{H}^1 P_{2\pi_1} \wedge \mathbf{H}^1 P_{2\pi_2}]^{[T_2,T_3]} \wedge [\mathbf{H}^{T_5-T_2} A_{\pi_1} \asymp \mathbf{H}^{T_5-T_2} A_{\pi_2}]^{[T_1,T_5]}) \odot [\mathbf{H}^1 L_{\pi_1} \wedge \mathbf{H}^1 L_{\pi_2}]^{[T_4,T_5]})$$

Requirement 5 (Current-state opacity): For current state opacity requirement, let the synthesized path be the secret and the initial states of all paths be the only information a system user can observe. This requirement guarantees that an intruder never determines whether a system is currently in a secret state. Hence, if there exists a path π_1 that ended in a secret state, there exists a non-secret path π_2 whose observation is the same as that of π_1 (we denote O as the set of observation). We consider the following time bounds for the synthesis of this requirement: $T_1 = 2\,s, T_2 = 3\,s, T_3 = 8\,s, T_4 = 9\,s, T_5 = 13\,s$. This objective can be expressed using HyperTWTL as:

$$\varphi_5 = \exists \pi_1 \exists \pi_2 . ([\mathbf{H}^1 I_{\pi_1} \wedge \mathbf{H}^1 I_{\pi_2}]^{[0,T_1]} \odot [\mathbf{H}^1 P_{1\pi_1} \wedge \mathbf{H}^1 P_{1\pi_2}]^{[T_2,T_3]} \odot [\mathbf{H}^1 P_{2\pi_1} \wedge \mathbf{H}^1 P_{2\pi_2}]^{[T_2,T_3]} \odot [\mathbf{H}^1 L_{\pi_1} \wedge \mathbf{H}^1 L_{\pi_2}]^{[T_4,T_5]}) \wedge [\mathbf{H}^1 A_{\pi_1} \not\asymp \mathbf{H}^1 A_{\pi_2}]^{[T_1,T_5]} \wedge [\mathbf{H}^{T_5-T_2} O_{\pi_1} \asymp \mathbf{H}^{T_5-T_2} O_{\pi_2}]^{[T_1,T_5]}$$

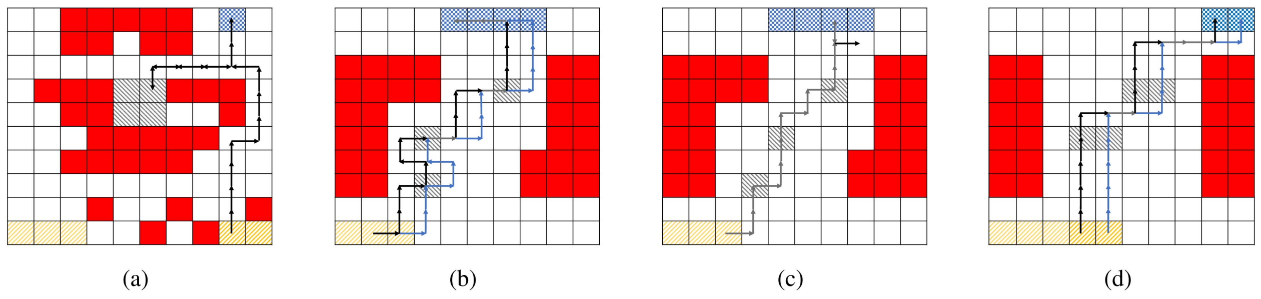

Fig. 1. (a) Strategy for shortest path φ_1. (b) Strategy for robustness under initial uncertainty φ_2. (c) Strategy for robustness under action uncertainty φ_3. (d) Strategy for initial-state opacity φ_4.

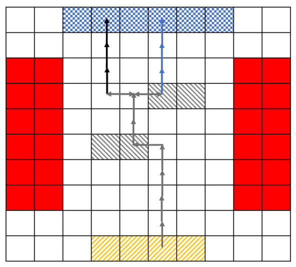

Fig. 2. Strategy for current-state opacity φ_5.

TABLE III
Comparison of Synthesis Times for Objectives Expressed as HyperTWTL and HyperLTL Specifications

Grid size	**Unrolling bound** ($\|\|\varphi\|\|$)	**HyperTWTL time** (seconds)	**HyperLTL time** (seconds)
10x10	20	2.77	5.04
20x20	40	9.81	14.13
40x40	60	44.36	63.75
60x60	80	285.55	359.32

The feasible paths on a 10×10 grid for all the 5 objectives above are shown in Fig. 1 where the yellow, blue, grey, red and white colors represent the initial, landing, service, obstacles and the allowable states respectively. For the shortest path objective, φ_1, the synthesized black path in Fig. 1(a) is the shortest path from the initial state through to the service state to the landing state. The feasible strategy for the initial state robustness objective from φ_2 is shown in Fig. 1(b). The synthesized path in black is initial-state robust as there exists a corresponding feasible strategy in blue that reaches the goal state from any of the initial states. In Fig. 1(c), the synthesized black path is action robust meaning the corresponding strategy is feasible for any single uncertain action. The synthesized path in black in Fig. 1(d) is the feasible strategy for the initial-state opacity that has the same observation as the path in blue. Both paths reach the landing state despite starting from a different initial state. Finally, from Fig. 2, the synthesized path in black is the feasible path for the current-state opacity that yields the same observation as the blue path but has different actions.

In the first set of experiments, we compare the performance of the proposed path planning approach for HyperTWTL objectives to a similar approach proposed for HyperLTL objectives in [33]. We formalized the shortest path requirement as HyperLTL and HyperTWTL specifications. We acknowledge the structural differences in the HyperLTL and HyperTWTL specifications since the latter can handle explicit time constraints. However, since there is no publicly available tool for verifying/synthesis of HyperSTL and HyperMTL specifications, we chose the implementation of HyperLTL in [33] as an arbitrary tool to compare our results. Using the computed bound in HyperTWTL as the unrolling bound for the HyperLTL, we compare the performance of the proposed approach for synthesizing requirements captured as HyperLTL and HyperTWTL specifications. The unrolling bounds ranging from $||\varphi|| = 20$ to $||\varphi|| = 80$ were used against varying grid sizes ranging from 10×10 to 60×60 to synthesize paths for the formalized HyperLTL and HyperTWTL specifications. The experiments are performed on an Windows 10 system with 64 GB RAM and Intel Core(TM) i9-10900 CPU (3.70 GHz). The two specifications that capture the robustness under initial uncertainty requirement as used in these experiments are as follows.

HyperTWTL: $\varphi = \exists\pi_1\forall\pi_2 \cdot [[\mathbf{H}^1\, I_{\pi_1} \not\asymp \mathbf{H}^1 I_{\pi_2}]^{[0,T_1]} \odot [\mathbf{H}^1 P_{1\pi_1} \wedge \mathbf{H}^1 P_{1\pi_2}]^{[T_2,T_3]} \odot [\mathbf{H}^1\, P_{2\pi_1} \wedge \mathbf{H}^1\, P_{2\pi_2}]^{[T_2,T_3]} \odot [\mathbf{H}^1\, P_{3\pi_1} \wedge \mathbf{H}^1\, P_{3\pi_2}]^{[T_2,T_3]} \odot [\mathbf{H}^1\, L_{\pi_1} \wedge \mathbf{H}^1\, L_{\pi_2}]^{[T_4,T_5]}] \wedge [\mathbf{H}^{T_5-T_2}\, A_{\pi_1} \asymp \mathbf{H}^{T_5-T_2}\, A_{\pi_2}]^{[T_2,T_5]}$

HyperLTL: $\varphi = \exists\pi_1\forall\pi_2 \cdot (I_{\pi_1} \not\asymp I_{\pi_2}) \wedge (P_{1\pi_1} \wedge P_{1\pi_2}) \wedge (P_{2\pi_1} \wedge P_{2\pi_2}) \wedge (P_{3\pi_1} \wedge P_{3\pi_2}) \wedge (L_{\pi_1} \wedge L_{\pi_2}) \wedge \Box_T(A_{\pi_1} \asymp A_{\pi_1})$

We considered the upper bounds $k = 10$, $k = 30$, $k = 50$, and $k = 70$ for the computation of the unrolling bounds in HyperTWTL specification. The respective synthesis time for this experiment is shown in Table III. We observe from Table III that for each HyperTWTL and HyperLTL formula, the computation time for the path synthesis increases with an increase in grid size as well as unrolling bound. However, the results shown in Table III suggest that it takes less time to synthesize policies for any planning objective expressed using HyperTWTL than HyperLTL. For instance, for a grid size of 10×10 with an unrolling bound of 20, it takes 2.77 seconds to synthesize a feasible strategy from the initial state to the landing state for the objective expressed as a HyperTWTL specification. However, it takes 5.04 seconds to synthesize a strategy for the same planning objective expressed as a HyperLTL specification. When the grid size and the unrolling bound are increased to 40×40 and 60 respectively, it takes 44.36 seconds to synthesize a strategy for an objective expressed as HyperTWTL specification while it

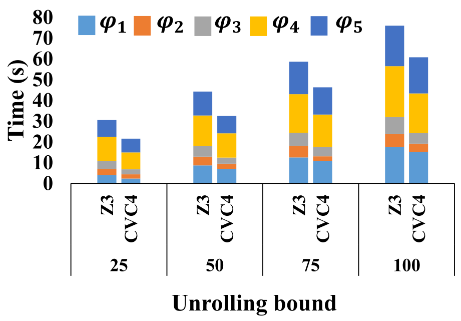

Fig. 3. Comparison of synthesis times for HyperTWTL objectives using Z3 and CVC4 Solvers.

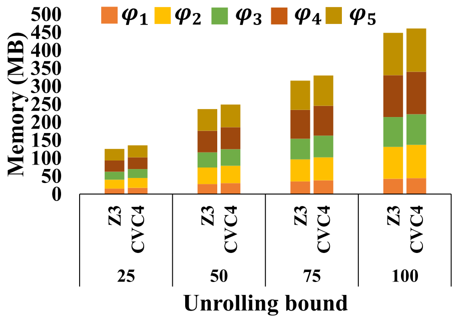

Fig. 4. Comparison of memory consumed for HyperTWTL objectives using Z3 and CVC4 Solvers.

takes 63.75 seconds to synthesize a feasible strategy for a path objective expressed as a HyperLTL specification. A similar trend is observed for the rest of the grid sizes and unrolling bounds.

In the second set of experiments, we evaluate the scalability and performance of the proposed path synthesis tool with Z3 and CVC4 solvers. While keeping the grid size constant to 10×10, we vary unrolling bounds from $\|\varphi = 25\|$ to $\|\varphi = 100$ to synthesize paths for HyperTWTL objectives $\varphi_1 - \varphi_5$. We then analyze the impact of $\|\varphi\|$ on the performance of the proposed tool. The respective synthesis time and memory consumed are shown in Figs. 3 and 4.

We observe from Fig. 3 CVC4 synthesizes paths within a shorter time than Z3. For instance, while synthesizing a feasible strategy for φ_1 for $\|\varphi\| = 25$, Z3 takes 3.91 seconds. In contrast, CVC4 takes 2.34 seconds while synthesizing a feasible strategy for the same objective for $\|\varphi\| = 25$. Similarly, while synthesizing a feasible strategy for φ_4 for $\|\varphi\| = 75$, Z3 takes 18.56 seconds whereas CVC4 takes 15.57. A similar trend is observed for the rest of the HyperTWTL objectives. From the results shown, we also observe a linear trend between SMT solvers' synthesis time and unrolling bounds. For instance, Z3 takes 3.91 seconds, 8.64 seconds, 12.52 seconds, and 17.53 seconds to synthesize paths for objective φ_1 for $\|\varphi\| = 25$, $\|\varphi\| = 50$, $\|\varphi\| = 75$, and $\|\varphi\| = 100$ respectively. Similarly, CVC4 takes 2.34 seconds, 6.98 seconds, 10.72 seconds and 15.23 seconds to synthesize paths for objective φ_1 for $\|\varphi\| = 25$, $\|\varphi\| = 50$, $\|\varphi\| = 75$, and $\|\varphi\| = 100$, respectively. Consequently, as shown in Fig. 4, we observe that Z3 consumes less memory than CVC4 for synthesizing paths for HyperTWTL objectives. For instance, while synthesizing a path for the objective φ_2 for $\|\varphi\| = 50$, CVC4 consumes 48.17 MB. In contrast, the memory consumed for synthesizing a path with the same property for $\|\varphi\| = 50$ using Z3 consumes 45.98 MB. Similarly, while synthesizing a path for φ_4 for $\|\varphi\| = 100$, CVC4 consumes 118.25 MB whereas Z3 consumes only 116.25 MB. Again, a similar trend of memory consumption is observed for the rest of the HyperTWTL objectives. Once again, We observe a linear trend between SMT solvers' memory consumption and unrolling bounds. For instance, Z3 consumes 31.55 MB, 59.78 MB, 80.15 MB and 116.25 MB, respectively for verifying φ_4 for $\|\varphi\| = 25$, $\|\varphi\| = 50$, $\|\varphi\| = 75$, and $\|\varphi\| = 100$. However, the memory consumption increases to 33.00 MB, 61.62 MB, 83.06 MB and 118.25 MB respectively for verifying the same property for $\|\varphi\| = 25$, $\|\varphi\| = 50$, $\|\varphi\| = 75$, and $\|\varphi\| = 100$ using CVC4.

VII. Related Works

Following the introduction of the concept of hyperproperties by Clarkson and Schneider [21], various hyperproperties formalisms have been proposed. These logics have been used in formalizing complex requirements in different applications. Very recently, control/strategy synthesis from hyperproperties has received a great attention from researchers. Interestingly the majority of these control/strategy synthesis literature are focused on HyperLTL. In [24], the authors studied the controller synthesis problem of finite-state systems for HyperLTL specifications. They demonstrated through a comprehensive analysis of the controller synthesis problem for various fragments of HyperLTL that the problem can be decided for HyperLTL specifications and finite-state plants. A reactive synthesis problem for hyperproperties expressed as HyperLTL specifications was also investigated in [41]. The authors also studied the bounded version of the synthesis problem for $\forall^*$ fragment of HyperLTL and presented a semi-decision procedure that constructs counterexamples of a given system up to a given bound. In [42], a bounded model checking (BMC) approach for HyperLTL is presented. The proposed approach reduces the BMC problem to a Quantified Boolean Formula (QBF) solving problem and then solves it using HyperQube tool. The authors demonstrated the HyperQube tool can be used for path planning in robotic systems. Probablistic hyperproperties was investigated in [43]. The authors extended the syntax and semantics of HyperPCTL with the notion of rewards to express the accumulated reward relation among different computations. Among the many applications, the authors demonstrated that the extended logic can be used to formalize path planning objectives in robotic applications. In [33], the authors presented a symbolic approach for synthesizing robotic planning strategies on discrete transition systems HyperLTL. Indeed, this work is the most relevant to our work. However, HyperLTL cannot express tasks with explicit time constraints which is addressed by introducing the HyperTWTL in this letter.

VIII. Conclusion

In this letter, we proposed HyperTWTL, which extends the semantics of TWTL with quantifiers and trace variables for specifying timed hyperproperties. Furthermore, given HyperTWTL specifications, we propose a symbolic strategy synthesis approach for robustness-, optimality- and privacy-aware robotic motion planning. This was achieved by reducing the planning problem to a first-order logic satisfiability problem. The proposed approach's feasibility, efficiency, and scalability were demonstrated using a surveillance mission case study and two industrial-strength SMT solvers. In the future, we plan to propose a real-time strategy synthesis approach and implement it on real robots to show the on-field effectiveness of our proposed methods.

References

[1] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-based probabilistic LTL motion planning with environment and motion uncertainties," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2386–2392, May 2021.

[2] M. Cai, H. Peng, Z. Li, and Z. Kan, "Receding horizon control-based motion planning with partially infeasible LTL constraints," *IEEE Control Syst. Lett.*, vol. 5, no. 4, pp. 1279–1284, Oct. 2021.

[3] M. Cai, S. Xiao, Z. Li, and Z. Kan, "Optimal probabilistic motion planning with potential infeasible LTL constraints," *IEEE Trans. Autom. Control*, vol. 68, no. 1, pp. 301–316, Jan. 2023.

[4] D. Tian et al., "Decentralized motion planning for multiagent collaboration under coupled LTL task specifications," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 6, pp. 3602–3611, Jun. 2022.

[5] C. K. Verginis, C. Vrohidis, C. P. Bechlioulis, K. J. Kyriakopoulos, and D. V. Dimarogonas, "Reconfigurable motion planning and control in obstacle cluttered environments under timed temporal tasks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 951–957.

[6] S. Saha and A. A. Julius, "Task and motion planning for manipulator arms with metric temporal logic specifications," *IEEE Robot. Automat. Lett.*, vol. 3, no. 1, pp. 379–386, Jan. 2018.

[7] Z. Lin et al., "Optimization-based motion planning and runtime monitoring for robotic agent with space and time tolerances," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1874–1879, 2020.

[8] Y. Zhou, D. Maity, and J. S. Baras, "Timed automata approach for motion planning using metric interval temporal logic," in *Proc. IEEE Eur. Control Conf.*, 2016, pp. 690–695.

[9] F. S. Barbosa et al., "Formal methods for robot motion planning with time and space constraints," in *Proc. FORMATS*. Berlin, Germany: Springer, 2021, pp. 1–14.

[10] K. Leahy et al., "Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints," *Auton. Robots*, vol. 40, no. 8, pp. 1363–1378, 2016.

[11] C. I. Vasile et al., "Time window temporal logic," *Theor. Comput. Sci.*, vol. 691, pp. 27–54, 2017.

[12] E. Bonnah and K. A. Hoque, "Runtime monitoring of time window temporal logic," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 5888–5895, Jul. 2022.

[13] A. S. Asarkaya, D. Aksaray, and Y. Yazicioğlu, "Temporal-logic-constrained hybrid reinforcement learning to perform optimal aerial monitoring with delivery drones," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2021, pp. 285–294.

[14] R. Peterson et al., "Distributed safe planning for satisfying minimal temporal relaxations of TWTL specifications," *Robot. Auton. Syst.*, vol. 142, 2021, Art. no. 103801.

[15] D. Aksaray et al., "Probabilistically guaranteed satisfaction of temporal logic constraints during reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 6531–6537.

[16] A. T. Büyükkoçak, D. Aksaray, and Y. Yazicioglu, "Distributed planning of multi-agent systems with coupled temporal logic specifications," in *Proc. AIAA Scitech Forum*, 2021, p. 1123.

[17] A. S. Asarkaya, D. Aksaray, and Y. Yazicioglu, "Persistent aerial monitoring under unknown stochastic dynamics in pick-up and delivery missions," in *Proc. AIAA Scitech Forum*, 2021, p. 1125.

[18] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Syst.*, vol. 2, no. 4, pp. 255–299, 1990.

[19] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proc. Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Berlin, Germany: Springer, 2004, pp. 152–166.

[20] I. Tkachev and A. Abate, "Formula-free finite abstractions for linear temporal verification of stochastic hybrid systems," in *Proc. 16th Int. Conf. Hybrid Syst.: Computation Control*, 2013, pp. 283–292.

[21] M. R. Clarkson et al., "Hyperproperties," *J. Comput. Secur.*, vol. 18, no. 6, pp. 1157–1210, 2010.

[22] S. Zdancewic and A. C. Myers, "Observational determinism for concurrent program security," in *Proc. IEEE 16th Comput. Secur. Found. Workshop*, 2003, pp. 29–43.

[23] J. A. Goguen and J. Meseguer, "Security policies and security models," in *Proc. IEEE Symp. Secur. Privacy*, 1982, pp. 11–11.

[24] B. Bonakdarpour and B. Finkbeiner, "Controller synthesis for hyperproperties," in *Proc. IEEE 33rd Comput. Secur. Found. Symp.*, 2020, pp. 366–379.

[25] B. Finkbeiner et al., "EAHyper: Satisfiability, implication, and equivalence checking of hyperproperties," in *Proc. Computer Aided Verification*. Berlin, Germany: Springer, 2017, pp. 564–570.

[26] B. Bonakdarpour et al., "Monitoring hyperproperties by combining static analysis and runtime verification," in *Proc. Int. Symp. Leveraging Appl. Formal Methods*, 2018, pp. 8–27.

[27] M. R. Garey et al., "Computers and intractability," in *A Guide to the Theory of np-Completeness*, New York, NY, USA: W. H. Freeman & Co., 1979.

[28] M. R. Clarkson et al., "Temporal logics for hyperproperties," in *Proc. POST.* Berlin, Germany: Springer, 2014, pp. 265–284.

[29] L. V. Nguyen et al., "Hyperproperties of real-valued signals," in *Proc. IEEE/ACM 15th Int. Conf. Formal Methods Models Syst. Des.*, 2017, pp. 104–113.

[30] B. Bonakdarpour et al., "Model checking timed hyperproperties in discrete-time systems," in *Proc. NASA Formal Methods*. Berlin, Germany: Springer, 2020, pp. 311–328.

[31] A. Pnueli, "The temporal logic of programs," in *Proc. IEEE 18th Annu. Symp. Found. Comput. Sci.*, 1977, pp. 46–57.

[32] H. M. Ho, R. Zhou, and T. M. Jones, "On verifying timed hyperproperties," in *Proc. 26th Int. Symp. Temporal Representation Reasoning (TIME)*, 2018, pp. 20:1–20:18.

[33] Y. Wang et al., "Hyperproperties for robotics: Planning via HyperLTL," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 8462–8468.

[34] ANYbotics, "Automation & digitalization at scale: ANYmal makes the case at BASF." Accessed: May 16, 2023. [Online]. Available: https://www.anybotics.com/anymal-makes-the-case-at-basf-chemical-plant/

[35] L. Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Proc. Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2008, pp. 337–340.

[36] M. Deters, A. Reynolds, T. King, C. Barrett, and C. Tinelli, "A tour of CVC4: How it works, and how to use it," in *Proc. IEEE Formal Methods Comput.-Aided Des.*, 2014, pp. 7–7.

[37] N. Rungta, "A billion SMT queries a day," in *Proc. Computer Aided Verification*. Berlin, Germany: Springer, 2022, pp. 3–18.

[38] N. Bjørner, "Z3 and SMT in industrial R&D," in *Proc. FM*. Berlin, Germany: Springer, 2018, pp. 675–678.

[39] D. Aksaray et al., "Dynamic routing of energy-aware vehicles with temporal logic constraints," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 3141–3146.

[40] P. Liberatore, "Complexity results on DPLL and resolution," *ACM Trans. Comput. Logic*, vol. 7, no. 1, pp. 84–107, 2006.

[41] B. Finkbeiner et al., "Synthesizing reactive systems from hyperproperties," in *Proc. Computer Aided Verification*. Berlin, Germany: Springer, 2018, pp. 289–306.

[42] T. Hsu et al., "Bounded model checking for hyperproperties," in *Proc. Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Germany: Springer, 2021, pp. 94–112.

[43] O. Dobe, L. Wilke, E. Ábrahám, E. Bartocci, and B. Bonakdarpour, "Probabilistic hyperproperties with rewards," in *Proc. NASA Formal Methods*. Berlin, Germany: Springer, 2022, pp. 656–673.