Non-Deterministic Planning for Hyperproperty Verification

Raven Beutner, Bernd Finkbeiner

CISPA Helmholtz Center for Information Security, Germany

Abstract

Non-deterministic planning aims to find a policy that achieves a given objective in an environment where actions have uncertain effects, and the agent – potentially – only observes parts of the current state. Hyperproperties are properties that relate multiple paths of a system and can, e.g., capture security and information-flow policies. Popular logics for expressing temporal hyperproperties – such as HyperLTL – extend LTL by offering selective quantification over executions of a system. In this paper, we show that planning offers a powerful intermediate language for the automated verification of hyperproperties. Concretely, we present an algorithm that, given a HyperLTL verification problem, constructs a non-deterministic multi-agent planning instance (in the form of a QDec-POMDP) that, when admitting a plan, implies the satisfaction of the verification problem. We show that for large fragments of HyperLTL, the resulting planning instance corresponds to a classical, FOND, or POND planning problem. We implement our encoding in a prototype verification tool and report on encouraging experimental results.

1 Introduction

AI planning is the task of finding a policy (aka. plan) that ensures that a specified goal is reached. In this paper, we present an exciting new application of planning: the automated verification of *hyperproperties*.

Hyperproperties and HyperLTL. Hyperproperties generalize traditional trace properties by relating *multiple* executions of a system (Clarkson and Schneider 2008). A trace property – specified, e.g., in LTL – reasons about *individual* executions in isolation, which falls short in many applications. For example, assume we model the dynamics of a system as a transition system over atomic propositions $\{o, h, l\}$, and want to specify that the output o of the system only depends on the low-security input l and does not leak information about the secret input h. We cannot specify this as a trace property in LTL; we need to relate multiple executions to observe how different inputs impact the output. HyperLTL extends LTL with explicit quantification over executions (Clarkson et al. 2014), and allows for the specification of such a property. For example, we can express observational determinism (OD) (Zdancewic and Myers 2003) as the following HyperLTL formula:

$$\forall \pi.\, \forall \pi'.\, (l_\pi \leftrightarrow l_{\pi'}) \rightarrow \mathsf{G}(o_\pi \leftrightarrow o_{\pi'}) \qquad \text{(OD)}$$

This formula states that on any *pair* of executions π, π' with identical low-security input, the output is (globally) the same. In other words, the output of the system behaves deterministically w.r.t. the low-security input. For non-deterministic systems, OD is often too strict, as any given low-security input might lead to multiple outputs. A relaxed notation – called non-inference (NI) (McLean 1994) – can be expressed in HyperLTL as follows:

$$\forall \pi.\, \exists \pi'.\, \mathsf{G}\left((o_\pi \leftrightarrow o_{\pi'}) \wedge (l_\pi \leftrightarrow l_{\pi'}) \wedge \neg h_{\pi'}\right) \qquad \text{(NI)}$$

That is, for any execution π, there *exists* another execution π' that has the same low-security behavior (via propositions o and l), despite having a fixed "dummy" high-security input (in our case, we require that h is always false, i.e., never holds on π'). If NI holds, a low-security attacker cannot distinguish any high-security input from the dummy input.

HyperLTL Verification as Planning. Our goal is to automatically verify that a finite-state system $\mathcal{T}$ satisfies a HyperLTL formula φ. We introduce a novel verification approach that leverages the advanced methods developed within the planning community. Concretely, we present a reduction that soundly converts a HyperLTL verification problem into a planning problem. Depending on the HyperLTL formula, our encoding uses several advanced features supported by modern planning frameworks, such as uncertain action effects (non-determinism) (Cimatti et al. 2003), partial observations (Bertoli et al. 2006), and multiple agents. We show that – by carefully combining these features – we obtain a planning problem that is sound w.r.t. the HyperLTL semantics: every plan can be translated back into a validity witness for the original verification problem. As a consequence, our encoding allows us to utilize mature planning tools for the automated verification of HyperLTL properties. We implement our encoding as a prototype and report on encouraging results using off-the-shelf planners.

2 High-Level Overview

Before proceeding with a formal construction, we provide a high-level intuition of our encoding. In HyperLTL, we can quantify over the executions of a system (as seen informally

in OD and NI). The overarching idea in our encoding is to let the planning agent(s) control all *existentially quantified* executions, such that any valid plan directly corresponds to a witness for the existentially quantified executions.

Verification as Planning. As an example, assume we want to verify that OD does *not* hold on a given system $\mathcal{T}$, i.e., we want to find concrete executions π, π' that violate the body of OD. We can interpret this as a classical (single-agent) planning problem: each planning state maintains two locations in $\mathcal{T}$, one for π and one for π', and, in each step, the actions update the locations for π, π' by moving along the transitions in $\mathcal{T}$. The planning objective is to construct executions for π, π' that *violate* $(l_\pi \leftrightarrow l_{\pi'}) \rightarrow \mathsf{G}(o_\pi \leftrightarrow o_{\pi'})$. Any successful plan (i.e., sequence of transitions) then directly corresponds to concrete paths π, π' disproving OD.

Non-Deterministic Planning. Verification becomes more interesting when the HyperLTL formula contains quantifier alternations, such as NI. Following the above intuition, a plan should provide a concrete witness for (the existentially quantified) π', but – this time – we need to consider *all possible* executions for (the universally quantified) π. Our idea is that we can approximate this behavior by viewing it as a fully observable non-deterministic (FOND) planning problem; intuitively, a plan controls the behavior of π' while the behavior of π is non-deterministic. That is, each action determines a successor location for π' but also non-deterministically updates the location of π. The agent's objective is to ensure that the resulting paths π, π', together, satisfy $\mathsf{G}\left((o_\pi \leftrightarrow o_{\pi'}) \wedge (l_\pi \leftrightarrow l_{\pi'}) \wedge \neg h_{\pi'}\right)$ (the LTL body of NI). Any plan (which is now *conditional* on the non-deterministic outcomes) thus defines a concrete execution for π', depending on the concrete execution for π.

Planning Under Partial Observations. In NI, π' is quantified *after* π, so the action sequence that defines the behavior of π' can be based on the behavior of π. This changes when quantifiers *trail* existential quantification, e.g., in a formula of the form $\exists\pi.\,\forall\pi'$. Here, we follow the same idea as before but ensure that the actions controlling π are *independent* of the current location of π', i.e., the agent(s) act in a *partially observable* non-deterministic (POND) domain.

3 Related Work

Non-Deterministic Planning. Non-deterministic planning provides a powerful intermediate language that encompasses problems such as reactive synthesis (Camacho et al. 2018), controller synthesis in MDPs, epistemic planning (Engesser and Miller 2020), and generalized planning (Hu and Giacomo 2011). Consequently, many methods and tools have been developed (Pereira et al. 2022; Messa and Pereira 2023; Camacho et al. 2017; Geffner and Geffner 2018; Rodriguez et al. 2021; Muise, McIlraith, and Beck 2012; Kuter et al. 2008), with some also supporting planning in partially observable domains (Bertoli et al. 2006; Cimatti et al. 2003; Bonet and Geffner 2011).

HyperLTL Verification. Model checking of HyperLTL on finite-state transition systems is decidable, and existing complete algorithms utilize expensive automata complementations or inclusion checks (Finkbeiner, Rabe, and Sánchez 2015; Beutner and Finkbeiner 2023). There also exist cheaper (but incomplete) methods based, e.g., on a bounded model-checking (Hsu, Sánchez, and Bonakdarpour 2021). For $\forall^*\exists^*$ HyperLTL properties (i.e., properties where no universal quantified appears below an existential quantifier), our encoding is closely related to game-based (or, equivalently, simulation-based) approaches (Beutner and Finkbeiner 2022a,b; Hsu et al. 2023), which interpret verification as a game between universal and existential quantifiers. In fact, non-deterministic planning can be seen as a specialized form of turn-based games (Kissmann and Edelkamp 2009). Crucially, the size of an (explicit-state) game-based approach scales *exponentially* in the number of quantified executions (Beutner and Finkbeiner 2022a), making it impractical for larger instances. In contrast, the planning-based approach in this paper can describe the problem in a factored representation and let the planner determine how to best explore the state space. Moreover, – by utilizing partial observations – our planning-based encoding is applicable to HyperLTL formulas with arbitrary quantifier prefixes, not only $\forall^*\exists^*$ formulas.

4 Preliminaries

For a set X, we write X^+ for the set of finite non-empty sequences over X, and X^ω for the set of infinite sequences.

4.1 Non-Deterministic Planning

As a basic planning model, we use Qualitative Dec-POMDPs (QDec-POMDP), a general framework that encompasses multiple agents, non-deterministic effects, and partial observations (Brafman, Shani, and Zilberstein 2013).

Definition 1. *A QDec-POMDP is a tuple $\mathcal{G} = (I, S, S_0, \{A_i\}, \delta, \{\Omega_i\}, \{\omega_i\}, G)$, where I is a finite set of agents; S is a finite set of states; $S_0 \subseteq S$ is a set of initial states; for each $i \in I$, A_i is a finite set of actions, and we define $\vec{A} := \otimes_{i\in I} A_i$ as the set of joint actions; $\delta : S \times \vec{A} \to 2^S$ is a (non-deterministic) transition function; for each $i \in I$, Ω_i is a finite set of observations; $\omega_i : S \to \Omega_i$ defines i's local observation; and $G \subseteq S$ is a set of goal states.*

We write $\{a_i\} \in \vec{A}$ for the joint action where each agent $i \in I$ chooses action a_i. A *local policy* for an agent $i \in I$, is a conditional plan that picks an action based on the history of observations, i.e., a function $f_i : \Omega_i^+ \to A_i$. We can represent a policy f_i as an (infinite) tree of degree $|\Omega_i|$ where nodes are labeled with elements from A_i. A *joint policy* $\{f_i\}$ assigns each agent $i \in I$ a local policy f_i. A finite path $p \in S^+$ is *compatible with* $\{f_i\}$ if and only if **(1)** $p(0) \in S_0$ (i.e., the path starts in an initial state), and **(2)** for every $0 \leq j < |p| - 1$, $p(j+1) \in \delta(p(j), \{a_i\})$, where $a_i := f_i(\omega_i(p(0)) \cdots \omega_i(p(j)))$ for every $i \in I$. That is, in the jth step, we compute the joint action $\{a_i\}$, where each a_i is determined by policy f_i based on the past observations made by i on the prefix $p(0) \cdots p(j)$. We write $\mathit{Exec}(\{f_i\}) \subseteq S^+$ for the set of all $\{f_i\}$-compatible paths.

The objective of the agents is to reach a goal state in G. Following Cimatti et al. (2003), we distinguish between dif-

ferent levels of reachability. A policy is a *weak plan* if *some* $p \in \mathit{Exec}(\{f_i\})$ reaches a state in G, i.e., $\{f_i\}$ can reach the goal provided the non-determinism is resolved favorably. A policy is a *strong plan* if there exists an $N \in \mathbb{N}$ such that *every* $p \in \mathit{Exec}(\{f_i\})$ with $|p| \geq N$ reaches G, i.e., the goal is guaranteed to be reached, irrespective of non-deterministic outcomes. Finally, a policy is a *strong cyclic plan* if, for every $p \in \mathit{Exec}(\{f_i\})$, either p reaches a state in G or there exists some $p' \in \mathit{Exec}(\{f_i\})$ that extends p (i.e., p is a *prefix* of p') and p' reaches a state in G.[1]

4.2 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), which are tuples $\mathcal{T} = (L, l_{init}, \mathbb{D}, \kappa, \ell)$ where L is a finite set of locations (we use "locations" to distinguish them from the "states" in a planning domain), $l_{init} \in L$ is an initial location, $\mathbb{D}$ is a finite set of *directions*, $\kappa : L \times \mathbb{D} \to L$ is the transition function, and $\ell : L \to 2^{AP}$ labels each location with an evaluation of the APs. Note that we use explicit directions in order to uniquely identify successor locations; we can easily model any transition function $L \to (2^L \setminus \{\emptyset\})$ using sufficiently many directions. A path in $\mathcal{T}$ is an infinite sequence $p \in L^\omega$ such that **(1)** $p(0) = l_{init}$, and **(2)** for every $j \in \mathbb{N}$, there exists some $d \in \mathbb{D}$ s.t. $p(j+1) = \kappa(p(j), d)$. We define $\mathit{Paths}(\mathcal{T}) \subseteq L^\omega$ as the set of all paths in $\mathcal{T}$.

4.3 HyperLTL

As the basic specification language for hyperproperties, we use HyperLTL, an extension of LTL with explicit quantification over (execution) paths (Clarkson et al. 2014). Let $\mathcal{V} = \{\pi, \pi', \ldots\}$ be a set of *path variables*. HyperLTL formulas are generated by the following grammar

$$\psi := a_\pi \mid \psi \wedge \psi \mid \neg\psi \mid \mathsf{X}\,\psi \mid \psi\,\mathsf{U}\,\psi$$
$$\varphi := \mathbb{Q}\pi.\,\varphi \mid \psi$$

where $a \in AP$, $\pi \in \mathcal{V}$, and $\mathbb{Q} \in \{\forall, \exists\}$ is a quantifier. We use the usual derived boolean constants and connectives $\mathit{true}, \mathit{false}, \vee, \rightarrow, \leftrightarrow$, and the temporal operators *eventually* ($\mathsf{F}\,\psi := \mathit{true}\,\mathsf{U}\,\psi$), and *globally* ($\mathsf{G}\,\psi := \neg\,\mathsf{F}\,\neg\psi$).

Given a TS $\mathcal{T} = (L, l_{init}, \mathbb{D}, \kappa, \ell)$, we evaluate a HyperLTL formula in the context of a path assignment $\Pi : \mathcal{V} \rightharpoonup L^\omega$ (mapping path variables to paths) and $j \in \mathbb{N}$ as follows:

$$\begin{aligned}
&\Pi, j \models_\mathcal{T} a_\pi && \text{iff} && a \in \ell\big(\Pi(\pi)(j)\big)\\
&\Pi, j \models_\mathcal{T} \psi_1 \wedge \psi_2 && \text{iff} && \Pi, j \models_\mathcal{T} \psi_1 \text{ and } \Pi, j \models_\mathcal{T} \psi_2\\
&\Pi, j \models_\mathcal{T} \neg\psi && \text{iff} && \Pi, j \not\models_\mathcal{T} \psi\\
&\Pi, j \models_\mathcal{T} \mathsf{X}\,\psi && \text{iff} && \Pi, j+1 \models_\mathcal{T} \psi\\
&\Pi, j \models_\mathcal{T} \psi_1\,\mathsf{U}\,\psi_2 && \text{iff} && \exists k \geq j.\, \Pi, k \models_\mathcal{T} \psi_2 \text{ and}\\
& && && \quad \forall j \leq l < k.\, \Pi, l \models_\mathcal{T} \psi_1\\
&\Pi, j \models_\mathcal{T} \mathbb{Q}\pi.\,\varphi && \text{iff} && \mathbb{Q}p \in \mathit{Paths}(\mathcal{T}).\, \Pi[\pi \mapsto p], j \models_\mathcal{T} \varphi
\end{aligned}$$

[1] A strong cyclic plan is one that always *preserves the possibility of reaching a goal state*, i.e., at every point during the plan's execution, the non-determinism can be resolved favorably such that the goal is reached. Our definition expresses exactly this: either p already reaches G or some extension of p *can* reach the goal. This definition is equivalent to the one of Cimatti et al. (2003).

The atomic formula a_π holds whenever a holds in the current position j on the path bound to π (as given by ℓ). Boolean and temporal operators are evaluated as expected by updating the current evaluation position j, and quantification adds paths to Π. We refer to Finkbeiner (2023) for details. We say $\mathcal{T}$ satisfies φ, written $\mathcal{T} \models \varphi$, if $\emptyset, 0 \models_\mathcal{T} \varphi$, where $\emptyset$ denotes the path assignment with empty domain.

Example 1. *As an example, consider the transition system $\mathcal{T}$ in Figure 1a and the HyperLTL formula $\varphi := \forall\pi_1.\,\exists\pi_2.\,\mathsf{G}(a_{\pi_1} \leftrightarrow \mathsf{X}\,a_{\pi_2})$. The formula expresses that for any path π_1, there exists some path π_2 that mirrors π_1 with a one-step delay. It is easy to see that $\mathcal{T} \models \varphi$.*

5 Verification via Planning

We want to automatically verify that $\mathcal{T} \models \varphi$. To this end, we present a novel encoding into a planning problem, thus leveraging the extensive research and tool development within the planning community. As already outlined in Section 2, our main idea is to interpret existential quantification in φ as being resolved by an agent that picks transitions in $\mathcal{T}$ to construct a path. Throughout this section, we assume that $\mathcal{T} = (L, l_{init}, \mathbb{D}, \kappa, \ell)$ is the fixed TS and $\varphi = \mathbb{Q}_1\pi_1 \ldots \mathbb{Q}_n\pi_n.\,\psi$ is the fixed HyperLTL formula over path variables $\pi_1, \ldots, \pi_n$. We distinguish between temporal reachability (Section 5.1) and temporal safety (Section 5.2).

5.1 Encoding For Temporal Reachability

We first consider the case in which the LTL body of φ expresses a *temporal reachability property* (i.e., "something good happens eventually").

DFA. A deterministic finite automaton (DFA) over some alphabet Σ is a tuple $\mathcal{A} = (Q, q_0, \varrho, F)$ where Q is a finite set of states, $q_0 \in Q$ is an initial state, $\varrho : Q \times \Sigma \to Q$ is a deterministic transition function, and $F \subseteq Q$ is a set of marked states. An infinite word $u \in \Sigma^\omega$ is accepted by $\mathcal{A}$ if the unique run *eventually* reaches some state in F. We say φ is a *temporal reachability formula* if ψ (the LTL body of φ) is recognized by a DFA, i.e., some DFA $\mathcal{A}_\psi = (Q_\psi, q_{0,\psi}, \varrho_\psi, F_\psi)$ over alphabet $2^{AP \times \{\pi_1, \ldots, \pi_n\}}$ accepts exactly those infinite words that satisfy ψ (recall that the atoms in ψ have the form $a_{\pi_i} \in AP \times \{\pi_1, \ldots, \pi_n\}$).

Planning Encoding. We write $\mathcal{V}_\exists := \{\pi_i \mid \mathbb{Q}_i = \exists\}$ for the set of existentially quantified path variables in φ, and, analogously, $\mathcal{V}_\forall$ for the set of universally quantified ones.

Definition 2. *Define the QDec-POMDP $\mathcal{G}^{reach}_{\mathcal{T},\varphi}$ as*

$$\mathcal{G}^{reach}_{\mathcal{T},\varphi} := (I, S, S_0, \{A_i\}, \delta, \{\Omega_i\}, \{\omega_i\}, G),$$

where

$$\begin{aligned}
I &:= \big\{i \mid \pi_i \in \mathcal{V}_\exists\big\},\\
S &:= \big\{\langle l_1, \ldots, l_n, q\rangle \mid q \in Q_\psi \wedge l_1, \ldots, l_n \in L\big\},\\
S_0 &:= \big\{\langle l_{init}, \ldots, l_{init}, q_{0,\psi}\rangle\big\},\\
A_i &:= \mathbb{D},\\
\Omega_i &:= \big\{\langle l_1, \ldots, l_i\rangle \mid l_1, \ldots, l_i \in L\big\},\\
G &:= \big\{\langle l_1, \ldots, l_n, q\rangle \mid q \in F_\psi \wedge l_1, \ldots, l_n \in L\big\},
\end{aligned}$$

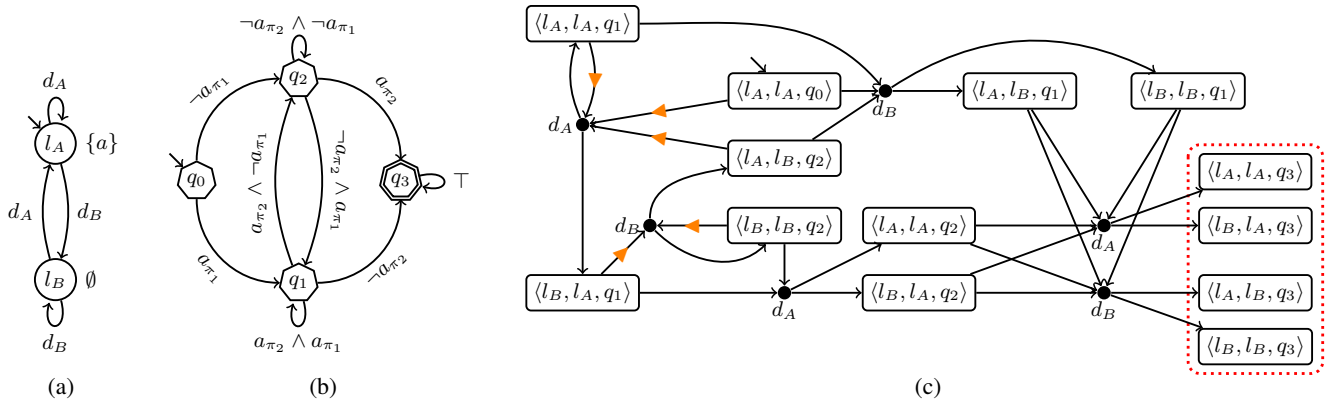

Figure 1: In Figure 1a, we depict a transition system $\mathcal{T}$ over $AP = \{a\}$ using directions $\mathbb{D} = \{d_A, d_B\}$. In Figure 1b, we give a DSA for the LTL body $\mathsf{G}(a_{\pi_1} \leftrightarrow \mathsf{X}\, a_{\pi_2})$ from Example 1, where we mark state q_3 as losing. In Figure 1c, we sketch the QDec-POMDP $\mathcal{G}^{safe}_{\mathcal{T},\varphi}$ constructed for the verification instance in Example 1 (see Example 2 for details).

the transition function δ is defined as

$$\delta\big(\langle l_1, \dots, l_n, q\rangle, \{d_i \in \mathbb{D}\}_{\pi_i \in \mathcal{V}_\exists}\big) := \Big\{\langle \kappa(l_1, d_1), \dots, \kappa(l_n, d_n), q'\rangle \mid \{d_i \in \mathbb{D}\}_{\pi_i \in \mathcal{V}_\forall} \wedge q' = \varrho_\psi\big(q, \textstyle\bigcup_{i=1}^{n} \{(a, \pi_i) \mid a \in \ell(l_i)\}\big)\Big\},$$

and the observation functions $\{\omega_i\}$ are defined as

$$\omega_i(\langle l_1, \dots, l_n, q\rangle) := \langle l_1, \dots, l_i\rangle.$$

Let us step through this definition. As already hinted in Section 2, we add one agent i for each existentially qualified path variable $\pi_i \in \mathcal{V}_\exists$. Each state has the form $\langle l_1, \dots, l_n, q\rangle$ and tracks a current location for each of the paths (where $l_i \in L$ is the current location for path π_i), and the current state of $\mathcal{A}_\psi$. Intuitively, the planning problem simulates $\pi_1, \dots, \pi_n$ by keeping track of their current location $(l_1, \dots, l_n)$, and letting the actions chosen by the agents (for existentially quantified paths) or the non-determinism (for universally quantified paths) fix the next location. 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 actions of each agent then directly correspond to directions in $\mathcal{T}$. When given a joint action $\{d_i\}_{\pi_i \in \mathcal{V}_\exists}$ (i.e., a direction for each existentially quantified path), the transition function considers *all possible* directions for universally quantified paths ($\{d_i\}_{\pi_i \in \mathcal{V}_\forall}$) and updates each location l_i based on the direction d_i. Existentially quantified paths thus follow the direction selected by the respective agent, and universally quantified ones follow a non-deterministically chosen direction. In each step, we also update the state of $\mathcal{A}_\psi$: For each $1 \leq i \leq n$, we collect all APs that hold in the current location ($\ell(l_i)$) and index them with π_i, thus obtaining a letter in $2^{AP \times \{\pi_1, \dots, \pi_n\}}$ which we feed to the transition function of $\mathcal{A}_\psi$. As argued in Section 2, each agent i controlling $\pi_i \in \mathcal{V}_\exists$ may only observe the paths $\pi_1, \dots, \pi_i$ quantified *before* π_i, so the observation set Ω_i of agent i consist exactly of tuples of the form $\langle l_1, \dots, l_i\rangle$ and the observation function ω_i projects each state to the observable locations. Lastly, the goal consists of all states in which the automaton has reached one of $\mathcal{A}_\psi$'s marked states.

Theorem 1. *Assume φ is a temporal reachability formula. If $\mathcal{G}^{reach}_{\mathcal{T},\varphi}$ admits a strong plan, then $\mathcal{T} \models \varphi$.*

Proof Sketch. We can use the policies in a strong plan for $\mathcal{G}^{reach}_{\mathcal{T},\varphi}$ to construct *Skolem functions* for existentially quantified paths in φ. The full proof is provided in the full version (Beutner and Finkbeiner 2024). □

Remark 1. *In the HyperLTL semantics, an existentially quantified path π_i is chosen based on* all *paths quantified before π_i. In our encoding, we abstract this selection with a step-wise action selection, so the agents can construct existentially quantified paths based only on the* prefixes *of previously quantified paths. This lack of information leads to incompleteness, i.e., in some cases $\mathcal{T} \models \varphi$, but $\mathcal{G}^{reach}_{\mathcal{T},\varphi}$ does* not *admit a strong plan. We can counteract this incompleteness by providing the agents with information about the future behavior of universally quantified executions using prophecies (Beutner and Finkbeiner 2022a).*

5.2 Encoding for Temporal Safety

We can also handle the case in which the LTL body of φ denotes a *temporal safety property*.

DSA. A deterministic safety automaton (DSA) over alphabet Σ is a DFA-like tuple $\mathcal{A} = (Q, q_0, \varrho, F)$. An infinite word $u \in \Sigma^\omega$ is accepted by $\mathcal{A}$ if the unique run *never* visits a state in F. We say φ is a *temporal safety formula* if ψ is recognized by a DSA $\mathcal{A}_\psi = (Q_\psi, q_{0,\psi}, \varrho_\psi, F_\psi)$. For example, the NI formula in Section 1 and the HyperLTL formula from Example 1 are temporal safety formulas.

Planning Encoding. Different from reachability properties, safety properties reason about infinite executions (and not only finite prefixes thereof). To encode this as a planning problem, we add special sink states s_{win} and s_{lose}, and mark s_{win} as the unique goal state. From any state $\langle l_1, \dots, l_n, q\rangle$ where $q \in F_\psi$, we then deterministically move to s_{lose}. From any state $\langle l_1, \dots, l_n, q\rangle$ where $q \notin F_\psi$, we extend the transitions from Definition 2 with an additional non-deterministic transition to s_{win}. The agents can thus never

ensure a visit to s_{win}, but a strong *cyclic* plan guarantees that we never visit a losing state in F_ψ. We denote the resulting QDec-POMPD with $\mathcal{G}^{safe}_{\mathcal{T},\varphi}$; a full description can be found in the full version (Beutner and Finkbeiner 2024).

Theorem 2. *Assume φ is a temporal safety formula. If $\mathcal{G}^{safe}_{\mathcal{T},\varphi}$ admits a strong cyclic plan, then $\mathcal{T} \models \varphi$.*

Example 2. *We consider the verification instance $(\mathcal{T}, \varphi)$ from Example 1. In Figure 1b, we depict a DSA for the LTL body of φ. In Figure 1c, we sketch the resulting QDec-POMDP $\mathcal{G}^{safe}_{\mathcal{T},\varphi}$. Each action from $\mathbb{D} = \{d_A, d_B\}$ for agent 2 (controlling the existentially quantified path π_2) non-deterministically leads to two successor states, which we visualize using immediate decision nodes labeled by directions. For example, in the initial state $\langle l_A, l_A, q_0\rangle$, the action (aka. direction) d_A non-deterministically leads to states $\langle l_A, l_A, q_1\rangle$ and $\langle l_B, l_A, q_1\rangle$. For simplicity, we omit s_{win} and s_{lose}: All states where the automaton has reached the losing state q_3 (surrounded by the red dashed box) transition to s_{lose}, and all other states have an additional non-deterministic transition to s_{win}. The QDec-POMDP $\mathcal{G}^{safe}_{\mathcal{T},\varphi}$ admits a strong cyclic plan by always following the action marked by the orange arrow, proving that $\mathcal{T} \models \varphi$.*

5.3 Encoding for Full HyperLTL

Our construction can also be extended to handle *full* HyperLTL by reducing to planning problems with temporal goals specified in LTL (Camacho et al. 2017; Camacho and McIlraith 2019). In this paper, we restrict our construction to the case of reachability and safety properties as **(1)** this suffices for almost all properties of interest, and **(2)** it allows us to employ the multiplicity of automated planners that yield strong (cyclic) plans for non-temporal objectives.

5.4 Factored Representation

In our construction, we used an explicit-state (flat) representation of the problem with $|L|^n \cdot |Q_\psi|$ many states. In practice, many planning formats (e.g., STRIPS, PDDL, SAS) allow for a *factored* description of the state space, using roughly $n \cdot |L| + |Q_\psi|$ many fluents that track the current location of each path *individually*. For example, the QDec-POMDP from Figure 1c can be represented compactly by tracking the locations of π_1 and π_2 individually. The possibility of using a factored representation is a core motivation for using planning tools for HyperLTL verification.

5.5 Classical, FOND, and POND Planning

In general, our encoding yields a planning problem that combines multiple agents, non-determinism, and partial observations. In many situations, however, the resulting problem does not require all these features: **(1)** For $\exists^*$ properties, the planning problem is classical, i.e., it consists of a single agent (controlling all paths), deterministic actions, and full information. **(2)** For $\forall^*\exists^*$ properties (e.g., NI), the problem involves a single agent (controlling all existentially quantified paths) acting under full information (FOND planning). **(3)** For $\forall^*\exists^*\forall^*$ properties, the problem involves a single agent acting under partial observations (POND planning).

			∃∃		∀∃	
Model	**Size PG**	**Size PDDL**	$t_{\texttt{PG}}$	$t_{\texttt{HyPlan}}$	$t_{\texttt{PG}}$	$t_{\texttt{HyPlan}}$
BAKERY3	31016.4	8.0/75.7	4.8	**1.2**	4.5	**0.9**
BAKERY5	614.6	7.7/19.0	0.6	**0.5**	**0.7**	1.1
MUTATION	1807.5	8.5/9.8	0.7	**0.3**	0.6	**0.4**
NI_C	37.5	7.7/13.5	**0.2**	0.3	0.5	**0.4**
NI_I	948.3	8.5/104.3	0.6	**0.4**	**0.7**	1.3
NRP_C	688.3	7.7/23.0	**0.5**	**0.5**	0.5	**0.4**
NRP_I	1018.6	7.7/22.2	0.5	**0.3**	**0.4**	**0.4**
SNARK_CON	105854.7	7.7/192.1	19.2	**6.9**	21.0	**5.1**
SNARK_SEQ	17415.6	8.0/32.4	3.3	**0.6**	5.6	**2.4**

Table 1: We compare `HyPlan` with a PG-based encoding. We list the size of the PG, the number of actions/objects in `HyPlan`'s PDDL encoding, and the verification times in seconds (averaged over multiple runs).

6 Implementation and Experiments

We have implemented our encoding for $\forall^*\exists^*$ HyperLTL formulas in a prototype called `HyPlan`. Our tool produces FOND planning instance in an extension of PDDL, featuring `(oneof p1 ... pn)` expressions in action effects; A format widely supported by many FOND planners. We compare `HyPlan` against the parity-game (PG) based encoding for $\forall^*\exists^*$ properties (Beutner and Finkbeiner 2022a). For our experiments, we collect the 10 NuSMV models from Hsu, Sánchez, and Bonakdarpour (2021) and generate random formulas of the form $\exists\pi.\,\exists\pi'.\,\mathsf{F}\,\psi$ and $\forall\pi.\,\exists\pi'.\,\mathsf{G}\,\psi$ where ψ is a temporal-operator-free formula. As remarked in Section 5.5, for the $\exists\exists$ properties, `HyPlan` produces classical planning problems, which we solve using `Scorpion` (Seipp and Helmert 2018). The $\forall\exists$ properties yield FOND planning problems, which we solve using `MyND` (Mattmüller et al. 2010). In Table 1, we report the average size of the encodings, as well as the time taken for the $\exists\exists$ and $\forall\exists$ properties. As noted in Section 3, the size of the PG is exponential in the number of paths, whereas the PDDL description is *factored* and delegates the exploration to the planner. Consequently, we observe that off-the-shelf planners perform competitively compared to explicit-state PG solvers.

7 Conclusion

We have presented a novel application of non-deterministic planning: the verification of hyperproperties. Our encoding is applicable to HyperLTL formulas with arbitrary quantifier prefixes, and our preliminary experiments show that off-the-shelf planners constitute a competitive alternative to explicit-state games. Moreover, further development of non-deterministic planners (for which our work provides additional incentives) directly improves our verification pipeline.

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

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2006. Strong planning under partial observability. *Artif. Intell.*

Beutner, R.; and Finkbeiner, B. 2022a. Prophecy Variables for Hyperproperty Verification. In *Computer Security Foundations Symposium, CSF 2022*.

Beutner, R.; and Finkbeiner, B. 2022b. Software Verification of Hyperproperties Beyond k-Safety. In *International Conference on Computer Aided Verification, CAV 2022*.

Beutner, R.; and Finkbeiner, B. 2023. AutoHyper: Explicit-State Model Checking for HyperLTL. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2023*.

Beutner, R.; and Finkbeiner, B. 2024. Non-Deterministic Planning for Hyperproperty Verification. *CoRR*.

Bonet, B.; and Geffner, H. 2011. Planning under Partial Observability by Classical Replanning: Theory and Experiments. In *International Joint Conference on Artificial Intelligence, IJCAI 2011*.

Brafman, R. I.; Shani, G.; and Zilberstein, S. 2013. Qualitative Planning under Partial Observability in Multi-Agent Domains. In *Conference on Artificial Intelligence, AAAI 2013*.

Camacho, A.; Baier, J. A.; Muise, C. J.; and McIlraith, S. A. 2018. Finite LTL Synthesis as Planning. In *International Conference on Automated Planning and Scheduling, ICAPS 2018*.

Camacho, A.; and McIlraith, S. A. 2019. Strong Fully Observable Non-Deterministic Planning with LTL and LTLf Goals. In *International Joint Conference on Artificial Intelligence, IJCAI 2019*.

Camacho, A.; Triantafillou, E.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2017. Non-Deterministic Planning with Temporally Extended Goals: LTL over Finite and Infinite Traces. In *Conference on Artificial Intelligence, AAAI 2017*.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.*

Clarkson, M. R.; Finkbeiner, B.; Koleini, M.; Micinski, K. K.; Rabe, M. N.; and Sánchez, C. 2014. Temporal Logics for Hyperproperties. In *International Conference on Principles of Security and Trust, POST 2014*.

Clarkson, M. R.; and Schneider, F. B. 2008. Hyperproperties. In *Computer Security Foundations Symposium, CSF 2008*.

Engesser, T.; and Miller, T. 2020. Implicit Coordination Using FOND Planning. In *Conference on Artificial Intelligence, AAAI 2020*.

Finkbeiner, B. 2023. Logics and Algorithms for Hyperproperties. *ACM SIGLOG News*.

Finkbeiner, B.; Rabe, M. N.; and Sánchez, C. 2015. Algorithms for Model Checking HyperLTL and HyperCTL*. In *International Conference on Computer Aided Verification, CAV 2015*.

Geffner, T.; and Geffner, H. 2018. Compact Policies for Fully Observable Non-Deterministic Planning as SAT. In *International Conference on Automated Planning and Scheduling, ICAPS 2018*.

Hsu, T.; Sánchez, C.; and Bonakdarpour, B. 2021. Bounded Model Checking for Hyperproperties. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2021*.

Hsu, T.; Sánchez, C.; Sheinvald, S.; and Bonakdarpour, B. 2023. Efficient Loop Conditions for Bounded Model Checking Hyperproperties. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2023*.

Hu, Y.; and Giacomo, G. D. 2011. Generalized Planning: Synthesizing Plans that Work for Multiple Environments. In *International Joint Conference on Artificial Intelligence, IJCAI 2011*.

Kissmann, P.; and Edelkamp, S. 2009. Solving Fully-Observable Non-deterministic Planning Problems via Translation into a General Game. In *Annual German Conference on AI, KI 2009*.

Kuter, U.; Nau, D. S.; Reisner, E.; and Goldman, R. P. 2008. Using Classical Planners to Solve Nondeterministic Planning Problems. In *International Conference on Automated Planning and Scheduling, ICAPS 2008*.

Mattmüller, R.; Ortlieb, M.; Helmert, M.; and Bercher, P. 2010. Pattern Database Heuristics for Fully Observable Nondeterministic Planning. In *International Conference on Automated Planning and Scheduling, ICAPS 2010*.

McLean, J. 1994. A general theory of composition for trace sets closed under selective interleaving functions. In *Symposium on Research in Security and Privacy, SP 1994*.

Messa, F.; and Pereira, A. G. 2023. A Best-First Search Algorithm for FOND Planning and Heuristic Functions to Optimize Decompressed Solution Size. In *International Conference on Automated Planning and Scheduling, ICAPS 2023*.

Muise, C. J.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-Deterministic Planning by Exploiting State Relevance. In *International Conference on Automated Planning and Scheduling, ICAPS 2012*.

Pereira, R. F.; Pereira, A. G.; Messa, F.; and Giacomo, G. D. 2022. Iterative Depth-First Search for FOND Planning. In *International Conference on Automated Planning and Scheduling, ICAPS 2022*.

Rodriguez, I. D.; Bonet, B.; Sardiña, S.; and Geffner, H. 2021. Flexible FOND Planning with Explicit Fairness Assumptions. In *International Conference on Automated Planning and Scheduling, ICAPS 2021*.

Seipp, J.; and Helmert, M. 2018. Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning. *J. Artif. Intell. Res.*

Zdancewic, S.; and Myers, A. C. 2003. Observational Determinism for Concurrent Program Security. In *Computer Security Foundations Workshop, CSFW 2003*.