The Logic in Computer Science Column

by

Yuri Gurevich

Microsoft Research
One Microsoft Way, Redmond WA 98052, USA
gurevich@microsoft.com

Temporal Hyperproperties

Bernd Finkbeiner
Universität des Saarlandes

Abstract

Hyperproperties generalize trace properties, which are sets of traces, to sets of sets of traces. The most prominent application of hyperproperties is information flow security: information flow policies characterize the secrecy and integrity of a system by comparing two or more execution traces, for example by comparing the outputs on execution traces that result from different values of a secret input. HyperLTL is an extension of linear-time temporal logic (LTL) with explicit quantification over traces. In this overview paper, we survey recent results on the expressiveness of HyperLTL and on the satisfiability and model checking problems. We also consider HyperCTL*, the extension of HyperLTL to branching time, and HyperFOLTL, the extension of HyperLTL with first-order quantification.

1 Introduction

Temporal logics like LTL specify properties of a reactive system as a set of *traces*. A trace is an infinite sequence that indicates, for some run of the system, which atomic propositions are true in the positions of the run. A system, given for example as a Kripke structure, is correct, if its traces are contained in the set of traces allowed by the temporal formula.

Not all relevant system properties can be stated by referring to individual traces, however. A prominent example of a system property that requires references to more than one trace is *observational determinism* [26]. Observational determinism is an information flow security policy that specifies that the behavior of the system must appear deterministic to an observer who sees a certain subset of the atomic propositions. In other words, the observer cannot deduce any information about any not directly observable inputs. To state observational determinism, we must refer to *pairs* of traces: if two traces agree on the observable inputs, they must also agree on the observable outputs.

Clarkson and Schneider coined the term *hyperproperties* [5] for such properties. While a trace property is a set of traces, a hyperproperty is a *set of sets*

of traces. It is easy to see that LTL cannot capture hyperproperties, because its formulas only refer to individual traces. Perhaps more surprising is the fact that even branching-time temporal logics like CTL* are not expressive enough. CTL* extends LTL with the path path quantifiers E (there exists a path) and A (for all paths). For example, one can state that a system both has a computation path on whose trace some condition φ is true *and* a computation path where φ is false. The limitation, however, is that nesting the path quantifiers "forgets" the paths chosen in earlier quantifiers: the formulas can therefore only relate a finite amount of information between two traces (namely, whether or not a certain subformula evaluates to true). Relating an *infinite* amount of information, such as the infinite sequence of outputs, is impossible.

The subject of this overview paper is the extension of the temporal logics to hyperproperties. The idea is to use, instead of the path quantifiers E and A, quantifiers over trace variables, so that the traces chosen by earlier quantifiers remain accessible in the subformula. A *HyperLTL* [4] formula starts with a quantifier prefix, which binds traces to trace variables, followed by an LTL formula where the atomic propositions are indexed by the trace variables. Suppose, for example, that the observable input to a system is the atomic proposition i and the output is the atomic proposition o. Observational determinism can then be expressed as the HyperLTL formula

$$\forall \pi.\, \forall \pi'.\, \square(i_\pi \leftrightarrow i_{\pi'}) \rightarrow \square(o_\pi \leftrightarrow o_{\pi'}),$$

where $\square$ is the usual "globally" operator of temporal logic: if two traces π and π' agree globally on i, they must also globally agree on o.

Many information flow properties from the literature can be expressed naturally in HyperLTL. HyperLTL has also found applications beyond security, for example for error resistant codes, specifying properties like the minimum Hamming distance between code words. To understand the expressiveness of HyperLTL, it is useful to compare HyperLTL to other common logics. *Epistemic temporal logic* extends LTL with the knowledge operator. HyperLTL and epistemic temporal logic have incomparable expressiveness: there are properties that can be expressed in HyperLTL but not in epistemic temporal logic and the other way around. Another interesting point of reference is *first-order logic*. For LTL, Kamp's seminal theorem [13] (in the formulation of Gabbay et al. [12]) states that LTL is expressively equivalent to first-order logic FO[<] over the natural numbers with order. In order to express hyperproperties in first-order logic, we use relational structures that consist of disjoint copies of the natural numbers with order, one for each trace. To be able to compare positions on different traces, we add the *equal-level predicate* E (cf. [23]), which relates the same time points on different traces. This logic, which we call FO[<, E], is strictly more expressive

than HyperLTL [11]. HyperLTL corresponds to the fragment of the logic where the quantifiers can only be used in a guarded manner, by either referring to the initial position of some trace, or to some position of a previously chosen trace, as opposed to arbitrary elements of the domain.

This paper gives an overview on recent results on the satisfiability and model checking problems of HyperLTL. It turns out that the differences between LTL and HyperLTL are surprisingly profound. While the models of an LTL formula form an ω-regular language, and satisfiability can therefore be checked with a standard emptiness check on ω-automata, the satisfiability of HyperLTL is undecidable. Model checking remains decidable, but becomes, with increasing quantifier alternation depth, significantly more expensive than for LTL.

From a practical point of view, the most important fragment is the alternation-free fragment, i.e., the fragment with only existential or only universal quantifiers. Many useful system properties can be expressed in this fragment. For example, observational determinism can be expressed using only universal trace quantifiers. As a rule of thumb, the alternation-free fragment is no more expensive than LTL. Satisfiability and model checking remain in PSPACE, just like for LTL. Satisfiability and model checking for the $\exists^*\forall^*$ fragment, which is relevant because it contains implications and equivalences between alternation-free formulas, is still decidable, albeit in EXPSPACE. Satisfiability becomes undecidable as soon as the fragment allows for a $\forall\exists$ combination [7]. The model checking problem is decidable for arbitrary HyperLTL formulas but becomes exponentially more expensive with every further quantifier alternation [10].

We conclude the overview with two important extensions to HyperLTL. The first extension is from linear to branching time. HyperLTL is a linear-time logic. If we allow arbitrary nestings between path quantifiers and temporal operators, we obtain a branching-time logic called HyperCTL* [9]. The second extension is to add first-order quantifiers. This makes it possible to quantify over arbitrary sets of agents, such as the users of a system [8]. For example, we can state that a system appears deterministic to every user, even though each user makes different observations.

2 HyperLTL

Fix a finite set AP of atomic propositions. A trace over AP is a map $t\colon \mathbb{N} \to 2^{\mathrm{AP}}$, denoted by $t(0)t(1)t(2)\cdots$. Let $(2^{\mathrm{AP}})^\omega$ denote the set of all traces over AP.

LTL. The formulas of linear-time temporal logic (LTL) [19] are generated by the following grammar:

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

where $a \in \mathrm{AP}$ is an *atomic proposition*, the Boolean connectives $\neg$ and $\wedge$ have the usual meaning, $\bigcirc$ is the temporal *next* operator, and $\mathcal{U}$ is the temporal *until* operator. We also consider the usual derived Boolean connectives, such as $\vee$, $\rightarrow$, and $\leftrightarrow$, and the derived temporal operators *eventually* $\Diamond\varphi \equiv \mathit{true}\,\mathcal{U}\,\varphi$, *globally* $\Box\varphi \equiv \neg\Diamond\neg\varphi$, and *weak until*: $\varphi\,\mathcal{W}\,\psi \equiv (\varphi\,\mathcal{U}\,\psi) \vee \Box\varphi$. The satisfaction of an LTL formula φ over a trace t, denoted by $t \models \varphi$, is defined as follows:

$$
\begin{array}{lll}
t \models a & \text{iff} & a \in t(0), \\
t \models \neg\varphi & \text{iff} & t \not\models \varphi, \\
t \models \varphi_1 \wedge \varphi_2 & \text{iff} & t \models \varphi_1 \text{ and } t \models \varphi_2, \\
t \models \bigcirc\varphi & \text{iff} & t[1,\infty] \models \varphi, \\
t \models \varphi_1 \mathcal{U} \varphi_2 & \text{iff} & \exists i \geq 0 :\ t[i,\infty] \models \varphi_2 \ \wedge\ \forall 0 \leq j < i :\ t[j,\infty] \models \varphi_1.
\end{array}
$$

For example, the LTL formula $\Box(a \rightarrow \Diamond b)$ specifies that every position in which a is true must eventually be followed by a position where b is true.

LTL formulas define ω-regular languages: each LTL formula can be translated into a Büchi automaton over the alphabet $\Sigma = 2^{\mathrm{AP}}$ that accepts precisely the traces that satisfy the formula [18, 24].

HyperLTL. The formulas of HyperLTL [4] are generated by the grammar

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

where a ranges over atomic propositions in AP and where π ranges over a given countable set $\mathcal{V}$ of *trace variables*. Further Boolean connectives and the temporal operators $\Diamond$, $\Box$, and $\mathcal{W}$ are derived as for LTL. A sentence is a closed formula, i.e., the formula has no free trace variables. A formula is in negation normal form if the only occurrences of $\neg$ occur in front of propositions a_π. Both LTL and HyperLTL formulas can always be brought into negation normal form.

The semantics of HyperLTL is defined with respect to a trace assignment, a partial mapping $\Pi\colon \mathcal{V} \rightarrow (2^{\mathrm{AP}})^\omega$. The assignment with empty domain is denoted by $\Pi_\emptyset$. Given a trace assignment Π, a trace variable π, and a trace t we denote by $\Pi[\pi \rightarrow t]$ the assignment that coincides with Π everywhere but at π, which is mapped to t. Furthermore, $\Pi[j,\infty]$ denotes the assignment mapping every π in Π's domain to $\Pi(\pi)(j)\Pi(\pi)(j+1)\Pi(\pi)(j+2)\cdots$. The satisfaction of a HyperLTL formula φ over a trace assignment Π and a set of traces T, denoted by $T, \Pi \models \varphi$, is defined as follows:

$$
\begin{array}{lll}
T, \Pi \models a_\pi & \text{iff} & a \in \Pi(\pi)(0), \\
T, \Pi \models \neg\psi & \text{iff} & T, \Pi \not\models \psi, \\
T, \Pi \models \psi_1 \wedge \psi_2 & \text{iff} & T, \Pi \models \psi_1 \text{ and } T, \Pi \models \psi_2, \\
T, \Pi \models \bigcirc \psi & \text{iff} & T, \Pi[1, \infty] \models \psi, \\
T, \Pi \models \psi_1 \,\mathcal{U}\, \psi_2 & \text{iff} & \exists i \geq 0 : T, \Pi[j, \infty] \models \psi_2 \\
 & & \wedge \forall 0 \leq j < i : T, \Pi[j', \infty] \models \psi_1, \\
T, \Pi \models \exists \pi.\, \varphi & \text{iff} & \exists t \in T : T, \Pi[\pi \to t] \models \psi, \\
T, \Pi \models \forall \pi.\, \varphi & \text{iff} & \forall t \in T : T, \Pi[\pi \to t] \models \psi.
\end{array}
$$

We say that a set T of traces satisfies a sentence φ, denoted by $T \models \varphi$, if $T, \Pi_\emptyset \models \varphi$.

System Properties. A *Kripke structure* is a tuple $K = (S, s_0, \delta, \text{AP}, L)$ consisting of a set of states S, an initial state s_0, a transition function $\delta : S \to 2^S$, a set of *atomic propositions* AP, and a *labeling function* $L : S \to 2^{\text{AP}}$. We require that each state has a successor, that is $\delta(s) \neq \emptyset$, to ensure that every execution of a Kripke structure can always be continued to infinity. A *path* of a Kripke structure is an infinite sequence $s_0 s_1 \ldots \in S^\omega$ such that s_0 is the initial state of K and $s_{i+1} \in \delta(s_i)$ for all $i \in \mathbb{N}$. By $\mathit{Paths}(K, s)$, we denote the set of all paths of K starting in state $s \in S$. A *trace* of a path $\sigma = s_0 s_1 \ldots$ is a sequence of labels $l_0 l_1 \ldots$ with $l_i = L(s_i)$ for all $i \in \mathbb{N}$. $\mathit{Tr}(K, s)$ is the set of all traces of paths of a Kripke structure K starting in state s. A Kripke structure K with initial state s_0 satisfies an LTL formula φ, denoted by $K \models \varphi$ iff for all traces $\pi \in \mathit{Tr}(K, s_0)$, it holds that $\pi \models \varphi$. Likewise, the Kripke structure satisfies a HyperLTL formula φ, also denoted by $K \models \varphi$, iff $\mathit{Tr}(K, s_0) \models \varphi$.

Examples. Many information flow security properties are hyperproperties. HyperLTL has also found applications beyond security, in particular in the specification of error resistant codes. In the following, we give a few examples from these areas. The examples are taken from [4, 10, 21], where further details and additional properties can be found.

Noninference [16] specifies that the behavior observable by a low-security observer must not change when all high-security inputs are replaced by a dummy input λ. Noninference can be expressed in HyperLTL as follows:

$$\forall \pi. \exists \pi'. \, (\square \lambda_{\pi'}) \;\wedge\; \pi =_L \pi'$$

where $\lambda_{\pi'}$ expresses that all of the high inputs in the current state of π' are λ, and $\pi =_L \pi'$ is short for $\bigwedge_{v \in V_L} v_\pi \leftrightarrow v_{\pi'}$, expressing that the variables V_L that are observable by some low-security observer have the same values in π and π'.

Generalized noninterference [15] is similar to observational determinism in that it requires that the outputs observed by a low-security observer should not be influenced by high-security inputs. Unlike observational determinism, it allows, however, for nondeterminism in the low-observable behavior:

$$\forall\pi.\forall\pi'.\exists\pi''.\ \pi =_{H,\mathsf{in}} \pi''\ \wedge\ \pi' =_{L} \pi''$$

The existentially quantified trace π'' is an interleaving of the high inputs of the universally quantified trace π and the low inputs and outputs of the universally quantified trace π'.

Quantitative information-flow policies limit the leakage of information to a certain rate. The following HyperLTL formula expresses that there is no tuple of $2^n + 1$ low-distinguishable traces (cf. [22, 25]):

$$\forall\pi_0.\ \ldots\ .\forall\pi_{2^n}.\left(\bigvee_i \pi_i \neq_{L,\mathsf{in}} \pi_0\right) \vee \bigvee_{i\neq j} \pi_i =_{L,\mathsf{out}} \pi_j$$

Error resistant codes transmit data over noisy channels. A typical correctness condition for such a code is that all code words have a minimal Hamming distance [10]. The following HyperLTL property guarantees specifies that all code words produced by an encoder have a minimal Hamming distance of d:

$$\forall\pi.\forall\pi'.\diamondsuit(\bigvee_{a\in I} a_\pi \neq a_{\pi'}) \Rightarrow \neg\mathrm{Ham}_O(d-1,\pi,\pi')$$

where the atomic propositions in I represent the input data, and the atomic propositions in O represent the output code words. The subformula $\mathrm{Ham}_O(d,\pi,\pi')$ is defined recursively as follows:

$$\begin{aligned}\mathrm{Ham}_O(-1,\pi,\pi') &= \mathit{false}\\ \mathrm{Ham}_O(d,\pi,\pi') &= (\textstyle\bigwedge_{a\in O} a_\pi = a_{\pi'})\,\mathcal{W}\,(\bigvee_{a\in O} a_\pi \neq a_{\pi'}\ \wedge\ \bigcirc\mathrm{Ham}_O(d-1,\pi,\pi')).\end{aligned}$$

3 Expressiveness

In order to better understand the expressiveness of HyperLTL, we relate the logic to several other common logics. The first comparison is with the standard temporal logics LTL and CTL*. We then consider epistemic temporal logic, which reasons about the *knowledge* of an agent. Finally, we relate HyperLTL to first-order logic in the spirit of Kamp's theorem for LTL.

3.1 HyperLTL vs. LTL and CTL*

We begin with the standard temporal logics LTL and CTL*. As discussed in the introduction, both logics cannot express hyperproperties, because they can only reason about a single trace at a time. An example of a property that can be expressed in HyperLTL, but in neither LTL nor CTL*, is observational determinism.

LTL. Since LTL is a sublogic of HyperLTL (consisting of the sentences with a single universal trace quantifier), HyperLTL obviously subsumes LTL. On the other hand, Alur et al. [2] showed that observational determinism is not an ω-regular tree property. Since every LTL formula can be translated to a Büchi automaton, observational determinism can therefore not be expressed in LTL.

Theorem 1. *[4] HyperLTL is strictly more expressive than LTL.*

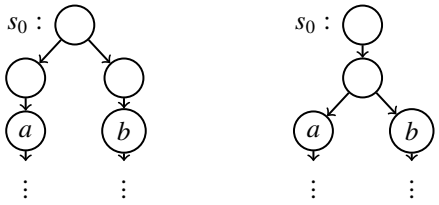
This fact can also be shown by the following direct argument (taken from [9]). Suppose that there is an LTL formula φ that expresses observational determinism. The set of traces T that satisfy the formula cannot be the full set of traces (with respect to some non-empty set of atomic propositions), because in that case all Kripke structures would satisfy the property. We pick a trace not in T and consider a Kripke structure that only allows this trace. Since this Kripke structure only has a single trace, it obviously satisfies observational determinism; but since that trace is not in T, it violates φ, contradicting our assumption that φ expresses observational determinism.

CTL*. CTL* is generated by the following grammar of state formulas Φ and path formulas φ:

$$\begin{array}{lllllllllll} \Phi & ::= & a & | & \neg\Phi & | & \Phi \wedge \Phi & | & \mathrm{A}\varphi & | & \mathrm{E}\varphi \\ \varphi & ::= & \Phi & | & \neg\varphi & | & \varphi \wedge \varphi & | & \bigcirc\varphi & | & \varphi\,\mathcal{U}\,\varphi \end{array}$$

CTL* state formulas Φ are interpreted over states and path formulas φ are interpreted over paths of a given Kripke structure. For a Kripke structure $K = (S, s_0, \delta, \mathrm{AP}, L)$ and a state $s \in S$, we define $s \models_K \mathrm{A}\varphi$ iff $\forall p \in \mathit{Paths}(K, s) : p \models \varphi$ and, symmetrically, $s \models_K \mathrm{E}\varphi$ iff $\exists p \in \mathit{Paths}(K, s) : p \models \varphi$. The semantics of temporal operators for paths corresponds to their interpretation over traces in LTL.

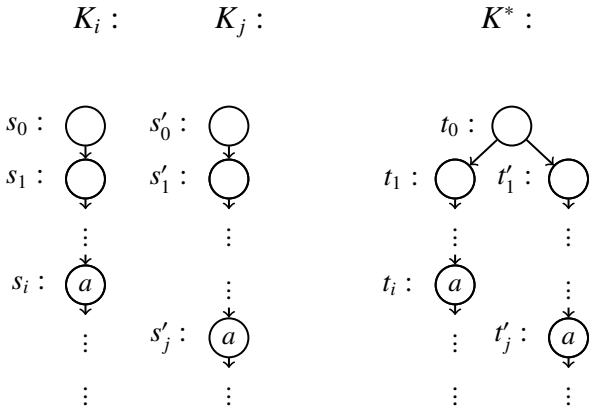
CTL* and HyperLTL have incomparable expressiveness. For example, the CTL* formula $\mathrm{A}\bigcirc(\mathrm{E}\bigcirc a) \wedge \mathrm{E}\bigcirc b$ distinguishes the following pair of Kripke structures, which cannot be distinguished by a HyperLTL formula:



On the other hand, CTL* cannot express observational determinism, which can be expressed in HyperLTL.

Theorem 2. *[4] HyperLTL and CTL* have incomparable expressiveness.*

We again show that observational determinism cannot be expressed in CTL* with a direct argument (which is again taken from [9]). Suppose that there is a CTL* formula Φ that expresses observational determinism. Consider the following family of observationally deterministic Kripke structures $K_1, K_2, \ldots$, where each K_i consists of a single branch from the initial state that only has one label a at step i:



All members of this family trivially satisfy observational determinism. We pick a pair K_i and K_j with $i \neq j$ of Kripke structures such that s_1 and s'_1 satisfy the same subformulas of Φ. Such a pair of Kripke structures must exist as φ has finitely many subformulas and the family of Kripke structures is infinite. We merge K and K' into one Kripke structure K^*, such that they share only the initial state as depicted above. By construction, states s_1, s'_1, t_1, and t'_1 all fulfill the same subformulas. The states t_0 and s_0 have the same label (none) and all their successors satisfy the same subformulas of φ. Hence, they also satisfy the same subformulas of Φ. In particular, K^* satisfies Φ but not observational determinism, which contradicts the assumption.

3.2 HyperLTL vs. Epistemic Temporal Logic

The epistemic temporal logics extend the temporal logics with a *knowledge operator* $\mathcal{K}_A\varphi$, which states that an agent A *knows* some fact φ. Knowing some fact means that this fact is true on all traces that are observationally equivalent from the agent's point of view.

In the following, we consider the epistemic temporal logic LTL_K [17, 6], interpreted over Kripke structures (cf. [21]). The syntax of LTL_K is that of LTL extended with the knowledge operator:

$$\varphi \;::=\; a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \mathcal{K}_A\varphi$$

An agent is identified by the set $A \subseteq \text{AP}$ of atomic propositions that are visible to the agent.

The satisfaction of a LTL_K formula over a trace of a Kripke structure K with initial state s_0 is given according to the following relation, where $t, i \models \varphi$ for a trace t and a natural number i indicates that φ holds on t at positition i:

$$\begin{array}{lll}
t, i \models a & \text{iff} & a \in t(i), \\
t, i \models \bigcirc\varphi & \text{iff} & t, i+1 \models \varphi, \\
t, i \models \varphi_1 \mathcal{U} \varphi_2 & \text{iff} & \exists k \geq i :\ t, k \models \varphi_2 \ \wedge\ \forall i \leq j < k :\ t, j \models \varphi_1, \\
t, i \models \mathcal{K}_A\varphi & \text{iff} & \forall t' \in Tr(K, s_0) :\ \text{if } t[0,i] =_A t'[0,i] \text{ then } t', i \models \varphi,
\end{array}$$

where $t[0,i] =_A t'[0,i]$ denotes that t and t' are equivalent on all atomic propositions in A on positions in $[0,i]$. A Kripke structure K satisfies a LTL_K formula φ, denoted by $K \models \varphi$, iff for all traces $t \in Tr(K, s_0)$ it holds that $t, 0 \models \varphi$.

There are many properties that can be expressed in both LTL_K and HyperLTL. For example, the LTL_K formula

$$\square\mathcal{K}_a b,$$

which states that an observer who sees a knows that b is true, can be expressed as the HyperLTL formula

$$\forall\pi.\,\forall\pi'.\, b_{\pi'} \mathcal{W} (a_\pi \nleftrightarrow a_{\pi'}).$$

However, there are LTL_K formulas that cannot be expressed in HyperLTL and vice versa[1].

[1]The original semantics of HyperLTL [4] differs slightly from the semantics presented in this paper. In the original semantics, HyperLTL subsumes LTL_K. The reason is that the original semantics allows for different sets of atomic propositions in the Kripke structure and in the HyperLTL formula. Propositions that occur in the formula but not in the Kripke structure can be used to simulate existential quantification over atomic propositions in the HyperLTL formula. The extension of HyperLTL with quantification over atomic propositions (strictly) subsumes LTL_K [21].

Theorem 3. *[3] HyperLTL and LTL_K have incomparable expressiveness.*

The following examples are due to Bozzelli et al. [3]: On the one hand, there is no HyperLTL formula equivalent to the LTL_K formula

$$\bigcirc\Diamond\mathcal{K}_\emptyset a.$$

Intuitively, the reason is that the formula relates at some point in time an *unbounded* number of traces. On the other hand, there is no LTL_K formula equivalent to the HyperLTL formula

$$\exists\pi.\exists\pi'.\ a_\pi \wedge \neg a_{\pi'} \wedge \bigcirc\Box(a_\pi \leftrightarrow a_{\pi'})).$$

3.3 HyperLTL vs. First-Order Logic

For LTL, Kamp's seminal theorem [13] (in the formulation of Gabbay et al. [12]) states that LTL is expressively equivalent to first-order logic FO[<] over the natural numbers with order. In order to formulate a corresponding "Kamp's theorem for HyperLTL," we encode sets of traces as relational structures. Following [11], we consider here relational structures that consist of disjoint copies of the natural numbers with order, one for each trace. To be able to compare positions on different traces, we add the *equal-level predicate* E (cf. [23]), which relates the same time points on different traces.

FO[<, E] is the first-order logic generated by the following grammar:

$$\varphi \ ::= \ x = y \mid x < y \mid P_a(x) \mid \mathrm{E}(x,y) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists x.\,\varphi$$

Here, x and y are first-order variables, and $a \in \mathrm{AP}$ is an atomic proposition.

Given a set $T \subseteq (2^{\mathrm{AP}})^\omega$ of traces over AP, the relational structure $\underline{T} = (T \times \mathbb{N}, <^{\underline{T}}, (P_a^{\underline{T}})_{a\in\mathrm{AP}}, \mathrm{E}^{\underline{T}})$ is defined as follows:

- $<^{\underline{T}} = \{((t,n),(t,n')) \mid t \in T \text{ and } n < n' \in \mathbb{N}\}$,
- $P_a^{\underline{T}} = \{(t,n) \mid t \in T, n \in \mathbb{N} \text{ and } a \in t(n)\}$, and
- $\mathrm{E}^{\underline{T}} = \{((t,n),(t',n)) \mid t,t' \in T \text{ and } n \in \mathbb{N}\}$.

Let $\mathcal{V}_1$ be the set of first-order variables. The semantics of FO[<, E] is defined with respect to a variable assignment, a partial mapping $\alpha : \mathcal{V}_1 \to T \times \mathbb{N}$. The assignmment with empty domain is denoted by $\alpha_\emptyset$. The satisfaction of a FO[<, E] formula φ over a relational structure $\underline{T}$ and a variable assignment α, denoted by $\underline{T}, \alpha \models \varphi$, is defined as follows:

$$
\begin{array}{lll}
\underline{T}, \alpha \models x = y & \text{iff} & \alpha(x) = \alpha(y), \\
\underline{T}, \alpha \models x < y & \text{iff} & \alpha(x) <^{\underline{T}} \alpha(y), \\
\underline{T}, \alpha \models P_a(x) & \text{iff} & \alpha(x) \in P_a^{\underline{T}}, \\
\underline{T}, \alpha \models \mathrm{E}(x, y) & \text{iff} & (\alpha(x), \alpha(y)) \in \mathrm{E}^{\underline{T}}, \\
\underline{T}, \alpha \models \neg\varphi & \text{iff} & \underline{T}, \alpha \not\models \varphi, \\
\underline{T}, \alpha \models \varphi \wedge \psi & \text{iff} & \underline{T}, \alpha \models \varphi \text{ and } \underline{T}, \alpha \models \psi, \\
\underline{T}, \alpha \models \exists x.\, \varphi & \text{iff} & \exists (t, n) \in T \times \mathbb{N} : \underline{T}, \alpha[x \mapsto (t, n)] \models \varphi,
\end{array}
$$

We say that a relational structure $\underline{T}$ satisfies a sentence φ, denoted by $\underline{T} \models \varphi$, if $\underline{T}, \alpha_\emptyset \models \varphi$. Every HyperLTL formula can be translated into an equivalent FO[<, E] formula. As an example, consider the HyperLTL formula

$$\forall \pi.\ \forall \pi'.\ \square(a_\pi \leftrightarrow a_{\pi'}).$$

The formula is equivalent to the FO[<, E] formula

$$\forall x.\ \forall y.\ \mathrm{E}(x, y) \rightarrow (P_a(x) \leftrightarrow P_a(y)).$$

FO[<, E] is, however, the more expressive logic. An example that can be expressed in FO[<, E] but not in HyperLTL is the property due to Bozzelli et al. [3], which we already used in the comparison between HyperLTL and epistemic temporal logic: there is a position n, such that the atomic proposition a is *false* at position n on all traces. This property is expressible in FO[<, E] as the following formula: $\exists x.\, \forall y.\, \mathrm{E}(x, y) \rightarrow \neg a$. There is no equivalent formula in HyperLTL [3].

HyperLTL corresponds exactly to a syntactic fragment of FO[<, E]. Let $\exists^M x.\, \varphi$ denote the FO[<, E] formula $\exists x.\ \min(x) \wedge \varphi$, where $\min(x) = \neg\exists y.\ \mathrm{Succ}(y, x)$ expresses that x is the first position of a trace and $\mathrm{Succ}(x, y) = x < y \wedge \neg\exists z.\, x < z < y$ expresses that y is the direct successor of x on some trace. Likewise, let $\forall^M x.\, \varphi$ denote $\forall x.\ \min(x) \rightarrow \varphi$, i.e., the quantifiers $\exists^M$ and $\forall^M$ only range over the first positions of a trace in $\underline{T}$. We use these quantifiers to mimic trace quantification in HyperLTL.

Furthermore, $\exists^G y \geq x.\, \varphi$ is shorthand for $\exists y.\, y \geq x \wedge \varphi$ and $\forall^G y \geq x.\, \varphi$ is shorthand for $\forall y.\, y \geq x \rightarrow \varphi$, i.e., the quantifiers $\exists^G$ and $\forall^G$ are guarded by a free variable x and range only over greater-or-equal positions on the same trace that x is on. We call the free variable x the *guard* of the quantifier.

We consider sentences of the form

$$\varphi = Q_1^M x_1. \cdots Q_k^M x_k.\ Q_1^G y_1 \geq x_{g_1}. \cdots Q_\ell^G y_\ell \geq x_{g_\ell}.\ \psi$$

with $Q \in \{\exists, \forall\}$, where we require the sets $\{x_1, \ldots, x_k\}$ and $\{y_1, \ldots, y_\ell\}$ to be disjoint, every guard x_{g_j} to be in $\{x_1, \ldots, x_k\}$, and ψ to be quantifier-free with free variables among the $\{y_1, \ldots, y_\ell\}$. This fragment is called HyperFO [11].

Theorem 4. *[11] HyperLTL and HyperFO are equally expressive. FO[<, E] is strictly more expressive than HyperLTL.*

4 Satisfiability

The HyperLTL satisfiability problem is to decide, for a given HyperLTL formula φ, whether or not there exists a set T of traces such that $T \models \varphi$. We begin with the satisfiability problem for the alternation-free fragment of HyperLTL. Universal formulas are particularly easy to decide, because we can restrict the models, without loss of generality, to singleton sets of traces: since all quantifiers are universal, every model with more than one trace could immediately be translated into another one where every trace except one is omitted. Hence, we can ignore the trace variables and interpret the HyperLTL formula as a plain LTL formula.

An existential formula may, in general, have more than one trace. For example, the models of $\exists \pi_1 \exists \pi_2.\ a_{\pi_1} \wedge \neg a_{\pi_2}$ have (at least) two traces. In order to reduce HyperLTL satisfiability again to LTL satisfiability, we *zip* such traces together. For this purpose, we introduce a fresh atomic proposition for every atomic proposition a and every path variable π that occur as an indexed proposition a_π in the formula. We obtain an equisatisfiable LTL formula by removing the quantifier prefix and replacing every occurrence of a_π with the new proposition.

Theorem 5. *[7] The satisfiability problem of the alternation-free fragment of HyperLTL is PSPACE-complete.*

The $\exists^*\forall^*$ fragment is especially interesting, because it includes implications between alternation-free formulas. The idea of the decision procedure is to eliminate the universal quantifiers by explicitly enumerating all possible interactions between the universal and existential quantifiers. This leads to an exponentially larger, but equisatisfiable existential formula.

Theorem 6. *[7] The satisfiability problem of the $\exists^*\forall^*$ fragment of HyperLTL is EXPSPACE-complete.*

Any extension beyond the already considered fragments makes the satisfiability problem undecidable. In the fragements considered so far, it was sufficient to consider models with a finite set of traces. It is easy to see that this is no longer the case if the formula has an existential quantifier in the scope of a universal quantifier. For example, any trace set that satisfies the formula $\forall \pi \exists \pi'.\ (\Diamond p) \wedge \Box(p \rightarrow (p \wedge \Box \neg p))$ contains at least one path for every natural number n, namely the path where the p occurs n steps after the first occurence of p on some path. In fact, it has been shown, via a reduction from Post's correspondence problem, that the satisfiability problem of the $\forall\exists$ fragment is undecidable [7].

Theorem 7. *[7] The satisfiability problem of the $\forall\exists$ fragment of HyperLTL is undecidable.*

5 Model Checking

The HyperLTL model checking problem is to decide, for a given Kripke structure K and a given HyperLTL formula ψ, whether or not $K \models \psi$. The model checking algorithm for HyperLTL described in the following is a simplified version of a model checking algorithm for the more expressive logic HyperCTL* [10] (see Section 6).

The complexity of the model checking problem depends strongly on the quantifier structure of ψ. We begin with the *alternation-free fragment* of HyperLTL, i.e., with the formulas that either contain only existential or only universal quantifiers. In the following, we assume without loss of generality that all quantifiers are existential. If the quantifiers are universal, we simply check the negated formula (which is existential): for an LTL formula φ over the indexed set of atomic propositions, $K \models \forall \pi_1 \dots \forall \pi_n.\ \varphi$ iff $K \not\models \exists \pi_1 \dots \pi_n.\ \neg\varphi$. The model checking procedure for an existential HyperLTL formula $\exists \pi_1 \dots \pi_n.\ \varphi$ consists of the following steps:

1. Convert the LTL formula φ over the indexed set of atomic propositions into an equivalent Büchi automaton over the alphabet $(2^{\mathrm{AP}})^n$. Each letter is a tuple of n sets of atomic propositions, where the ith element of the tuple represents the atomic propositions of trace π_i.

2. Eliminate existential quantifiers. Suppose we have already constructed a Büchi automaton over alphabet $(2^{\mathrm{AP}})^k$ for a subformula φ. In order to construct the automaton for the subformula $\psi = \exists \pi.\ \varphi$, we reduce the alphabet from $(2^{\mathrm{AP}})^k$ to $(2^{\mathrm{AP}})^{k-1}$. We intersect the automaton constructed for φ with an automaton accepting all the traces of K such that the kth element of the tuple is chosen according to the trace of K. Then, we project the language onto the tuples with the first $k-1$ elements. This elimination is repeated until all quantifiers have been eliminated and the language of the resulting automaton is over the one-letter alphabet (consisting of the empty tuple).

3. Check for emptiness. Once all existential quantifiers have been eliminated, the formula is satisfied iff the language of the automaton is non-empty.

The complexity of the model checking problem for alternation-free HyperLTL is the same as for LTL: PSPACE-complete in the size of the formula and NLOGSPACE-complete in the size of the Kripke structure.

Theorem 8. *[10] The model checking problem for the alternation-free fragment of HyperLTL is PSPACE-complete in the size of the formula and NLOGSPACE-complete in the size of the Kripke structure.*

We now extend the model checking algorithm to the full logic. As before, we assume without loss of generality that the outermost quantifier is existential

(otherwise we complement the formula by dualizing all quantifiers and negating the LTL subformula). We again construct a Büchi automaton that is non-empty iff the HyperLTL formula is satisfied. The construction of the Büchi automaton is by induction over the number of quantifier alternations (i.e., over the number of alternations from existential to universal and from universal to existential quantifiers). The alternation-free fragment is the base case. If the quantifiers are existential, we proceed as described above; if the quantifiers are universal, we complement the formula, resulting in existential quantifiers, build the Büchi automaton as described above, and then complement the automaton. For the induction step, suppose the outer quantifiers are existential: then we eliminate the existential quantifiers as described above. If the outer quantifiers are universal, we consider the complemented formula, which turns the outer quantifiers existential, and complement the resulting automaton.

Each complementation increases the size of the automaton exponentially. Let $g_c(k, n)$ be a tower of exponentiations of height k, defined simply as $g_c(0, n) = n$ and $g_c(k, n) = c^{g_c(k-1,n)}$. We define NSPACE($g(k, n)$) to be the languages that are accepted by a nondeterministic Turing machine that runs in SPACE $O(g_c(k, n))$ for some $c > 1$. For convenience, we define NSPACE($g(-1, n)$) to be NLOGSPACE. The complexity of the model checking problem is then characterized by the following theorem.

Theorem 9. *[10, 21] Given a Kripke structure K and a HyperCTL* formula φ with alternation depth k, the model checking problem for K and φ is complete for NSPACE($g(k, |\varphi|)$) and NSPACE($g(k-1, |K|)$).*

6 From Linear to Branching Time

Temporal logics are identified as either linear-time or branching-time based on the equivalences on Kripke structures they induce. LTL is a linear-time logic, because it induces trace equivalence as an equivalence relation on Kripke structures. A logic *induces* an equivalence relation on Kripke structures that distinguishes two Kripke structures iff there is a formula in the logic that is satisfied by one of the two Kripke structures, but not by the other. Two Kripke structures K and K' are called *trace equivalent* if $Tr(K, s_0) = Tr(K', s'_0)$. By contrast, CTL* induces bisimulation, a finer equivalence relation on Kripke structures. A *bisimulation* for a pair of Kripke structures $K = (S, s_0, \delta, \mathrm{AP}, L)$ and $K' = (S', s'_0, \delta', \mathrm{AP}', L')$ is an equivalence relation $R \subseteq S \times S'$ on their states, such that it holds for all pairs $(s, s') \in R$ that $L(s) = L'(s')$ and for all successors $t \in \delta(s)$ of s, there exists a successor $t' \in \delta'(s')$ of s' such that $(t, t') \in R$, and vice versa.

As discussed in Section 3.1, HyperLTL is more expressive than LTL. The

induced equivalence, however, is also trace equivalence. HyperLTL, thus, is also a linear-time temporal logic.

Theorem 10. *[9] HyperLTL induces trace equivalence.*

HyperCTL*. Extending the path quantifiers of CTL* by *path variables* leads to the logic HyperCTL*, which subsumes both HyperLTL and CTL*. The formulas of HyperCTL* are generated by the following grammar:

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

We require that temporal operators only occur inside the scope of path quantifiers.

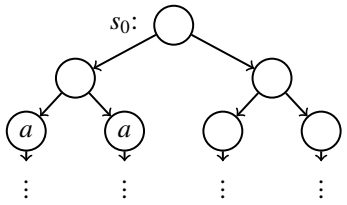
The semantics of HyperCTL* is given in terms of assignments of variables to *paths*, which are defined analogously to trace assignments. Given a Kripke structure K, the satisfaction of HyperCTL* formulas, denoted by $K, \Pi \models \varphi$, is defined as follows:

$$\begin{array}{lll}
K, \Pi \models a_\pi & \text{iff} & a \in L(\Pi(\pi)(0)), \\
K, \Pi \models \neg\varphi & \text{iff} & \Pi, K \not\models \varphi, \\
K, \Pi \models \varphi_1 \vee \varphi_2 & \text{iff} & K, \Pi \models \varphi_1 \text{ or } K, \Pi \models \varphi_2, \\
K, \Pi \models \bigcirc\varphi & \text{iff} & K, \Pi[1, \infty] \models \varphi, \\
K, \Pi \models \varphi_1 \,\mathcal{U}\, \varphi_2 & \text{iff} & \exists i \geq 0 :\ K, \Pi[i, \infty] \models \varphi_2 \text{ and} \\
 & & \forall 0 \leq j < i :\ K, \Pi[j, \infty] \models \varphi_1, \\
K, \Pi \models \exists \pi.\, \varphi & \text{iff} & \exists p \in \mathit{Paths}(K, \Pi(\varepsilon)(0)) : \\
 & & K, \Pi[\pi \mapsto p,\ \varepsilon \mapsto p] \models \varphi,
\end{array}$$

where ε is a special path variable that denotes the path most recently added to Π (i.e., closest in scope to π). For the empty assignment $\Pi_\emptyset$, we define $\Pi_\emptyset(\varepsilon)(0)$ to yield the initial state. A Kripke structure $K = (S, s_0, \delta, \mathrm{AP}, L)$ satisfies a HyperCTL* formula φ, denoted with $K \models \varphi$, iff $K, \Pi_\emptyset \models \varphi$. Like CTL*, HyperCTL* induces bisimulation and is, hence, a branching-time temporal logic.

Theorem 11. *[9] HyperCTL* induces bisimulation.*

HyperCTL* combines the expressiveness of HyperLTL with that of CTL*. Consider, for example, the following Kripke structure:



An observer who sees a can infer which branch was taken in the first nondeterministic choice, but not which leaf node was taken in the second nondeterministic choice. This is expressed by the HyperCTL* formula

$$\forall \pi.\ \bigcirc \forall \pi'.\ \bigcirc (a_\pi \leftrightarrow a_{\pi'}).$$

7 From Propositional to First-Order HyperLTL

Just like propositional HyperLTL is an extension of propositional LTL, first-order HyperLTL (HyperFOLTL) is an extension of first-order LTL (HyperFOLTL). First-order LTL (FOLTL) extends propositional LTL with sorts, constants, and predicate symbols.

FOLTL. A *signature* $\Sigma = (S, C, \mathcal{R}, ar)$ consists of a non-empty and finite set of sorts, finite and disjoint sets C and $\mathcal{R}$ of constant and predicate symbols, and an arity function $ar : C \cup \mathcal{R} \to S^*$, with $|ar(c)| = 1$ for any $c \in C$, where S^* denotes the set of finite sequences of sorts. For each sort s, we let $\mathcal{V}_s$ be a countably infinite set of variables. We let $\mathcal{V}_1 := \bigcup_{s \in S} \mathcal{V}_s$.

FOLTL *formulas* over the signature $\Sigma = (S, C, \mathcal{R}, ar)$ are given by the grammar

$$\varphi \ ::= \ t = t' \ \mid \ R(t_1, \dots, t_k) \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \exists x\colon s.\, \varphi \ \mid \ \bigcirc \varphi \ \mid \ \varphi \,\mathcal{U}\, \varphi$$

where t, t', and the t_is range over $C \cup \mathcal{V}_1$, R ranges over $\mathcal{R}$, s ranges over S, and x ranges over $\mathcal{V}_1$.

A *structure* $\mathcal{S}$ over the signature $\Sigma = (S, C, \mathcal{R}, ar)$ consists of a S-indexed family of (finite or infinite) *universes* $U_s \neq \emptyset$ and *interpretations* $R^{\mathcal{S}} \in U_{s_1} \times \dots U_{s_k}$, for each $R \in C \cup \mathcal{R}$ of sort $(s_1, \dots, s_k)$. We let $U := \bigcup_{s \in S} U_s$. A *temporal structure* over Σ is a sequence $\bar{\mathcal{S}} = (\mathcal{S}_0, \mathcal{S}_1, \dots)$ of structures over Σ such that all structures $\mathcal{S}_i$, with $i \geq 0$, have the same universe family, denoted $(U_s)_{s \in S}$, and rigid constant interpretations, i.e. $c^{\mathcal{S}_i} = c^{\mathcal{S}_0}$, for all $c \in C$ and $i > 0$.

Given a structure, a *valuation* is a mapping $\nu : \mathcal{V}_1 \to U$ with $\nu(x) \in U_s$ for any $x \in \mathcal{V}_s$. For a valuation ν and tuples $\bar{x} = (x_1, \dots, x_n)$ and $\bar{d} = (d_1, \dots, d_n)$, where $x_i \in \mathcal{V}_s$ and $d_i \in U_s$ for some sort s, for each i, we write $\nu[\bar{x} \mapsto \bar{d}]$ for the valuation that maps each x_i to d_i and leaves the other variables' valuation unaltered. By $\nu(\bar{x})$ we denote the tuple $(\nu(x_1), \dots, \nu(x_n))$. We extend this notation by applying a valuation ν also to constant symbols $c \in C$, with $\nu(c) = c^{\mathcal{S}}$.

Let $\bar{\mathcal{S}}$ be a temporal structure over the signature Σ, with $\bar{\mathcal{S}} = (\mathcal{S}_0, \mathcal{S}_1, \dots)$, φ a formula over Σ, and ν a valuation. We define the satisfaction of φ over $\bar{\mathcal{S}}$ and μ,

denoted by $\bar{\mathcal{S}}, \nu \models \varphi$, as follows:

$$
\begin{array}{lll}
\bar{\mathcal{S}}, \nu \models t = t' & \text{iff} & \nu(t) = \nu(t') \\
\bar{\mathcal{S}}, \nu \models R(\bar{t}) & \text{iff} & \nu(\bar{t}) \in R^{S_0} \\
\bar{\mathcal{S}}, \nu \models \neg\psi & \text{iff} & \bar{\mathcal{S}}, \nu \not\models \psi \\
\bar{\mathcal{S}}, \nu \models \psi \vee \psi' & \text{iff} & \bar{\mathcal{S}}, \nu \models \psi \text{ or } \bar{\mathcal{S}}, \nu \models \psi' \\
\bar{\mathcal{S}}, \nu \models \exists x.\, \psi & \text{iff} & \bar{\mathcal{S}}, \nu[x \mapsto d] \models \psi, \text{ for some } d \in U \\
\bar{\mathcal{S}}, \nu \models \bigcirc \psi & \text{iff} & \bar{\mathcal{S}}[1, \infty], \nu \models \psi \\
\bar{\mathcal{S}}, \nu \models \psi \,\mathcal{U}\, \psi' & \text{iff} & \text{for some } j \geq 0,\ \bar{\mathcal{S}}[j, \infty], \nu \models \psi', \text{ and} \\
 & & \quad \bar{\mathcal{S}}[k, \infty], \nu \models \psi, \text{ for all } k \text{ with } 0 \leq k < j
\end{array}
$$

HyperFOLTL. HyperFOLTL *formulas* over signature Σ and trace variables $\mathcal{V}$ are generated by the following grammar:

$$\psi ::= \exists \pi.\, \psi \mid \forall \pi.\, \psi \mid \varphi$$

where $\pi \in \mathcal{V}$ is a trace variable and φ is a FOLTL formula over Σ. HyperFOLTL formulas thus start with a prefix of trace quantifiers consisting of at least one quantifier and then continue with a subformula that contains only first-order quantifiers, no trace quantifiers.

The semantics of a HyperFOLTL formula ψ is given with respect to a set $\mathcal{T}$ of temporal structures, a valuation $\alpha : \mathcal{V}_1 \to U$ of the first-order variables, and a valuation $\Pi : \mathcal{V} \to \mathcal{T}$ of the trace variables. The *satisfaction* of a HyperFOLTL formula φ, denoted by $\mathcal{T}, \alpha, \beta \models \psi$, is then defined as follows:

$$
\begin{array}{lll}
\mathcal{T}, \alpha, \Pi \models \exists \pi.\, \psi & \text{iff} & \mathcal{T}, \alpha, \Pi[\pi \mapsto t] \models \psi, \text{ for some } t \in \mathcal{T}, \\
\mathcal{T}, \alpha, \Pi \models \forall \pi.\, \psi & \text{iff} & \mathcal{T}, \alpha, \Pi[\pi \mapsto t] \models \psi, \text{ for all } t \in \mathcal{T}, \\
\mathcal{T}, \alpha, \Pi \models \psi & \text{iff} & \bar{\mathcal{S}}, \alpha \models \psi,
\end{array}
$$

where φ is an HyperFOLTL formula, ψ is an FOLTL formula, and the temporal structure $\bar{\mathcal{S}}$ is such that for all $R \in \mathcal{R}$, $i \in \mathbb{N}$, and, $\pi \in \Pi$, the interpretation $R_\pi^{\mathcal{S}_i}$ is $R^{\beta(\pi)(i)}$ if π in the domain of Π, and $\emptyset$ otherwise.

For example, observational determinism can be formalized by the following HyperFOLTL formula

$$\forall \pi, \pi'. \forall x : \mathit{user}.\ (\Box I_\pi(x) \leftrightarrow I_{\pi'}(x)) \ \rightarrow\ (O_\pi(x) \leftrightarrow O_{\pi'}(x)),$$

where $I(x)$ represents an input observed by user x and $O(x)$ represents an output observed by user x. The formula states that, on any two traces and for any user, if the inputs are always the same, then the low outputs are also always the same. That is, from the point of view of each user, the observable behavior of the program is determined by its input.

As the satisfiability of HyperLTL is undecidable, the same holds for HyperFOLTL. There is, however, again a substantial decidable fragment. To define this fragment, we will consider the projection of a sorted FOLTL formula on a sort s, defined as the FOLTL formula obtained by removing all quantifications and terms of a sort different from s [1]. The $\exists^*$FOLTL fragment of sorted FOLTL consists of those closed formulas φ in negation normal form such that, for each sort s, the projection of φ on s is a formula of the form

$$\exists x_1, \ldots, x_k. \varphi'_s$$

with $k \geq 0$ and φ'_s a FOLTL formula containing no existential quantifiers.

The $\exists^*_\pi \forall^*_\pi \exists^*$FOLTL fragment of HyperFOLTL consists of all formulas of the form $\exists \pi_1, \ldots \pi_k. \forall \pi'_1 \ldots \pi'_\ell. \varphi$ with $k \geq 0$, $\ell \geq 0$, and φ an FOLTL formula in $\exists^*$FOLTL.

Theorem 12. *[8] The satisfiability problems of the* $\exists^*$FOLTL *fragment of* FOLTL *and the* $\exists^*_\pi \forall^*_\pi \exists^*$FOLTL *fragment of* HyperFOLTL *are decidable.*

This result has been used to verify secrecy properties of a web-based conference management system with arbitrarily many users [8]. In this case study, the possible behaviors of the conference management system where encoded as a FOLTL formula φ_S. Assumptions on the behavior of the users of the system were encoded as a HyperFOLTL formula φ_A. Verifying that a HyperFOLTL specification ψ is satisfied then amounts to checking that the HyperFOLTL formula

$$\varphi_S \wedge \varphi_Q \wedge \neg\psi$$

is unsatisfiable.

8 Conclusions

HyperLTL is the natural extension of linear-time temporal logic to hyperproperties. In this paper, we have surveyed recent results on HyperLTL, in particular on the satisfiability and model checking problems. The additional expressiveness of alternation-free HyperLTL comes at very little extra cost over the trace logic LTL: the model checking and satisfiability problems remain in the same complexity class. With quantifier alternation, model checking becomes significantly more expensive and satisfiability becomes undecidable as soon as the fragment contains $\forall\exists$ formulas.

There are plenty of open questions in this area. Conceptually, a key limitation of HyperLTL is that it traverses the traces synchronously. This is adequate for the specification and verification of hardware, but not necessarily for software,

in particular for concurrent software, where asynchronous behavior is more common. In terms of basic algorithms, the complexity of the satisfiability problem of HyperCTL* is still open. While the undecidability of HyperLTL implies that HyperCTL* is also, in general, undecidable (this was established in [4]), the obvious question is whether it is possible to establish decidable fragments in a similar fashion as for HyperLTL.

Another interesting algorithmic problem is the synthesis of reactive systems from HyperLTL (and HyperCTL*) specifications. In synthesis, we ask for the existence of an implementation, which is usually understood as an infinite tree that branches according to the possible inputs to a system and whose nodes are labeled with the outputs of the system. Since HyperLTL can express partial observability, the synthesis problem for HyperLTL naturally generalizes the well-studied synthesis under incomplete information [14] and the synthesis of distributed systems [20].

Finally, a major open problem is to find a temporal logic that is expressively equivalent to FO[<, E]. In Section 3.3, we have argued that HyperLTL is less expressive than FO[<, E], because HyperLTL cannot express the property described by Bozzelli et al. [3], which relates at some point in time an unbounded number of traces. Since LTL_K can express such properties, epistemic temporal logics similar to LTL_K might be candidates for logics that are expressively equivalent to FO[<, E].

Acknowledgements. The work reported in this paper has previously appeared in various publications [4, 9, 10, 7, 11, 8]. I am indebted to my coauthors, Michael R. Clarkson, Christopher Hahn, Masoud Koleini, Kristopher K. Micinski, Christian Müller, Markus N. Rabe, César Sánchez, Helmut Seidl, Eugen Zalinescu, and Martin Zimmermann. This work was partially supported by the German Research Foundation (DFG) in the Collaborative Research Center 1223.

References

[1] Aharon Abadi, Alexander Rabinovich, and Mooly Sagiv. Decidable fragments of many-sorted logic. *Journal of Symbolic Computation*, 45(2):153–172, 2010.

[2] Rajeev Alur, Pavol Černý, and Steve Zdancewic. Preserving secrecy under refinement. In *Proc. of ICALP'06*, pages 107–118, 2006.

[3] Laura Bozzelli, Bastien Maubert, and Sophie Pinchinat. Unifying hyper and epistemic temporal logics. In Andrew M. Pitts, editor, *FoSSaCS 2015*, volume 9034 of *Lecture Notes in Computer Science*, pages 167–182. Springer, 2015. doi: 10.1007/978-3-662-46678-0_11.

[4] Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. Temporal logics for hyperproperties. In Martín Abadi and Steve Kremer, editors, *POST 2014*, volume 8414 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2014. doi:10.1007/978-3-642-54792-8_15.

[5] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010. doi:10.3233/JCS-2009-0393.

[6] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[7] Bernd Finkbeiner and Christopher Hahn. Deciding hyperproperties. In Josée Desharnais and Radha Jagadeesan, editors, *CONCUR 2016*, volume 59 of *LIPIcs*, pages 13:1–13:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CONCUR.2016.13.

[8] Bernd Finkbeiner, Christian Müller, Helmut Seidl, and Eugen Zalinescu. Verifying Security Policies in Multi-agent Workflows with Loops. In *15th ACM Conference on Computer and Communications Security (CCS'17)*. ACM Press, 2017. doi:10.1145/3133956.3134080.

[9] Bernd Finkbeiner and Markus N. Rabe. The linear-hyper-branching spectrum of temporal logics. *it - Information Technology*, 56:273–279, November 2014.

[10] Bernd Finkbeiner, Markus N. Rabe, and César Sánchez. Algorithms for model checking HyperLTL and HyperCTL*. In Daniel Kroening and Corina S. Pasareanu, editors, *CAV 2015 (Part I)*, volume 9206 of *Lecture Notes in Computer Science*, pages 30–48. Springer, 2015. doi:10.1007/978-3-319-21690-4_3.

[11] Bernd Finkbeiner and Martin Zimmermann. The First-Order Logic of Hyperproperties. In *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2017/7003.

[12] Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal basis of fairness. In Paul W. Abrahams, Richard J. Lipton, and Stephen R. Bourne, editors, *POPL 1980*, pages 163–173. ACM Press, 1980. doi:10.1145/567446.567462.

[13] Hans W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Computer Science Department, University of California at Los Angeles, USA, 1968.

[14] Orna Kupfermant and Moshe Y Vardit. Synthesis with incomplete informatio. In *Advances in Temporal Logic*, pages 109–127. Springer, 2000.

[15] Daryl McCullough. Noninterference and the composability of security properties. In *Proc. IEEE Symposium on Security and Privacy*, pages 177–186, April 1988.

[16] John McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proc. IEEE Symposium on Security and Privacy*, pages 79–93, May 1994.

[17] Ron Van Der Meyden. Axioms for knowledge and time in distributed systems with perfect recall. In *in: Proceedings of the Ninth IEEE Symposium on Logic in Computer Science*, pages 448–457, 1993.

[18] David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *Proc. of LICS'88*, pages 422–427. IEEE CS Press, 1988. doi:10.1109/LICS.1988.5139.

[19] Amir Pnueli. The Temporal Logic of Programs. In *FOCS 1977*, pages 46–57, 1977.

[20] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 746–757, 1990. URL: http://dx.doi.org/10.1109/FSCS.1990.89597, doi:doi:10.1109/FSCS.1990.89597.

[21] Markus N. Rabe. *A Temporal Logic Approach to Information-flow Control*. PhD thesis, Saarland University, 2016.

[22] G. Smith. On the foundations of quantitative information flow. In *Proc. Conference on Foundations of Software Science and Computation Structures*, pages 288–302, March 2009.

[23] Wolfgang Thomas. Path logics with synchronization. In Kamal Lodaya, Madhavan Mukund, and R. Ramanujam, editors, *Perspectives in Concurrency Theory*, pages 469–481. IARCS-Universities, Universities Press, 2009.

[24] Moshe Y. Vardi. Alternating automata and program verification. In *Computer Science Today*, volume 1000 of *LNCS*, pages 471–485. Springer, 1995. doi:10.1007/BFb0015261.

[25] H. Yasuoka and T. Terauchi. On bounding problems of quantitative information flow. In *Proc. European Symposium on Research in Computer Security*, pages 357–372, September 2010.

[26] S. Zdancewic and A. C. Myers. Observational determinism for concurrent program security. In *Proc. of CSFW*, 2003.