The Hierarchy of Hyperlogics

Norine Coenen, Bernd Finkbeiner, Christopher Hahn and Jana Hofmann
Reactive Systems Group, Saarland University
Saarbrücken, Germany
Email: {lastname}@react.uni-saarland.de

***Abstract*—Hyperproperties, which generalize trace properties by relating multiple traces, are widely studied in information-flow security. Recently, a number of logics for hyperproperties have been proposed, and there is a need to understand their decidability and relative expressiveness. The new logics have been obtained from standard logics with two principal extensions: temporal logics, like LTL and CTL*, have been generalized to hyperproperties by adding variables for traces or paths. First-order and second-order logics, like monadic first-order logic of order and MSO, have been extended with the equal-level predicate. We study the impact of the two extensions across the spectrum of linear-time and branching-time logics, in particular for logics with quantification over propositions. The resulting hierarchy of hyperlogics differs significantly from the classical hierarchy, suggesting that the equal-level predicate adds more expressiveness than trace and path variables. Within the hierarchy of hyperlogics, we identify new boundaries on the decidability of the satisfiability problem. Specifically, we show that while HyperQPTL and HyperCTL* are both undecidable in general, formulas within their $\exists^*\forall^*$ fragments are decidable.**

1. Introduction

Temporal logics are classified into linear-time and branching-time logics: While *linear-time* temporal logics like LTL [1] describe properties of individual traces, *branching-time* temporal logics like CTL* [2] describe properties of computation trees, where the branches can be inspected by quantifying existentially or universally over paths. *Hyperlogics* add a second, orthogonal, dimension to this classification [3]: while the standard temporal logics only refer to a *single* trace or path at a time, the temporal hyperlogics HyperLTL and HyperCTL* relate *multiple* traces or paths to each other [4]. This makes it possible to express information-flow properties such as noninterference [5] and observational determinism [6]. Technically, the temporal hyperlogics extend the standard logics with trace or path variables. By quantifying over multiple variables, the formula can refer to several traces or paths at the same time. For example, the HyperLTL formula

$$\forall\pi.\forall\pi'.\ \square \bigwedge_{a\in AP} a_\pi \leftrightarrow a_{\pi'} \tag{1}$$

expresses that *all pairs* of traces must agree on the values of the atomic propositions (given as a set AP) at all times.

A different method for the construction of hyperlogics has been introduced for first-order and second-order logics such as monadic first-order logic of order (FO[$<$]) and full monadic second-order logic (MSO). The extension consists of adding the *equal-level predicate* E (cf. [7], [8]), which relates the same time points on different traces. The HyperLTL formula (1), for example, is equivalent to the FO[$<, E$] formula

$$\forall x.\forall y.\ E(x,y) \rightarrow \bigwedge_{a\in AP} (P_a(x) \leftrightarrow P_a(y)).$$

It is, so far, poorly understood how these two extensions compare in terms of expressiveness. A natural point of reference is Kamp's seminal theorem [9], which states (in the formulation of Gabbay et al. [10]) that LTL is expressively equivalent to FO[$<$]. However, the potential analogue of Kamp's theorem for hyperlogics, that HyperLTL might be equivalent to FO[$<, E$], is known *not* to be true [8].

In this paper, we initiate a comprehensive study of the spectrum of hyperlogics, guided by the known results for the standard logics (Figure 1a and Figure 1b). In addition to the equivalence of LTL and FO[$<$] established by Kamp's theorem, it is known that quantified propositional temporal logic (QPTL) [11] and monadic second-order logic of one successor (S1S) are expressively equivalent [12]. Moreover, previous work [13] showed that CTL* and monadic path logic (MPL) [14], as well as quantified computation tree logic (QCTL*) [15] and MSO are expressively equivalent [16].

Figures 1c and 1d show the results for the linear and branching-time hyperlogics, respectively. For linear time, the most striking difference to the hierarchy of the standard logics is that S1S is no longer equivalent to QPTL when lifted to hyperlogics: S1S[E] is *strictly more expressive* than HyperQPTL, which, in turn, is strictly more expressive than FO[$<, E$]. For branching time, we have that HyperQCTL* is still expressively equivalent to MSO[E]. However, MPL,

This work was partially supported by the German Research Foundation (DFG) as part of the Collaborative Research Center "Methods and Tools for Understanding and Controlling Privacy" (CRC 1223) and the Collaborative Research Center "Foundations of Perspicuous Software Systems" (TRR 248, 389792660), and by the European Research Council (ERC) Grant OSARES (No. 683300).

978-1-7281-3608-0/19/$31.00

which is equivalent to CTL* in the standard hierarchy (Figure 1b), falls strictly *between* HyperCTL* and HyperQCTL* when equipped with the equal-level predicate (MPL[E]).

The choice of logics considered in our expressiveness study is motivated by practical interest in certain hyperproperties. Branching-time hyperlogics, for example, are useful to state that a system can generate secret information [3], e.g., there is, at some point, a branching into observably equivalent paths that differ in the values of a secret:

$$\exists \pi. \Diamond \exists \pi'. \ (\Box \bigwedge_{a \in P} a_\pi \leftrightarrow a_{\pi'}) \wedge (\bigcirc \bigvee_{a \in S} a_\pi \nleftrightarrow a_{\pi'}),$$

where the set of atomic propositions divides into the two disjoint sets of publicly observable propositions P and secret propositions S.

An example for the usefulness of quantification over atomic propositions is the extension of LTL to QPTL, which improves the expressiveness from non-counting properties to general ω-regular properties. In HyperQPTL, the extension of HyperLTL with quantification over atomic propositions, it additionally becomes possible to express properties like *promptness* [17], which states that there is a bound, common for all traces, on the number of steps until an eventuality a is satisfied. Promptness can be expressed in HyperQPTL by using a quantified atomic proposition q, such that the first occurrence of q represents the bound:

$$\exists q. \forall \pi. \ \Diamond q \wedge (\neg q \, \mathcal{U} \, a_\pi).$$

Comparing the impact of adding quantification over trace and path variables to temporal logics vs. adding the equal-level predicate to first-order and second-order logics, our analysis indicates that the equal-level predicate adds more expressive power to a first-order (or second-order) logic than trace and path quantification adds to a temporal logic. Regarding the decidability of the logics, we show that, in the linear-time hierarchy, the boundary of the decidable hyperproperties can be characterized as the $\exists^*\forall^*$ HyperQPTL fragment. (The $\exists^*\forall^*$ fragment consists of all formulas with an arbitrary number of existential trace quantifiers followed by an arbitrary number of universal trace quantifiers.) In the branching-time hierarchy, we show that $\exists^*\forall^*$ HyperCTL* formulas can still be decided. The decidability results are summarized in Table 1.

The remainder of this paper is structured as follows. Section 2 covers preliminaries, including the basic temporal logics LTL and CTL*. In Section 3, we provide proofs for the expressiveness results from Figure 1. In Section 4, we show new decidability bounds in the linear-time and branching-time hyperlogic hierarchies as indicated in Table 1.

2. Preliminaries

We formally define traces and trees, and introduce some basic notation for trace and path manipulation. We define the temporal logics LTL and CTL*, which serve as the foundation for the extensions studied in this paper.

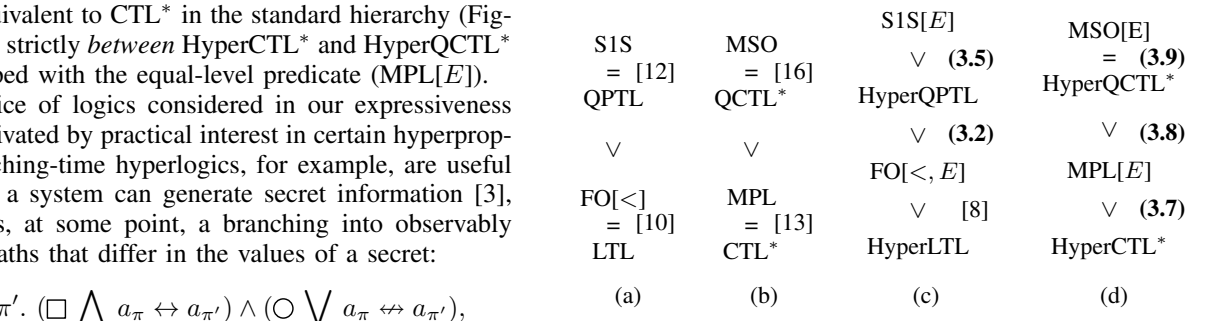

Figure 1: The linear-time hierarchies of standard logics (a) and hyperlogics (c), and the branching-time hierarchies of standard logics (b) and hyperlogics (d). Novel results are annotated with the corresponding theorem number.

Logic	Result	Hyperlogic	Result
LTL	decidable [18]	HyperLTL	undecidable [19] $\exists^*\forall^*$ decidable [19]
QPTL	decidable [11]	HyperQPTL	undecidable **(4.3)** $\exists^*\forall^*$ decidable **(4.2)**
CTL*	decidable [20]	HyperCTL*	undecidable **(4.8)** $\exists^*\forall^*$ decidable **(4.7)**

Table 1: Satisfiability results. Novel results are annotated with the corresponding theorem number.

2.1. Traces and Trees

Let AP be a set of atomic propositions. We call an infinite sequence over subsets of atomic propositions $t \in (2^{AP})^\omega$ a *trace*. A set of traces $T \subseteq (2^{AP})^\omega$ is also called a *trace property* while a set of sets of traces $H \subseteq 2^{((2^{AP})^\omega)}$ is called a *hyperproperty*. Trace manipulation is defined as follows: for a trace t and a natural number $i \geq 0$, we denote the i-th element of the trace by $t[i]$. For a natural number $j \geq i$, $t[i,j]$ denotes the sequence $t[i]t[i+1]\dots t[j-1]t[j]$. Moreover, $t[i,\infty]$ denotes the infinite suffix of t starting at position i. For two traces t and t', we define a zipping operation as follows: $zip(t,t') = (t[0],t'[0])(t[1],t'[1])\dots$.

A *tree* $\mathcal{T}$ is defined as a partially-ordered infinite set of nodes S, where all nodes share a common minimal element $r \in S$, called the *root* of the tree. Moreover, for every node $s \in S$, the set of its *ancestors* $\{s' | s' < s\}$ is totally-ordered. We say that s' is the *direct ancestor* of s, if $s' < s$, and there is no s'' such that $s' < s'' < s$. A Σ-*labeled tree* is defined as a tree $\mathcal{T}$ equipped with a function $L : S \to \Sigma$, that labels every node with an element from a finite set Σ. For the case that $\Sigma = 2^{AP}$, we say that the tree is AP-*labeled*. A *path* through a tree $\mathcal{T}$ is a sequence $\sigma = s_0, s_1, \dots$ of direct ancestors in $\mathcal{T}$, i.e., for all s_i, s_{i+1}, node s_i is the direct ancestor of s_{i+1}. A path is called *initial* if s_0 is the root node, which we omit if it is clear from the context. We use the same path manipulation operations as for traces. The set of *paths originating in node* $s \in S$ is denoted by $Paths(\mathcal{T}, s)$. If s is the root node, we simply write $Paths(\mathcal{T})$.

2.2. LTL and CTL*

Linear-time Temporal Logic (LTL) [1] and Computation Tree Logic (CTL*) [2] are the most studied temporal logics for linear-time and branching-time properties, respectively.

Definition 2.1 (LTL)**.** LTL is a linear-time logic that combines the usual Boolean connectives with temporal modalities $\bigcirc$ (next) and $\mathcal{U}$ (until). The syntax is given by the following grammar:

$$\varphi ::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc\psi \mid \psi\,\mathcal{U}\,\psi,$$

where $a \in AP$, which is the set of atomic propositions. We allow the standard Boolean connectives $\wedge$, $\rightarrow$, $\leftrightarrow$ as well as the derived LTL modalities release $\varphi\,\mathcal{R}\,\psi \equiv \neg(\neg\varphi\,\mathcal{U}\,\neg\psi)$, eventually $\diamondsuit\varphi \equiv \mathit{true}\,\mathcal{U}\,\varphi$, and globally $\square\varphi \equiv \neg\diamondsuit\neg\varphi$. Given a trace $t \in (2^{AP})^\omega$, the semantics of an LTL formula is defined as follows:

$$\begin{aligned}
t &\models a && \text{iff } a \in t[0]\\
t &\models \neg\varphi && \text{iff } t \not\models \varphi\\
t &\models \varphi_1 \vee \varphi_2 && \text{iff } t \models \varphi_1 \text{ or } t \models \varphi_2\\
t &\models \bigcirc\varphi && \text{iff } t[1,\infty] \models \varphi\\
t &\models \varphi_1\,\mathcal{U}\,\varphi_2 && \text{iff } \exists i \geq 0.\ t[i,\infty] \models \varphi_2\\
& && \quad \wedge\, \forall 0 \leq j < i.\ t[j,\infty] \models \varphi_1.
\end{aligned}$$

Definition 2.2 (CTL*)**.** CTL* is a branching-time logic (i.e. the model is a tree) that extends LTL (Def. 2.1) with a path quantifier E meaning "there exists a path". The syntax, where φ denotes state formulas and ψ denotes path formulas, is given as follows:

$$\begin{aligned}
\varphi &::= a \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathrm{E}\psi\\
\psi &::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid \bigcirc\psi \mid \psi\,\mathcal{U}\,\psi,
\end{aligned}$$

where $a \in AP$ is an atomic proposition. The semantics of CTL* is defined over an AP-labeled tree $\mathcal{T}$ with nodes S and labeling function L. Given a node $s \in S$ and a path p in $\mathcal{T}$, we define the semantics of CTL* state and path formulas as follows:

$$\begin{aligned}
s &\models_{\mathcal{T}} a && \text{iff } a \in L(s)\\
s &\models_{\mathcal{T}} \neg\varphi && \text{iff } s \not\models_{\mathcal{T}} \varphi\\
s &\models_{\mathcal{T}} \varphi_1 \vee \varphi_2 && \text{iff } s \models_{\mathcal{T}} \varphi_1 \text{ or } s \models_{\mathcal{T}} \varphi_2\\
s &\models_{\mathcal{T}} \mathrm{E}\psi && \text{iff } \exists p \in \mathit{Paths}(\mathcal{T}, s).\ p \models_{\mathcal{T}} \psi\\
p &\models_{\mathcal{T}} \varphi && \text{iff } p[0] \models_{\mathcal{T}} \varphi\\
p &\models_{\mathcal{T}} \neg\psi && \text{iff } p \not\models_{\mathcal{T}} \psi\\
p &\models_{\mathcal{T}} \psi_1 \vee \psi_2 && \text{iff } p \models_{\mathcal{T}} \psi_1 \text{ or } p \models_{\mathcal{T}} \psi_2\\
p &\models_{\mathcal{T}} \bigcirc\psi && \text{iff } p[1,\infty] \models_{\mathcal{T}} \psi\\
p &\models_{\mathcal{T}} \psi_1\,\mathcal{U}\,\psi_2 && \text{iff } \exists i \geq 0.\ p[i,\infty] \models_{\mathcal{T}} \psi_2\\
& && \quad \wedge\, \forall 0 \leq j < i.\ p[j,\infty] \models_{\mathcal{T}} \psi_1.
\end{aligned}$$

For a tree $\mathcal{T}$ and a CTL* formula φ, we write $\mathcal{T} \models \varphi$ if $\mathcal{T}$ has root r, such that $r \models_{\mathcal{T}} \varphi$.

3. Expressivity

We examine the expressive power of the hyperlogics by going bottom-up through the linear-time and branching-time hierarchies of hyperlogics depicted in Figure 1.

3.1. The Linear-Time Hierarchy

In the following, we first show that, like for the standard logics, HyperQPTL is strictly more expressive than FO[$<, E$]. We then establish that, indeed, S1S[E] is strictly more expressive than HyperQPTL.

Definition 3.1 (HyperLTL)**.** HyperLTL [4] extends LTL (Def. 2.1) with explicit trace quantification. Let $\mathcal{V} = \{\pi_1, \pi_2, \ldots\}$ be an infinite set of trace variables. HyperLTL formulas are defined by the grammar:

$$\begin{aligned}
\varphi &::= \forall\pi.\,\varphi \mid \exists\pi.\,\varphi \mid \psi\\
\psi &::= a_\pi \mid \neg\psi \mid \psi \vee \psi \mid \bigcirc\psi \mid \psi\,\mathcal{U}\,\psi,
\end{aligned}$$

where $a \in AP$ and $\pi \in \mathcal{V}$. Here, $\forall\pi.\,\varphi$ and $\exists\pi.\,\varphi$ denote universal and existential trace quantification, and a_π requires the atomic proposition a to hold on trace π. The semantics of HyperLTL is defined with respect to a set of traces T. Let $\Pi : \mathcal{V} \to T$ be a trace assignment that maps trace variables to traces in T. We can update a trace assignment Π, denoted by $\Pi[\pi \mapsto t]$, where π maps to t and all other trace variables are as in Π. The satisfaction relation $\models_T$ for HyperLTL over a set of traces T is defined as follows:

$$\begin{aligned}
\Pi, i &\models_T a_\pi && \text{iff } a \in \Pi(\pi)[i]\\
\Pi, i &\models_T \neg\varphi && \text{iff } \Pi, i \not\models_T \varphi\\
\Pi, i &\models_T \varphi_1 \vee \varphi_2 && \text{iff } \Pi, i \models_T \varphi_1 \text{ or } \Pi, i \models_T \varphi_2\\
\Pi, i &\models_T \bigcirc\varphi && \text{iff } \Pi, i+1 \models_T \varphi\\
\Pi, i &\models_T \varphi_1\,\mathcal{U}\,\varphi_2 && \text{iff } \exists j \geq i.\ \Pi, j \models_T \varphi_2\\
& && \quad \wedge\, \forall i \leq k < j.\ \Pi, k \models_T \varphi_1\\
\Pi, i &\models_T \exists\pi.\,\varphi && \text{iff } \exists t \in T.\ \Pi[\pi \mapsto t], i \models_T \varphi\\
\Pi, i &\models_T \forall\pi.\,\varphi && \text{iff } \forall t \in T.\ \Pi[\pi \mapsto t], i \models_T \varphi.
\end{aligned}$$

We say that a trace set T satisfies a HyperLTL formula φ, written as $T \models \varphi$, if $\emptyset, 0 \models_T \varphi$, where $\emptyset$ denotes the empty trace assignment.

Definition 3.2 (HyperQPTL)**.** HyperQPTL [21] extends HyperLTL (Def. 3.1) with explicit quantification over atomic propositions. We add atomic formulas q, which are independent of the trace variables, and prenex propositional quantification $\exists q.\varphi$ to the syntax. HyperQPTL inherits the semantics of HyperLTL with two additional rules for the new syntactic constructs:

$$\begin{aligned}
\Pi, i &\models_T \exists q.\,\varphi \text{ iff } \exists t \in (2^{\{q\}})^\omega.\ \Pi[\pi_q \mapsto t], i \models_T \varphi\\
\Pi, i &\models_T q \qquad \text{iff } q \in \Pi(\pi_q)[i].
\end{aligned}$$

Definition 3.3 (FO[$<, E$])**.** FO[$<, E$] extends FO[$<$] with the equal-level predicate E, which relates points in time. The syntax of FO[$<, E$] is obtained by extending the syntax of FO[$<$] with $E(x, y)$. Given a set of atomic propositions AP

and a set V_1 of first-order variables, we define the syntax of FO[$<, E$] formulas as follows:

$$\tau ::= P_a(x) \mid x < y \mid x = y \mid E(x,y)$$
$$\varphi ::= \tau \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists x.\varphi,$$

where $a \in AP$ and $x, y \in V_1$.

While FO[$<$] formulas are interpreted over a trace t, we interpret an FO[$<, E$] formula φ over a set of traces T, writing $T \models \varphi$ if T satisfies φ. As first described in [8], we assign first-order variables with elements from the domain $T \times \mathbb{N}$. The $<$ relation is defined as the set $\{(t, n_1), (t, n_2) \in (T \times \mathbb{N})^2 \mid n_1 < n_2\}$ and the equal-level predicate is defined as $\{(t_1, n), (t_2, n) \in (T \times \mathbb{N})^2\}$. Note that $x < y$ holds in FO[$<, E$] iff y is a successor of x on the same trace.

Lemma 3.1. *HyperQPTL is at least as expressive as FO[$<, E$].*

Proof. We give a linear translation from FO[$<, E$] to HyperQPTL. We encode each first-order variable x as a combination of a trace variable π_x and a propositional variable q_x, where we enforce q_x to hold exactly once. Let φ be an FO[$<, E$] formula over AP in prenex normal form. We construct the HyperQPTL formula hq(φ) as follows:

$$\begin{aligned}
\text{hq}(P_a(x)) &= \Diamond(q_x \wedge a_{\pi_x}) \\
\text{hq}(x < y) &= \Diamond(q_x \wedge \bigcirc\Diamond q_y) \wedge \Box(\bigwedge_{a \in AP} a_{\pi_x} \leftrightarrow a_{\pi_y}) \\
\text{hq}(x = y) &= \Diamond(q_x \wedge q_y) \wedge \Box(\bigwedge_{a \in AP} a_{\pi_x} \leftrightarrow a_{\pi_y}) \\
\text{hq}(E(x,y)) &= \Diamond(q_x \wedge q_y) \\
\text{hq}(\neg\varphi_1) &= \neg\text{hq}(\varphi_1) \\
\text{hq}(\varphi_1 \vee \varphi_2) &= \text{hq}(\varphi_1) \vee \text{hq}(\varphi_2) \\
\text{hq}(\exists x.\varphi_1) &= \exists \pi_x.\ \exists q_x.\ (\neg q_x)\,\mathcal{U}(q_x \wedge \bigcirc\Box\neg q_x) \\
&\quad \wedge \text{hq}(\varphi_1).
\end{aligned}$$

Note that since T is a set of traces, two traces are equal in HyperQPTL iff they globally agree on all atomic propositions. Since we assume φ to be in prenex normal form, hq(φ) is a valid HyperQPTL formula. A straight-forward induction proves that for all trace sets T, $T \models \varphi$ iff $T \models \text{hq}(\varphi)$. □

Theorem 3.2. *HyperQPTL is strictly more expressive than FO[$<, E$].*

Proof. With Lemma 3.1, we are left to show that there are properties that HyperQPTL can express, but FO[$<, E$] cannot. We apply a similar technique as in [21]: Consider the class of models T with only a single trace. In this class, HyperQPTL is expressively equivalent to QPTL and FO[$<, E$] is expressively equivalent to FO[$<$], which is equivalent to LTL [9]. It is known, however, that QPTL is strictly more expressive than LTL since QPTL can express any ω-regular language [22], which LTL cannot [23]. Therefore, HyperQPTL must be strictly more expressive than FO[$<, E$]. □

Definition 3.4 (S1S[E])**.** Similar to the definition of FO[$<, E$], we extend S1S with the equal-level predicate and interpret it over sets of traces. Let AP be a set of atomic propositions, $V_1 = \{x_1, x_2, \ldots\}$ be a set of first-order variables, and $V_2 = \{X_1, X_2, \ldots\}$ a set of second-order variables. The syntax of S1S[E] formulas φ is defined as follows:

$$\tau ::= x \mid min(x) \mid S(\tau)$$
$$\varphi ::= \tau \in X \mid \tau = \tau \mid E(\tau,\tau) \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \exists X.\varphi,$$

where $x \in V_1$ is a first-order variable, S denotes the successor relation, and $min(x)$ indicates the minimal element of the traces addressed by x. Furthermore, $E(\tau,\tau)$ is the equal-level predicate and $X \in V_2 \cup \{X_a \mid a \in AP\}$. We interpret S1S[$E$] formulas over a set of traces T. As in the case of FO[$<, E$], the domain of the first-order variables is $T \times \mathbb{N}$. Let $\mathcal{V}_1 : V_1 \to T \times \mathbb{N}$ and $\mathcal{V}_2 : V_2 \to 2^{(T \times \mathbb{N})}$ be the first-order and second-order valuation, respectively. The value of a term is defined as:

$$\begin{aligned}
[x]_{\mathcal{V}_1} &= \mathcal{V}_1(x) \\
[min(x)]_{\mathcal{V}_1} &= (proj_1(\mathcal{V}_1(x)), 0) \\
[S(\tau)]_{\mathcal{V}_1} &= (proj_1([\tau]_{\mathcal{V}_1}), proj_2([\tau]_{\mathcal{V}_1}) + 1),
\end{aligned}$$

where $proj_1$ and $proj_2$ denote the projection to the first and second component, respectively. Let φ be an S1S[E] formula with free first-order and second-order variables $V_1' \subseteq V_1$ and $V_2' \subseteq V_2 \cup \{X_a \mid a \in AP\}$, respectively. We define the satisfaction relation $\mathcal{V}_1, \mathcal{V}_2 \models \varphi$ with respect to two valuations $\mathcal{V}_1, \mathcal{V}_2$ assigning all free variables in V_1' and V_2' as follows:

$$\begin{aligned}
&\mathcal{V}_1, \mathcal{V}_2 \models_T \tau \in X && \text{iff } [\tau]_{\mathcal{V}_1} \in \mathcal{V}_2(X) \\
&\mathcal{V}_1, \mathcal{V}_2 \models_T \tau_1 = \tau_2 && \text{iff } [\tau_1]_{\mathcal{V}_1} = [\tau_2]_{\mathcal{V}_1} \\
&\mathcal{V}_1, \mathcal{V}_2 \models_T E(\tau_1, \tau_2) && \text{iff } proj_2([\tau_1]_{\mathcal{V}_1}) = proj_2([\tau_2]_{\mathcal{V}_1}) \\
&\mathcal{V}_1, \mathcal{V}_2 \models_T \neg\varphi && \text{iff } \mathcal{V}_1, \mathcal{V}_2 \not\models_T \varphi \\
&\mathcal{V}_1, \mathcal{V}_2 \models_T \varphi_1 \vee \varphi_2 && \text{iff } \mathcal{V}_1, \mathcal{V}_2 \models_T \varphi_1 \text{ or } \mathcal{V}_1, \mathcal{V}_2 \models_T \varphi_2 \\
&\mathcal{V}_1, \mathcal{V}_2 \models_T \exists x.\varphi && \text{iff } \exists (t,n) \in T \times \mathbb{N}. \\
&&& \quad \mathcal{V}_1[x \mapsto (t,n)], \mathcal{V}_2 \models_T \varphi \\
&\mathcal{V}_1, \mathcal{V}_2 \models_T \exists X.\varphi && \text{iff } \exists A \subseteq T \times \mathbb{N}. \\
&&& \quad \mathcal{V}_1, \mathcal{V}_2[X \mapsto A] \models_T \varphi,
\end{aligned}$$

where $\mathcal{V}_i[x \mapsto v]$ updates a valuation.

We call an S1S[E] formula φ closed if every free variable is a second-order variable of the form X_a with $a \in AP$. We say that a trace set T over AP satisfies a closed S1S[E] formula φ, written $T \models \varphi$, if $\emptyset, \mathcal{V}_2 \models_T \varphi$, where $\emptyset$ denotes the empty first-order valuation and $\mathcal{V}_2$ assigns each free X_a in φ to the set $\{(t,n) \in T \times \mathbb{N} \mid a \in t[n]\}$.

Lemma 3.3. *S1S[E] is at least as expressive as HyperQPTL.*

Proof. We describe a linear translation from HyperQPTL to S1S[E], which is similar to the translation from HyperLTL to FO[$<, E$] described in [8]. Assume two sets of first-order variables $Tr = \{x_\pi, x_{\pi'}, \ldots\}$ for trace variables and $Ti =$

$\{y_1, y_2, \ldots\}$ to indicate time. Additionally, we represent propositional variables with second-order variables. Given a HyperQPTL formula φ over atomic propositions AP and a time variable y_i, we inductively construct the S1S[E] formula with free second-order variables $\{X_a \mid a \in AP\}$ as follows:

$$\begin{aligned}
\text{se}(a_\pi, y_i) &= \exists x.\ x \geq x_\pi \vee x < x_\pi \wedge E(y_i, x) \\
&\qquad \wedge x \in X_a \\
&\text{where } x_\pi \text{ is the trace variable for } \pi \\
\text{se}(q, y_i) &= \exists x_p.\ E(y_i, x_q) \wedge x_q \in X_q \\
&\text{where } X_q \text{ is the propositional variable for } q \\
\text{se}(\neg\varphi_1, y_i) &= \neg\text{se}(\varphi_1, y_i) \\
\text{se}(\varphi_1 \vee \varphi_2, y_i) &= \text{se}(\varphi_1, y_i) \vee \text{se}(\varphi_2, y_i) \\
\text{se}(\bigcirc\varphi_1, y_i) &= \text{se}(\varphi_1, S(y_j)) \\
\text{se}(\varphi_1 \,\mathcal{U}\, \varphi_2, y_i) &= \exists y_j \geq y_i.\ \text{se}(\varphi_2, y_j) \\
&\qquad \wedge (\forall y_k.\ y_i \leq y_k < y_j \rightarrow \text{se}(\varphi_1, y_k)) \\
\text{se}(\exists\pi.\varphi_1, y_i) &= \exists x_\pi.\ \text{se}(\varphi_1, y_i) \\
&\text{where } x_\pi \text{ is now used as trace variable for } \pi \\
\text{se}(\exists q.\varphi_1, y_i) &= \exists X_q.\ \text{se}(\varphi_1, y_i) \\
&\text{where } X_q \text{ is now used as propositional variable for } q.
\end{aligned}$$

For a HyperQPTL formula φ, we define

$$\text{se}(\varphi) \coloneqq \exists y_0.\, (\neg\exists y.\, y < y_0) \wedge \text{se}(\varphi, y_0).$$

Using induction, it follows that for each φ and trace set T, $T \models \varphi$ iff $T \models \text{se}(\varphi)$. □

Lemma 3.4. *The S1S[E] model checking problem is undecidable.*

Proof. The S1S[E] model checking problem is to decide for a formula φ and a regular trace set T, whether $T \models \varphi$. A trace set is regular, if it can be described by a Kripke structure. We prove this Lemma by a reduction from 2-counter machines (2CM), which are known to be Turing complete. We describe a simple trace set T such that given a 2CM $\mathcal{M}$ with an initial configuration s_0, we can construct an S1S[E] formula $\varphi_{\mathcal{M},s_0}$ such that $\mathcal{M}$ halts iff $T \models \varphi_{\mathcal{M},s_0}$. A 2CM consists of a finite set of instructions $l_1 : instr_1; \ldots; l_{k-1} : instr_{k-1}; l_k : instr_{halt}$, where the last instruction is the instruction to halt and all other instructions are of one of the following forms:

- $c_i \coloneqq c_i + 1$; `goto` l_j (for $i \in \{1,2\}$ and $1 \leq j \leq k$)
- `if` $c_i = 0$ `then goto` l_j `else` $c_i \coloneqq c_i - 1$; `goto` $l_{j'}$ (for $i \in \{1,2\}$ and $1 \leq j, j' \leq k$).

A 2CM configuration s is a triple (i, m, n), which indicates that the values of the two counters are currently m and n and that the next instruction to be executed is l_i. We call each configuration in which i denotes the halting instruction a halting configuration s_{halt}. Furthermore, we say that a 2CM $\mathcal{M}$ halts for a given initial configuration s_0 if there is a finite sequence $s_0, s_1, \ldots, s_{halt}$ such that for all two successive configurations s_i, s_{i+1}, the latter one is a result of applying the instruction specified in s_i to configuration s_i. The main ideas to encode the halting problem of a 2CM $\mathcal{M}$ with initial configuration s_0 into S1S[E] are the following:

- We can express in S1S[E] that a set X_t contains exactly all nodes of a trace $t \in T$.
- Each 2CM configuration s is encoded as a trace t_s over atomic propositions c_1, c_2 and l which are true exactly once on the trace. A state where the first counter has value m is encoded as a trace t with $c_1 \in t[m]$.
- We choose the trace set T to be the infinite trace set containing a trace t_s for all possible 2CM configurations $s \in \{(i, m, n) \in \mathbb{N}^3\}$. Note that T is clearly regular.
- We can give an S1S[E] formula $succ(X_{t_s}, X_{t_{s'}})$, which is true for traces $t_s, t_{s'}$ iff configuration s' is the result of applying the instruction l_i to configuration $s = (i, m, n)$.
- Given a machine $\mathcal{M}$ with initial configuration s_0, we give an S1S[E] formula $halting(X)$ which is true iff X encodes a halting computation of $\mathcal{M}$. The formula $halting(X)$ is a conjunct of the following requirements:
  - X is a union of finitely many encodings X_{t_s}. This can be formulated in S1S[E] by expressing that there is an upper bound on the positions where c_1, c_2, and l occur on traces in X.
  - X is predecessor closed with respect to the instructions of the machine, i.e., if X_{t_s} is a subset of X, then either X_{t_s} is the trace encoding of the initial configuration s_0, or there is a $X_{t_{s'}} \subseteq X$ such that $succ(X_{t_{s'}}, X_{t_s})$.
  - There is a halting configuration in X, i.e., there is a trace $X_{t_s} \subseteq X$ where l holds at position k in t_s.

Using the ideas presented above, we can define $\varphi_{\mathcal{M},s_0}$ as a formula that checks whether there is a subset X of T which encodes a halting computation of $\mathcal{M}$ starting in s_0, i.e., $\varphi_{\mathcal{M},s_0} \coloneqq \exists X.\ halting(X)$. □

Theorem 3.5. *S1S[E] is strictly more expressive than HyperQPTL.*

Proof. This follows from Lemma 3.3 and Lemma 3.4, since the HyperQPTL model checking problem is decidable [21]. □

3.2. The Branching-Time Hierarchy

The question studied in this section is whether the equi-expressiveness of CTL* and MPL, and of QCTL* and MSO, translates to the corresponding hyperlogics. We establish that MPL[E] (even if restricted to bisimulation-invariant properties) is strictly more expressive than HyperCTL*. We then show that HyperQCTL* is more expressive than MPL[E]. Lastly, we show the rather surprising result that MSO[E] is not more expressive than HyperQCTL*. In fact, the two logics are equally expressive.

Definition 3.5 (HyperCTL*)**.** HyperCTL* [4] generalizes CTL* (Def. 2.2) by adding explicit path variables and quantification. Quantification in HyperCTL* ranges over the paths in a tree. Let $\pi \in \mathcal{V}$ be a *path variable* from an

infinite supply of path variables $\mathcal{V}$ and let $\exists \pi.\ \varphi$ be the explicit existential *path quantification*. HyperCTL* formulas are generated by the following grammar:

$$\varphi ::= a_\pi \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc \varphi \mid \varphi \,\mathcal{U}\, \varphi \mid \exists \pi.\ \varphi.$$

The semantics of a HyperCTL* formula are defined with respect to a tree $\mathcal{T}$ and a *path assignment* $\Pi : \mathcal{V} \to Paths(\mathcal{T})$, which is a partial mapping from path variables to actual paths in the tree. The satisfaction relation $\models_\mathcal{T}$ is given as follows:

$$\begin{aligned}
&\Pi, i \models_\mathcal{T} a_\pi && \text{iff } a \in L(\Pi(\pi)[i]) \\
&\Pi, i \models_\mathcal{T} \neg\varphi && \text{iff } \Pi, i \not\models_\mathcal{T} \varphi \\
&\Pi, i \models_\mathcal{T} \varphi_1 \vee \varphi_2 && \text{iff } \Pi, i \models_\mathcal{T} \varphi_1 \text{ or } \Pi, i \models_\mathcal{T} \varphi_2 \\
&\Pi, i \models_\mathcal{T} \bigcirc \varphi && \text{iff } \Pi, i+1 \models_\mathcal{T} \varphi \\
&\Pi, i \models_\mathcal{T} \varphi_1 \,\mathcal{U}\, \varphi_2 && \text{iff } \exists j \geq i.\ \Pi, j \models_\mathcal{T} \varphi_2 \\
&&& \quad \wedge \forall i \leq k < j.\ \Pi, k \models_\mathcal{T} \varphi_1 \\
&\Pi, i \models_\mathcal{T} \exists \pi.\varphi && \text{iff } \exists p \in Paths(\mathcal{T}).\ p[0,i] = \varepsilon[0,i] \\
&&& \quad \wedge \Pi[\pi \mapsto p, \varepsilon \mapsto p], i \models_\mathcal{T} \varphi,
\end{aligned}$$

where we use ε to denote the last path that was added to the path assignment Π. We say that a tree $\mathcal{T}$ satisfies a HyperCTL* formula φ, written as $\mathcal{T} \models \varphi$, if $\emptyset, 0 \models_\mathcal{T} \varphi$, where $\emptyset$ denotes the empty path assignment.

Without loss of generality, we assume that HyperCTL* formulas are given in negation normal form (NNF) where negations only occur directly in front of atomic propositions. This can be achieved by a straight-forward extension of the syntax and semantics with *conjunction*, the *universal path quantifier* $\forall \pi.\ \varphi$ and the temporal modality $\mathcal{R}$ (release), which has the following semantics [24]:

$$\begin{aligned}
\Pi, i \models_\mathcal{T} \varphi \,\mathcal{R}\, \psi \text{ iff } &\forall j \geq i.\ \Pi, j \models_\mathcal{T} \psi \vee \\
&(\exists j \geq i.\ \Pi, j \models_\mathcal{T} \varphi \\
&\wedge \forall i \leq k \leq j.\ \Pi, k \models_\mathcal{T} \psi).
\end{aligned}$$

Definition 3.6 (MPL[E])**.** MPL equipped with the equal-level predicate (MPL[E]) is, syntactically, FO[$<, E$] (Def. 3.3) with additional second-order set variables $\{X, Y, \ldots\}$, i.e., with atomic formulas $x \in X$ and second-order quantification $\exists X.\varphi$. MPL[E], interpreted over trees $\mathcal{T}$, maps first-order variables to nodes in the tree and second-order variables to sets of nodes. $x < y$ indicates that x is an ancestor of y. Atomic formulas $x \in X$ and $x = y$ are interpreted as expected as set membership and equality on nodes. The equal-level predicate $E(x, y)$ denotes that two nodes x and y are on the same level, i.e., have the same number of ancestors. As for MPL, we require that MPL[E]'s second-order quantification is restricted to full paths, i.e., each quantified second-order variable X is mapped to a set whose nodes constitute exactly one path of the tree. We write $\mathcal{T} \models \varphi$ if the tree $\mathcal{T}$ satisfies the MPL[E] formula φ.

Lemma 3.6. *MPL[E] is at least as expressive as HyperCTL*.*

Proof. We can give a linear translation from HyperCTL* to MPL[E], which is very similar to the one in Lemma 3.3. As before, we assume sets of first-order variables $Tr = \{x_1, x_2, \ldots\}$ for trace variables and $Ti = \{y_1, y_2, \ldots\}$ to indicate time. Since we are now in a branching-time setting, we translate the quantification of paths with MPL[E] second-order quantification. □

Theorem 3.7. *MPL[E] is strictly more expressive than HyperCTL*, even if restricted to bisimulation-invariant properties.*

Proof. We proceed by arguing that in contrast to HyperCTL*, MPL[E] can simulate the epistemic knowledge modality known from KLTL and KCTL* [25], [26]. Epistemic logics describe multiple agents and the local knowledge they gain from observing different aspects of the system. Consider the extension of HyperCTL* with the modality $\mathcal{K}_{A,\pi}\varphi$ (HyperKCTL*), stating that the agent who can observe only the propositions $A \subseteq AP$ on path π knows that φ holds. The modality has the following semantics:

$$\begin{aligned}
\Pi, i \models_\mathcal{T} \mathcal{K}_{A,\pi}\varphi \text{ iff } &\forall p \in Paths(\mathcal{T}).\ p[0,i] =_A \Pi(\pi)[0,i] \\
&\rightarrow \Pi[\pi \mapsto p], i \models_\mathcal{T} \varphi.
\end{aligned}$$

Here, $=_A$ is used to state that two paths are A-observationally equivalent, i.e., equal when projected to the set A. One can extend the translation from HyperCTL* to MPL[E] to also handle $\mathcal{K}_{A,\pi}\varphi$. The idea is to quantify a new second-order set for variable π which agrees with the previously quantified set on all atomic propositions in A. This shows that MPL[E] subsumes HyperCTL* extended with the knowledge operator. Note that due to the branching-time character of HyperKCTL*, all properties expressible in that logic must be bisimulation invariant. Thus, also the fragment of MPL[E] that is equivalent to HyperKCTL* can only characterize bisimulation-invariant properties. It is, however, known that HyperCTL* does not subsume KCTL* [27], which is a syntactic subset of HyperKCTL*. Hence, MPL[E] must be strictly more expressive than HyperCTL*.

□

We now show that HyperQCTL* is more expressive than MPL[E]. This result coincides with the fact that QCTL* is known to be more expressive than MPL, which is equivalent to CTL* [13], [20].

Definition 3.7 (HyperQCTL*)**.** HyperQCTL* extends HyperCTL* (Def. 3.5) with quantification over atomic propositions. We add atomic formulas q_π and propositional quantification $\exists q.\varphi$ to the syntax of HyperCTL* to obtain HyperQCTL*. Let $\mathcal{T}$ be an AP-labeled tree with labeling function L. The satisfaction relation of HyperQCTL* extends the one of HyperCTL* by adding a rule for the propositional quantification to the definition of the semantics:

$$\begin{aligned}
\Pi, i \models_\mathcal{T} \exists q.\varphi \text{ iff } &\exists L' : S \to 2^{AP \cup \{q\}}.\ \forall s \in S. \\
&L'(s) =_{AP \setminus \{q\}} L(s) \wedge \Pi, i \models_{\mathcal{T}[L'/L]} \varphi.
\end{aligned}$$

Here, L' is the updated labeling function which (re)assigns the atomic proposition q to nodes in $\mathcal{T}$. We write $\mathcal{T}[L'/L]$ for the tree $\mathcal{T}$ labeled with elements from $AP \cup \{q\}$ with

the new labeling function L'. Note that compared to the definition of HyperQPTL, we do not just add an independent q-trace to the model. Instead, q is (re-)assigned at every node in the tree.

We say that a tree $\mathcal{T}$ satisfies a HyperQCTL* formula φ, written $\mathcal{T} \models \varphi$, if $\emptyset, 0 \models_{\mathcal{T}} \varphi$.

Theorem 3.8. *HyperQCTL* is strictly more expressive than MPL[E].*

Proof. The proof proceeds as the one for Theorem 3.2. Consider the model of *linear trees*, i.e., trees in which each node has a unique successor. For this class of models, MPL[E] is equivalent to FO[$<$], since the equal-level predicate collapses to equality and second-order quantification in MPL[E] can only quantify the unique single path. Likewise, HyperQCTL* collapses to QPTL. But it is known that QPTL can express all ω-regular properties [22] and FO[$<$], which is equivalent to LTL, cannot [23]. Thus, HyperQCTL* has to be more expressive than MPL[E]. □

Lastly, we prove the equivalence of MSO[E] and HyperQCTL*. The syntax and semantics of MSO[E] is the same as for MPL[E]. The only difference is that the second-order quantification is not restricted to range over full paths.

Theorem 3.9. *HyperQCTL* and MSO[E] are expressively equivalent.*

Proof. We give linear translations in both directions. To translate an MSO[E] formula into HyperQCTL*, we can use propositional quantification to mimic second-order quantification of a set of nodes in the tree. Furthermore, to translate first-order quantification into HyperQCTL*, we quantify a path and an atomic proposition, of which we require that it holds exactly once on that path (as in Lemma 3.1). This allows us to translate the equal-level predicate $E(x, y)$ by checking whether the atomic proposition for x holds at the same time on the path for x as the atomic proposition for y on the path for y. For the other direction, we translate a HyperQCTL* formula into an MSO[E] formula by using distinct first-order variables to indicate time and paths (as in Lemma 3.3). Quantification of atomic propositions can be mimicked by second-order quantification. □

4. Satisfiability

The satisfiability problem of a standard linear-time logic is to decide, for a given formula φ, whether there exists a trace t such that $t \models \varphi$. For a linear-time hyperlogic, we ask whether a trace set T exists such that $T \models \varphi$. For branching-time logics, both for standard and for hyperlogics, we ask whether there is a tree $\mathcal{T}$ such that $\mathcal{T} \models \varphi$.

The satisfiability problems of LTL and CTL* are decidable [18], [20]. In the linear-time spectrum, it is known that the fragment of $\exists^*\forall^*$ HyperLTL is decidable, and that already a single $\forall\exists$ trace quantifier alternation leads to undecidability [19]. We show in Section 4.1 that the same rule also applies to the more expressive linear-time hyperlogic HyperQPTL: the $\exists^*\forall^*$ HyperQPTL fragment is decidable whereas a $\forall\exists$ trace quantifier alternation causes undecidability.

In the branching-time spectrum, no decidable hyperlogics have been identified so far. In Section 4.2, we first consider the satisfiability problem of the existential fragment of HyperCTL*. We then obtain the general result that the $\exists^*\forall^*$ fragment is decidable and that a single $\forall\exists$ quantifier alternation leads to undecidability.

4.1. Linear-Time Satisfiability

We define the fragments of HyperQPTL on the basis of the formulas *trace* quantification (analogously to HyperLTL [19]). Note that formulas in these fragments may contain arbitrary propositional quantification.

Lemma 4.1. *The satisfiability problem of HyperQPTL formulas in the $\forall^*$ fragment is decidable.*

Proof. We employ a similar reasoning as in [19]. First note that a $\forall^*$ HyperQPTL formula is satisfiable iff it is satisfiable by a model containing only a single trace. Since all quantifiers are universal, a model with more than one trace can always be reduced to a model with exactly one trace. We therefore reduce the satisfiability problem of $\forall^*$ HyperQPTL to the satisfiability problem of QPTL, which is decidable [11]. Given a $\forall^*$ HyperQPTL formula φ, consider the QPTL formula ψ obtained by removing the trace quantifiers from φ and also removing all trace variables from atomic propositions a_π. The resulting formula is equisatisfiable to φ. □

Theorem 4.2. *The satisfiability problem of HyperQPTL formulas in the $\exists^*\forall^*$ fragment is decidable.*

Proof. Let a HyperQPTL formula ψ of the form $\vec{Q}_0.\ \exists\pi_0, \ldots, \pi_n.\ \vec{Q}_1.\ \forall\pi'_0, \ldots, \pi'_m.\ \vec{Q}_2.\ \varphi$ over $AP = \{a^0, \ldots, a^{k-1}\}$ be given. $\vec{Q}_i$ denotes arbitrary propositional quantification. In the following, we assume that ψ has at least one existential trace quantifier. If this is not the case, Lemma 4.1 handles the formula. We construct an equisatisfiable QPTL formula ψ_{QPTL} for ψ. As first step, we eliminate the universal quantification by explicitly enumerating every possible interaction between the universal and existential quantifiers, a technique already used to prove the decidability of the $\exists^*\forall^*$ HyperLTL fragment [19]:

$$\vec{Q}_0 \exists \pi_0, \ldots, \pi_n . \vec{Q}_1 \bigwedge_{j_1=1}^{n} \cdots \bigwedge_{j_m=1}^{n} \vec{Q}_2 . \varphi[\pi_{j_1}/\pi'_1, \ldots, \pi_{j_m}/\pi'_m],$$

where $\varphi[\pi_j/\pi'_i]$ denotes that the trace variable π'_i in φ is replaced by π_j. Like this, every combination of trace assignments for the universal quantification is covered and the resulting formula is equisatisfiable and of size $\mathcal{O}(n^m)$. The formula is technically not a HyperQPTL formula yet, since $\vec{Q}_2$ is in the scope of a conjunction. An equisatisfiable HyperQPTL formula without $\vec{Q}_2$ can be constructed by introducing a fresh existential quantifier in the quantifier prefix

for each existential quantification in a conjunct (denoted by $\vec{\exists}$) and by simply moving the universal quantification in $\vec{Q}_2$ in front of the conjunction, into the quantifier prefix (denoted by $\vec{\forall}$). We use φ' to denote the body of the resulting HyperQPTL formula. As second step, we eliminate the existential trace quantification by replacing each trace quantifier $\exists \pi_i$ with k existential quantifiers over propositions, one for every atomic proposition in AP:

$$\psi_{QPTL} \coloneqq \vec{Q}_0 \bigexists_{\pi=\pi_0}^{\pi_n} a_\pi^0, \ldots, a_\pi^{k-1} \vec{Q}_1 \vec{\exists}\ \vec{\forall}.\ \varphi'.$$

By the semantics of HyperQPTL, the resulting formula is equisatisfiable to ψ: Assume ψ is satisfied by a trace set T_ψ. We construct a set of witnesses for ψ_{QPTL} by splitting every witness $t \in T_\psi$ into k traces t^j ($j < k$), one for every atomic proposition $a^j \in AP$. For all such j, we require the traces $t^j \in (2^{\{a_\pi^j\}})^\omega$ to agree with t on the translated atomic proposition. The resulting trace set $\{t^j \mid 0 \leq j < k\}$ satisfies ψ_{QPTL} by construction. Constructing T_ψ from a set of witnesses for a formula ψ_{QPTL} obtained from the above construction works analogously by computing, for every point in time k, the union of all the witness positions $t^j[k]$ of witnesses for ψ_{QPTL} for each π_i. □

Theorem 4.3. *The satisfiability problem for HyperQPTL formulas in the $\forall\exists$ fragment is undecidable.*

Proof. HyperQPTL subsumes HyperLTL as a syntactic fragment. Since the satisfiability problem for HyperLTL is undecidable for formulas with at least one trace quantifier alternation starting with a universal quantifier [19], this also holds for HyperQPTL. □

We have thus identified the $\exists^*\forall^*$ fragment as the largest decidable fragment of the linear-time hyperlogic HyperQPTL.

4.2. Branching-Time Satisfiability

In the following, we study the satisfiability problem of HyperCTL*. We assume formulas to be given in negated normal form. A HyperCTL* formula in NNF is in the $\exists^*$ and $\forall^*$ fragment, respectively, if it contains exclusively universal or exclusively existential path quantifiers. The union of the two fragments is the alternation-free fragment. The formula is in the $\exists^*\forall^*$ fragment, if there is no existential path quantifier in the scope of a universal path quantifier. It is in the $\forall\exists$ fragment if there is exactly one existential path quantifier in the scope of a single universal path quantifier.

Lemma 4.4. *The satisfiability problem of HyperCTL* formulas in the $\forall^*$ fragment is decidable.*

Proof. The proof is similar to the proof for Lemma 4.1. We reduce the satisfiability problem of $\forall^*$ HyperCTL* to the satisfiability problem of CTL* and consider models that are linear trees, i.e., trees having only a single path. □

We now prove in two steps that the $\exists^*$ fragment is decidable, regardless of the temporal modalities and the nesting depth of the existential quantifiers. We start with formulas of a specific form and then generalize the result to the full fragment. Subsequently, we establish that the satisfiability problem for HyperCTL* formulas in the full $\exists^*\forall^*$ fragment remains decidable.

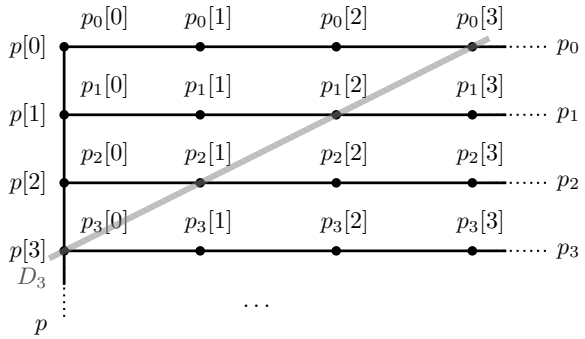

Figure 2: The witness p for π and the sequence of witnesses p_i for π'' arranged in a comb-like structure. Nodes $p_3[0], p_2[1], p_1[2]$, and $p_0[3]$ reside on diagonal D_3.

Lemma 4.5. *The satisfiability problem for HyperCTL* formulas of the form $\varphi \coloneqq \exists\pi.(\exists\pi'.\psi')\,\mathcal{R}(\exists\pi''.\psi'')$, where ψ' and ψ'' are quantifier free, is decidable.*

Proof. The key idea of the proof is to show that every model of a φ-shaped formula has a finite representation. More concretely, we show that we can represent an arbitrary model $\mathcal{T}$ satisfying φ as a tree $\mathcal{T}_{fin}$ of bounded size. We then show that $\mathcal{T}_{fin}$ can be extended to an infinite tree $\tilde{\mathcal{T}}$ which satisfies φ. We conclude by describing a naive decision procedure which enumerates all bounded trees $\mathcal{T}_{fin}$ and checks whether they can be extended into an infinite model $\tilde{\mathcal{T}}$ for φ.

Intuition. We first give some intuition on how to construct $\mathcal{T}_{fin}$ out of $\mathcal{T}$. Assume a formula $\exists\pi.\square(\exists\pi''.\psi)$ (which belongs to the fragment described in the statement) and a model $\mathcal{T}$ satisfying it. In this model, there needs to be a path p witnessing π, and at each point in time i, there must be a path p_i, which branches off of p and serves as a witness for π'' at point in time i. Extracting these witnesses from the model results in a comb-like structure as depicted in Figure 2.

Through formula ψ, each pair of nodes $p[i+j]$ and $p_i[j]$ are related with each other, e.g., if $\psi = \square(a_\pi \leftrightarrow a_{\pi'})$, then all $p[i+j]$ and $p_i[j]$ must agree on a. The nodes $p[i+j]$ and $p_i[j]$ always reside on the same diagonal in the comb. Like this, all nodes on the same diagonal are related with each other through p. When transforming the witnesses, it is therefore important to only consider one diagonal as a whole and to not alter just a single node on it. Diagonal D_3 is also depicted in Figure 2. Next, note that ψ can be transformed into a Büchi automaton which accepts each pair of witness paths $(p[i,\infty], p_i)$. We label each p_i in the comb with the corresponding accepting automaton run. Now, the crucial observation is that if two diagonals in the comb are labeled with the same set of automaton states, then we can just *cut out* the part between those two diagonals and still

have accepting runs, i.e., the resulting paths p and all p_i are still witnesses for π and π''. The proof proceeds by repeatedly cutting out nonessential parts of the comb until it has a suitable prefix of bounded size which we call $\mathcal{T}_{fin}$.

Formal Proof. We assume w.l.o.g. that no $\bigcirc$-modality occurs in the formula; any $\bigcirc$-modality can only add an offset to the operations which we describe in the following. Assume a tree $\mathcal{T}$ over nodes S with labeling function L that satisfies φ, i.e., there exist a path p through $\mathcal{T}$ serving as witness for π. By the semantics of the $\mathcal{R}$-modality, either (Case 1) an infinite sequence $p_0, p_1, p_2, \dots$ of witnesses for π'' or (Case 2) a finite sequence $p_0, p_1, \dots, p_n$ of witnesses for π'' and a final witness p' for π'.

Case 1. We first construct a Büchi automaton $A_{\psi''}$ that accepts all possible pairs of paths (p, p_i) that satisfy ψ''. Then, we formally describe the comb structure and how it can be labeled with the accepting runs of $A_{\psi''}$ on each $(p[i, \infty], p_i)$. Furthermore, we give the necessary definitions of *frontiers* and *cuts* in order to define how to cut parts out of the comb without changing the fact that its paths constitute witnesses for φ. We show how to construct $\mathcal{T}_{fin}$ by repeatedly cutting out nonessential parts of the comb and give a maximal bound for its size. Lastly, we describe how to extend $\mathcal{T}_{fin}$ to an infinite model $\tilde{\mathcal{T}}$.

Automaton construction. Since the formula ψ'' is quantifier-free, it is an LTL formula where each atomic proposition a_π is interpreted as a unique LTL proposition. Let $A'_{\psi''} = (Q', q'_0, \Sigma', \delta', F')$ be the nondeterministic Büchi automaton obtained from this LTL interpretation of ψ'' [28]. We transform $A'_{\psi''}$ into a nondeterministic Büchi automaton $A_{\psi''} = (Q, q_0, \Sigma, \delta, F)$, which reasons separately over π and π'':

- $Q : Q'$, $q_0 : q'_0$
- $\Sigma : S \times S$
- $\delta : Q \times \Sigma \to 2^Q$ where $q' \in \delta(q, (s_0, s_1))$ iff $q' \in \delta'(q, A)$ and $L(s_0) = \{a \in AP \mid a_\pi \in A\}$ and $L(s_1) = \{a \in AP \mid a_{\pi''} \in A\}$, where $A \subseteq \{a_\pi, a_{\pi''} \mid a \in AP\}$.
- $F : F'$

Note that $A_{\psi''}$ reasons over pairs of paths, while $A'_{\psi''}$ reasons over traces. The automaton $A_{\psi''}$ yields accepting runs r_i for all pairs of witnesses $(p[i, \infty], p_i)$. We can thus *associate* with each node $p_i[j]$ the automaton state $r_i[j]$.

Frontiers. We arrange the witness paths p and $p_0, p_1, p_2, \dots$ in a comb-like structure as shown in Figure 2. For all $k \in \mathbb{N}$, we address the sequence of nodes $p_0[k-0], \dots, p_k[k-k]$ as the k-th diagonal of the comb, denoted by D_k. We use the usual sequence notation for diagonals, e.g., $D_k[i]$ to address the i-th element of the sequence. We call the set of automaton states associated with nodes in D_k *frontier* F_k, formally $F_k \coloneqq \{r_i[k-i] \mid i \leq k\}$. Note that for every k, $F_k \subseteq Q$.

Cuts. We now establish how to safely remove parts of the comb structure, i.e., in such a way that the witness paths in the altered comb still have accepting runs in $A_{\psi''}$. To this end, we define the *cut operation* and refine it to a *preserving cut*, which requires the definition of an additional property which we call *frontier-preserving cuttable*.

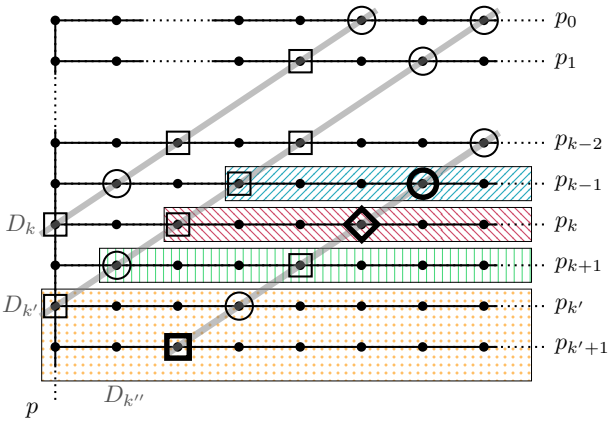

Figure 3: A comb structure with highlighted diagonals D_k, $D_{k'}$, and $D_{k''}$, together with their associated automaton states (depicted as square, circle and diamond). States at representative positions are printed in bold, suffices used for the cut are highlighted.

For two diagonals D_k and $D_{k'}$ with $F_k = F_{k'}$, a cut modifies the comb in such a way that the suffix of every node in D_k is replaced by the suffix of a node in $D_{k'}$, where both nodes have to be associated with the same automaton state. Formally, we replace every $p_i[k-i, \infty]$ with some $p_{i'}[k'-i', \infty]$, requiring that $D_k[i]$ and $D_{k'}[i']$ are associated with the same state. Additionally, to preserve the relation of the modified paths with p, we replace the sub-comb with origin in $p[k]$ with the sub-comb with origin in $p[k']$. Note that because of the requirement that $D_k[i]$ and $D_{k'}[i']$ are associated with the same state, the modified witness paths still have accepting runs through $A_{\psi''}$.

For $k \leq k'$, we say that two diagonals D_k, $D_{k'}$ with $F_k = F_{k'}$ are frontier-preserving cuttable (for short: cuttable) if for every $q \in F_{k'}$, q is either associated with at least as many nodes on D_k as on $D_{k'}$, or it is associated with $|Q|$ nodes on D_k.

For $k \leq k' \leq k''$, a cut preserving $F_{k''}$ is a cut between two cuttable diagonals D_k and $D_{k'}$, such that the set $F_{k''}$ is not modified by the operation. For each $q \in F_{k''}$, pick a position $i_q \leq k''$ as a representative such that the state associated with $D_{k''}[i_q]$ is q. All states q with representative position $i_q \geq k'$ will not be affected by the cut. For representative positions $i_q < k'$, ensure that when choosing suffices from $D_{k'}$ for the cut, each suffix $p_{i_q}[k' - i_q, \infty]$ is chosen at least once. This is possible since we require D_k and $D_{k'}$ to be cuttable. Like this, we ensured that all representative states are not deleted by the cut. Figure 3 and Figure 4 show the choice of representative positions in a comb and the resulting preserving cut.

Construct $\mathcal{T}_{fin}$. We describe how to perform a series of preserving cuts to ensure that sufficiently many accepting states can be found in a bounded-size prefix $\mathcal{T}_{fin}$ of the comb. First, note that there are at most $2^{|Q|}$ different frontiers. Furthermore, there are at most $c = (|Q|+1)^{|Q|}$ many different equivalence classes of the cuttable property, i.e.

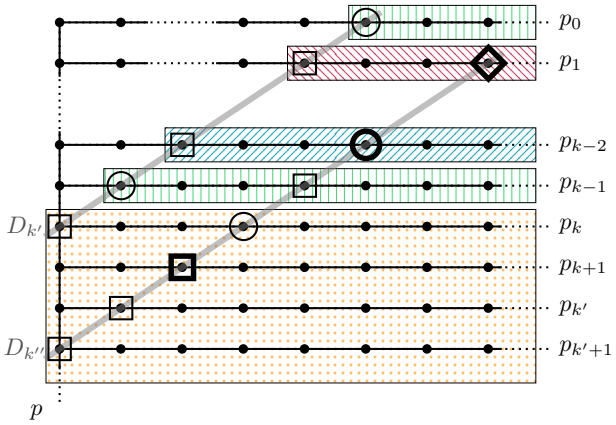

Figure 4: The result of the cutting operation prepared in Figure 3. The highlights show which suffix was shifted to which node in the comb.

for $c+1$ many diagonals, at least two are cuttable. We say that a diagonal D_k is *close* to $D_{k'}$ if $|k'-k| \leq c$. By the pigeonhole principle, for every two diagonals $D_k, D_{k'}$ and state set $F_{k''}$ with $k \leq k' \leq k''$, we can perform a number of cuts on diagonals situated between D_k and $D_{k'}$, each preserving $F_{k''}$, such that at the end, $D_{k'}$ is close to D_k and the set $F_{k''}$ did not change.

There are only finitely many different frontiers in the infinite comb, so at least one frontier occurs on infinitely many diagonals. We call that frontier F_ω. Pick the smallest number $inf \in \mathbb{N}$ such that $F_{inf} = F_\omega$ and cut diagonal D_{inf} as close as possible to D_0 while preserving F_{inf}. Note that the first $|Q|^{|Q|}$ diagonals are, in general, not cuttable; therefore, in the worst case, D_{inf} will be cut close to $D_{|Q|^{|Q|}}$. As a result of these cuts, frontier F_ω might not occur infinitely often anymore. More concretely, diagonals which were previously associated with frontier F_ω will now have frontiers which are a subset of F_ω. Since there are only finitely many different subsets of any finite set, we know that there exists at least one frontier $F_{\omega'} \subseteq F_\omega$ that occurs on infinitely many diagonals.

For every automaton state $q \in F_{\omega'}$, there exists by construction an $i_q \leq inf$ such that $D_{inf}[i_q]$ is associated with q. We call the set of all i_q the set of designated positions P. Now, find the smallest $inf' > inf$, such that $F_{inf'} = F_{\omega'}$, and for all $i \in P$, r_i has an accepting state between inf and inf'. Such an inf' exist because of the Büchi acceptance condition. We now perform a series of cuts to cut $D_{inf'}$ as close to D_{inf} as possible, each of which preserves $F_{inf'}$. Find the $i \in P$ whose accepting state is closest to D_{inf} and cut the corresponding diagonal close to D_{inf}. Continue with the $i' \in P$ whose accepting state comes next and cut it close to the last diagonal that was cut close. Proceed, until the diagonal of the last accepting state of designated position was cut close. Finally, cut $D_{inf'}$ close to that last diagonal.

We choose $\mathcal{T}_{fin}$ to be the finite prefix of the resulting comb up to (and including) $D_{inf'}$. The depth of $\mathcal{T}_{fin}$ is

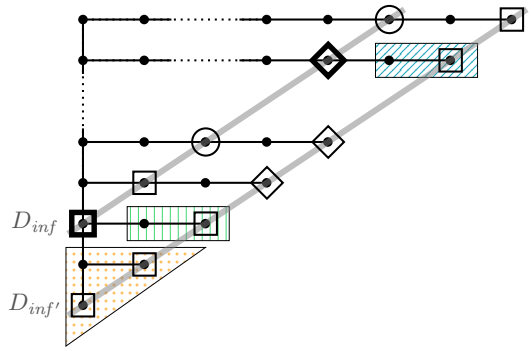

Figure 5: The finite prefix $\mathcal{T}_{fin}$ with diagonals D_{inf} and $D_{inf'}$. States in designated positions on D_{inf} are printed in bold. Suffices that will be copied to extend the prefix comb are highlighted.

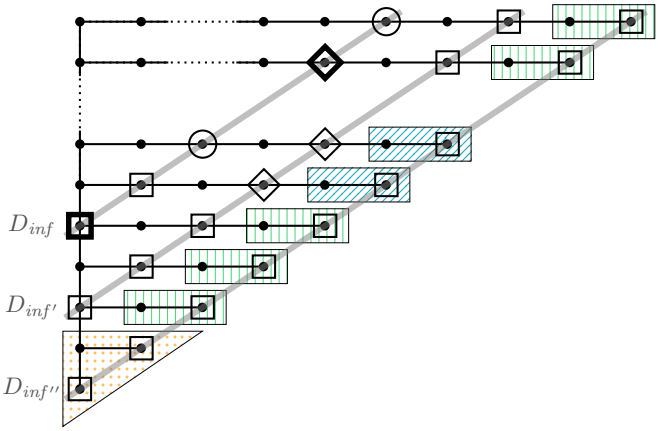

Figure 6: The resulting larger finite prefix after extending every witness path in Figure 5 once. The highlights show which part of $\mathcal{T}_{fin}$ was used to extend the prefix.

bounded by $b = |Q|^{|Q|} + (2 + |Q|) \cdot (|Q| + 1)^{|Q|}$. This is because $\mathcal{T}_{fin}$ consists of a prefix of diagonals up to the first cuttable diagonal (at the most $|Q|^{|Q|}$ many), followed by D_{inf}. Then, $|P| \leq |Q|$ many diagonals have been cut close and the distance between them is at the most $(|Q|+1)^{|Q|}$. Lastly, $D_{inf'}$ was cut close, again with a maximal distance of $(|Q|+1)^{|Q|}$.

Construct $\tilde{\mathcal{T}}$. We now extend $\mathcal{T}_{fin}$ into an infinite tree $\tilde{\mathcal{T}}$ also satisfying φ. By construction, for each $i \in P$, run r_i has an accepting state between D_{inf} and $D_{inf'}$. Furthermore, for each $q \in F_{inf'}$, there is a designated position $i_q \in P$ such that $D_{inf}[i_q]$ is associated with q. We extend $\mathcal{T}_{fin}$ by extending each node in $D_{inf'}$ as follows: For each $i \leq inf'$ with q associated to $D_{inf'}[i]$ and designated position i_q, we append a copy of $p_{i_q}[inf - i_q + 1, inf' - i_q]$ to $p_i[inf' - i]$. Additionally, we copy the sub-comb starting in node $p[inf+1]$ and append it to node $p[inf']$, thus completing the extension. By construction, we now have a larger finite comb ending in a diagonal $D_{inf''}$ with $F_{inf''} \subseteq F_{inf'}$. Figure 5 shows a possible prefix comb $\mathcal{T}_{fin}$ and Figure 6 shows how it is extended. Repeating this process indefinitely, we get

an infinite, ultimately periodic model $\tilde{\mathcal{T}}$ where each pair $(p[i,\infty], p_i)$ of witness paths in the comb of $\tilde{\mathcal{T}}$ is accepted by $A_{\psi''}$. It is thus a model for φ.

Case 2. In the case where the release modality is witnessed by path p for π and a sequence of paths $\{p_0, p_1, \ldots, p_n, p'\}$ for π'' and π', we proceed very similar to Case 1. We again arrange the witnesses in a comb-like graph, with the only difference that at $p[n]$, there are the two witnesses p_n and p' branching from p. In order to get the same structure as in Case 1, we zip p' and p_n into one witness path $\bar{p}_n$. Furthermore, for all $m > n$, we add dummy witnesses $p_\top = \emptyset^\omega$ branching from p at $p[m]$.

Automata construction. As in Case 1, we associate the paths with the corresponding automaton runs. For $(p[i,\infty], p_i)$ with $i < n$, we use the automaton $A_{\psi''}$, as in Case 1. For $(p[i,\infty], p_\top)$ with $i > n$, we use the automaton $A_\top$, which unconditionally accepts every pair of traces. For $(p[n,\infty], \bar{p}_n)$, we construct a new automaton $A_{\psi' \wedge \psi''}$ based on the LTL automaton $A'_{\psi' \wedge \psi''}$ for $\psi' \wedge \psi''$, similar to the construction of $A_{\psi''}$. The automaton $A_{\psi' \wedge \psi''}$ has the alphabet $\Sigma = S \times (S \times S)$ and the transition function $\delta : Q \times \Sigma \to 2^Q$, where $q' \in \delta(q, (s_0, (s_1, s_2)))$ iff $q' \in \delta'(q, A)$, and $L(s_0) = \{a \in AP \mid a_\pi \in A\}$, $L(s_1) = \{a \in AP \mid a_{\pi'} \in A\}$, and $L(s_2) = \{a \in AP \mid a_{\pi''} \in A\}$. We denote the set of states of automaton $A_{\psi''}$ with Q and the set of states of automaton $A_{\psi' \wedge \psi''}$ with $Q_{\psi' \wedge \psi''}$.

Construct $\mathcal{T}_{fin}$. Again, we construct a tree $\mathcal{T}_{fin}$ of bounded depth. First, cut diagonal D_n close to diagonal D_0. Following D_n, there are again finitely many different cuttable diagonals (containing states from all three automata). Proceeding as in Case 1, construct $\mathcal{T}_{fin}$ such that after D_n, there are two diagonals D_{inf} and $D_{inf'}$ with sufficiently many accepting states in between. The bound b' on the depth of $\mathcal{T}_{fin}$ is obtained analogously to the bound in Case 1. We only remark that n is bounded by $|Q|^{|Q|} + (|Q|+1)^{|Q|}$, and the maximal number of different cuttable diagonals is described in terms of the number of states of all three automata, i.e., $(|Q|+|Q_{\psi' \wedge \psi''}|+|Q_\top|+1)^{(|Q|+|Q_{\psi' \wedge \psi''}|+|Q_\top|)}$. We conclude by noting that b' can be used as an over-approximation of bound b in Case 1. Finally, as in Case 1, we construct an infinite satisfying tree $\tilde{\mathcal{T}}$ using $\mathcal{T}_{fin}$.

Decision Algorithm. Enumerate all comb-like prefixes $\mathcal{T}_{fin}$ of bounded depth b' to find a suitable prefix (for either of both cases). Whether a prefix is suitable or not can be decided by labeling it with corresponding runs from the automata of Cases 1 and 2 and checking whether it contains a segment between two diagonals D_{inf} and $D_{inf'}$ which qualifies to be extended into a model $\tilde{\mathcal{T}}$ as described above. When associating the comb prefix with runs from the automata, we have to take into account all finitely-many points in time n where ψ'' could be released by ψ'. If some prefix $\mathcal{T}_{fin}$ is suitable, φ is satisfiable (namely by the described tree $\tilde{\mathcal{T}}$). As shown above, there is a suitable finite prefix of bounded depth b' whenever φ is satisfiable. □

Corollary 4.1. *The satisfiability problem for HyperCTL* formulas of the form $\exists\pi.(\exists\pi''.\psi'')\,\mathcal{U}(\exists\pi'.\psi')$, where ψ' and ψ'' are quantifier free, is decidable.*

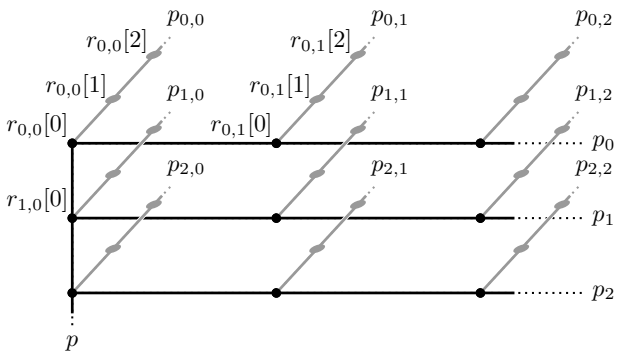

Figure 7: A 3-dimensional comb graph resulting from the arrangement of witnesses for a HyperCTL* formula $\exists\pi.\square(\exists\pi'.\square(\exists\pi''.\varphi))$. The innermost witnesses $p_{i,j}$ are labeled with the automaton states of the corresponding automaton runs.

Proof. We proceed similarly to Case 2 in the proof above. The only difference is that formula ψ'' does not have to hold at the same point in time n where formula ψ' holds. Therefore, the resulting comb does not have two witnesses branching from $p[n]$ that we have to zip. We use an automaton $A_{\psi'}$ instead of $A_{\psi' \wedge \psi''}$ to obtain the run r_n for $(p[n,\infty], p')$. □

We lift the arguments of the above proof to arbitrary formulas in the existential fragment of HyperCTL*.

Lemma 4.6. *The satisfiability problem for HyperCTL* formulas in the $\exists^*$ fragment is decidable.*

Proof. Define the existential quantifier depth of a $\exists^*$HyperCTL* formula as the maximal number of alternations between existential quantifiers and the temporal modalities $\mathcal{R}$ and $\mathcal{U}$ in the syntax tree. The witnesses of a formula with quantifier depth d can be arranged as a d-dimensional comb. We assume, again, w.l.o.g. that no $\bigcirc$-modality occurs in the formula. Lemma 4.5 and Corollary 4.1 cover the case where the comb is 2-dimensional. We now lift the arguments to the general case. Given a d-dimensional comb, we associate the innermost witnesses with the corresponding runs on the d-tuple automata, which we build as before from the inner LTL formulas. A 3-dimensional comb and the corresponding automaton runs are exemplarily depicted in Figure 7.

In the d-dimensional case, diagonals are hyperplanes. We represent the k-th plane D_k in the d-dimensional comb by a nested sequence of depth $d-1$.

$$
\begin{aligned}
D_k &:= [D_{k,0}, \ldots, D_{k,k}] \\
D_{k,i_1} &:= [D_{k,i_1,0} \ldots, D_{k,i_1,k-i_1}] \\
&\vdots \\
D_{k,i_1,\ldots,i_{d-2}} &:= [p_{i_1,\ldots,i_{d-2},0}[s], p_{i_1,\ldots,i_{d-2},1}[s-1], \ldots \\
&\qquad p_{i_1,\ldots,i_{d-2},s}[0]], \text{ where } s = k - (i_1 + \ldots + i_{d-2})
\end{aligned}
$$

We additionally define d-dimensional frontiers as nested sets of automata states.

$$\begin{aligned} F_k &\coloneqq \{F_{k,i_1} \mid i_1 \leq k, i_1 \in \mathbb{N}\} \\ F_{k,i_1} &\coloneqq \{F_{k,i_1,i_2} \mid i_1 + i_2 \leq k, i_2 \in \mathbb{N}\} \\ &\vdots \\ F_{k,i_1,\ldots,i_{d-2}} &\coloneqq \{r_{i_1,\ldots,i_{d-2},i_{d-1}}[i_d] \mid \\ &\qquad i_{d_1} + i_d = k - (i_1 + \ldots + i_{d-2})\} \end{aligned}$$

As an example, in Fig. 7, $[p_{0,0}[1], p_{0,1}[0]]$, and $[p_{1,0}[0]]$ constitute plane D_1. The corresponding frontier is giving by $F_1 = \{F_{1,0}, F_{1,1}\} = \{\{r_{0,0}[1], r_{0,1}[0]\}, \{r_{1,0}[0]\}\}$.

We define the *cuttable* property recursively, with the definition for the 2-dimensional case (c.f. Lemma 4.5) as base case. For indices $i_1, \ldots, i_l$ we also write $\bar{i}$. Two sub-planes $D_{k,\bar{i}}$, $D_{k',\bar{i}'}$ with $F_{k,\bar{i}} = F_{k',\bar{i}'}$ are *cuttable* if for every two $F_{k',\bar{i}',j'} \in F_{k',\bar{i}'}$ and $F_{k,\bar{i},j} \in F_{k,\bar{i}}$ with $F_{k',\bar{i}',j'} = F_{k,\bar{i},j}$, sub-frontier $F_{k,\bar{i},j}$ is associated with at least as many subsequences in $D_{k,\bar{i}}$ as $F_{k',\bar{i}'}$ is associated with in $D_{k',\bar{i}'}$, or at least with $|Q|$ many (where Q is the set of automata states). Furthermore, if $F_{k,\bar{i},j}$ is associated with $D_{k,\bar{i},j}$, and $F_{k',\bar{i}',j'}$ is associated with $D_{k',\bar{i}',j'}$, the corresponding sub-planes must be cuttable again.

For the sake of explainability and compactness, we extend the *cut* operation only to the 3-dimensional case. The d-dimensional case generalizes the definition. A plane in the 3-dimensional comb is a sequence of sequences $D_{k,i}$ of nodes. Each sequence $D_{k,i}$ contains nodes that reside on paths branching from p_i. To cut plane $D_{k'}$ to D_k, we require that $F_k = F_{k'}$. Pick for each set $F_{k,i}$ an equal set $F_{k',i'}$ and cut the diagonal $D_{k',i'}$ to $D_{k,i}$, as described for the 2-dimensional case in the proof of Lemma 4.5. To preserve the relations of paths in the third dimension ($p_{i,j}$) with the paths in the first and second dimension (p and p_i), also replace each 2-dimensional sub-comb with origin at $p_i[k-i]$ with the sub-comb at $p_{i'}[k'-i']$; and the 3-dimensional sub-comb with origin at $p[k]$ with the sub-comb $p[k']$. Using the lifted definition of cuttable, it is also possible to define *preserving cuts* by first declaring a set of representative positions and then choosing the sets in such a way that no representative positions are deleted during the cut.

With the same arguments as in the 2-dimensional case and using the lifted (preserving) cut operation, we can create a 3-dimensional comb prefix $\mathcal{T}_{fin}$ of bounded size which can then be extended to a satisfying model $\widetilde{\mathcal{T}}$. Note that the bound in the 3-dimensional case is exponentially larger than the bound in the 2-dimensional case due to the more complicated definition of cuttable. □

Theorem 4.7. *The satisfiability problem for HyperCTL* formulas in the $\exists^*\forall^*$ fragment is decidable.*

Proof. The proof is similar to the proof for Theorem 4.2. Let an $\exists^*\forall^*$ HyperCTL* formula φ be given with at least one existential quantifier, the case with no existential quantifier is handled by Lemma 4.4. We again eliminate the universal quantification by explicitly enumerating every possible interaction between the universal and existential quantifiers. The resulting formula is in the $\exists^*$ fragment, which can be decided (cf. Lemma 4.6). □

Theorem 4.8. *The satisfiability problem for HyperCTL* formulas in the $\forall\exists$ fragment is undecidable.*

Proof. HyperCTL* subsumes HyperLTL as a syntactic fragment. As in Theorem 4.3, the undecidability of the $\forall\exists$ HyperCTL* fragment follows from the undecidability of the $\forall\exists$ HyperLTL fragment [19]. □

We have thus identified the largest decidable fragment of HyperCTL*, which is, as for HyperLTL and HyperQPTL, the $\exists^*\forall^*$ fragment.

5. Conclusion and Future Work

In this paper, we have developed a comprehensive expressiveness hierarchy of linear-time and branching-time hyperlogics. In the linear-time spectrum, we studied the relationships between HyperLTL, FO[$<, E$], HyperQPTL and S1S[E]. In the branching-time spectrum, we studied HyperCTL*, MPL[E], HyperQCTL* and MSO[E]. Our results show significant differences between the hierarchy of the standard logics and the hierarchy of the hyperlogics. Most significantly, the equivalences between QPTL and S1S, and CTL* and MPL do not translate to the corresponding hyperlogics: HyperQPTL is not equivalent to S1S[E], and HyperCTL* is not equivalent to MPL[E]. The only equivalence that translates to the hierarchy of hyperlogics is the equivalence between QCTL* and MSO: HyperQCTL* and MSO[E] are equivalent. The reason for this equivalence is that on *trees*, it is possible to simulate the equal-level predicate and the second-order quantification by using propositional HyperQCTL* quantification (as described in the proof of Theorem 3.9).

Our results also identify the decidability boundary of the satisfiability problem. The previous result here was that HyperLTL formulas in the $\exists^*\forall^*$ fragment are decidable [19]. We showed that the same holds for the more expressive logic HyperQPTL and that, hence, propositional quantification does not influence the decidability of the satisfiability problem in the linear-time hierarchy. In the branching-time hierarchy, we identified the largest decidable fragment of HyperCTL*, which is, again, the $\exists^*\forall^*$ fragment.

Our results indicate that the equal-level predicate adds, in most cases, more expressiveness than trace or path variables. This raises the question if alternatives to the equal-level predicate can be found that would lift logics like FO[$<$] to hyperlogics while preserving the equivalences to the temporal logics. Another direction for future work concerns the extension of the temporal logics with knowledge modalities [29]. It is known that KLTL and KCTL*, the extensions of LTL and CTL* with knowledge modalities under perfect recall semantics, are not subsumed by HyperLTL and HyperCTL*, respectively [27]. The question how extensions of HyperLTL and HyperCTL* with knowledge modalities fit into the expressiveness hierarchy is open.

References

[1] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, 1977, pp. 46–57. [Online]. Available: https://doi.org/10.1109/SFCS.1977.32

[2] E. A. Emerson and J. Y. Halpern, ""sometimes" and "not never" revisited: on branching versus linear time temporal logic," *J. ACM*, vol. 33, no. 1, pp. 151–178, 1986. [Online]. Available: https://doi.org/10.1145/4904.4999

[3] B. Finkbeiner and M. N. Rabe, "The linear-hyper-branching spectrum of temporal logics," *it - Information Technology*, vol. 56, pp. 273–279, November 2014.

[4] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez, "Temporal logics for hyperproperties," in *Principles of Security and Trust - Third International Conference, POST 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, 2014, pp. 265–284. [Online]. Available: https://doi.org/10.1007/978-3-642-54792-8_15

[5] J. A. Goguen and J. Meseguer, "Security policies and security models," in *1982 IEEE Symposium on Security and Privacy, Oakland, CA, USA, April 26-28, 1982*, 1982, pp. 11–20. [Online]. Available: https://doi.org/10.1109/SP.1982.10014

[6] S. Zdancewic and A. C. Myers, "Observational determinism for concurrent program security," in *16th IEEE Computer Security Foundations Workshop (CSFW-16 2003), 30 June - 2 July 2003, Pacific Grove, CA, USA*, 2003, p. 29. [Online]. Available: https://doi.org/10.1109/CSFW.2003.1212703

[7] W. Thomas, "Path logics with synchronization," in *Perspectives in Concurrency Theory*, K. Lodaya, M. Mukund, and R. Ramanujam, Eds. IARCS-Universities, Universities Press, 2009, pp. 469–481.

[8] B. Finkbeiner and M. Zimmermann, "The first-order logic of hyperproperties," in *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, 2017, pp. 30:1–30:14. [Online]. Available: https://doi.org/10.4230/LIPIcs.STACS.2017.30

[9] H. W. Kamp, "Tense logic and the theory of linear order," Ph.D. dissertation, Computer Science Department, University of California at Los Angeles, USA, 1968.

[10] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi, "On the temporal analysis of fairness," in *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '80. New York, NY, USA: ACM, 1980, pp. 163–173. [Online]. Available: http://doi.acm.org/10.1145/567446.567462

[11] A. P. Sistla, "Theoretical issues in the design and verification of distributed systems," Ph.D. dissertation, 1983.

[12] Y. Kesten and A. Pnueli, "A complete proof systems for QPTL," in *Proceedings, 10th Annual IEEE Symposium on Logic in Computer Science, San Diego, California, USA, June 26-29, 1995*, 1995, pp. 2–12. [Online]. Available: https://doi.org/10.1109/LICS.1995.523239

[13] F. Moller and A. M. Rabinovich, "On the expressive power of CTL," in *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. IEEE Computer Society, 1999, pp. 360–368. [Online]. Available: https://doi.org/10.1109/LICS.1999.782631

[14] K. R. Abrahamson, "Decidability and expressiveness of logics of processes," Ph.D. dissertation, Seattle, WA, USA, 1980, aAI8109709.

[15] T. French, "Decidability of quantifed propositional branching time logics," in *AI 2001: Advances in Artificial Intelligence, 14th Australian Joint Conference on Artificial Intelligence, Adelaide, Australia, December 10-14, 2001, Proceedings*, 2001, pp. 165–176. [Online]. Available: https://doi.org/10.1007/3-540-45656-2_15

[16] F. Laroussinie and N. Markey, "Quantified CTL: expressiveness and complexity," *Logical Methods in Computer Science*, vol. 10, no. 4, 2014. [Online]. Available: https://doi.org/10.2168/LMCS-10(4:17)2014

[17] O. Kupferman, N. Piterman, and M. Y. Vardi, "From liveness to promptness," *Formal Methods in System Design*, vol. 34, no. 2, pp. 83–103, Apr 2009. [Online]. Available: https://doi.org/10.1007/s10703-009-0067-z

[18] A. P. Sistla and E. M. Clarke, "The complexity of propositional linear temporal logics," *J. ACM*, vol. 32, no. 3, pp. 733–749, 1985. [Online]. Available: https://doi.org/10.1145/3828.3837

[19] B. Finkbeiner and C. Hahn, "Deciding hyperproperties," in *27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada*, 2016, pp. 13:1–13:14. [Online]. Available: https://doi.org/10.4230/LIPIcs.CONCUR.2016.13

[20] E. A. Emerson and A. P. Sistla, "Deciding full branching time logic," *Information and Control*, vol. 61, no. 3, pp. 175–201, 1984. [Online]. Available: https://doi.org/10.1016/S0019-9958(84)80047-9

[21] M. N. Rabe, "A temporal logic approach to information-flow control," Ph.D. dissertation, Saarland University, 2016. [Online]. Available: http://scidok.sulb.uni-saarland.de/volltexte/2016/6387/

[22] A. P. Sistla, M. Y. Vardi, and P. Wolper, "The complementation problem for büchi automata with applications to temporal logic (extended abstract)," in *Automata, Languages and Programming, 12th Colloquium, Nafplion, Greece, July 15-19, 1985, Proceedings*, 1985, pp. 465–474. [Online]. Available: https://doi.org/10.1007/BFb0015772

[23] R. McNaughton and S. A. Papert, *Counter-Free Automata (MIT research monograph no. 65)*. The MIT Press, 1971.

[24] C. Baier and J. Katoen, *Principles of model checking*. MIT Press, 2008.

[25] R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi, *Reasoning About Knowledge*. MIT press, 2004.

[26] J. Y. Halpern and M. Y. Vardi, "The complexity of reasoning about knowledge and time. i. lower bounds," *J. Comput. Syst. Sci.*, vol. 38, no. 1, pp. 195–237, 1989. [Online]. Available: https://doi.org/10.1016/0022-0000(89)90039-1

[27] L. Bozzelli, B. Maubert, and S. Pinchinat, "Unifying hyper and epistemic temporal logics," in *Foundations of Software Science and Computation Structures - 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, 2015, pp. 167–182. [Online]. Available: https://doi.org/10.1007/978-3-662-46678-0_11

[28] M. Y. Vardi and P. Wolper, "Reasoning about infinite computations," *Inf. Comput.*, vol. 115, no. 1, pp. 1–37, 1994. [Online]. Available: https://doi.org/10.1006/inco.1994.1092

[29] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi, *Reasoning About Knowledge*. MIT Press, 1995.

Appendix A. Proof of Lemma 3.6

A.1. MPL[E] Subsumes HyperCTL*

Like in Lemma 3.3, we use the idea from [8] where $\{y_1, y_2, \ldots\}$ are first-order variables used to indicate time. We use second-order quantification to quantify traces. Given a HyperKCTL* formula φ, we inductively construct an MPL[E] formula.

$$
\begin{aligned}
\mathrm{mpe}(a_\pi, y_i) &= \exists x.\ x \in X_\pi \wedge E(x, y_i) \wedge P_a(x) \\
&\quad \text{where } X_\pi \text{ is the second-order variable used for path } \pi \\
\mathrm{mpe}(\neg\varphi_1, y_i) &= \neg\mathrm{mpe}(\varphi_1, y_i) \\
\mathrm{mpe}(\varphi_1 \vee \varphi_2, y_i) &= \mathrm{mpe}(\varphi_1, y_i) \vee \mathrm{mpe}(\varphi_2, y_i) \\
\mathrm{mpe}(\bigcirc \varphi_1, y_i) &= \exists y_j > y_i.\ \neg(\exists y_k.\ y_i < y_k < y_j) \\
&\qquad \wedge \mathrm{mpe}(\varphi_1, y_j) \\
\mathrm{mpe}(\varphi_1 \,\mathcal{U}\, \varphi_2, y_i) &= \exists y_j.\ y_i \leq y_j \wedge \mathrm{mpe}(\varphi_2, y_j) \\
&\qquad \wedge \forall y_k.\ y_i \leq y_k < y_j \\
&\qquad \rightarrow \mathrm{mpe}(\varphi_1, y_k) \\
\mathrm{mpe}(\exists\pi.\varphi_1, y_i) &= \exists X_\pi.\ \mathit{prefix}(X_\pi, X_\varepsilon, y_i) \\
&\qquad \wedge \mathrm{mpe}(\varphi_1, y_i) \\
&\quad \text{where } X_\pi \text{ is now used as trace variable for } \pi \\
&\text{where} \\
\mathit{prefix}(X_\pi, X_\varepsilon, y_i) \coloneqq\ & \forall y_i' \leq y_i.\ \forall x.\ E(x, y_i') \rightarrow \\
& \quad (x \in X_\pi \leftrightarrow x \in X_\varepsilon) \\
& \quad \rightarrow x =_A x'
\end{aligned}
$$

where we use X_ε to denote the second-order variable that was quantified most recently (i.e. closest in the scope to X_π). The MPL[E] formula $\mathrm{mpe}(\varphi, 0)$ is equivalent to the HyperCTL* formula φ, which can be shown by a straightforward induction.

A.2. MSO[E] Can Express the Knowledge Modality

Extend the translation from HyperCTL* to MPL[E] given above with an additional rule for $\mathcal{K}_{A,\pi}$.

$$
\begin{aligned}
\mathrm{mpe}(\mathcal{K}_{A,\pi}.\varphi_1, y_i) &= \forall X_\pi'.\ \mathit{eqPre}(X_\pi, X_\pi', y_i, A) \\
&\qquad \rightarrow \mathrm{mpe}(\varphi_1, y_i) \\
&\quad \text{where } X_\pi' \text{ is now used as trace variable for } \pi \\
&\text{where} \\
\mathit{eqPre}(X_\pi, X_\pi', y_i, A) \coloneqq\ & \forall x \in X_\pi. \forall x' \in X_\pi'.\ (\exists y_i' \leq y_i. \\
& \quad E(x, y_i') \wedge E(x', y_i')) \\
& \quad \rightarrow x =_A x'
\end{aligned}
$$

Appendix B. Proof of Theorem 3.9

B.1. HyperQCTL* Subsumes MSO[E]

Let φ be a MSO[E] formula over AP. We define an equivalent HyperQCTL* formula $\mathrm{hqc}(\varphi)$ as follows.

$$
\begin{aligned}
\mathrm{hqc}(P_a(x)) &= \Diamond(q^x_{\pi_x} \wedge a_{\pi_x}) \\
\mathrm{hqc}(x \in X) &= \Diamond(q^x_{\pi_x} \wedge q^X_{\pi_x}) \\
\mathrm{hqc}(x < y) &= \Diamond(q^x_{\pi_x} \wedge \bigcirc\Diamond q^y_{\pi_y}) \\
\mathrm{hqc}(x = y) &= \Diamond(q^x_{\pi_x} \wedge q^y_{\pi_y}) \wedge \forall q.\Box(q_{\pi_x} \leftrightarrow q_{\pi_y}) \\
\mathrm{hqc}(E(x, y)) &= \Diamond(q^x_{\pi_x} \wedge q^y_{\pi_y}) \\
\mathrm{hqc}(\neg\varphi_1) &= \neg\mathrm{hqc}(\varphi_1) \\
\mathrm{hqc}(\varphi_1 \vee \varphi_2) &= \mathrm{hqc}(\varphi_1) \vee \mathrm{hqc}(\varphi_2) \\
\mathrm{hqc}(\exists x.\varphi_1) &= \exists\pi_x.\ \exists q^x.\ (\neg q^x_{\pi_x})\,\mathcal{U}(q^x_{\pi_x} \wedge \bigcirc\Box(\neg q^x_{\pi_x})) \\
&\qquad \wedge \mathrm{hqc}(\varphi_1) \\
\mathrm{hqc}(\exists X.\varphi_1) &= \exists q^X.\ \mathrm{hqc}(\varphi_1)
\end{aligned}
$$

Note that we used the fact that two paths are equal in HyperQCTL* iff a universal quantification of atomic proposition q always assigns q to be globally equivalent on the two paths. Using a straightforward induction, we can show that for every tree $\mathcal{T}$, $\mathcal{T} \models \varphi$ iff $\mathcal{T} \models \mathrm{hqc}(\varphi)$.

B.2. MSO[E] Subsumes HyperQCTL*

Given a HyperQCTL* formula φ, we inductively construct an MSO[E] formula ψ. The construction is very similar to the one described in Appendix A. We only need to make sure that the second-order quantification X_π, which encodes a quantified path π, encodes a full path of the tree.

$$
\begin{aligned}
\mathrm{mse}(a_\pi, y_i) &= \exists x.\ x \in X_\pi \wedge E(x, y_i) \wedge P_a(x) \\
&\quad \text{where } X_\pi \text{ is the second-order variable used for path } \pi \\
&\quad \text{and } a \text{ is not quantified by a propositional quantifier.} \\
\mathrm{mse}(q_\pi, y_i) &= \exists x.\ x \in X_\pi \wedge E(x, y_i) \wedge x \in X_q \\
&\quad \text{where } X_\pi \text{ is the second-order variable used for path } \pi \\
&\quad \text{and } q \text{ is quantified by a propositional quantifier.} \\
\mathrm{mse}(\neg\varphi_1, y_i) &= \neg\mathrm{mse}(\varphi_1, y_i) \\
\mathrm{mse}(\varphi_1 \vee \varphi_2, y_i) &= \mathrm{mse}(\varphi_1, y_i) \vee \mathrm{mse}(\varphi_2, y_i) \\
\mathrm{mse}(\bigcirc \varphi_1, y_i) &= \exists y_j.\ y_i < y_j \wedge \neg(\exists y_k.\ y_i < y_k < y_j) \\
&\qquad \wedge \mathrm{mse}(\varphi_1, y_j) \\
\mathrm{mse}(\varphi_1 \,\mathcal{U}\, \varphi_2, y_i) &= \exists y_j.\ y_i \leq y_j \wedge \mathrm{mse}(\varphi_2, y_j) \\
&\qquad \wedge \forall y_k.\ y_i \leq y_k < y_j \rightarrow \mathrm{mse}(\varphi_1, y_k) \\
\mathrm{mse}(\exists\pi.\varphi_1, y_i) &= \exists X_\pi.\ \mathit{path}(X_\pi) \wedge \\
&\qquad \mathit{prefix}(X_\pi, X_\varepsilon, y_i) \wedge \mathrm{mse}(\varphi_1, y_i) \\
&\quad \text{where } X_\pi \text{ is now used as trace variable for } \pi \\
\mathrm{mse}(\exists q.\varphi_1, y_i) &= \exists X_q.\mathrm{mse}(\varphi_1, y_i) \\
&\quad \text{where } X_q \text{ is now used as propositional variable for } q
\end{aligned}
$$

where

$$
\begin{aligned}
path(X_\pi) &:= (\exists x \in X_\pi.\ \neg\exists x'.x' < x) \wedge \\
&\quad \forall x \in X_\pi.\ \exists x' \in X_\pi.\ succ(x, x') \wedge \\
&\quad \neg(\exists x'' \in X_\pi.\ succ(x, x'')) \\
succ(x, x') &:= x < x' \wedge \neg(\exists x''.\ x < x'' < x') \\
prefix(X_\pi, X_\varepsilon, y_i) &:= \forall y_i' \leq y_i.\ \forall x.\ E(x, y_i') \rightarrow \\
&\quad (x \in X_\pi \leftrightarrow x \in X_\varepsilon)
\end{aligned}
$$

We use X_ε to denote the second-order variable that was quantified most recently (i.e. closest in the scope to X_π) and also encodes a path. Note that a set X_π encodes a full path of the tree if it contains the root node and a unique direct successor for every node in the set.