ECAI 2025
I. Lynce et al. (Eds.)

doi:10.3233/FAIA251000

On Conformant Planning and Model-Checking of $\exists^*\forall^*$ Hyperproperties

Raven Beutner and Bernd Finkbeiner

CISPA Helmholtz Center for Information Security, Germany

Abstract. We study the connection of two problems within the planning and verification community: Conformant planning and model-checking of hyperproperties. Conformant planning is the task of finding a sequential plan that achieves a given objective independent of non-deterministic action effects during the plan's execution. Hyperproperties are system properties that relate multiple execution traces of a system and, e.g., capture information-flow and fairness policies. In this paper, we show that model-checking of $\exists^*\forall^*$ hyperproperties is closely related to the problem of computing a conformant plan. Firstly, we show that we can efficiently reduce a hyperproperty model-checking instance to a conformant planning instance, and prove that our encoding is sound and complete. Secondly, we establish the converse direction: Every conformant planning problem is, itself, a hyperproperty model-checking task.

1 Introduction

In this paper, we identify two problems from two different research communities that seem unrelated at first glance, yet share the same computational challenge: Conformant planning and model-checking of $\exists^*\forall^*$ hyperproperties.

Conformant Planning Conformant planning is the task of finding a plan given uncertainty about the effect of action, and without any sensing ability during the plan's execution. That is, the same plan (i.e., sequence of actions) should achieve the goal, regardless of which non-deterministic action effects occur during the plan's execution [28].

Hyperproperties Hyperproperties [17] are system properties that relate multiple executions in a system and can thus capture properties that cannot be expressed by reasoning over individual traces. As an example, we consider a simple information-flow property. Assume we model the behavior of a system as a transition system over variables $\{o, h, l\}$, and want to specify that the output (o) of the system does not leak information about the secret input (h). We cannot specify such a property by reasoning about traces in isolation (e.g., in LTL). Instead, we need to relate multiple executions to observe how different inputs impact the output; a hyperproperty. HyperLTL [18] extends LTL with quantification over executions and can thereby express hyperproperties. For example, we can express a simple information-flow policy – called *non-inference* (NI) [32] – in HyperLTL as follows

$$\forall \pi_1.\, \exists \pi_2.\, \square(o_{\pi_1} = o_{\pi_2} \wedge l_{\pi_1} = l_{\pi_2}) \wedge \square(h_{\pi_2} = \dagger), \qquad \text{(NI)}$$

where $\square$ denotes LTL's *globally* operator. This formula states that for any execution (trace) π_1, there exists some execution π_2 that **(1)** globally has the same low-security observations as π_1 (i.e., output o and low-security input l globally agree between π_1 and π_2), and **(2)** the high-security input on π_2 equals some dummy value (denoted $\dagger$). If (NI) holds, an attacker thus cannot distinguish any high-security input sequence from the sequence of dummy values.

Two Sides of the Same Coin Conformant planning and model-checking of HyperLTL are computationally expensive problems within their respective communities. At first glance, they seem unrelated, and diverse solution concepts exist in both communities. Techniques employed in conformant planning often rely on an efficient heuristic search over *belief states*, i.e., sets of states that contain exactly those states that are currently plausible [29, 10]. In contrast, techniques employed in model-checking of hyperproperties employ more direct approaches using automata [4], symbolic execution [21], program logics [22, 20], or bounded unrolling [30]. Despite their differences, we demonstrate that the core algorithmic challenge between the two problems is shared. Concretely, we present efficient translations that reduce a conformant planning problem to an equivalent HyperLTL model-checking problem, and vice versa. Our ultimate hope is that this observation will lead to new solutions by adapting successful concepts from conformant planning to HyperLTL model-checking, and vice versa.

Hyperproperty Model-Checking as Conformant Planning Our first contribution is a novel encoding of HyperLTL verification into conformant planning. Our encoding is applicable to $\exists^*\forall^*$ HyperLTL formulas, i.e., formulas where an arbitrary number of existential trace quantifiers is followed by an arbitrary number of universal quantifiers. We can massage every formula with at most one quantifier alternation into a $\exists^*\forall^*$ formula. For example, (NI) is a $\forall\exists$ formula, but we can simply check the *negated* formula, which is a $\exists\forall$ formula:

$$\exists \pi_1.\, \forall \pi_2.\, \Diamond(o_{\pi_1} \neq o_{\pi_2} \vee l_{\pi_1} \neq l_{\pi_2}) \vee \Diamond(h_{\pi_2} \neq \dagger) \qquad \text{(NI}_\neg\text{)}$$

where $\Diamond$ denotes LTL's *eventually* operator. To check if some transition system $\mathcal{T}$ satisfies (NI$_\neg$), we view model-checking as a conformant planning problem. Each state in the planning problem maintains two system locations of $\mathcal{T}$, one for path π_1 and one for π_2. The idea is that each plan (i.e., sequence of actions) should define a unique path for π_1, i.e., each action updates the location for π_1 along some transition of $\mathcal{T}$. At the same time, the location of π_2 is updated *non-deterministically*. Any plan, therefore, defines a unique witness path for π_1, while the plan's executions non-deterministically explore *all* possible paths for π_2. The planning goal is to ensure that all executions of the plan eventually reach a goal state where $o_{\pi_1} \neq o_{\pi_2} \vee l_{\pi_1} \neq l_{\pi_2}$ or $h_{\pi_2} \neq \dagger$. We prove that if the resulting planning instance admits a conformant plan, $\mathcal{T}$ satisfies (NI$_\neg$).

Note how the conformant nature of the plan (i.e., the fact that the plan cannot depend on the nondeterministic outcomes) is critical to ensure that the witness trace for π_1 does not depend on π_2. Crucially, our encoding is not only sound (if a conformant plan exists, the HyperLTL formula is satisfied) but also *complete*, i.e., if $\mathcal{T}$ satisfies ($\text{NI}_\neg$), there exists a conformant plan. This is in sharp contrast to previous planning-based encodings of hyperproperties [5] (we discuss this in Section 2). We show that our encoding works both with an explicit-state representation (as in [5]), but also applies to symbolically represented planning problems (e.g., STRIPS planning).

Conformant Planning is a Hyperproperty After demonstrating that we can use conformant planning for model-checking $\exists^*\forall^*$ hyperproperties, we prove that conformant planning itself *is* a hyperproperty. Conformant planning requires a plan that achieves the goal independent of the effects of non-deterministic actions. The existence of a conformant plan thus corresponds to the satisfaction of the (informal) $\exists\forall$ HyperLTL formula $\exists\pi_1.\,\forall\pi_2.\,\big(\square\, sameAction(\pi_1,\pi_2)\big) \rightarrow \diamondsuit\, goal(\pi_2)$ over a transition system that generates all possible paths in the planning problem. I.e., there exists some path π_1, such that all paths π_2 with the same sequence of actions ($\square\, sameAction(\pi_1,\pi_2)$) eventually reach the goal ($\diamondsuit\, goal(\pi_2)$). While the close connection between conformant plans and quantification has been explored extensively before (e.g., in the form of SAT or QBF encodings [35, 36]; cf. Section 2), HyperLTL can directly capture the temporal nature of plans *without* bounding the length. We implement this translation from PDDL to HyperLTL model-checking instances in a prototype. Our results show that current HyperLTL verification tools struggle with the resulting instances, thus **(1)** creating a challenging set of benchmarks for future evaluation, and **(2)** highlighting the importance of heuristics in model-checking of $\exists^*\forall^*$ hyperproperties (an entirely unexplored research area).

Supplementary Materials Full proofs of all results can be found in the full version [8].

2 Related Work

Hyperproperty Model-Checking Finite-state model-checking of HyperLTL is decidable [18] but expensive: checking a formula with k quantifier alternations is k-fold exponential [34]. Complete algorithms rely on expensive automata complementation or language inclusion checks [25, 4]. Approximations for this expensive problem include QBF-based bounded unrolling [30], or a strategy-based instantiation of existential quantification [19, 2, 3]. In the former, the system is unrolled up to a fixed depth, so the quantification over traces reduces to a QBF formula. In the latter, we interpret verification of a $\forall^*\exists^*$ formula as a game between the universal and existential quantifiers, and attempt to find a winning strategy for the existential player. In contrast to these abstractions, our planning-based encoding precisely captures the HyperLTL semantics (i.e., it is sound and complete), at the cost of encoding into another computationally expensive problem: conformant planning [9].

Hyperproperties and Planning The connection between (classical) planning and formal verification has been explored in various forms [27, 26, 23, 16]. The work most closely related to ours is [5], showing that every HyperLTL model-checking problem can be translated soundly to a (contingent) multi-agent planning problem represented as a QDecPOMDP [11]. In [5]'s approach, verifying a $\forall^*\exists^*$ property abstracts to a FOND-planning problem by searching for a policy that resolves existentially-quantified traces, similar to the game-based verification approaches discussed above [19, 2]. This abstraction is *incomplete* (cf. [5, Remark 1]), i.e., a property might hold, but no witnessing contingent plan exists. The encoding of the present paper uses related ideas (i.e., we also simulate multiple paths in a planning problem) but identifies sensorless behavior – i.e., *conformant* instead of (fully-observable) *contingent* planning – as the key missing gadget. Consequently, our encoding for $\exists^*\forall^*$ properties (which by negation also applies to $\forall^*\exists^*$ formulas) is sound-and-*complete*. Conformant planning is thus a drop-in replacement for model-checking $\exists^*\forall^*$ properties; contingent (FOND) planning is only an abstraction. Moreover, we also study the reverse direction and show that conformant planning is, itself, a $\exists\forall$ hyperproperty. Obtaining a similar result for the QDecPOMDP seems challenging.

Knowledge and Games The connection between (missing) knowledge of an agent and verification of hyperproperties has been explored extensively in the context of game-based verification for HyperLTL [6, 7, 37]. We can view conformant planning as a special form of a two-player game, played between the planning agent and the environment, without any observations. In this light, we can see our planning-based encoding as a specialized verification game for $\exists^*\forall^*$ properties (compared to, e.g., the game from [6, 7]). A clear advantage to a planning-based approach is the fact that existing planning frameworks and tools often study symbolically represented domains (e.g., STRIPS or PDDL planning), whereas formal games under imperfect information are mostly studied in an explicit-state setting.

Conformant Planning and QBF Most conformant planning approaches employ heuristic search over *belief states*, i.e., sets of world states [10, 29], represented, e.g., as CNFs or BDDs [15]. Another line of research related to the present paper are QBF-based approaches to conformant planning [35, 36]. Similar to our encoding into HyperLTL, these approaches also exploit the connection between $\exists\forall$ quantification and conformant plans, either by directly using QBF or by repeated queries to a SAT solver. Given the temporal nature of the planning problem, a QBF encoding usually employs a bound on the length of the plan, similar to SAT-based classical planning [31]. In our framework, we can encode the temporal requirements directly using temporal logics (HyperLTL), allowing us to encode *unbounded* reachability properties or even more complex temporal requirements. The resulting HyperLTL model-checking query can then be handled by a wide range of techniques, some of which, themself, rely on QBF-solving [30].

3 Preliminaries

In this section, we introduce planning problems, transition systems, and HyperLTL. Planning problems are typically defined in a factored representation (e.g., STRIPS, PDDL) using Boolean propositions (or fluents) to represent the current state of the planning problem. Likewise, systems are typically symbolically defined by a set of Boolean variables (e.g., circuits or NuSmv models). To begin, we work with an *explicit-state* representation of the planning domain and system, simplifying our encoding. In Section 6, we then show how we can extend our encodings to symbolically represented planning problems and systems.

3.1 Conformant Planning

Definition 1. *A (non-deterministic) planning problem is a tuple* $\mathcal{P} = (S, s_0, G, \mathcal{O})$*, where* S *is a finite set of states,* $s_0 \in S$ *is an initial*

state, $G \subseteq S$ is a set of goal states, and $\mathcal{O}$ is a finite set of actions. Each action $a \in \mathcal{O}$ has the form $a = \langle pre_a, \mathit{eff}_a \rangle$, where $pre_a \subseteq S$ is a set of states defining the states in which the action can be applied, and $\mathit{eff}_a : pre_a \to (2^S \setminus \{\emptyset\})$ is the non-deterministic effect function mapping each state $s \in pre_a$ to a non-empty set of potential outcomes $\mathit{eff}_a(s) \subseteq S$.

Note how our definition assumes that there exists a unique initial state, which is w.l.o.g., as in our setting, action effects can be non-deterministic. The update function eff_a easily allows us to model conditional effects, which are crucial in conformant planning [29].

A plan is then a sequence of actions $\langle a_1, \dots, a_n \rangle$. Given a set of states $T \subseteq S$, we inductively define $[\![T, \langle a_1, \dots, a_n \rangle]\!] \subseteq S$ as the set of states reachable after executing plan $\langle a_1, \dots, a_n \rangle$. For the empty plan $\langle\rangle$, we have $[\![T, \langle\rangle]\!] := T$. For a non-empty plan we define

$$[\![T, \langle a_1, \dots, a_n \rangle]\!] := [\![\bigcup_{s \in T} \mathit{eff}_{a_1}(s), \langle a_2, \dots, a_n \rangle]\!]$$

if $T \subseteq pre_{a_1}$. That is, we consider all possible states $s \in T$ and all possible effects when applying a_1 ($\bigcup_{s \in T} \mathit{eff}_{a_1}(s)$), and then inductively execute the remaining plan $\langle a_2, \dots, a_n \rangle$. Note that $[\![T, \langle a_1, \dots, a_n \rangle]\!]$ is undefined if action a_1 is not applicable in some state in T [29]. A plan $\langle a_1, \dots, a_n \rangle$ is a *conformant plan* if $[\![\{s_0\}, \langle a_1, \dots, a_n \rangle]\!]$ is defined and $[\![\{s_0\}, \langle a_1, \dots, a_n \rangle]\!] \subseteq G$. That is, executing the plan from the initial state s_0 always results in a goal state, independent of the non-deterministic action effects. We assume that for every $s \in G$ and $a \in \mathcal{O}$ with $s \in pre_a$, $\mathit{eff}_a(s) \subseteq G$.

3.2 Transition Systems

We assume that AP is a fixed set of *atomic propositions*. As the basic system model, we use finite-state transition systems (TS).

Definition 2. *A transition system (TS) is a tuple $\mathcal{T} = (L, l_{init}, \mathbb{D}, \kappa, \ell)$, where L is a finite set of locations (following [5], we use "locations" to distinguish them from planning "states"), $l_{init} \in L$ is an initial location, $\mathbb{D}$ is a finite set of* directions*, $\kappa : L \times \mathbb{D} \to L$ is a transition function, and $\ell : L \to 2^{AP}$ is a labeling.*

A path in $\mathcal{T}$ is an infinite sequence $p \in L^\omega$ of locations such that **(1)** $p(0) = l_{init}$, and **(2)** for every $i \in \mathbb{N}$, there exists some direction $d \in \mathbb{D}$ with $p(i+1) = \kappa(p(i), d)$. We define $Paths(\mathcal{T}) \subseteq L^\omega$ as the set of all paths in $\mathcal{T}$.

3.3 HyperLTL

HyperLTL extends LTL with explicit quantification over system executions [18], thus lifting it from a logic expressing trace properties to one expressing hyperproperties. In this paper, we focus on $\exists^*\forall^*$ formulas, i.e., formulas where any number of existential quantifiers is followed by any number of universal quantifiers. Such formulas are generated by the following grammar

$$\psi := a_{\pi_i} \mid \psi \wedge \psi \mid \neg\psi \mid \bigcirc \psi \mid \psi \,\mathcal{U}\, \psi$$
$$\varphi := \exists \pi_1 \dots \exists \pi_n . \forall \pi_{n+1} \dots \forall \pi_{n+m} . \psi$$

where $\pi_1, \dots, \pi_{n+m}$ are so-called *path variables*, and $a \in AP$ is an atomic proposition. Here $\bigcirc$ and $\mathcal{U}$ denote LTL's *next* and *until* operator, respectively. Within the LTL body, we use the usual derived boolean constants and connectives $\mathit{true}, \mathit{false}, \vee, \rightarrow, \leftrightarrow$, and the temporal operators *eventually* ($\Diamond \psi := \mathit{true}\,\mathcal{U}\,\psi$), and *globally* ($\Box \psi := \neg \Diamond \neg \psi$). To define the semantics, we use a path assignment $\Pi : \{\pi_1, \dots, \pi_{n+m}\} \to Paths(\mathcal{T})$, which maps each path variable to a path. Given a path assignment Π, we can evaluate the LTL body ψ at some position $i \in \mathbb{N}$ as follows:

$$\begin{array}{lll}
\Pi, i \models a_\pi & \text{iff} & a \in \ell\big(\Pi(\pi)(i)\big) \\
\Pi, i \models \psi_1 \wedge \psi_2 & \text{iff} & \Pi, i \models \psi_1 \text{ and } \Pi, i \models \psi_2 \\
\Pi, i \models \neg\psi & \text{iff} & \Pi, i \not\models \psi \\
\Pi, i \models \bigcirc \psi & \text{iff} & \Pi, i+1 \models \psi \\
\Pi, i \models \psi_1 \,\mathcal{U}\, \psi_2 & \text{iff} & \exists k \geq i . \Pi, k \models \psi_2 \text{ and} \\
 & & \quad \forall i \leq j < k . \Pi, j \models \psi_1
\end{array}$$

Boolean and temporal operators are evaluated as for LTL by updating the current evaluation position i. The atomic formula a_π holds whenever a holds in the current position i on the path bound to π (as given by $\mathcal{T}$'s labeling ℓ). The quantifier prefix then quantifies over paths in the system to construct a path assignment, and evaluates this assignment on the LTL formula. Formally, $\mathcal{T} \models \exists \pi_1 \dots \exists \pi_n . \forall \pi_{n+1} \dots \forall \pi_{n+m} . \psi$, iff

$$\exists p_1, \dots, p_n \in Paths(\mathcal{T}) . \forall p_{n+1}, \dots, p_{n+m} \in Paths(\mathcal{T}) . \big[\pi_1 \mapsto p_1, \dots, \pi_{n+m} \mapsto p_{n+m}\big], 0 \models \psi .$$

For more details on HyperLTL, we refer to [24].

4 Hyperproperty Model-Checking as Planning

In this section, we show that we can interpret the verification of an $\exists^*\forall^*$ HyperLTL formula as a conformant planning problem. For this, assume that $\mathcal{T} = (L, l_{init}, \mathbb{D}, \kappa, \ell)$ is a fixed TS and $\varphi = \exists \pi_1 \dots \exists \pi_n . \forall \pi_{n+1} \dots \forall \pi_{n+m} . \psi$ is a fixed $\exists^*\forall^*$ HyperLTL formula. We want to check if $\mathcal{T} \models \varphi$. As sketched in the introduction, our main idea is to construct a planning problem such that each plan corresponds to concrete paths for $\pi_1, \dots, \pi_n$. For the plan to be successful, the concrete paths generated by that plan should be valid choices for the existentially quantified paths in φ. That is, the paths satisfy ψ (the LTL body of φ) no matter what paths we pick for $\pi_{n+1}, \dots, \pi_{n+m}$. We ensure the latter by exploring *all* possible choices for $\pi_{n+1}, \dots, \pi_{n+m}$ using non-deterministic action effects.

Temporal Reachability For now, we assume that ψ – the LTL body of φ – expresses a reachability property. Note that most properties studied in practice have the form $\forall^*\exists^* \Box \psi$ for some propositional formula ψ (see, e.g., [24, 3, 20]), so their negation will have the form $\exists^*\forall^* . \Diamond \neg\psi$, which is a reachability property. We will later discuss how we can encode arbitrary $\exists^*\forall^*$ properties using more expressive planning objectives (beyond reachability).

DFA To track the reachability property expressed by ψ, we use a deterministic finite automaton (DFA). This automaton tracks whether a word satisfies the LTL body of φ, and thus operates on letters from alphabet $2^{AP \times \{\pi_1, \dots, \pi_{n+m}\}}$ (recall that the atoms in the LTL formula ψ have the form $a_{\pi_i} \in AP \times \{\pi_1, \dots, \pi_{n+m}\}$). Formally, a DFA is a tuple $\mathcal{A} = (Q, q_0, \delta, F)$ where Q is a finite set of states, $q_0 \in Q$ is an initial state, δ maps each pair $(q, q') \in Q \times Q$ to a Boolean formula over $AP \times \{\pi_1, \dots, \pi_{n+m}\}$, and $F \subseteq Q$ is a set of accepting states. When reading a letter $\sigma \in 2^{AP \times \{\pi_1, \dots, \pi_{n+m}\}}$, we can transition from q to q' iff the evaluation defined by σ (i.e., the assignment mapping $(a, \pi_i) \in AP \times \{\pi_1, \dots, \pi_{n+m}\}$ to true iff $(a, \pi_i) \in \sigma$) satisfies $\delta(q, q')$, written $\sigma \models \delta(q, q')$. As $\mathcal{A}$ is deterministic, we can assume that for every $q \in Q$ and every $\sigma \in 2^{AP \times \{\pi_1, \dots, \pi_{n+m}\}}$, there exists a unique $q' \in Q$ with $\sigma \models \delta(q, q')$. Each infinite word $u \in (2^{AP \times \{\pi_1, \dots, \pi_{n+m}\}})^\omega$ thus

generates a unique run of $\mathcal{A}$. Formally, we define $run_{\mathcal{A},u} \in Q^\omega$ as the unique run with $run_{\mathcal{A},u}(0) = q_0$, and for every $i \in \mathbb{N}$, $u(i) \models \delta_\psi(run_{\mathcal{A},u}(i), run_{\mathcal{A},u}(i+1))$. The word u is accepted by $\mathcal{A}$ if the unique run $run_{\mathcal{A},u}$ eventually reaches some state in F. In the following, we assume that $\mathcal{A}_\psi = (Q_\psi, q_{0,\psi}, \delta_\psi, F_\psi)$ is a DFA that accepts exactly those infinite words that satisfy ψ. We can assume, w.l.o.g., that all accepting states in F_ψ are sink states with a self-loop.

Planning Encoding We can now define a conformant planning problem associated with $\mathcal{T}, \varphi$.

Definition 3. *Define the conformant planning problem* $\mathcal{P}_{\mathcal{T},\varphi} := \big(S, s_0, G, \mathcal{O}\big)$*, where*

$$S := \big\{\langle l_1, \dots, l_{n+m}, q\rangle \mid l_1, \dots, l_{n+m} \in L, q \in Q_\psi\big\},$$
$$s_0 := \langle l_{init}, \dots, l_{init}, q_{0,\psi}\rangle,$$
$$G := \big\{\langle l_1, \dots, l_{n+m}, q\rangle \mid l_1, \dots, l_{n+m} \in L, q \in F_\psi\big\},$$
$$\mathcal{O} := \{\langle pre_{\vec{d}}, eff_{\vec{d}}\rangle \mid \vec{d} \in \mathbb{D}^n\}$$

and for each action $\vec{d} = (d_1, \dots, d_n) \in \mathbb{D}^n$ *we define* $pre_{\vec{d}} := S$ *and* $eff_{\vec{d}} : S \to S$ *by*

$$eff_{\vec{d}}\big(\langle l_1, \dots, l_{n+m}, q\rangle\big) :=$$
$$\Big\{\langle \kappa(l_1, d_1), \dots, \kappa(l_{n+m}, d_{n+m}), q'\rangle \mid d_{n+1}, \dots, d_{n+m} \in \mathbb{D} \wedge$$
$$\Big(\bigcup_{i=1}^{n+m} \big\{(a, \pi_i) \mid a \in \ell(l_i)\big\}\Big) \models \delta_\psi(q, q')\Big\}.$$

The idea behind our definition is that the planning instance will simulate $n + m$ paths (for the path variables $\pi_1, \dots, \pi_{n+m}$ used in φ) incrementally (guided by a plan). For this, each planning state $\langle l_1, \dots, l_{n+m}, q\rangle$ tracks the current location of each path $(l_1, \dots, l_{n+m})$ and the current state of $\mathcal{A}_\psi$ (thus tracking whether the simulated paths satisfy ψ). We start each π_i in the initial location l_{init} and start the run of $\mathcal{A}_\psi$ in the initial state $q_{0,\psi}$. The goal consists of all states where the automaton has reached one of $\mathcal{A}_\psi$'s accepting states. The crux is that the actions only define the behavior of the existentially quantified paths $\pi_1, \dots, \pi_n$, while nondeterminism determines the state sequence for universally quantified paths $(\pi_{n+1}, \dots, \pi_{n+m})$. Formally, each action in $\mathcal{P}_{\mathcal{T},\varphi}$ is a vector $\vec{d} = (d_1, \dots, d_n) \in \mathbb{D}^n$ of n directions (matching the number of existentially quantified path variables in φ). Each action can be applied in all states (i.e., $pre_{\vec{d}} = S$). When applying action $\vec{d}$ in a state $\langle l_1, \dots, l_{n+m}, q\rangle$, locations $l_1, \dots, l_n$ (i.e., the locations that correspond to existentially quantified paths) are updated by following the directions $d_1, \dots, d_n$ (i.e., the ith location is updated to $\kappa(l_i, d_i)$). In contrast, the locations of universally quantified paths $(l_{n+1}, \dots, l_{n+m})$ are updated non-deterministically, i.e., we consider all possible directions $d_{n+1}, \dots, d_{n+m}$ in the definition of $eff_{\vec{d}}$. Every plan can thus precisely determine the state sequence of existentially quantified paths, while universally quantified paths explore all possible paths.

In each step, we also update the state of $\mathcal{A}_\psi$ to track whether the state sequence simulated so far satisfies the LTL body ψ. For each $1 \le i \le n + m$, we collect all APs that hold in the current location and index them with π_i, thus obtaining a letter $\bigcup_{i=1}^{n+m} \big\{(a, \pi_i) \mid a \in \ell(l_i)\big\}$ in $2^{AP \times \{\pi_1, \dots, \pi_{n+m}\}}$. We then transition to the unique state $q' \in Q_\psi$, with $\bigcup_{i=1}^{n+m} \big\{(a, \pi_i) \mid a \in \ell(l_i)\big\} \models \delta_\psi(q, q')$.

Note that the size of $\mathcal{P}_{\mathcal{T},\varphi}$ is polynomial in the size of $\mathcal{T}$ and exponential in $n+m$ (as usual for self-compositions [1]). Concretely, $\mathcal{P}_{\mathcal{T},\varphi}$ has $\mathcal{O}(|S|^{n+m})$ many states.

Soundness and Completeness We can show that our encoding is sound, i.e., the existence of a conformant plan implies that the hyperproperty is satisfied. Moreover, our key contribution is the observation that *conformance* (i.e., sensorless behavior) is the key technical gadget that allows us to precisely express the semantics of HyperLTL, leading to completeness:

Theorem 1 (Soundness and Completeness)**.** *There exists a conformant plan for* $\mathcal{P}_{\mathcal{T},\varphi}$ *if and only if* $\mathcal{T} \models \varphi$.

Proof Sketch. For the first direction, assume that $\langle a_1, \dots, a_N\rangle$ is a conformant plan for $\mathcal{P}_{\mathcal{T},\varphi}$. We obtain n finite prefixes of length N by simulating the directions used in each action (recall that each action a_i is an n-tuple of directions). By extending these finite prefixes into infinite paths, we obtain concrete witness paths $p_1, \dots, p_n$ for $\pi_1, \dots, \pi_n$. Indeed, no matter what paths $p_{n+1}, \dots, p_{n+m}$ we consider for the m universally quantified paths in φ, there exists *some* execution of plan $\langle a_1, \dots, a_N\rangle$ that traverses the prefixes of $p_{n+1}, \dots, p_{n+m}$. As the plan is conformant, the paths $p_1, \dots, p_{n+m}$ together must thus visit an accepting state of $\mathcal{A}_\psi$, and thus satisfy ψ; $p_1, \dots, p_n$ are witnesses for $\pi_1, \dots, \pi_n$, so $\mathcal{T} \models \varphi$.

For the second direction, assume $\mathcal{T} \models \varphi$. As $\mathcal{T} \models \varphi$, there exist witness paths $p_1, \dots, p_n \in Paths(\mathcal{T})$ for the existentially quantified paths $\pi_1, \dots, \pi_n$. We choose some plan that – within $\mathcal{P}_{\mathcal{T},\varphi}$ – generates exactly the paths $p_1, \dots, p_n$, and claim that it is conformant. Indeed, every execution of this plan traverses exactly paths $p_1, \dots, p_n$ in the first n system copies, and traverses some paths $p_{n+1}, \dots, p_{n+m} \in Paths(\mathcal{T})$ in the remaining m system copies. As $p_1, \dots, p_n$ are witness traces for φ, all such combinations satisfy ψ and thus eventually reach an accepting state in $\mathcal{A}_\psi$. Every execution of the plan thus eventually visits a goal state (after a *bounded* number of steps since $\mathcal{T}$ is finite-state); the plan is conformant.

A full proof can be found in the full version [8]. □

Beyond Reachabilty So far, our encoding is limited to formulas where the LTL body expresses a reachability property, as reachability (of the goal) is the standard goal description used in (conformant) planning. The idea underlying our encoding can also be extended to handle arbitrary temporal requirements by using more expressive goal conditions. Our encoding could thus be easily extended to full HyperLTL, i.e., for every $\exists^*\forall^*$ HyperLTL formula, we can construct a planning problem (*with LTL-defined planning objective*) that admits a conformant plan iff the HyperLTL formula is satisfied. There exist many approaches that study non-deterministic planning under temporal objectives (e.g., LTL) [14, 13, 12, 33], so far mostly in a fully-observable setting.

5 Conformant Planning as a Hyperproperty

In the previous section, we showed that we can solve $\exists^*\forall^*$ HyperLTL model-checking by viewing it as a conformant planning problem. In this section, we show the reverse: conformant planning is a $\exists^*\forall^*$ hyperproperty. We, again, first work with an explicit state representation and lift our encoding to symbolic systems in Section 6. Let $\mathcal{P} = (S, s_0, G, \mathcal{O})$ be a fixed planning problem.

We will first construct a TS over atomic propositions $AP := \{act_a \mid a \in \mathcal{O}\} \cup \{goal\}$, whose paths precisely correspond to all possible plan executions in $\mathcal{P}$. The atomic propositions then allow us to **(1)** access the last action played (i.e., act_a should hold iff action a was the action used in the previous step), and **(2)** determine if the current state is a goal state (via AP $goal$). To record the last action, each location will be of the form (s, a), where $s \in S$ is the

current planning state, and $a \in \mathcal{O}$ is the action that was last played. Moreover, we add locations of the form $(\lightning, a)$ (where $a \in \mathcal{O}$), to indicate that action a was last played but was not applicable. As we do not need to uniquely identify transitions, we omit directions and directly view the transition function as a function $\kappa : L \to 2^L \setminus \{\emptyset\}$. Formally, we define the explicit-state TS $\mathcal{T}_\mathcal{P}$ as follows:

Definition 4. *Define the TS $\mathcal{T}_\mathcal{P} := (L, l_{init}, \kappa, \ell)$, where*

$$L := \big\{(s,a) \mid s \in S, a \in \mathcal{O}\big\} \cup \big\{(\lightning, a) \mid a \in \mathcal{O}\big\},$$

and $l_{init} := (s_0, a_0)$, where $a_0 \in \mathcal{O}$ is an arbitrary action. For the transition function we define

$$\kappa\big((s,a)\big) := \Big\{(s', a') \mid a' \in \mathcal{O} \wedge s \in \mathit{pre}_{a'} \wedge s' \in \mathit{eff}_{a'}(s)\Big\} \cup \Big\{(\lightning, a') \mid a' \in \mathcal{O} \wedge s \notin \mathit{pre}_{a'}\Big\}$$
$$\kappa\big((\lightning, a)\big) := \big\{(\lightning, a') \mid a' \in \mathcal{O}\big\}.$$

The labeling function is defined by:

$$\ell\big(s,a\big) := \begin{cases} \big\{act_a, goal\big\} & \text{if } s \in G \\ \big\{act_a\big\} & \text{otherwise} \end{cases}$$
$$\ell\big(\lightning, a\big) := \big\{act_a\big\}.$$

It is easy to see that every sequence of state-action pairs $(s_0,a_0)(s_1,a_1)\cdots(s_n,a_n) \in (S \times \mathcal{O})^*$ is the prefix of some path in $\mathit{Paths}(\mathcal{T}_\mathcal{P})$ iff $s_0, s_1, \ldots, s_n$ is a sequence of states in $\mathcal{P}$ under plan $\langle a_1, \ldots, a_n\rangle$. The TS $\mathcal{T}_\mathcal{P}$ thus generates all possible plan executions in $\mathcal{P}$, and records the last actions in each location. Note that once we reach an error location of the form $(\lightning, a)$ (by playing a non-applicable action), we can never reach a location where AP *goal* holds.

Definition 5. *Define the HyperLTL formula $\varphi_\mathcal{P}$ by*

$$\varphi_\mathcal{P} := \exists \pi_1.\, \forall \pi_2.\, \Diamond\, goal_{\pi_2} \vee \Diamond\Big(\bigvee_{a \in \mathcal{O}} (act_a)_{\pi_1} \not\Leftrightarrow (act_a)_{\pi_2}\Big).$$

That is, we require some plan (modeled as a sequence π_1 of state-action pairs) such that all paths π_2 with the same action sequence (i.e., all paths that follow the same plan as encoded in path π_1) eventually reach the goal. Phrased differently, any path π_2 must either reach the goal or eventually use a different action than used on π_1. Note that $\varphi_\mathcal{P}$ also implies that π_1 eventually reaches the goal; by instantiating π_2 with the same path used for π_1, the second disjunct will never be satisfied.

Theorem 2 (Soundness and Completeness)**.** *There exists a conformant plan for $\mathcal{P}$ if and only if $\mathcal{T}_\mathcal{P} \models \varphi_\mathcal{P}$.*

Proof Sketch. For the first direction, assume that $\mathcal{P}$ admits a conformant plan $\langle a_1, \ldots, a_N\rangle$. To show $\mathcal{T}_\mathcal{P} \models \varphi_\mathcal{P}$, we need to provide a witness trace for π_1. We create this witness path p_1 by choosing any path in $\mathcal{T}_\mathcal{P}$ where the first N actions (which are recorded in each location of $\mathcal{T}_\mathcal{P}$) are exactly $a_1, \ldots, a_N$. To show that p_1 is a witness path for π_1 consider any possible path $p_2 \in \mathit{Paths}(\mathcal{T})$ for the universally quantified π_2 in $\varphi_\mathcal{P}$. There are two options: Either the first N actions in p_2 are exactly $a_1, \ldots, a_N$, or at some position, the action differs. In the latter case, the second disjunct in the body of $\varphi_\mathcal{P}$ is satisfied. In the former case, the action sequence is exactly $\langle a_1, \ldots, a_N\rangle$, so, by construction of $\mathcal{T}_\mathcal{P}$, the state sequence is some execution in $\mathcal{P}$ under plan $\langle a_1, \ldots, a_N\rangle$. As $\langle a_1, \ldots, a_N\rangle$ is a conformant plan, this already implies that the state sequence visits a goal state, so the first disjunct in the body of $\varphi_\mathcal{P}$ is satisfied.

For the reverse direction, assume that $\mathcal{T}_\mathcal{P} \models \varphi_\mathcal{P}$, and let $p_1 \in \mathit{Paths}(\mathcal{T}_\mathcal{P})$ be a witness path for π_1. We can extract a conformant plan by projecting on the actions recorded in p_1. To show that this plan is conformant, consider an arbitrary execution under that plan. If paired with the sequence of actions, we obtain a path $p_2 \in \mathit{Paths}(\mathcal{T}_\mathcal{P})$ which we can use for the universally quantified π_2; As $\mathcal{T}_\mathcal{P} \models \varphi_\mathcal{P}$, and p_1 is a witness for π_1, $[\pi_1 \mapsto p_1, \pi_2 \mapsto p_2]$ satisfies the body of $\varphi_\mathcal{P}$. As the action sequence of p_1 and p_2 is the same, the second disjunct in $\varphi_\mathcal{P}$'s is never satisfied, so the first disjunct must hold. This already implies that p_2 visits a goal state, and, as $\mathcal{T}$ is finite-state, the length until a goal state is visited is bounded across all executions. As this holds for any execution of the plan, the plan is conformant.

A full proof can be found in the full version [8]. □

6 STRIPS Planning and Symbolic Systems

In the previous sections, we worked with an explicit-state representation of the planning domain and the transition system. In practice, planning problems are typically represented symbolically using (Boolean) variables (also called propositions or fluents) to describe the current state (e.g., STRIPS or PDDL). Converting such a planning problem to the explicit-state description used in Definition 1, results in an exponential blowup, making it infeasible in practice. Likewise, many systems used in HyperLTL verification are described symbolically (e.g., circuits or NuSMV models). In this section, we show that the ideas of the above encodings seamlessly apply to such symbolically-represented systems and planning problems, i.e., we can directly encode HyperLTL model-checking on symbolic systems as (conformant) STRIPS-style planning, and vice versa; *without* first obtaining an explicit-state representation.

6.1 STRIPS Conformant Planning

We consider a STRIPS-like non-deterministic planning domain:

Definition 6. *A (non-deterministic) STRIPS planning problem is a tuple $\mathcal{P} = (P, I, G, \mathcal{O})$, where P is a finite set of Boolean propositions (also called fluents), $I \subseteq P$ is a set of propositions (defining the initial state), $G \subseteq P$ is a set of propositions defining goal states, and $\mathcal{O}$ is a finite set of actions. Each action $a \in \mathcal{O}$ has the form $a = \langle \mathit{pre}_a, \mathit{eff}_a\rangle$, where $\mathit{pre}_a \subseteq P$ defines all propositions which must hold in order for the action to be applicable, and $\mathit{eff}_a = \{e_1, \ldots, e_k\}$ is a finite set of conditional effects with $e_i = \langle \mathit{con}, \mathit{add}, \mathit{del}\rangle$, where con is a Boolean formula over P (the condition), $\mathit{add} \subseteq P$ is the add list, and $\mathit{del} \subseteq P$ is the delete list.*

A state in $\mathcal{P}$ is a set $s \subseteq P$ defining which propositions are set to true. An action $a = \langle \mathit{pre}_a, \mathit{eff}_a\rangle$ can be applied in s if $\mathit{pre}_a \subseteq s$. When applying action a in state s, we non-deterministically execute one of the (applicable) conditional effects in eff_a. Here, a conditional effect $\langle \mathit{con}, \mathit{add}, \mathit{del}\rangle \in \mathit{eff}_a$ is applicable if $s \models \mathit{con}$, i.e., the Boolean formula con is satisfied by the assignment to propositions P defined by s. Given a state s and action a, we define $\mathit{apply}_a(s) \subseteq 2^P$ as the effect of applying a in s:

$$\Big\{(s \setminus \mathit{del}) \cup \mathit{add} \mid \langle \mathit{con}, \mathit{add}, \mathit{del}\rangle \in \mathit{eff}_a \wedge s \models \mathit{con}\Big\}.$$

As in Section 3.1, we extend this to plans. A plan is a sequence $\langle a_1, \ldots, a_n\rangle$ of actions. Given a set of states $T \subseteq 2^P$, we inductively define $[\![T, \langle a_1, \ldots, a_n\rangle]\!] \subseteq 2^P$ as the set of states reachable

after executing plan $\langle a_1, \dots, a_n \rangle$. For the empty plan $\langle\rangle$, we define $[\![T, \langle\rangle]\!] := T$. For a non-empty plan we define

$$[\![T, \langle a_1, \dots, a_n \rangle]\!] := [\![\bigcup_{s \in T} apply_{a_1}(s), \langle a_2, \dots, a_n \rangle]\!]$$

if for every $s \in T$, we have $pre_{a_1} \subseteq s$. As before, a plan $\langle a_1, \dots, a_n \rangle$ is a *conformant plan* if $[\![\{I\}, \langle a_1, \dots, a_n \rangle]\!]$ is defined and for every $s \in [\![\{I\}, \langle a_1, \dots, a_n \rangle]\!]$ we have $s \cap G \neq \emptyset$. That is, all executions of the plan, starting in the initial state I, result in goal states, i.e., states where at least one goal proposition holds. Our definition differs slightly from the one used by Hoffmann and Brafman [29] in that we non-deterministically pick one of the conditional effects, allowing us to model non-deterministic action effects. In contrast, Hoffmann and Brafman [29] consider deterministic actions but non-deterministic initial state(s). In our definition, we can always model deterministic effects by ensuring that at most one conditional effect is applicable.

6.2 Symbolic Transition Systems

Similarly, we can consider a symbolic description of transition systems. Here, we assume that the states are represented via Boolean variables, and each direction is assigned a list of guarded commands, which indirectly bounds the branching degree of the system to $|\mathbb{D}|$.

Definition 7. *A symbolic transition system (STS) is a tuple $\mathcal{T} = (X, v_{init}, \mathbb{D})$, where X is a finite set of Boolean variables, $v_{init} \subseteq X$ is an initial assignment to X, and $\mathbb{D}$ is a finite set of* directions. *Each direction $d \in \mathbb{D}$ has the form $d = (g, pos, neg)$ where g is a Boolean formula over X (called the guard), $pos \subseteq X$ is a set of variables which will be set to true, and $neg \subseteq X$ is a set of variables which will be set to false.*

A state is a variable evaluation $v \subseteq X$. A state v' is a successor of v, written $v \to_{\mathcal{T}} v'$, if there exists a $d = (g, pos, neg) \in \mathbb{D}$ such that $v \models g$ and $v' = (v \setminus neg) \cup pos$. That is, the guard to the direction is satisfied (interpreting a state $v \subseteq X$ as the obvious assignment $X \to \mathbb{B}$), all positive variables in pos are set to true, and all negative variables in neg are set to false (similar to the add and delete list in STRIPS planning).

As before, a path in $\mathcal{T}$ is an infinite sequence $p \in (2^X)^\omega$ such that **(1)** $p(0) = v_{init}$, and **(2)** for every $i \in \mathbb{N}$, $p(i) \to_{\mathcal{T}} p(i+1)$. We define $Paths(\mathcal{T}) \subseteq (2^X)^\omega$ as the set of all paths in $\mathcal{T}$. Note that our definition omits atomic propositions (APs) as we can directly view the Boolean state variables as APs. That is, we assume that the atomic formulas within the LTL body are of the form x_{π_i} where $x \in X$ and $\pi_i \in \{\pi_1, \dots, \pi_{n+m}\}$. Given a HyperLTL formula φ, we define $\mathcal{T} \models \varphi$ as expected.

6.3 Hyperproperty Model-Checking as Planning

We can now extend our encoding from Section 4 to symbolic systems and symbolic planning domains. Ultimately, given an STS $\mathcal{T}$ and HyperLTL formula φ, we want to construct a STRIPS planning problem (cf. Definition 6) that admits a conformant plan iff $\mathcal{T} \models \varphi$ (*without* first translating the STS to an explicit-state TS).

In the following, assume that $\mathcal{T} = (X, v_{init}, \mathbb{D})$ is the fixed STS and $\varphi = \exists \pi_1 \dots \exists \pi_n . \forall \pi_{n+1} \dots \forall \pi_{n+m} . \psi$ is a fixed $\exists^*\forall^*$ HyperLTL formula. As before, we assume that ψ – the LTL body of φ – expresses a reachability property. Let $\mathcal{A}_\psi = (Q_\psi, q_{0,\psi}, \delta_\psi, F_\psi)$ be a DFA over letters from $2^{X \times \{\pi_1, \dots, \pi_{n+m}\}}$ (recall that we use the Boolean variables in $\mathcal{T}$ as APs) that accepts exactly those infinite words that satisfy ψ.

Indexed Variables In our planning encoding, we maintain the current location of $n+m$ system copies. When using an explicit-state representation as in Definition 3, we could simply use an $(n+m)$-tuple of system locations. In our symbolic setting, the current location of a system is defined by a variable evaluation over X, so to track the $n+m$ system copies, we use an indexed set of variables. Formally, we will represent $n+m$ system locations within each planning state by using propositions from $X \times \{\pi_1, \dots, \pi_{n+m}\}$. Variables $\{(x, \pi_i) \mid x \in X\}$ then define the current location of the ith system copy. Given a set of variables $A \subseteq X$, and $\pi_i \in \{\pi_1, \dots, \pi_{n+m}\}$, we define $A_{\circ \pi_i} := \{(x, \pi_i) \mid x \in A\}$ as the indexed set of variables. Likewise, given a Boolean formula g over variables from X, define $g_{\circ \pi_i}$ as the formula over $X \times \{\pi_1, \dots, \pi_{n+m}\}$ where each variable x is replaced by (x, π_i).

Definition 8. *Define the STRIPS planning problem $\mathcal{P}_{\mathcal{T},\varphi}$ as $\mathcal{P}_{\mathcal{T},\varphi} := \big(P, I, G, \mathcal{O}\big)$, where*

$$\begin{aligned} P &:= (X \times \{\pi_1, \dots, \pi_{n+m}\}) \cup Q_\psi \\ I &:= \big\{(x, \pi_i) \mid x \in v_{init}, 1 \leq i \leq n+m\big\} \cup \{q_{0,\psi}\} \\ G &:= F_\psi \\ \mathcal{O} &:= \big\{\langle pre_{\vec{d}}, eff_{\vec{d}} \rangle \mid \vec{d} \in \mathbb{D}^n\big\} \end{aligned}$$

and for each action $\vec{d} \in \mathbb{D}^n$ we define $pre_{\vec{d}} := \emptyset$ and the conditional effects $eff_{\vec{d}}$ are defined by

$$eff_{\vec{d}} := \Big\{ e_{\vec{d} \uplus \vec{d'}, q, q'} \mid \vec{d'} \in \mathbb{D}^m, q, q' \in Q_\psi \Big\}.$$

For each direction vector $\vec{d} \uplus \vec{d'} = (d_1, \dots, d_{n+m}) \in \mathbb{D}^{n+m}$, where $d_i = (g_i, pos_i, neg_i)$, we define the conditional effect $e_{\vec{d} \uplus \vec{d'}, q, q'}$ as $e_{\vec{d} \uplus \vec{d'}, q, q'} := \langle con, add, del \rangle$, where

$$\begin{aligned} con &:= q \wedge \delta_\pi(q, q') \wedge \bigwedge_{i=1}^{n+m} (g_i)_{\circ \pi_i} \\ add &:= \{q'\} \cup \bigcup_{i=1}^{n+m} (pos_i)_{\circ \pi_i} \\ del &:= \{q\} \cup \bigcup_{i=1}^{n+m} (neg_i)_{\circ \pi_i}. \end{aligned}$$

The idea behind this encoding is similar to the explicit-state encoding in Definition 3: The action chosen in each step determines how the n existentially quantified systems are updated, while the m universally quantified systems are updated non-deterministically. Each location of $\mathcal{T}$ is defined by the Boolean variables in X, so our planning problem uses propositions of the form (x, π_i) defining the value of variable x in the system copy for π_i. Additionally, we track the current state of $\mathcal{A}_\psi$, by adding each state in Q_ψ as a proposition. Initially, a proposition (x, π_i) is set iff variable x is set to true in $\mathcal{T}$'s initial state. The goal then consists of all states where some proposition in F_ψ is set, i.e., where $\mathcal{A}_\psi$ is in some accepting state. Similar to the encoding in Definition 3, the actions consist of n-tuples of directions. The idea is that each action $\vec{d} = (d_1, \dots, d_n) \in \mathbb{D}^n$ defines how each of the n existentially quantified systems is updated, while the m universally quantified systems are updated non-deterministically. Once action $\vec{d}$ is fixed, the conditional effects in $eff_{\vec{d}}$ are executed non-deterministically (if applicable). Here, each conditional effect in $eff_{\vec{d}}$ has the form $e_{\vec{d} \uplus \vec{d'}, q, q'}$ where $\vec{d'} \in \mathbb{D}^m$ gives the direction for each of the m universally quantified systems (the conditional effects thus consider all possible updates to universally quantified

systems). At the same time, we consider all possible combinations q, q' to update the state of $\mathcal{A}_\psi$. Each effect thus determines directions for all universal copies and determines how $\mathcal{A}_\psi$ is updated. Of the conditional effects in $\mathit{eff}_{\vec{d}}$, only a few are actually applicable: Namely, those where the guard of all directions is satisfied and $\mathcal{A}_\psi$ actually moves from q to q'. The condition of each conditional effect ensures this: Effect $e_{\vec{d} \uplus \vec{d'}, q, q'}$ is only applicable if **(1)** q holds (i.e., we are currently in state q), **(2)** $\delta_\psi(q, q')$ (i.e., $\mathcal{A}_\psi$ moves from state q to q' in the current system states; Recall that $\delta_\psi(q, q')$ is a Boolean formula over $X \times \{\pi_1, \dots, \pi_{n+m}\}$ and all these variables are propositions in the planning problem), and **(3)** all the guards of all directions are satisfied. To express the latter condition, we assume that $\vec{d} \uplus \vec{d'} = (d_1, \dots, d_{n+m})$ is the direction vector for all system copies (obtained by merging the n directions fixed by the action and the m directions fixed in the effect), and g_i is the guard of the ith direction (recall that g_i is a Boolean formula over X). In our encoding, we track the current state of the ith system copy via the indexed variables $\{(x, \pi_i) \mid x \in X\}$, so, within guard g_i, we replace each variable x with (x, π_i) (i.e., $(g_i)_{\circ \pi_i}$). When applying the conditional effect $e_{\vec{d} \uplus \vec{d'}, q, q'}$, we move $\mathcal{A}_\psi$ from q to q', i.e., we add the proposition q' and delete the proposition q. At the same time, we update each system copy based on the chosen direction, i.e., the ith copy is updated based on direction d_i. Formally, this amounts to adding all variables set to true by d_i (i.e., adding $(\mathit{pos}_i)_{\circ \pi_i}$) and removing all variables set to false by d_i (i.e., deleting $(\mathit{neg}_i)_{\circ \pi_i}$).

The resulting (STRIPS-like) planning problem models the same behavior as the explicit-state construction from Definition 3. As a result, we get the following direct corollary of Theorem 1:

Corollary 3 (Soundness and Completeness)**.** *There exists a conformant plan for $\mathcal{P}_{\mathcal{T},\varphi}$ if and only if $\mathcal{T} \models \varphi$.*

Note that we can construct $\mathcal{P}_{\mathcal{T},\varphi}$ in polynomial time in the size $\mathcal{T}$. Concretely, the number of fluents is $\mathcal{O}((n+m) \cdot |X|)$, the number of actions is $\mathcal{O}(|\mathbb{D}|^n)$, and the overall number of conditional effects is $\mathcal{O}(|\mathbb{D}|^{n+m})$ (as usual for a self-compositions [1]).

6.4 Conformant Planning as a Hyperproperty

We now sketch how we can extend the encoding from Section 5 to STRIPS planning problems and symbolically transition systems. Given a STRIPS planning problem $\mathcal{P} = (P, I, G, \mathcal{O})$ (cf. Definition 6), we can construct an STS $\mathcal{T}_\mathcal{P}$ (cf. Definition 7) that has the same behavior as the explicit-state construction in Definition 4. To accomplish this, we track each proposition in P as a variable in $\mathcal{T}_\mathcal{P}$. Moreover, we add additional variables $\{\mathit{act}_a \mid a \in \mathcal{O}\}$ (used to record the last action that was played), *goal* (set whenever we reach a goal proposition from G), and $\lightning$ (set whenever we have previously played a non-applicable action, similar to the $(\lightning, a)$ states in Definition 4). The initial state of $\mathcal{T}_\mathcal{P}$ sets exactly those propositions contained in I (i.e., $\mathcal{P}$'s initial state). For each action $a = \langle \mathit{pre}_a, \mathit{eff}_a \rangle \in \mathcal{O}$ with k conditional effects $\mathit{eff}_a = \{e_1, \dots, e_k\}$, we add k directions $d_{(a,e_1)}, \dots, d_{(a,e_k)}$ and a special direction $d_{(\lightning,a)}$. The idea is that direction $d_{(a,e_i)}$ models the effect of applying the ith conditional effect of action a. The guard of direction $d_{(a,e_i)}$ ensures that **(1)** all propositions in pre_a are true (i.e., the precondition of action a is satisfied), **(2)** the condition of the conditional effect e_i is satisfied, and **(3)** variable $\lightning$ is currently set to false (i.e., we have, so far, not played a non-applicable action). When applying direction $d_{(a,e_i)}$, we change the propositions in P according to e_i's add and delete list (i.e., direction $d_{(a,e_i)}$ sets all variables in e_i's add list to true and all variables in e_i's delete list to false). Moreover, direction $d_{(a,e_i)}$ sets variable act_a to true and all variables $\mathit{act}_{a'}$ with $a' \neq a$ to false. The special direction $d_{(\lightning,a)}$ can only be applied if pre_a does not hold. If applied, $d_{(\lightning,a)}$ sets variables act_a and $\lightning$ to true. As in Definition 4, applying a non-applicable action thus results in a state where $\lightning$ is set to true, prohibiting the future use of directions of the form $d_{(a,e_i)}$. It is not hard to see that the resulting STS generates all possible executions of $\mathcal{P}$; for every action $a \in \mathcal{O}$ and every possible (non-deterministically chosen) conditional effect of a, some direction mirrors the same effect in $\mathcal{T}_\mathcal{P}$. The resulting STS $\mathcal{T}_\mathcal{P}$ thus models the same behavior as the explicit-state construction in Definition 4. As a direct corollary of Theorem 2, we get:

Corollary 4. *The planning problem $\mathcal{P}$ admits a conformant plan if and only if $\mathcal{T}_\mathcal{P} \models \varphi_\mathcal{P}$.*

Note that we can construct $\mathcal{T}_\mathcal{P}$ in linear time in the size of $\mathcal{P}$. It uses $\mathcal{O}(|P| + |\mathcal{O}|)$ variables, and the number of directions in $\mathcal{T}_\mathcal{P}$ equals the number of total conditional effects in $\mathcal{P}$.

Implementation We have implemented our symbolic encoding in a prototype tool that translates PDDL 2.1 planning problems to HyperLTL verification instances. Our results show that we translate benchmarks within seconds, indicating that our encoding maintains the core computational challenge. Current HyperLTL verification tools effectively explore all possible witness paths, leading to poor performance on the resulting instances. As we discuss in the next section, we believe that this emphasizes the importance of transferring solutions between both disciplines, and, e.g., study (heuristic) guided approach to HyperLTL model-checking.

7 Conclusion and Future Work

In this paper, we have shown that conformant plans and $\exists^*\forall^*$ hyperproperties are closely related. In particular, conformance is the missing technical link that aligns non-deterministic planning with $\exists^*\forall^*$ model-checking in a sound-and-*complete* (and bi-directional) way. We hope that this formal connection eventually leads to an improvement in solutions for both problems.

We empathize that we do not believe that our encodings – when applied naïvely – result in instances that are well-suited for tools in either domain. Instead, we view our results as a strong indicator that the fundamental *concepts* developed within either community can be customized to problems in the other community. For example, the use of heuristics in the verification of hyperproperties is an entirely unexplored, yet very fruitful, direction for future research. Current verification approaches to $\exists^*\forall^*$ explore all possible witness paths using, e.g., automata or QBF encodings. As our experiments in Section 6.4 confirm, such approaches are difficult to scale: even medium-sized conformant planning problems can, when translated into a HyperLTL verification problem, not be solved with current verification tools. This is unsurprising: The planning community has demonstrated that exhaustive (non-guided) searches scale poorly but can be improved drastically by employing a (heuristic) guided search. Section 4 demonstrates that the verification of quantifier alternations in HyperLTL formulas is essentially a planning problem, providing strong evidence that heuristic-guided exploration could lead to much better scalability; even for HyperLTL problems beyond the planning domain. Using the encoding from Section 6, we could directly translate a HyperLTL verification problem to a planning problem. However, we believe that it is much more efficient to directly integrate heuristics into HyperLTL-specific tools by, e.g., guiding the system exploration during bounded model-checking.

Acknowledgments This work was supported by the European Research Council (ERC) Grant HYPER (101055412), and by the German Research Foundation (DFG) as part of TRR 248 (389792660).

References

[1] G. Barthe, P. R. D'Argenio, and T. Rezk. Secure information flow by self-composition. In *Computer Security Foundations Workshop, CSFW 2004*, 2004. doi: 10.1109/CSFW.2004.17.

[2] R. Beutner and B. Finkbeiner. Prophecy variables for hyperproperty verification. In *Computer Security Foundations Symposium, CSF 2022*, 2022. doi: 10.1109/CSF54842.2022.9919658.

[3] R. Beutner and B. Finkbeiner. Software verification of hyperproperties beyond k-safety. In *International Conference on Computer Aided Verification, CAV 2022*, 2022. doi: 10.1007/978-3-031-13185-1_17.

[4] R. Beutner and B. Finkbeiner. AutoHyper: Explicit-state model checking for HyperLTL. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2023*, 2023. doi: 10.1007/978-3-031-30823-9_8.

[5] R. Beutner and B. Finkbeiner. Non-deterministic planning for hyperproperty verification. In *International Conference on Automated Planning and Scheduling, ICAPS 2024*, 2024. doi: 10.1609/ICAPS.V34I1.31457.

[6] R. Beutner and B. Finkbeiner. Multiplayer games with incomplete information for hyperproperty verification. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025*, 2025. doi: 10.5555/3709347.3743896.

[7] R. Beutner and B. Finkbeiner. On hyperproperty verification, quantifier alternations, and games under partial information. In *Formal Methods in Computer-Aided Design, FMCAD 2025*, 2025.

[8] R. Beutner and B. Finkbeiner. On conformant planning and model-checking of $\exists^*\forall^*$ hyperproperties. *CoRR*, 2025.

[9] B. Bonet. Conformant plans and beyond: Principles and complexity. *Artif. Intell.*, 2010. doi: 10.1016/J.ARTINT.2009.11.001.

[10] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *International Conference on Artificial Intelligence Planning Systems, AIPS 2000*, 2000.

[11] R. I. Brafman, G. Shani, and S. Zilberstein. Qualitative planning under partial observability in multi-agent domains. In *Conference on Artificial Intelligence, AAAI 2013*, 2013. doi: 10.1609/AAAI.V27I1.8643.

[12] D. Calvanese, G. D. Giacomo, and M. Y. Vardi. Reasoning about actions and planning in LTL action theories. In *International Conference on Principles and Knowledge Representation and Reasoning, KR 2002*, 2002.

[13] A. Camacho and S. A. McIlraith. Strong fully observable non-deterministic planning with LTL and LTLf goals. In *International Joint Conference on Artificial Intelligence, IJCAI 2019*, 2019. doi: 10.24963/IJCAI.2019/767.

[14] A. Camacho, E. Triantafillou, C. J. Muise, J. A. Baier, and S. A. McIlraith. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *Conference on Artificial Intelligence, AAAI 2017*, 2017. doi: 10.1609/AAAI.V31I1.11058.

[15] A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. *J. Artif. Intell. Res.*, 2000. doi: 10.1613/JAIR.774.

[16] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.*, 2003. doi: 10.1016/S0004-3702(02)00374-0.

[17] M. R. Clarkson and F. B. Schneider. Hyperproperties. In *Computer Security Foundations Symposium, CSF 2008*, 2008. doi: 10.1109/CSF.2008.7.

[18] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez. Temporal logics for hyperproperties. In *International Conference on Principles of Security and Trust, POST 2014*, 2014. doi: 10.1007/978-3-642-54792-8_15.

[19] N. Coenen, B. Finkbeiner, C. Sánchez, and L. Tentrup. Verifying hyperliveness. In *International Conference on Computer Aided Verification, CAV 2019*, 2019. doi: 10.1007/978-3-030-25540-4_7.

[20] A. Correnson and B. Finkbeiner. Coinductive proofs for temporal hyperliveness. *Proc. ACM Program. Lang.*, (POPL), 2025. doi: 10.1145/3704889.

[21] A. Correnson, T. Nießen, B. Finkbeiner, and G. Weissenbacher. Finding $\forall\exists$ hyperbugs using symbolic execution. *Proc. ACM Program. Lang.*, (OOPSLA2), 2024. doi: 10.1145/3689761.

[22] T. Dardinier and P. Müller. Hyper hoare logic: (dis-)proving program hyperproperties. *Proc. ACM Program. Lang.*, (PLDI), 2024. doi: 10.1145/3656437.

[23] S. Edelkamp and M. Helmert. MIPS: the model-checking integrated planning system. *AI Mag.*, 2001. doi: 10.1609/AIMAG.V22I3.1574.

[24] B. Finkbeiner. Logics and algorithms for hyperproperties. *ACM SIGLOG News*, 2023. doi: 10.1145/3610392.3610394.

[25] B. Finkbeiner, M. N. Rabe, and C. Sánchez. Algorithms for model checking HyperLTL and HyperCTL*. In *International Conference on Computer Aided Verification, CAV 2015*, 2015. doi: 10.1007/978-3-319-21690-4_3.

[26] F. Giunchiglia and P. Traverso. Planning as model checking. In *European Conference on Planning, ECP 1999*, 1999. doi: 10.1007/10720246_1.

[27] D. Gnad, J. Eisenhut, A. Lluch-Lafuente, and J. Hoffmann. Model checking ω-regular properties with decoupled search. In *International Conference on Computer Aided Verification, CAV 2021*, 2021. doi: 10.1007/978-3-030-81688-9_19.

[28] R. P. Goldman and M. S. Boddy. Expressive planning and explicit knowledge. In *International Conference on Artificial Intelligence Planning Systems, AIPS 1996*, 1996.

[29] J. Hoffmann and R. I. Brafman. Conformant planning via heuristic forward search: A new approach. *Artif. Intell.*, 2006. doi: 10.1016/J.ARTINT.2006.01.003.

[30] T. Hsu, C. Sánchez, and B. Bonakdarpour. Bounded model checking for hyperproperties. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2021*, 2021. doi: 10.1007/978-3-030-72016-2_6.

[31] H. A. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic and stochastic search. In *Conference on Artificial Intelligence, AAAI 1996*, 1996.

[32] J. McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Symposium on Research in Security and Privacy, SP 1994*, 1994. doi: 10.1109/RISP.1994.296590.

[33] F. Patrizi, N. Lipovetzky, G. D. Giacomo, and H. Geffner. Computing infinite plans for LTL goals using a classical planner. In *International Joint Conference on Artificial Intelligence, IJCAI 2011*, 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-334.

[34] M. N. Rabe. *A temporal logic approach to information-flow control*. PhD thesis, Saarland University, 2016.

[35] J. Rintanen. Constructing conditional plans by a theorem-prover. *J. Artif. Intell. Res.*, 1999. doi: 10.1613/JAIR.591.

[36] J. Rintanen. Asymptotically optimal encodings of conformant planning in QBF. In *Conference on Artificial Intelligence, AAAI 2007*, 2007.

[37] S. Winter and M. Zimmermann. Prophecies all the way: Game-based model-checking for HyperQPTL beyond $\forall^*\exists$. In *International Conference on Concurrency Theory, CONCUR 2025*, 2025. doi: 10.4230/LIPICS.CONCUR.2025.37.