Asynchronous Extensions of HyperLTL

Laura Bozzelli
University of Napoli "Federico II"
Napoli, Italy

Adriano Peron
University of Napoli "Federico II"
Napoli, Italy

César Sánchez
IMDEA Software Institute
Madrid, Spain

***Abstract*—Hyperproperties are a modern specification paradigm that extends trace properties to express properties of sets of traces. Temporal logics for hyperproperties studied in the literature, including HyperLTL, assume a synchronous semantics and enjoy a decidable model checking problem. In this paper, we introduce two asynchronous and orthogonal extensions of HyperLTL, namely *Stuttering HyperLTL* (HyperLTL$_S$) and *Context HyperLTL* (HyperLTL$_C$). Both of these extensions are useful, for instance, to formulate asynchronous variants of information-flow security properties. We show that for these logics, model checking is in general undecidable. On the positive side, for each of them, we identify a fragment with a decidable model checking that subsumes HyperLTL and that can express meaningful asynchronous requirements. Moreover, we provide the exact computational complexity of model checking for these two fragments which, for the HyperLTL$_S$ fragment, coincides with that of the strictly less expressive logic HyperLTL.**

I. Introduction

Model checking is a well-established formal method technique to automatically check for global correctness of finite-state systems [1], [2]. Properties for model checking are usually specified in classic *regular* temporal logics such as LTL, CTL, and CTL* [3], [4], which provide temporal modalities for describing the ordering of events along individual execution traces of a system (trace properties). These logics lack mechanisms to relate distinct traces, which is required to express important information-flow security policies. Examples include properties that compare observations made by an external low-security agent along traces resulting from different values of not directly observable inputs. These security requirements go, in general, beyond regular properties.

In the last decade, a novel specification paradigm has been introduced that generalizes traditional regular trace properties by properties of sets of traces, the so called *hyperproperties* [5]. Hyperproperties relate distinct traces and are useful to formalize information-flow security policies like noninterference [6], [7] and observational determinism [8]. Hyperproperties also have applications in other settings, such as the symmetric access to critical resources in distributed protocols [9]. Many temporal logics for hyperproperties have been proposed in the literature [10]–[16] for which model checking is decidable, including HyperLTL [11], HyperCTL* [11], HyperQPTL [13], [15], and HyperPDL$-\Delta$ [16] which extend LTL, CTL*, QPTL [17], and PDL [18], respectively, by explicit first-order quantification over traces and trace variables to refer to multiple traces at the same time.

In all these logics, the mechanism for comparing distinct traces is synchronous and consists in evaluating the temporal modalities by a lockstepwise traversal of all the traces assigned to the quantified trace variables. This represents a limitation in various scenarios [19], [20] where properties of interest are instead asynchronous, since these properties require to relate traces at distinct time points which can be arbitrarily far from each other. Recently, two powerful and expressively equivalent formalisms have been introduced [20] for specifying asynchronous linear-time hyperproperties. The first one, called H$_\mu$, is based on a fixpoint calculus, while the second one exploits parity multi-tape Alternating Asynchronous Word Automata (AAWA) [20] for expressing the quantifier-free part of a specification. AAWA allow to specify very expressive non-regular multi-trace properties. As a matter of fact, model checking against H$_\mu$ or its AAWA-based counterpart is undecidable even for the (quantifier) alternation-free fragment. In [20], two decidable subclasses of parity AAWA are identified which lead to H$_\mu$ fragments with decidable model checking. Both subclasses express only ω-regular languages over the synchronous product of tuples of traces with fixed arity. In particular, the first subclass captures all the multi-trace regular properties and the corresponding H$_\mu$ fragment (the so called k-synchronous fragment for a given $k \geq 1$) is strictly more expressive than HyperLTL, while the second subclass is non-elementarily more succinct than the first subclass and leads to a H$_\mu$ fragment which seems expressively incomparable with HyperLTL.

Our contribution: In this paper, we introduce two novel, more expressive, extensions of HyperLTL for the specification of asynchronous linear-time hyperproperties, obtained by adding intuitive logical features that provide natural modeling facilities. The first formalism, that we call *Stuttering HyperLTL* (HyperLTL$_S$), is useful in information-flow security settings where an observer is not time-sensitive, i.e. the observer cannot distinguish consecutive time points along an execution having the same observation. This requires asynchronously matching sequences of observations along distinct execution traces. The novel feature of HyperLTL$_S$ consists in temporal modalities parameterized by finite sets Γ of LTL formulas. These modalities are evaluated along sub-traces of the given traces which are obtained by removing "redundant" positions with respect to the pointwise evaluation of the LTL formulas

This work was funded in part by the Madrid Regional Government under project "S2018/TCS-4339 (BLOQUES-CM)", by Spanish National Project "BOSCO (PGC2018-102210-B-100)".

978-1-6654-4895-6/21/$31.00

in Γ. We show that model checking against the alternation-free fragment of HyperLTL$_S$ is already undecidable. On the positive side, we identify a meaningful fragment, called *simple HyperLTL$_S$*, with a decidable model-checking problem, which strictly subsumes HyperLTL and allows to express asynchronous variants of relevant security properties such as noninterference [6] and observational determinism [8]. Moreover, model checking against simple HyperLTL$_S$ has the same computational complexity as model checking for HyperLTL and is expressively incomparable with the two H$_\mu$ fragments previously described. In particular, unlike these two fragments and HyperLTL, quantifier-free formulas of simple HyperLTL$_S$ can express some non-regular multi-trace properties.

The second logic that we introduce in this paper, called *Context HyperLTL* (HyperLTL$_C$), allows to specify complex combinations of asynchronous and synchronous requirements. HyperLTL$_C$ extends HyperLTL by unary modalities parameterized by a non-empty subset C of trace variables (*context*) which restrict the evaluation of the temporal modalities to the traces associated with the variables in C. Like HyperLTL$_S$, model checking against HyperLTL$_C$ is undecidable. In this case we exhibit a fragment of HyperLTL$_C$ which is, in a certain sense, maximal with respect to the decidability of model checking, and extends HyperLTL by allowing the comparison of different traces at time points of bounded distance. This fragment is subsumed by k-synchronous H$_\mu$, and we establish that for a fixed quantifier alternation depth, model checking this fragment is exponentially harder than model checking HyperLTL.

With regard to expressiveness issues, both HyperLTL$_C$ and HyperLTL$_S$ are subsumed by H$_\mu$. On the other hand, questions concerning the comparison of the expressive power of HyperLTL$_S$ and HyperLTL$_C$ are left open: we conjecture that (simple) HyperLTL$_S$ and HyperLTL$_C$ are expressively incomparable.

Related work: Another linear-time temporal logic, called asynchronous HyperLTL (AHyperLTL), for pure asynchronous hyperproperties and useful for asynchronous security analysis has been recently introduced in [21]. This logic, which is expressively incomparable with HyperLTL, adds an additional quantification layer over the so called trajectory variables. Intuitively, a *trajectory* describes an asynchronous interleaving of the traces in the current multi-trace where single steps of distinct traces can overlap, and temporal modalities, indexed by trajectory variables, are evaluated along the associated trajectories. The logic has an undecidable model-checking problem, but [21] identifies practical fragments with decidable model-checking, and reports an empirical evaluation.

Other known logics for linear-time hyperproperties are the first-order logic with equal-level predicate FOL[<,E] [22] and its monadic second-order extension S1S[E] [15]. We conjecture that these logics are expressively incomparable with H$_\mu$, HyperLTL$_C$, and HyperLTL$_S$. For instance, we believe that S1S[E] cannot express counting properties requiring that two segments along two different traces at an unbounded distance from each other have the same length. This kind of requirements can be instead expressed in HyperLTL$_C$ and H$_\mu$. Proving these conjectures are left for future work.

II. Preliminaries

Let $\mathbb{N}$ be the set of natural numbers. Given $i, j \in \mathbb{N}$, we write $[i, j]$ for the set of natural numbers h such that $i \leq h \leq j$, $[i, j)$ for the set of natural numbers h such that $i \leq h < j$, and $[i, \infty]$ for the set of natural numbers h such that $h \geq i$.

We fix a *finite* set AP of atomic propositions. A *trace* is an infinite word over 2^{AP}. A *pointed trace* is a pair (π, i) consisting of a trace π and a position $i \in \mathbb{N}$ along π.

For a word w over some alphabet Σ, $|w|$ is the length of w ($|w| = \infty$ if w is infinite), for each $0 \leq i < |w|$, $w(i)$ is the $(i+1)^{th}$ symbol of w, and w^i is the suffix of w from position i, i.e., the word $w(i)w(i+1)\ldots$

Given $n, h \in \mathbb{N}$ and integer constants $c > 1$, $\mathsf{Tower}_c(h, n)$ denotes a tower of exponentials of base c, height h, and argument n: $\mathsf{Tower}_c(0, n) = n$ and $\mathsf{Tower}_c(h+1, n) = c^{\mathsf{Tower}_c(h,n)}$. For each $h \in \mathbb{N}$, we denote by h-EXPSPACE the class of languages decided by deterministic Turing machines bounded in space by functions of n in $O(\mathsf{Tower}_c(h, n^d))$ for some integer constants $c > 1$ and $d \geq 1$. Note that 0-EXPSPACE coincides with PSPACE.

A. Linear-time Temporal Logic (LTL)

We recall syntax and semantics of LTL [3]. Formulas θ of LTL over the set AP of atomic propositions are defined as follows:

$$\theta ::= p \mid \neg\theta \mid \theta \wedge \theta \mid \mathsf{X}\theta \mid \theta \,\mathsf{U}\, \theta$$

where $p \in \mathsf{AP}$ and X and U are the "next" and "until" temporal modalities respectively. The logic is interpreted over pointed traces (π, i). The satisfaction relation $(\pi, i) \models \theta$, meaning that formula θ holds at position i along π, is inductively defined as follows (we omit the semantics for the Boolean connectives which is standard):

$$\begin{array}{ll} (\pi,i) \models p & \Leftrightarrow p \in \pi(i) \\ (\pi,i) \models \mathsf{X}\theta & \Leftrightarrow (\pi, i+1) \models \theta \\ (\pi,i) \models \theta_1 \,\mathsf{U}\, \theta_2 & \Leftrightarrow \text{for some } j \geq i : (\pi,j) \models \theta_2 \text{ and} \\ & \quad (\pi,k) \models \theta_1 \text{ for all } i \leq k < j \end{array}$$

A trace π is a model of θ, written $\pi \models \theta$, if $(\pi, 0) \models \theta$.

B. Linear-time Hyper Specifications

In this section, we consider an abstract notion of linear-time hyper specifications which are interpreted over sets of traces. For the rest of the discussion, we fix a finite ordered set VAR of trace variables.

A *pointed trace assignment* Π is a partial mapping over VAR, assigning to each trace variable x in its domain *Dom*(Π) a pointed trace. The assignment Π is initial if for each $x \in$ *Dom*(Π), $\Pi(x)$ is of the form $(\pi, 0)$ for some trace π. For a trace variable $x \in \mathsf{VAR}$ and a pointed trace (π, i), we denote by $\Pi[x \mapsto (\pi, i)]$ the pointed trace assignment having domain *Dom*$(\Pi) \cup \{x\}$ that behaves as Π on the variables in *Dom*$(\Pi) \setminus \{x\}$ and assigns to x the pointed trace (π, i).

A *multi-trace specification* $\mathcal{S}(x_1, \ldots, x_n)$ is a specification (in some formalism) parameterized by a subset $\{x_1, \ldots, x_n\}$ of VAR whose semantics is represented by a set Υ of pointed trace assignments with domain $\{x_1, \ldots, x_n\}$. Depending on the given formalism, one can restrict to consider only *initial* pointed trace assignments. We write $\Pi \models S(x_1, \ldots, x_n)$ for the trace assignments Π in Υ.

Given a class $\mathcal{C}$ of multi-trace specifications, linear-time hyper expressions ξ over $\mathcal{C}$ are defined as follows:

$$\xi ::= \exists x.\xi \mid \forall x.\xi \mid S(x_1, \ldots, x_n)$$

where $x, x_1, \ldots, x_n \in \mathsf{VAR}$, $S(x_1, \ldots, x_n)$ is a multi-trace specification in the class $\mathcal{C}$, $\exists x$ is the *hyper* existential trace quantifier for variable x, and $\forall x$ the hyper universal trace quantifier for x. Informally, the expression $\exists x.\xi$ requires that for some trace π in the given set of traces, ξ holds when x is mapped to $(\pi, 0)$, while $\forall x.\xi$ requires that all traces π, ξ holds when x is mapped to $(\pi, 0)$. We say that an expression ξ is a *sentence* if every variable x_i in the multi-trace specification $S(x_1, \ldots, x_n)$ of ξ is in the scope of a quantifier for the trace variable x_i, and distinct occurrences of quantifiers are associated with distinct trace variables. The *quantifier alternation depth* of ξ is the number of switches between $\exists$ and $\forall$ quantifiers in the quantifier prefix of ξ.

For instance, HyperLTL sentences [11] are linear-time hyper sentences over the class of multi-trace specifications obtained by LTL formulas by replacing atomic propositions p with relativized versions $p[x]$, where $x \in \mathsf{VAR}$. Intuitively, $p[x]$ asserts that p holds at the pointed trace assigned to x.

Given a linear-time expression ξ with multi-trace specification $S(x_1, \ldots, x_n)$, a set $\mathcal{L}$ of traces, and an initial pointed trace assignment Π such that $\mathit{Dom}(\Pi)$ contains the variables in $\{x_1, \ldots, x_n\}$ which are not in the scope of a quantifier, and the traces referenced by Π are in $\mathcal{L}$, the satisfaction relation $(\mathcal{L}, \Pi) \models \xi$ is inductively defined as follows:

$$\begin{array}{ll}
(\mathcal{L}, \Pi) \models \exists x.\xi & \Leftrightarrow \text{ for some trace } \pi \in \mathcal{L}: \\
 & \quad (\mathcal{L}, \Pi[x \mapsto (\pi, 0)]) \models \xi \\
(\mathcal{L}, \Pi) \models \forall x.\xi & \Leftrightarrow \text{ for each trace } \pi \in \mathcal{L}: \\
 & \quad (\mathcal{L}, \Pi[x \mapsto (\pi, 0)]) \models \xi \\
(\mathcal{L}, \Pi) \models S(x_1, \ldots, x_n) & \Leftrightarrow \Pi \models S(x_1, \ldots, x_n)
\end{array}$$

If ξ is a sentence, we write $\mathcal{L} \models \xi$ to mean that $(\mathcal{L}, \Pi_\emptyset) \models \xi$, where $\Pi_\emptyset$ is the empty assignment.

C. Kripke Structures and Asynchronous Word Automata

Kripke structures. A *Kripke structure* (*over* AP) is a tuple $\mathcal{K} = \langle S, S_0, E, V \rangle$, where S is a set of states, $S_0 \subseteq S$ is the set of initial states, $E \subseteq S \times S$ is a transition relation which is total in the first argument (i.e. for each $s \in S$ there is a $t \in S$ with $(s, t) \in E$), and $V : S \to 2^{\mathsf{AP}}$ is an AP*-valuation* assigning to each state s the set of propositions holding at s. The Kripke structure $\mathcal{K}$ is finite if S is finite.

A *path* $\nu = t_0, t_1, \ldots$ of $\mathcal{K}$ is an infinite word over S such that $t_0 \in S_0$ is an initial state and for all $i \geq 0$, $(t_i, t_{i+1}) \in E$. The path $\nu = t_0, t_1, \ldots$ induces the trace $V(t_0)V(t_1)\ldots$. A finite path of $\mathcal{K}$ is a non-empty finite infix of some path of $\mathcal{K}$. A *trace* of $\mathcal{K}$ is a trace induced by some path of $\mathcal{K}$. We denote by $\mathcal{L}(\mathcal{K})$ the set of traces of $\mathcal{K}$. We also consider *fair finite Kripke structures* $(\mathcal{K}, F)$, that is, finite Kripke structures $\mathcal{K}$ equipped with a subset F of $\mathcal{K}$-states. A path ν of $\mathcal{K}$ is F*-fair* if ν visits infinitely many times states in F. We denote by $\mathcal{L}(\mathcal{K}, F)$ the set of traces of $\mathcal{K}$ associated with the F-fair paths of $\mathcal{K}$. We consider the following decision problems for a given class $\mathcal{C}$ of multi-trace specifications:

- *Model checking problem*: checking for a given finite Kripke structure $\mathcal{K}$ and a linear-time hyper sentence ξ over $\mathcal{C}$, whether $\mathcal{L}(\mathcal{K}) \models \xi$ (we also write $\mathcal{K} \models \xi$).
- *Fair model checking problem*: checking for a given fair finite Kripke structure $(\mathcal{K}, F)$ and a linear-time hyper sentence ξ over $\mathcal{C}$, whether $\mathcal{L}(\mathcal{K}, F) \models \xi$.

Note that model checking reduces to fair model checking for the special case where F coincides with the set of $\mathcal{K}$-states.

Labeled Trees. A tree T is a prefix closed subset of $\mathbb{N}^*$. Elements of T are called nodes and the empty word ε is the root of T. For $x \in T$, a child of x in T is a node of the form $x \cdot n$ for some $n \in \mathbb{N}$. A path of T is a maximal sequence π of nodes such that $\pi(0) = \varepsilon$ and $\pi(i)$ is a child in T of $\pi(i-1)$ for all $0 < i < |\pi|$. For an alphabet Σ, a Σ-labeled tree is a pair $\langle T, \mathit{Lab} \rangle$ consisting of a tree and a labelling $\mathit{Lab} : T \to \Sigma$ assigning to each node in T a symbol in Σ.

Asynchronous Word Automata. We consider a variant of the framework of alternating asynchronous word automata introduced in [20], a class of finite-state automata for the asynchronous traversal of multiple infinite words. Given a set X, $\mathbb{B}^+(X)$ denotes the set of *positive* Boolean formulas over X, that is, Boolean formulas built from elements in X using $\vee$ and $\wedge$ (we also allow the formulas `true` and `false`). Let $n \geq 1$. A Büchi nAAWA over a finite alphabet Σ is a tuple $\mathcal{A} = \langle \Sigma, q_0, Q, \rho, F \rangle$, where Q is a finite set of (control) states, $q_0 \in Q$ is the initial state, $\rho : Q \times \Sigma^n \to \mathbb{B}^+(Q \times [1, n])$ is the transition function, and $F \subseteq Q$ is a set of accepting states. Intuitively, an nAAWA has access to n infinite input words over Σ and at each step, it activates multiple copies. For each copy, there is exactly one input word whose current input symbol is consumed, so the reading head of such word moves one position to the right.

In particular, the target of a move of $\mathcal{A}$ is encoded by a pair $(q, i) \in A \times [1, n]$, where q indicates the target state while the direction i indicates on which input word to progress.

Formally, a run of $\mathcal{A}$ over an n-tuple $\overline{w} = (w_1, \ldots, w_n)$ of infinite words over Σ is a $(Q \times \mathbb{N}^n)$-labeled tree $r = \langle T_r, \mathit{Lab}_r \rangle$, where each node of T_r labelled by $(q, \wp)$ with $\wp = (i_1, \ldots, i_n)$ describes a copy of the automaton that is in state q and reads the $(i_h + 1)^{th}$ symbol of the input word w_h for each $h \in [1, n]$. Moreover, we require that

- $r(\varepsilon) = (q_0, (0, \ldots, 0))$, that is, initially, the automaton is in state q_0 reading the first position of each input word);
- for each $\tau \in T_r$ with $\mathit{Lab}_r(\tau) = (q, (i_1 \ldots, i_n))$, there is a set $\{(q_1, d_1), \ldots, (q_k, d_k)\} \subseteq Q \times [1, n]$ for some $k \geq 0$ satisfying $\delta(q, (w_1(i_1), \ldots, w_n(i_n)))$ such that τ has

k children $\tau_1, \ldots, \tau_k$ and $Lab_r(\tau_j) = (q_j, (i_1, \ldots, i_{d_j} + 1, \ldots, i_n))$ for all $1 \leq j \leq k$.

The run r is accepting if each infinite path ν visits infinitely often nodes labeled by some accepting state in F. We denote by $\mathcal{L}(\mathcal{A})$ the set of n-tuples $\overline{w}$ of infinite words over Σ such that there is an accepting run of $\mathcal{A}$ over $\overline{w}$.

For each $k \geq 1$, we also consider *k-synchronous* Büchi nAAWA [20], which are Büchi nAAWA such that for each run r and for each node of r with label $(q, \wp)$, the position vector $\wp = (i_1, \ldots, i_n)$ satisfies $|i_\ell - i_{\ell'}| \leq k$ for all $\ell, \ell' \in [1, k]$. Intuitively, a k-synchronous nAAWA can never be ahead more than k steps in one direction with respect to the others. Note that AAWA over 2^{AP} can be seen as multi-trace specifications. It is known [20] that model checking against linear-time hyper sentences over Büchi AAWA is undecidable, and the problem becomes decidable when one restricts to consider k-synchronous Büchi AAWA. In particular, the following holds.

Proposition II.1 ([20]). *Let $d \in \mathbb{N}$. The (fair) model checking problem against linear-time hyper sentences of quantifier alternation depth d over the class of k-synchronous Büchi nAAWA over 2^{AP} (k, n, AP being input parameters of the problem instances) is $(d+1)$-*EXPSPACE*-complete, and for a fixed formula, it is $(d-1)$-*EXPSPACE*-complete for $d > 0$ and* NLOGSPACE*-complete otherwise.*

III. STUTTERING HYPERLTL

In this section we introduce an asynchronous extension of HyperLTL that we call *stuttering HyperLTL* (HyperLTL$_S$ for short). The novel logic is obtained by exploiting relativized versions of the temporal modalities with respect to finite sets Γ of LTL formulas. Intuitively, these modalities are evaluated along sub-traces of the given traces which are obtained by removing "redundant" positions with respect to the pointwise evaluation of the LTL formulas in Γ. The rest of this section is organized as follows. In Subsection III-A we introduce a generalization of the classical notion of stuttering. Then, in Subsection III-B we define the syntax and semantics of HyperLTL$_S$ and provide some examples of specifications in this logic. Finally, we investigate the model checking problem against HyperLTL$_S$. In Subsection III-C, we show that the problem is in general undecidable, and in Subsection III-D, we identify a meaningful fragment of HyperLTL$_S$ for which model checking is shown to be decidable.

A. LTL-Relativized Stuttering

Classically, a trace is stutter-free if there are no consecutive positions having the same propositional valuation unless the valuation is repeated ad-infinitum. We can associate to each trace a unique stutter-free trace by removing "redundant" positions. In this subsection, we generalize these notions with respect to the pointwise evaluation of a finite set of LTL formulas.

Definition III.1 (LTL stutter factorization). *Let Γ be a finite set of LTL formulas and π a trace. The* Γ-stutter factorization *of π is the unique increasing sequence of positions $\{i_k\}_{k \in [0, m_\infty]}$ for some $m_\infty \in \mathbb{N} \cup \{\infty\}$ such that the following holds for all $j < m_\infty$:*

- *$i_0 = 0$ and $i_j < i_{j+1}$;*
- *for each $\theta \in \Gamma$, the truth value of θ along the segment $[i_j, i_{j+1})$ does not change, i.e. for all $h, k \in [i_j, i_{j+1})$, $(\pi, h) \models \theta$ iff $(\pi, k) \models \theta$, and the same holds for the infinite segment $[m_\infty, \infty]$ in case $m_\infty \neq \infty$;*
- *the truth value of some formula in Γ changes along adjacent segments, i.e. for some $\theta \in \Gamma$ (depending on j), $(\pi, i_j) \models \theta$ iff $(\pi, i_{j+1}) \not\models \theta$.*

Thus, the Γ-stutter factorization $\{i_k\}_{k \in [0, m_\infty]}$ of π partitions the trace in adjacent non-empty segments such that the valuation of formulas in Γ does not change within a segment, and changes in moving from a segment to the adjacent ones. This factorization induces in a natural way a trace obtained by selecting the first positions of the finite segments and all the positions of the unique infinite segment, if any. Formally, the Γ-*stutter trace of* π, denoted by $stfr_\Gamma(\pi)$, is defined as follows:

- $stfr_\Gamma(\pi) \stackrel{\text{def}}{=} \pi(i_0)\pi(i_1)\ldots$ if $m_\infty = \infty$;
- $stfr_\Gamma(\pi) \stackrel{\text{def}}{=} \pi(i_0)\pi(i_1)\ldots\pi(i_{m_\infty - 1}) \cdot \pi^{i_{m_\infty}}$ if $m_\infty \neq \infty$.

As an example, assume that $\mathsf{AP} = \{p, q, r\}$ and let $\Gamma = \{p \,\mathsf{U}\, q\}$. Given $h, k \geq 1$, let $\pi_{h,k}$ be the trace $\pi_{h,k} = p^h q^k r^\omega$. These traces have the same Γ-stutter trace given by pr^ω. This is because for all $h', k' \geq 1$, both $p^{h'} q^{k'} r^\omega$ and $q^{k'} r^\omega$ satisfy $p \,\mathsf{U}\, q$, while r^ω does not. Hence, the $\{p \,\mathsf{U}\, q\}$-factorization of $\pi_{h,k}$ consists of two segments: the first one is $p^h q^k$ (the first position has valuation p) and the second one is r^ω.

We say that a trace π is Γ-*stutter free* if it coincides with its Γ-stutter trace, i.e. $stfr_\Gamma(\pi) = \pi$. Note that if $\Gamma = \emptyset$, each trace is $\emptyset$-stutter free, i.e. $stfr_\emptyset(\pi) = \pi$.

For each finite set Γ of LTL formulas, we define the successor function $succ_\Gamma$ as follows. The function maps a pointed trace (π, i) to the trace (π, ℓ) where ℓ is the first position of the segment in the Γ-stutter factorization of π following the i-segment if the i-segment is not the last one; otherwise, ℓ is $i + 1$. Formally,

Definition III.2 (Relativized Successor). *Let Γ be a finite set of LTL formulas, π a trace with Γ-stutter factorization $\{i_k\}_{k \in [0, m_\infty]}$, and $i \geq 0$. The Γ-successor of the pointed trace (π, i), denoted by $succ_\Gamma(\pi, i)$, is the trace (π, ℓ) where position ℓ is defined as follows: if there is $j < m_\infty$ such that $i \in [i_j, i_{j+1})$, then $\ell = i_{j+1}$; otherwise (note that in this case $m_\infty \neq \infty$ and $i \geq i_{m_\infty}$), $\ell = i + 1$.*

B. Syntax and Semantics of Stuttering HyperLTL

Stuttering HyperLTL (HyperLTL$_S$) formulas over the given finite set AP of atomic propositions and finite set VAR of trace variables are linear-time hyper expressions over multi-trace specifications ψ, called *HyperLTL$_S$ quantifier-free formulas*, where ψ is defined by the following syntax:

$$\psi ::= \top \mid p[x] \mid \neg\psi \mid \psi \wedge \psi \mid \mathsf{X}_\Gamma \psi \mid \psi \,\mathsf{U}_\Gamma\, \psi$$

where $p \in \mathsf{AP}$, $x \in \mathsf{VAR}$, Γ is a finite set of LTL formulas over AP, and X_Γ and U_Γ are the stutter-relativized versions of the LTL temporal modalities.

When Γ is empty, we omit the subscript Γ in the temporal modalities. Informally, $p[x]$ asserts that p holds at the pointed trace assigned to x, while the relativized temporal modalities X_Γ and U_Γ are evaluated by a lockstepwise traversal of the Γ-stutter traces associated with the currently quantified traces. We also exploit the standard logical connectives $\vee$ (disjunction) and $\rightarrow$ (implication) as abbreviations, and the *relativized eventually* modality $\mathsf{F}_\Gamma\psi \stackrel{\text{def}}{=} \top\,\mathsf{U}_\Gamma\,\psi$ and its dual $\mathsf{G}_\Gamma\psi \stackrel{\text{def}}{=} \neg\mathsf{F}_\Gamma\neg\psi$ (*relativized always*). The size $|\xi|$ of a HyperLTL$_S$ (quantifier-free) formula ξ is the number of distinct sub-formulas of ξ *plus* the number of distinct sub-formulas of those LTL formulas occurring in the subscripts of the temporal modalities.

For each finite set Γ of LTL formulas, we denote by HyperLTL$_S[\Gamma]$ the syntactical fragment of HyperLTL$_S$ where the subscript of each temporal modality is Γ. Note that standard HyperLTL corresponds to the fragment HyperLTL$_S[\emptyset]$. In the following, for each HyperLTL$_S$ formula φ, we denote by HyperLTL(φ) the HyperLTL formula obtained from φ by replacing each relativized temporal modality in φ with its $\emptyset$-relativized version.

Semantics of HyperLTL$_S$ Quantifier-free Formulas. Given a finite set Γ of LTL formulas, we extend in a natural way the relativized successor function $succ_\Gamma$ to pointed trace assignments Π as follows: the Γ-*successor* $succ_\Gamma(\Pi)$ *of* Π is the pointed trace assignment with domain $Dom(\Pi)$ associating to each $x \in Dom(\Pi)$ the Γ-successor $succ_\Gamma(\Pi(x))$ of the pointed trace $\Pi(x)$. For each $j \in \mathbb{N}$, we use $succ_\Gamma^j$ for the function obtained by j applications of the function $succ_\Gamma$.

Given a HyperLTL$_S$ quantifier-free formula ψ and a pointed trace assignment Π such that $Dom(\Pi)$ contains the trace variables occurring in ψ, the satisfaction relation $\Pi \models \psi$ is inductively defined as follows (we omit the semantics of the Boolean connectives which is standard):

$$\begin{array}{ll}
\Pi \models p[x] & \Leftrightarrow \Pi(x) = (\pi, i) \text{ and } p \in \pi(i) \\
\Pi \models \mathsf{X}_\Gamma\psi & \Leftrightarrow succ_\Gamma(\Pi) \models \psi \\
\Pi \models \psi_1\,\mathsf{U}_\Gamma\,\psi_2 & \Leftrightarrow \text{for some } i \geq 0 : succ_\Gamma^i(\Pi) \models \psi_2 \text{ and} \\
 & \quad succ_\Gamma^k(\Pi) \models \psi_1 \text{ for all } 0 \leq k < i
\end{array}$$

In the following, given a set Γ of LTL formulas, we also consider the model checking problem against the fragment HyperLTL$_S[\Gamma]$ of HyperLTL$_S$. For this fragment, by the semantics of HyperLTL$_S$, we deduce the following fact, where for a set $\mathcal{L}$ of traces, $stfr_\Gamma(\mathcal{L})$ denotes the set of Γ-stutter traces over the traces in $\mathcal{L}$, i.e. $stfr_\Gamma(\mathcal{L}) \stackrel{\text{def}}{=} \{stfr_\Gamma(\pi) \mid \pi \in \mathcal{L}\}$.

Remark III.1. *A set $\mathcal{L}$ of traces is a model of a HyperLTL$_S[\Gamma]$ sentence φ if and only if $stfr_\Gamma(\mathcal{L})$ is a model of the HyperLTL sentence HyperLTL(φ).*

Let LTL$_S$ be the extension of LTL obtained by adding the stutter-relativized versions of the LTL temporal modalities. Note that LTL$_S$ formulas correspond to *one-variable* HyperLTL$_S$ quantifier-free formulas. We can show that LTL$_S$ has the same expressiveness as LTL, as established by the following Proposition III.1 (missing proofs of all the claims in this paper can be found in [23]). On the other hand, HyperLTL$_S$ *quantifier-free* formulas are in general more expressive than HyperLTL *quantifier-free* formulas. Indeed, multiple traces of fixed arity (i.e., the number of distinct variables in the quantifier-free formula) can be seen as single traces where one considers a copy of propositions for each trace. With this encoding, quantifier-free HyperLTL can express only LTL properties. On the other hand, quantifier-free HyperLTL$_S$ can express powerful non-regular properties. For instance, let $\mathsf{AP} = \{p\}$ and for all $n, m, k \geq 1$, let $\pi_{k,m}$ and π'_n be the traces defined as: $\pi_{k,m} = \emptyset^k(\{p\}\emptyset)^m\emptyset^\omega$ and $\pi'_n = \emptyset(\{p\}\emptyset)^n\emptyset^\omega$. Evidently, the language $\mathcal{L} = \{(\pi_{k,n}, \pi'_n) \mid k, n \geq 1\}$ is not regular. On the other hand, $\mathcal{L}$ can be easily captured by a two-variable quantifier-free formula in HyperLTL$_S[\mathsf{AP}]$.

Proposition III.1. *Given a LTL$_S$ formula, one can construct in polynomial time an equivalent LTL formula.*

We now show that HyperLTL$_S$ is strictly less expressive than the fixpoint calculus H$_\mu$ introduced in [20]. Indeed, H$_\mu$ cannot be embedded into HyperLTL$_S$ since for singleton trace sets, H$_\mu$ characterizes the class of ω-regular languages, while HyperLTL$_S$ corresponds to LTL, which consequently, captures only a strict subclass of ω-regular languages. Moreover, by the following result and the fact that parity AAWA are equivalent to H$_\mu$ quantifier-free formulas [20], we obtain that HyperLTL$_S$ is subsumed by H$_\mu$.

Proposition III.2. *Given a HyperLTL$_S$ quantifier-free formula ψ with trace variables $x_1, \ldots, x_n$, one can build in polynomial time a Büchi nAAWA $\mathcal{A}_\psi$ such that $\mathcal{L}(\mathcal{A}_\psi)$ is the set of n-tuples $(\pi_1, \ldots, \pi_n)$ of traces so that $(\{x_1 \mapsto (\pi_1, 0), \ldots, x_1 \mapsto (\pi_n, 0)\}) \models \psi$.*

Proof. By exploiting the dual R_Γ (*relativized release*) of the until modality U_Γ, we can assume without loss of generality that ψ is in *negation normal form*, so negation is applied only to relativized atomic propositions. Given a finite set Γ of LTL formulas, let ξ_Γ be the following LTL formula

$$\xi_\Gamma = \bigwedge_{\xi \in \Gamma} \mathsf{G}(\xi \leftrightarrow \mathsf{X}\xi) \vee \bigvee_{\xi \in \Gamma} (\xi \leftrightarrow \neg\mathsf{X}\xi)$$

The LTL formula ξ_Γ has as models the traces π such that the first segment in the factorization of π is either infinite or has length 1. For each $i \in [1, n]$, we can easily construct in linear time (in the number of distinct sub-formulas in Γ) a Büchi nAAWA $\mathcal{A}_{\Gamma,i}$ accepting the n-tuples $(\pi_1, \ldots, \pi_n)$ of traces so that the i^{th} component π_i is a model of ξ_Γ. Similarly, we can also define $\overline{\mathcal{A}}_{\Gamma,i}$ accepting the n-tuples $(\pi_1, \ldots, \pi_n)$ of traces so that the i^{th} component π_i is not a model of ξ_Γ.

Let Υ be the set of subscripts Γ occurring in the temporal modalities of ψ. Then by exploiting the automata $\mathcal{A}_{\Gamma,i}$ and $\overline{\mathcal{A}}_{\Gamma,i}$ where $\Gamma \in \Upsilon$ and $i \in [1, n]$, we construct a Büchi nAAWA $\mathcal{A}_\psi$ satisfying Proposition III.2 as follows. Given an input multi-trace $(\pi_1, \ldots, \pi_n)$, the behaviour of the automaton $\mathcal{A}_\psi$ is subdivided in phases. At the beginning of each phase with current position vector $\wp = (j_1, \ldots, j_n)$, $\mathcal{A}_\psi$ keeps track in its state of the currently processed sub-formula θ of ψ.

By the transition function, θ is processed in accordance with the 'local' characterization of the semantics of the Boolean connectives and the relativized temporal modalities. Whenever θ is of the form $\theta_1 \mathsf{U}_\Gamma \theta_2$ or $\theta_1 \mathsf{R}_\Gamma \theta_2$, or θ is argument of a sub-formula of the form $\mathsf{X}_\Gamma \theta$, and $\mathcal{A}_\psi$ has to check that θ holds at the position vector $(succ_\Gamma(\pi_1, j_1), \ldots, succ_\Gamma(\pi_1, j_1))$, $\mathcal{A}_\psi$ moves along the directions $1, \ldots, n$ in turns. During the movement along direction $i \in [1, n]$, the automaton is in state (θ, i, Γ) and guesses that either (i) the next input position is in the current segment of the Γ-factorization of π_i and this segment is not the last one, or (ii) the previous condition does not hold, hence, the next input position corresponds to $succ_\Gamma(\pi_1, j_1)$. In the first (resp., second case) case, it activates in parallel a copy of the auxiliary automaton $\overline{\mathcal{A}}_{\Gamma,i}$ (resp., $\mathcal{A}_{\Gamma,i}$) for checking that the guess is correct and moves one position to the right along π_i. Moreover, in the first case, $\mathcal{A}_\psi$ remains in state (θ, i, Γ), while in the second case, the automaton changes direction by moving to the state $(\theta, i+1, \Gamma)$ if $i < n$, and starts a new phase by moving at state θ otherwise. □

Examples of Specifications. Stuttering HyperLTL can express relevant information-flow security properties for asynchronous frameworks such as distributed systems or cryptographic protocols. These properties specify how information may propagate from input to outputs by comparing distinct executions of a system possibly at different points of time. Assume that each user is classified either at a low security level, representing public information, or at a high level, representing secret information. Moreover, let LI be a set of propositions for describing inputs of low users, LO propositions that describe outputs of low users, and HI be a set of propositions for representing inputs of high users. As a first example, we consider the asynchronous variant of the *noninterference* property, as defined by Goguen and Meseguer [6], asserting that the observations of low users do not change when all high inputs are removed. In an asynchronous setting, a user cannot infer that a transition occurred if consecutive observations remain unchanged. In other terms, steps observed by a user do not correspond to the same number of steps in different executions of the system. Thus, since a low user can only observe the low output propositions, we require that for each trace π, there is a trace π' with no high inputs such that the LO-stutter traces of π and π' coincide, that is, π and π' are indistinguishable to a low user. This can be expressed in HyperLTL$_S$ as follows, where proposition $p_\emptyset$ denotes absence of high input.

$$\forall x.\, \exists y.\, \mathsf{G} p_\emptyset[y] \wedge \mathsf{G}_{LO} \bigwedge_{p \in LO} (p[x] \leftrightarrow p[y])$$

Assuming that the observations are not time-sensitive, noninterference cannot in general be expressed in HyperLTL unless one only considers systems where all the traces are LO-stutter free. Another relevant example is *generalized noninterference* as formulated in [7] which allows nondeterminism in the low-observable behavior and requires for all system traces π and π', the existence of an interleaved trace π'' whose high inputs are the same as π and whose low outputs are the same as π'. This property can be expressed in HyperLTL$_S$ as follows:

$$\forall x.\, \forall y.\, \exists z.\, \mathsf{G}_{HI} \bigwedge_{p \in HI} (p[y] \leftrightarrow p[z]) \wedge \mathsf{G}_{LO} \bigwedge_{p \in LO} (p[y] \leftrightarrow p[z])$$

Another classical security policy is *observational determinism* specifying that traces which have the same initial low inputs are indistinguishable to a low user. The following HyperLTL$_S$ formula captures observational determinism with equivalence of traces up to stuttering as formulated in [8].

$$\forall x.\, \forall y.\, \bigwedge_{p \in LI} (p[x] \leftrightarrow p[y]) \rightarrow \mathsf{G}_{LO} \bigwedge_{p \in LO} (p[x] \leftrightarrow p[y])$$

Lastly, an interesting feature of HyperLTL$_S$ is the possibility of combining asynchrony and synchrony constraints. We illustrate this ability by considering an unbounded time requirement which has application in the analysis of procedural software: "*whenever a procedure A is invoked, the procedure terminates, but there is no bound on the running time of A that upper-bounds the duration of A on all traces*". In other words, for every candidate bound k there is a trace in which A is invoked and terminates with a longer duration. We assume, crucially, that procedure A can be activated at most once along an execution, and we let c_A characterize the call to A and r_A the return from A. This requirement can be expressed in HyperLTL$_S$ as follows.

$$\forall x.\, \exists y.\, \mathsf{F} c_A[x] \rightarrow \left(\mathsf{F} c_A[y] \wedge \mathsf{X}_{\{\mathsf{F} c_A\}} \begin{pmatrix} \neg r_A[x] \wedge \neg r_A[y] \\ \mathsf{U}_{\{\mathsf{F} c_A\}} \\ r_A[x] \wedge \neg r_A[y] \end{pmatrix} \right)$$

Essentially, we claim there is always a call to A that runs for a longer period of time than any candidate maximum duration. Note that the occurrence of the relativized until $\mathsf{U}_{\{\mathsf{F} c_A\}}$ in the previous formula can be equivalently replaced by the standard until U. We have used the relativized until since the previous formula is in the fragment investigated in Subsection III-D below which enjoys a decidable model checking problem.

C. Undecidability of Model Checking HyperLTL$_S$

In this section, we establish the following negative result.

Theorem III.1. *The model checking problem for HyperLTL$_S$ is undecidable even for the HyperLTL$_S$ fragment where the quantifier alternation depth is 0 and the stutter-relativized temporal modalities just use two sets of LTL-formulas where one is empty and the other one consists of atomic propositions only.*

Theorem III.1 is proved by a reduction from the Post's Correspondence Problem (PCP, for short) [24]. We fix an instance $\mathcal{I}$ of PCP which is a tuple

$$\mathcal{I} = \langle \langle u_1^1, \ldots, u_n^1 \rangle, \langle u_1^2, \ldots, u_n^2 \rangle \rangle$$

where $n \geq 1$ and for each $1 \leq i \leq n$, u_i^1 and u_i^2 are non-empty finite words over a finite alphabet Σ. Let $[n] = \{1, \ldots, n\}$. A *solution of* $\mathcal{I}$ is a non-empty sequence $i_1, i_2, \ldots, i_k$ of integers in $[n]$ such that $u_{i_1}^1 \cdot u_{i_2}^1 \cdot \ldots \cdot u_{i_k}^1 = u_{i_1}^2 \cdot u_{i_2}^2 \cdot \ldots \cdot u_{i_k}^2$. PCP

consists in checking for a given instance $\mathcal{I}$, whether $\mathcal{I}$ admits a solution. This problem is known to be undecidable [24].

Assumption. We assume without loss of generality that each word u_i^ℓ of $\mathcal{I}$, where $i \in [n]$ and $\ell = 1, 2$, has length at least 2. Indeed, if this assumption does not hold, we consider the instance $\mathcal{I}'$ of PCP obtained from $\mathcal{I}$ by replacing each word u_i^ℓ of the form $a_1 \dots a_n$ with the word $a_1 a_1 \dots a_n a_n$ (i.e., we duplicate each symbol occurring in u_i^ℓ). Evidently $\mathcal{I}'$ has a solution if and only if $\mathcal{I}$ has a solution.

In order to encode the PCP instance $\mathcal{I}$ into an instance of the model checking problem for HyperLTL$_S$, we exploit the following set AP of atomic propositions, where $\#, p_1, \dots, p_n, q_1, q_2$ are fresh symbols not in Σ.

$$\mathsf{AP} \stackrel{\text{def}}{=} \Sigma \cup \{\#\} \cup \{p_1 \dots, p_n\} \cup \{q_1, q_2\}$$

Intuitively, for each $i \in [n]$ and $\ell = 1, 2$, propositions p_i and q_ℓ are exploited to mark each symbol of the word u_i^ℓ of the instance $\mathcal{I}$, while proposition $\#$ is used to mark only the last symbol of u_i^ℓ. Thus, for the word u_i^ℓ, we denote by $[u_i^\ell, p_i, q_\ell]$ the finite word over 2^{AP} of length $|u_i^\ell|$ obtained from u_i^ℓ by marking each symbol of u_i^ℓ with propositions p_i and q_ℓ and, additionally, by marking the last symbol of u_i^ℓ with proposition $\#$. Formally, $[u_i^\ell, p_i, q_\ell]$ is the finite word over 2^{AP} having length $|u_i^\ell|$ such that for each $0 \leq h < |u_i^\ell|$, $[u_i^\ell, p_i, q_\ell](h) = \{u_i^\ell(h), p_i, q_\ell\}$ if $h < |u_i^\ell| - 1$, and $[u_i^\ell, p_i, q_\ell](h) = \{u_i^\ell(h), p_i, q_\ell, \#\}$ otherwise.

Given a non-empty sequence $i_1, i_2, \dots, i_k$ of integers in $[n]$ and $\ell = 1, 2$, we encode the word $u_{i_1}^\ell \cdot u_{i_2}^\ell \cdot \dots \cdot u_{i_k}^\ell$ by the trace, denoted by $\pi_{i_1,\dots,i_k}^\ell$, defined as:

$$\pi_{i_1,\dots,i_k}^\ell \stackrel{\text{def}}{=} \{\#\} \cdot [u_{i_1}^\ell, p_{i_1}, q_\ell] \cdot \dots \cdot [u_{i_k}^\ell, p_{i_k}, q_\ell] \cdot \{\#\}^\omega$$

Let Γ be the set of atomic propositions given by $\Gamma = \{\#, p_1, \dots, p_n\}$. We crucially observe that since each word of $\mathcal{I}$ has length at least 2, the projection of the Γ-stutter trace $\mathit{stfr}_\Gamma(\pi_{i_1,\dots,i_k}^\ell)$ of $\pi_{i_1,\dots,i_k}^\ell$ over Γ is given by

$$\{\#\} \cdot \{p_{i_1}\} \cdot \{p_{i_1}, \#\} \cdot \dots \cdot \{p_{i_k}\} \cdot \{p_{i_k}, \#\} \cdot \{\#\}^\omega$$

Hence, we obtain the following characterization of non-emptiness of the set of $\mathcal{I}$'s solutions, where a *well-formed trace* is a trace of the form $\pi_{i_1,\dots,i_k}^\ell$ for some non-empty sequence $i_1, i_2, \dots, i_k$ of integers in $[n]$ and $\ell = 1, 2$.

Proposition III.3. *$\mathcal{I}$ has some solution* if and only *if there are two well-formed traces π_1 and π_2 satisfying the following conditions, where $\Gamma = \{\#, p_1, \dots, p_n\}$:*

1) *for each $\ell = 1, 2$, π_ℓ does not contain occurrences of propositions $q_{3-\ell}$, i.e. for each $h \in \mathbb{N}$, $q_{3-\ell} \notin \pi_\ell(h)$;*
2) *the projections of π_1 and π_2 over Σ coincide, i.e. for each $h \in \mathbb{N}$ and $p \in \Sigma$, $p \in \pi_1(h)$ iff $p \in \pi_2(h)$;*
3) *the projections of $\mathit{stfr}_\Gamma(\pi_1)$ and $\mathit{stfr}_\Gamma(\pi_2)$ over Γ coincide, i.e. for each $h \in \mathbb{N}$ and $p \in \Gamma$, $p \in \mathit{stfr}_\Gamma(\pi_1)(h)$ iff $p \in \mathit{stfr}_\Gamma(\pi_2)(h)$.*

By exploiting Proposition III.3, we construct a finite Kripke structure $\mathcal{K}_\mathcal{I}$ and a HyperLTL$_S$ sentence $\varphi_\mathcal{I}$ over AP whose quantifier alternation depth is 0 and whose temporal modalities are parameterized either by the empty set or by $\Gamma = \{\#, p_1, \dots, p_n\}$ such that $\mathcal{I}$ has a solution if and only if $\mathcal{K}_\mathcal{I} \models \varphi_\mathcal{I}$. Note that Theorem III.1 then follows directly by the undecidability of PCP.

First, we easily deduce the following result concerning the construction of the Kripke structure $\mathcal{K}_\mathcal{I}$.

Proposition III.4. *One can build in time polynomial in the size of $\mathcal{I}$ a finite Kripke structure $\mathcal{K}_\mathcal{I}$ over AP satisfying the following conditions:*

- *the set of traces of $\mathcal{K}_\mathcal{I}$ contains the set of well-formed traces;*
- *each trace of $\mathcal{K}_\mathcal{I}$ having a suffix where $\#$ always holds is a well-formed trace.*

Finally, the HyperLTL$_S$ sentence $\varphi_\mathcal{I}$ is defined as follows, where $\Gamma = \{\#, p_1, \dots, p_n\}$:

$$\begin{aligned}\varphi_\mathcal{I} ::= \quad & \exists x_1.\, \exists x_2.\, \mathsf{FG}(\#[x_1] \wedge \#[x_2]) \wedge \\ & \mathsf{G}(\neg q_2[x_1] \wedge \neg q_1[x_2]) \wedge \\ & \bigwedge_{p \in \Sigma} \mathsf{G}(p[x_1] \leftrightarrow p[x_2]) \wedge \bigwedge_{p \in \Gamma} \mathsf{G}_\Gamma(p[x_1]] \leftrightarrow p[x_2])\end{aligned}$$

Assume that $\varphi_\mathcal{I}$ is interpreted over the Kripke structure $\mathcal{K}_\mathcal{I}$ of Proposition III.4. Then, by Proposition III.4, the first conjunct in the body of $\varphi_\mathcal{I}$ ensures that the two traces π_1 and π_2 of $\mathcal{K}_\mathcal{I}$ selected by the existential quantification are well-formed traces. Moreover, the other three conjuncts in the body of $\varphi_\mathcal{I}$ correspond to Conditions (1)–(3) of Proposition III.3 over the selected traces π_1 and π_2. Hence, $\mathcal{K}_\mathcal{I} \models \varphi_\mathcal{I}$ if and only if there are two well-formed traces that satisfy Conditions (1)–(3) of Proposition III.3 if and only if $\mathcal{I}$ admits a solution. This concludes the proof of Theorem III.1.

D. A Decidable Fragment of HyperLTL$_S$

In the previous section, we have shown that the model checking problem is undecidable for the HyperLTL$_S$ sentences whose relativized temporal modalities exploit two distinct sets of LTL formulas. In this section, we establish that the use of a unique finite set Γ of LTL formulas as a subscript of the temporal modalities in the given formula leads to a decidable model checking problem. In particular, we consider the fragment of HyperLTL$_S$, we call *simple HyperLTL$_S$*, whose quantifier-free formulas ψ satisfy the following requirement: there exists a finite set Γ of LTL formulas (depending on ψ) such that ψ is a Boolean combination of quantifier-free formulas in HyperLTL$_S[\Gamma]$ and *one-variable* HyperLTL$_S$ quantifier-free formulas. Simple HyperLTL$_S$ strictly subsumes HyperLTL and can express interesting asynchronous security properties like asynchronous noninterference [6] and observational determinism [8]. In particular, the HyperLTL$_S$ sentences at the end of Subsection III-B used for expressing noninterference (but not generalized noninterference), observational determinism, and the unbounded time procedural requirement are simple HyperLTL$_S$ formulas.

We solve the (fair) model checking for simple HyperLTL$_S$ by a reduction to HyperLTL model checking, which is known

to be decidable [11]. Our reduction is exponential in the size of the given sentence. As a preliminary step, we first show, by an adaptation of the standard automata-theoretic approach for LTL [25], that the problem for a simple HyperLTL$_S$ sentence φ can be reduced in exponential time to the fair model checking against a sentence in the fragment HyperLTL$_S[\Gamma]$ for some set Γ of *atomic propositions* depending on φ.

Reduction to the Fragment HyperLTL$_S[\Gamma]$ with Γ being Propositional. In order to prove the reduction from simple HyperLTL$_S$ to HyperLTL$_S[\Gamma]$ for propositional Γ (formally expressed in Theorem III.2 below), we need some preliminary results. Recall that a Nondeterministic Büchi Automaton (NBA) is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_0, \Delta, Acc \rangle$, where Σ is a finite alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states, $\Delta \subseteq Q \times \Sigma \times Q$ is the transition function, and $Acc \subseteq Q$ is the set of *accepting* states. Given a infinite word w over Σ, a run of $\mathcal{A}$ over w is an infinite sequence of states $q_0, q_1, \ldots$ such that $q_0 \in Q_0$ and for all $i \geq 0$, $(q_i, w(i), q_{i+1}) \in \Delta$. The run is accepting if for infinitely many i, $q_i \in Acc$. The language $\mathcal{L}(\mathcal{A})$ accepted by $\mathcal{A}$ consists of the infinite words w over Σ such that there is an accepting run over w.

Fix a non-empty set Γ of LTL formulas over AP. The closure $cl(\Gamma)$ of Γ is the set of LTL formulas consisting of the sub-formulas of the formulas $\theta \in \Gamma$ and their negations (we identify $\neg\neg\theta$ with θ). Note that $\Gamma \subseteq cl(\Gamma)$. Without loss of generality, we can assume that $\mathsf{AP} \subseteq \Gamma$. Precisely, AP can be taken as the set of propositions occurring in the given simple HyperLTL$_S$ sentence and $cl(\Gamma)$ contains all the propositions in AP and their negations. For each formula $\theta \in cl(\Gamma) \setminus \mathsf{AP}$, we introduce a fresh atomic proposition not in AP, denoted by $at(\theta)$. Moreover, for allowing a uniform notation, for each $p \in \mathsf{AP}$, we write $at(p)$ to mean p itself. Let AP_Γ be the set AP extended with these new propositions. By a straightforward adaptation of the well-known translation of LTL formulas into equivalent NBA [25], we obtain the following result, where for an infinite word w over 2^{AP_Γ}, $(w)_{\mathsf{AP}}$ denotes the projection of w over AP.

Proposition III.5. *Given a finite set Γ of LTL formulas over AP, one can construct in single exponential time an NBA $\mathcal{A}_\Gamma$ over 2^{AP_Γ} with $2^{O(|\mathsf{AP}_\Gamma|)}$ states satisfying the following:*

1) *let $w \in \mathcal{L}(\mathcal{A}_\Gamma)$: then for all $i \geq 0$ and $\theta \in cl(\Gamma)$, $at(\theta) \in w(i)$ if and only if $((w)_{\mathsf{AP}}, i) \models \theta$.*
2) *for each trace π (i.e., infinite word over 2^{AP}), there exists $w \in \mathcal{L}(\mathcal{A}_\Gamma)$ such that $\pi = (w)_{\mathsf{AP}}$.*

Let $\mathcal{K} = \langle S, S_0, E, V \rangle$ be a finite Kripke structure over AP and $F \subseteq S$. Next, we consider the synchronous product of the fair Kripke structure $(\mathcal{K}, F)$ with the NBA $\mathcal{A}_\Gamma = \langle 2^{\mathsf{AP}_\Gamma}, Q, Q_0, \Delta, Acc \rangle$ over 2^{AP_Γ} of Proposition III.5 associated with Γ. More specifically, we construct a Kripke structure $\mathcal{K}_\Gamma$ over AP_Γ and a subset F_Γ of $\mathcal{K}_\Gamma$-states such that $\mathcal{L}(\mathcal{K}_\Gamma, F_\Gamma)$ is the set of words $w \in \mathcal{L}(\mathcal{A}_\Gamma)$ whose projections over AP are in $\mathcal{L}(\mathcal{K}, F)$. Formally, the Γ-*extension of* $(\mathcal{K}, F)$ is the fair Kripke structure $(\mathcal{K}_\Gamma, F_\Gamma)$ where $\mathcal{K}_\Gamma = \langle S_\Gamma, S_{0,\Gamma}, E_\Gamma, V_\Gamma \rangle$ and F_Γ are defined as follows:

- S_Γ is the set of tuples $(s, B, q, \ell) \in S \times 2^{\mathsf{AP}_\Gamma} \times Q \times \{1, 2\}$ such that $V(s) = B \cap \mathsf{AP}$;
- $S_{0,\Gamma} = S_\Gamma \cap (S_0 \times 2^{\mathsf{AP}_\Gamma} \times Q_0 \times \{1\})$;
- E_Γ consists of the following transitions:
  - $((s, B, q, 1), (s', B', q', \ell))$ such that $(s, s') \in E$, $(q, B, q') \in \Delta$, and $\ell = 2$ if $s \in F$ and $\ell = 1$ otherwise;
  - $((s, B, q, 2), (s', B', q', \ell))$ such that $(s, s') \in E$, $(q, B, q') \in \Delta$, and $\ell = 1$ if $q \in Acc$ and $\ell = 2$ otherwise.
- for each $(s, B, q, \ell) \in S_\Gamma$, $V_\Gamma((s, B, q, \ell)) = B$;
- $F_\Gamma = \{(s, B, q, 2) \in S_\Gamma \mid q \in Acc\}$.

By construction and Proposition III.5(2), we easily obtain the following result.

Proposition III.6. *For each infinite word w over 2^{AP_Γ}, $w \in \mathcal{L}(\mathcal{K}_\Gamma, F_\Gamma)$ if and only if $w \in \mathcal{L}(\mathcal{A}_\Gamma)$ and $(w)_{\mathsf{AP}} \in \mathcal{L}(\mathcal{K}, F)$. Moreover, for each $\pi \in \mathcal{L}(\mathcal{K}, F)$, there exists $w \in \mathcal{L}(\mathcal{K}_\Gamma, F_\Gamma)$ such that $(w)_{\mathsf{AP}} = \pi$.*

For each $\Gamma' \subseteq \Gamma$, let Γ'_{prop} be the set of propositions in AP_Γ associated with the formulas in Γ', in other words $\Gamma'_{prop} \stackrel{\text{def}}{=} \{at(\theta) \mid \theta \in \Gamma'\}$. By Propositions III.5–III.6, we deduce the following result which allows to reduce the fair model checking against a simple HyperLTL$_S$ sentence to the fair model checking against a HyperLTL$_S$ sentence in the fragment HyperLTL$_S[\Gamma'_{prop}]$ for some set Γ'_{prop} of atomic propositions.

Lemma III.1. *The following holds:*

1) *For each $\theta \in \Gamma$ and $w \in \mathcal{L}(\mathcal{K}_\Gamma, F_\Gamma)$, $at(\theta) \in w(0)$ iff $(w)_{\mathsf{AP}} \models \theta$.*
2) *For all $\Gamma' \subseteq \Gamma$ and $w \in \mathcal{L}(\mathcal{K}_\Gamma, F_\Gamma)$, $(\mathit{stfr}_{\Gamma'_{prop}}(w))_{\mathsf{AP}} = \mathit{stfr}_{\Gamma'}(\pi)$ where $\pi = (w)_{\mathsf{AP}}$.*

Proof. Property 1 directly follows from Proposition III.5(1) and Proposition III.6. Now, let us consider Property 2. Let $\Gamma' \subseteq \Gamma$, $w \in \mathcal{L}(\mathcal{K}_\Gamma, F_\Gamma)$, and $\pi = (w)_{\mathsf{AP}}$. By Proposition III.6, $w \in \mathcal{L}(\mathcal{A}_\Gamma)$. Moreover, by Property (1) of Proposition III.5, for all $i \geq 0$ and $\theta \in \Gamma'$, $at(\theta) \in w(i)$ if and only if $(\pi, i) \models \theta$. Since $\Gamma'_{prop} \stackrel{\text{def}}{=} \{at(\theta) \mid \theta \in \Gamma'\}$, it follows that $(\mathit{stfr}_{\Gamma'_{prop}}(w))_{\mathsf{AP}} = \mathit{stfr}_{\Gamma'}(\pi)$, and the result follows. □

We can now prove the desired result.

Theorem III.2. *Given a simple HyperLTL$_S$ sentence φ and a fair finite Kripke structure $(\mathcal{K}, F)$ over AP, one can construct in single exponential time in the size of φ, a HyperLTL$_S$ sentence φ' having the same quantifier prefix as φ and a fair finite Kripke structure $(\mathcal{K}', F')$ over an extension AP' of AP such that $|\varphi'| = O(|\varphi|)$, φ' is in the fragment HyperLTL$_S[\mathsf{AP}'']$ for some $\mathsf{AP}'' \subseteq \mathsf{AP}'$, $|\mathcal{K}'| = O(|\mathcal{K}| * 2^{O(|\varphi|)})$, and $\mathcal{L}(\mathcal{K}', F') \models \varphi'$ if and only if $\mathcal{L}(\mathcal{K}, F) \models \varphi$.*

Proof. By hypothesis, there is a finite set Γ' of LTL formulas such that φ is of the form

$$Q_n x_n . Q_{n-1} x_{n-1} . \ldots . Q_1 x_1 . \psi$$

where $n \geq 1$, $Q_i \in \{\exists, \forall\}$ for all $i \in [1,n]$, and ψ is a Boolean combination of quantifier-free formulas in a set $\Upsilon_1 \cup \Upsilon_{\Gamma'}$, where Υ_1 consists of *one-variable* HyperLTL$_S$ quantifier-free formulas and $\Upsilon_{\Gamma'}$ consists of quantifier-free formulas in HyperLTL$_S[\Gamma']$. By Proposition III.1, we can assume without loss of generality that the formulas in Υ_1 are *one-variable* HyperLTL quantifier-free formulas. Let LTL(Υ_1) be the set of LTL formulas corresponding to the formulas in Υ_1 (i.e., for each $\theta \in \Upsilon_1$, we take the LTL formula obtained from θ by removing the unique variable occurring in θ). We define:

- $\Gamma \stackrel{\text{def}}{=} \text{LTL}(\Upsilon_1) \cup \Gamma'$. We assume that Γ is not empty; otherwise the result is obvious.
- $(\mathcal{K}', F') \stackrel{\text{def}}{=} (\mathcal{K}_\Gamma, F_\Gamma)$, where $(\mathcal{K}_\Gamma, F_\Gamma)$ is the Γ-extension of $(\mathcal{K}, F)$;
- $\varphi' \stackrel{\text{def}}{=} Q_n x_n. \ldots . Q_1 x_1. \psi'$, where ψ' is defined as follows: by hypothesis, ψ can be seen as a propositional formula ψ_p over the set of atomic formulas $\Upsilon_1 \cup \Upsilon_{\Gamma'}$. Then, ψ' is obtained from ψ_p by replacing (i) each formula $\xi \in \Upsilon_1$ with the x-relativized proposition in AP_Γ given by $at(\text{LTL}(\xi))[x]$, where LTL(ξ) is the LTL formula associated with ξ and x is the unique variable occurring in ξ, and (ii) each formula $\xi \in \Upsilon_{\Gamma'}$ with the formula obtained from ξ by replacing each relativized temporal modality in ξ with its Γ'_{prop}-relativized version.

We show that $\mathcal{L}(\mathcal{K}_\Gamma, F_\Gamma) \models \varphi'$ if and only if $\mathcal{L}(\mathcal{K}, F) \models \varphi$. Hence, Theorem III.2 directly follows. For each $i \in [1,n]$, let $\varphi_i \stackrel{\text{def}}{=} Q_i x_i. \ldots . Q_1 x_1. \psi$ and $\varphi'_i \stackrel{\text{def}}{=} Q_i x_i. \ldots . Q_1 x_1. \psi'$. Moreover, we write φ_0 (resp., φ'_0) to mean formula ψ (resp., ψ'). The result directly follows from the following claim.

Claim. Let $0 \leq i \leq n$ and $w_1, \ldots, w_{n-i} \in \mathcal{L}(\mathcal{K}_\Gamma, F_\Gamma)$. Then, $(\mathcal{L}(\mathcal{K}_\Gamma, F_\Gamma), \{x_1 \mapsto (w_1, 0), \ldots, x_{n-i} \mapsto (w_{n-i}, 0)\}) \models \varphi'_i$ if and only if $(\mathcal{L}(\mathcal{K}, F), \{x_1 \mapsto ((w_1)_{\mathsf{AP}}, 0), \ldots, x_{n-i} \mapsto ((w_{n-i})_{\mathsf{AP}}, 0)\}) \models \varphi_i$.

Proof of the Claim. For the base case ($i = 0$), the result directly follows from construction and Lemma III.1. For the induction step, the result directly follows from the induction hypothesis and the second part of Proposition III.6. □

Fair Model Checking against HyperLTL$_S[\Gamma]$ with $\Gamma \subseteq \mathsf{AP}$. By Theorem III.2, we can restrict to consider the fair model checking against the fragments HyperLTL$_S[\Gamma]$ where Γ is a non-empty finite set of atomic propositions. We show that this problem can be reduced in polynomial time to a variant of model checking against HyperLTL.

Definition III.3 (LTL-conditioned model checking)**.** *For a Kripke structure $\mathcal{K}$ and a LTL formula θ, we denote by $\mathcal{L}(\mathcal{K}, \theta)$ the set of traces of $\mathcal{K}$ which satisfy θ. The* LTL-conditioned model checking problem against HyperLTL *is checking for a finite Kripke structure $\mathcal{K}$, a LTL formula θ and a HyperLTL sentence φ, whether $\mathcal{L}(\mathcal{K}, \theta) \models \varphi$.*

LTL-conditioned model checking against HyperLTL can be easily reduced in linear time to HyperLTL model checking (for details see [23]).

Proposition III.7. *Given an LTL formula θ and a HyperLTL sentence φ, one can construct in linear time a HyperLTL sentence φ_θ having the same quantifier prefix as φ such that for each Kripke structure $\mathcal{K}$, $\mathcal{L}(\mathcal{K}, \theta) \models \varphi$ iff $\mathcal{L}(\mathcal{K}) \models \varphi_\theta$.*

Let $(\mathcal{K}, F)$ be a fair finite Kripke structure with $\mathcal{K} = \langle S, S_0, E, V \rangle$ and φ be a HyperLTL$_S[\Gamma]$ sentence with $\Gamma \subseteq \mathsf{AP}$ and $\Gamma \neq \emptyset$. Let *acc* be a fresh proposition not in AP. Starting from $\mathcal{K}$, F, and Γ, we construct in polynomial time a finite Kripke structure $\widehat{\mathcal{K}}$ over $\widehat{\mathsf{AP}} = \mathsf{AP} \cup \{acc\}$ and an LTL formula $\widehat{\theta}$ over $\widehat{\mathsf{AP}}$ such that the projections over AP of the traces of $\widehat{\mathcal{K}}$ satisfying $\widehat{\theta}$ correspond to the traces in $stfr_\Gamma(\mathcal{L}(\mathcal{K}, F))$. By Remark III.1 and since φ does not contain occurrences of the special proposition *acc*, we obtain that $\mathcal{L}(\mathcal{K}, F)$ is a model of the HyperLTL$_S[\Gamma]$ sentence φ *iff* $\mathcal{L}(\widehat{K}, \widehat{\theta})$ is a model of the HyperLTL sentence HyperLTL(φ).

Intuitively, the Kripke structure $\widehat{\mathcal{K}}$ is obtained from $\mathcal{K}$ by adding edges which keep track of the states associated with the starting positions of adjacent segments along the Γ-stutter factorizations of (the traces of) the F-fair paths of $\mathcal{K}$. Formally, let $R_\Gamma(\mathcal{K})$ and $R_\Gamma(\mathcal{K}, F)$ be the sets of state pairs in $\mathcal{K}$ defined as follows:

- $R_\Gamma(\mathcal{K})$ consists of the pairs $(q, q') \in S \times S$ such that $V(q) \cap \Gamma \neq V(q') \cap \Gamma$ and there is a finite path of $\mathcal{K}$ of the form $q \cdot \rho \cdot q'$ such that $V(q) \cap \Gamma = V(\rho(i)) \cap \Gamma$ for all $0 \leq i < |\rho|$.
- $R_\Gamma(\mathcal{K}, F)$ is defined similarly but, additionally, we require that the finite path $q \cdot \rho \cdot q'$ visits some accepting state in F.

The finite sets $R_\Gamma(\mathcal{K})$ and $R_\Gamma(\mathcal{K}, F)$ can be easily computed in polynomial time by standard closure algorithms. By exploiting the sets $R_\Gamma(\mathcal{K})$ and $R_\Gamma(\mathcal{K}, F)$, we define the finite Kripke structure $\widehat{\mathcal{K}} = \langle \widehat{S}, \widehat{S_0}, \widehat{E}, \widehat{V} \rangle$ over $\widehat{\mathsf{AP}} = \mathsf{AP} \cup \{acc\}$ as follows:

- $\widehat{S} = S \times \{0,1\}$ and $\widehat{S_0} = S_0 \times \{0\}$.
- $\widehat{E}$ consists of the edges $((s,\ell),(s,\ell'))$ such that one of the following holds:
  - *either* $(s,s') \in E \cup R_\Gamma(\mathcal{K})$ and ($\ell' = 1$ iff $s' \in F$),
  - *or* $(s,s') \in R_\Gamma(\mathcal{K}, F)$ and $\ell' = 1$.
- $\widehat{V}((s,1)) = V(s) \cup \{acc\}$ and $\widehat{V}((s,0)) = V(s)$.

Let $\widehat{\theta}$ be the LTL formula over $\widehat{\mathsf{AP}}$ defined as follows:

$$\mathsf{G}\mathsf{F}acc \wedge \mathsf{G}\Big(\bigvee_{p \in \Gamma} (p \leftrightarrow \neg \mathsf{X} p) \vee \bigwedge_{p \in \Gamma} \mathsf{G}(p \leftrightarrow \mathsf{X} p)\Big)$$

The first conjunct in the definition of $\widehat{\theta}$ ensures that proposition *acc* holds infinitely often while the second conjunct captures the traces that are Γ-strutter free. By construction, we easily obtain the following result.

Proposition III.8. *$stfr_\Gamma(\mathcal{L}(\mathcal{K}, F))$ coincides with the set of projections over AP of the traces in $\mathcal{L}(\widehat{\mathcal{K}}, \widehat{\theta})$.*

Proof. Let $\pi \in stfr_\Gamma(\mathcal{L}(\mathcal{K}, F))$. Hence, there is a F-fair path ν of $\mathcal{K}$ such that $\pi = stfr_\Gamma(V(\nu))$ where $V(\nu)$ is the trace associated with ν. By construction, there is a trace $\widehat{\pi}$ of $\widehat{\mathcal{K}}$ such that $acc \in \widehat{\pi}(i)$ for infinitely many i and the projection of $\widehat{\pi}$ over AP coincides with π. By construction of the LTL

formula $\widehat{\theta}$, $\widehat{\pi}$ satisfies $\widehat{\theta}$. Hence, $\widehat{\pi} \in \mathcal{L}(\widehat{\mathcal{K}}, \widehat{\theta})$. The converse direction is similar. □

By Proposition III.8 and Remark III.1, we obtain that for each HyperLTL$_S$[Γ] sentence φ, $\mathcal{L}(\mathcal{K}, F) \models \varphi$ iff $\mathcal{L}(\widehat{K}, \widehat{\theta}) \models$ HyperLTL(φ).

By [13] the model checking problem of a finite Kripke structure $\mathcal{K}$ against a HyperLTL sentence φ of quantifier alternation depth d can be done in nondeterministic space bounded by $O(\mathsf{Tower}_2(d, |\varphi| \log(|\mathcal{K}|)))$. Thus, since simple HyperLTL$_S$ subsumes HyperLTL, by Theorem III.2 and Proposition III.7, we obtain the main result of this section, where the lower bounds correspond to the known ones for HyperLTL [13].

Theorem III.3. *For each $d \in \mathbb{N}$, (fair) model checking against simple HyperLTL$_S$ sentences of quantifier alternation depth d is d-*EXPSPACE*-complete, and for a fixed formula, it is $(d-1)$-*EXPSPACE*-complete for $d > 0$ and* NLOGSPACE*-complete otherwise.*

IV. CONTEXT HYPERLTL

In this section, we introduce an alternative logical framework for specifying asynchronous linear-time hyperproperties. The novel framework, we call *context* HyperLTL (HyperLTL$_C$ for short), extends HyperLTL by unary modalities $\langle C \rangle$ parameterized by a non-empty subset C of trace variables—called the *context*—which restrict the evaluation of the temporal modalities to the traces associated with the variables in C. Formally, HyperLTL$_C$ formulas over the given finite set AP of atomic propositions and finite set VAR of trace variables are linear-time hyper expressions over multi-trace specifications ψ, called *HyperLTL$_C$ quantifier-free formulas*, where ψ is defined by the following syntax:

$$\psi ::= \top \mid p[x] \mid \neg\psi \mid \psi \wedge \psi \mid \mathsf{X}\psi \mid \psi \,\mathsf{U}\, \psi \mid \langle C \rangle \psi$$

where $p \in \mathsf{AP}$, $x \in \mathsf{VAR}$, and $\langle C \rangle$ is the context modality with $\emptyset \neq C \subseteq \mathsf{VAR}$. A *context* is a non-empty subset of trace variables in VAR. The size $|\xi|$ of a HyperLTL$_C$ (quantifier-free) formula ξ is the number of distinct sub-formulas of ξ. A context C is *global for a formula* ξ if C contains all the trace variables occurring in ξ.

Semantics of HyperLTL$_C$ quantifier-free formulas. Let Π be a pointed trace assignment. Given a context C and an offset $i \geq 0$, we denote by $\Pi +_C i$ the pointed trace assignment with domain $\mathit{Dom}(\Pi)$ defined as follows:

- for each $x \in \mathit{Dom}(\Pi) \cap C$ with $\Pi(x) = (\pi, h)$, $[\Pi +_C i](x) = (\pi, h + i)$;
- for each $x \in \mathit{Dom}(\Pi) \setminus C$, $[\Pi +_C i](x) = \Pi(x)$.

Intuitively, the positions of the pointed traces associated with the variables in C advance of the offset i, while the positions of the other pointed traces remain unchanged.

Given a HyperLTL$_C$ quantifier-free formula ψ, a context C, and a pointed trace assignment Π such that $\mathit{Dom}(\Pi)$ contains the trace variables occurring in ψ, the satisfaction relation $(\Pi, C) \models \psi$ is inductively defined as follows (we omit the semantics of the Boolean connectives which is standard):

$$\begin{array}{ll}
(\Pi, C) \models p[x] & \Leftrightarrow \Pi(x) = (\pi, i) \text{ and } p \in \pi(i) \\
(\Pi, C) \models \mathsf{X}\psi & \Leftrightarrow (\Pi +_C 1, C) \models \psi \\
(\Pi, C) \models \psi_1 \,\mathsf{U}\, \psi_2 & \Leftrightarrow \text{for some } i \geq 0 : (\Pi +_C i, C) \models \psi_2 \\
 & \quad \text{and } (\Pi +_C k, C) \models \psi_1 \text{ for all } k < i \\
(\Pi, C) \models \langle C' \rangle \psi & \Leftrightarrow (\Pi, C') \models \psi
\end{array}$$

We write $\Pi \models \psi$ to mean that $(\Pi, \mathsf{VAR}) \models \psi$.

Examples of specifications. The logic Context HyperLTL extends HyperLTL by allowing to specify complex combinations of asynchronous and synchronous constraints. As an example, we consider the property [20] that a HyperLTL quantifier-free formula $\psi(x_1, \ldots, x_n)$ holds along the traces bound by variables $x_1 \ldots, x_n$ after an initialization phase. Note that this phase can take a different number of steps on each trace. The previous requirement can be expressed by an HyperLTL$_C$ quantifier-free formula as follows, where proposition in characterizes the initialization phase:

$$\begin{array}{l}
\langle \{x_1\} \rangle (in[x_1] \,\mathsf{U}\, (\neg in[x_1] \wedge \langle \{x_2\} \rangle (\ldots \\
\quad \langle \{x_n\} \rangle (in[x_n] \,\mathsf{U}\, (\neg in[x_n] \wedge \langle \{x_1, \ldots, x_n\} \rangle \psi)) \ldots)))
\end{array}$$

As another example, illustrating the high expressiveness of HyperLTL$_C$, we consider the following hyper-bounded-time response requirement: "*for every trace there is a bound k such that each request q is followed by a response p within k steps.*" This can be expressed in HyperLTL$_C$ as follows:

$$\begin{array}{l}
\forall x. \forall y. \left[\mathsf{F} q[x] \wedge \bigwedge\limits_{r \in \mathsf{AP}} \mathsf{G}(r[x] \leftrightarrow r[y]) \right] \longrightarrow \\
\quad \langle \{y\} \rangle \mathsf{F} \left(\begin{array}{l} q[y] \wedge \langle \{x\} \rangle \mathsf{G} \left(\begin{array}{l} q[x] \rightarrow \\ \{x, y\} (\neg p[y] \,\mathsf{U}\, p[x]) \end{array} \right) \end{array} \right)
\end{array}$$

Note that x and y refer to the same trace and the context modalities are exploited to synchronously compare distinct segments along the same trace, that correspond to different request-response intervals. This ability is not supported by Stuttering HyperLTL. On the other hand, we conjecture that unlike HyperLTL$_S$, HyperLTL$_C$ cannot express asynchronous variants of security properties such as noninterference and observational determinism (see Subsection III-B).

It is worth noting that the *global promptness version* (in the style of Prompt LTL [26]) of the previous bounded-time response requirement is expressible in HyperLTL$_C$ as well. In this setting, one need to check for a uniform bound k on the response time in all the traces of the system. This can be formalized by an HyperLTL$_C$ formula obtained by the formula above by replacing the quantifier prefix $\forall x \forall y$ with $\exists y \forall x$ and by removing the equality constraint on the traces for x and y.

A. Undecidability of model checking against HyperLTL$_C$

In this section, we establish that model checking against HyperLTL$_C$ is in general undecidable. Let $\mathcal{F}_0$ and $\mathcal{F}_1$ be the fragments of HyperLTL$_C$ consisting of the formulas such that the number of trace variables is 2, the nesting depth of context modalities is 2, and, additionally, (i) in $\mathcal{F}_0$ the quantifier

alternation depth is 0, and (ii) in $\mathcal{F}_1$ the quantifier alternation depth is 1 and each temporal modality in the scope of a non-global context is F.

Theorem IV.1. *Model checking against HyperLTL$_C$ is undecidable even for the fragments $\mathcal{F}_0$ and $\mathcal{F}_1$.*

Theorem IV.1 is proved by a polynomial-time reduction from the *halting problem for Minsky 2-counter machines* [27]. Such a machine is a tuple $M = \langle Q, q_{init}, q_{halt}, \Delta \rangle$, where Q is a finite set of (control) locations, $q_{init} \in Q$ is the initial location, $q_{halt} \in Q$ is the halting location, and $\Delta \subseteq Q \times L \times Q$ is a transition relation over the instruction set $L = \{\mathsf{inc}, \mathsf{dec}, \mathsf{zero}\} \times \{1, 2\}$. We adopt the following notational conventions. For an instruction $op = (\_, c) \in L$, let $c(op) \stackrel{\text{def}}{=} c \in \{1, 2\}$ be the *counter* associated with op. For a transition $\delta \in \Delta$ of the form $\delta = (q, op, q')$, we define $from(\delta) \stackrel{\text{def}}{=} q$, $op(\delta) \stackrel{\text{def}}{=} op$, $c(\delta) \stackrel{\text{def}}{=} c(op)$, and $to(\delta) \stackrel{\text{def}}{=} q'$. Without loss of generality, we assume that for each transition $\delta \in \Delta$, $from(\delta) \neq q_{halt}$.

An M-configuration is a pair (q, ν) consisting of a location $q \in Q$ and a counter valuation $\nu : \{1, 2\} \to \mathbb{N}$. A computation of M is a non-empty *finite* sequence $(q_1, \nu_1), \ldots, (q_k, \nu_k)$ of configurations such that for each $1 \leq i < k$, $(q_i, op, q_{i+1}) \in \Delta$ for some instruction $op \in L$ (depending on i) and the following holds, where $c \in \{1, 2\}$ is the counter associated with the instruction op: (i) $\nu_{i+1}(c') = \nu_i(c')$ if $c' \neq c$; (ii) $\nu_{i+1}(c) = \nu_i(c) + 1$ if $op = (\mathsf{inc}, c)$; (iii) $\nu_{i+1}(c) = \nu_i(c) - 1$ if $op = (\mathsf{dec}, c)$ (in particular, it has to be $\nu_i(c) > 0$); and (iv) $\nu_{i+1}(c) = \nu_i(c) = 0$ if $op = (\mathsf{zero}, c)$. M *halts* if there is a computation starting at the *initial* configuration (q_{init}, ν_{init}), where $\nu_{init}(1) = \nu_{init}(2) = 0$, and leading to some halting configuration (q_{halt}, ν). The halting problem is to decide whether a given machine M halts, and it is undecidable [27]. We prove the following result, from which Theorem IV.1 directly follows.

Proposition IV.1. *One can build a finite Kripke Structure $\mathcal{K}_M$ and a HyperLTL$_C$ sentence φ_M in the fragment $\mathcal{F}_0$ (resp., $\mathcal{F}_1$) such that M halts* iff $\mathcal{K}_M \models \varphi_M$.

Proof. Here, we focus on $\mathcal{F}_0$. First, we define an encoding of a computation of M as a trace where the finite set AP of atomic propositions is given by $\mathsf{AP} \stackrel{\text{def}}{=} \Delta \cup \{1, 2, beg_1, beg_2\}$.

Intuitively, in the encoding of an M-computation, we keep track of the transition used in the current step of the computation. Moreover, for each $c \in \{1, 2\}$, the propositions in $\{c, beg_c\}$ are used for encoding the current value of counter c. In particular, for $c \in \{1, 2\}$, a *c-code for the M-transition* $\delta \in \Delta$ is a finite word w_c over 2^{AP} of the form $\{\delta, beg_c\} \cdot \{\delta, c\}^h$ for some $h \geq 0$ such that $h = 0$ if $op(\delta) = (\mathsf{zero}, c)$. The c-code w_c encodes the value for counter c given by h (or equivalently $|w_c| - 1$). Note that only the occurrences of the symbols $\{\delta, c\}$ encode units in the value of counter c, while the symbol $\{\delta, beg_c\}$ is only used as left marker in the encoding. A *configuration-code w for the M-transition* $\delta \in \Delta$ is a finite word over 2^{AP} of the form $w = \{\delta\} \cdot w_1 \cdot w_2$ such that for each counter $c \in \{1, 2\}$, w_c is a c-code for transition δ. The configuration-code w encodes the M-configuration $(from(\delta), \nu)$, where $\nu(c) = |w_c| - 1$ for all $c \in \{1, 2\}$. Note that if $op(\delta) = (\mathsf{zero}, c)$, then $\nu(c) = 0$.

A *computation*-code is a trace of the form $\pi = w_{\delta_1} \cdots w_{\delta_k} \cdot \emptyset^\omega$, where $k \geq 1$ and for all $1 \leq i \leq k$, w_{δ_i} is a configuration-code for transition δ_i, and whenever $i < k$, it holds that $to(\delta_i) = from(\delta_{i+1})$. Note that by our assumptions $to(\delta_i) \neq q_{halt}$ for all $1 \leq i < k$. The computation-code π is *initial* if the first configuration-code w_{δ_1} encodes the initial configuration, and it is *halting* if for the last configuration-code w_{δ_k} in π, it holds that $to(\delta_k) = q_{halt}$. For all $1 \leq i \leq k$, let (q_i, ν_i) be the M-configuration encoded by the configuration-code w_{δ_i} and $c_i = c(\delta_i)$. The computation-code π is *good* if, additionally, for all $1 \leq j < k$, the following holds: (i) $\nu_{j+1}(c) = \nu_j(c)$ if either $c \neq c_j$ or $op(\delta_j) = (\mathsf{zero}, c_j)$ (*equality requirement*); (ii) $\nu_{j+1}(c_j) = \nu_j(c_j) + 1$ if $op(\delta_j) = (\mathsf{inc}, c_j)$ (*increment requirement*); (iii) $\nu_{j+1}(c_j) = \nu_j(c_j) - 1$ if $op(\delta_j) = (\mathsf{dec}, c_j)$ (*decrement requirement*).
Clearly, M halts *iff* there exists an initial and halting good computation-code. By construction, it is a trivial task to define a Kripke structure $\mathcal{K}_M$ satisfying the following.

Claim. One can construct in polynomial time a finite Kripke structure $\mathcal{K}_M$ over AP such that the set of traces of $\mathcal{K}_M$ which visit some empty position (i.e., a position with label the empty set of propositions) corresponds to the set of initial and halting computation-codes.

We now define a HyperLTL$_C$ sentence φ_M in the fragment $\mathcal{F}_0$ that, when interpreted on the Kripke structure $\mathcal{K}_M$, captures the traces π of $\mathcal{K}_M$ which visit some empty position (hence, by the previous claim, π is an initial and halting computation-code) and satisfy the goodness requirement.

$$\varphi_M \stackrel{\text{def}}{=} \exists x_1.\, \exists x_2.\mathsf{G} \bigwedge_{p \in \mathsf{AP}} (p[x_1] \leftrightarrow p[x_2]) \wedge \mathsf{F} \bigwedge_{p \in \mathsf{AP}} \neg p[x_1] \wedge \psi_{good}$$

where the HyperLTL$_C$ quantifier-free sub-formula ψ_{good} is defined in the following. Intuitively, when interpreted on the Kripke structure $\mathcal{K}_M$ of the previous claim, formula φ_M asserts the existence of two traces π_1 and π_2 bounded to the trace variables x_1 and x_2, respectively, such that (i) π_1 and π_2 coincide (this is ensured by the first conjunct); (ii) π_1 is an initial and halting computation-code (this is ensured by the previous claim and the second conjunct); (iii) π_1 satisfies the goodness requirement by means of the conjunct ψ_{good}.

We now define the quantifier-free formula ψ_{good}. Let $\Delta_{halt} \stackrel{\text{def}}{=} \{\delta \in \Delta \mid to(\delta) = q_{halt}\}$ be the set of transitions having as a target location the halting location. In the definition of ψ_{good}, we crucially exploit the context modalities. Essentially, for each position $i \geq 0$ along π_1 and π_2 corresponding to the initial position of a c-code for a transition $\delta \notin \Delta_{halt}$ within a configuration code w_δ, we exploit:

- temporal modalities in the scope of the context modality $\langle\{x_2\}\rangle$ for moving the current position along trace π_2 (the trace bounded by x_2) to the beginning of the c-code of the configuration code w' following w_δ,

- and then we use the temporal modalities in the scope of the global context $\{x_1, x_2\}$ for synchronously ensuring that for the c-codes associated to the consecutive configuration codes w_δ and w', the equality, increment, and decrement requirements are fulfilled.

Formally, the HyperLTL$_C$ quantifier-free formula ψ_{good} is defined as follows:

$$\psi_{good} \stackrel{\text{def}}{=} \mathsf{G} \bigwedge_{\delta \in \Delta \setminus \Delta_{halt}} \bigwedge_{c \in \{1,2\}} \Big[(\delta[x_1] \wedge beg_c[x_1]) \longrightarrow \langle \{x_2\} \rangle \mathsf{X} \Big(\neg beg_c[x_2] \,\mathsf{U}\, \big(beg_c[x_2] \wedge \langle \{x_1, x_2\} \rangle (\psi_=(\delta, c) \wedge \psi_{inc}(\delta, c) \wedge \psi_{dec}(\delta, c))\big)\Big)\Big]$$

where the sub-formulas $\psi_=(\delta, c)$, $\psi_{inc}(\delta, c)$, and $\psi_{dec}(\delta, c)$ capture the equality, increment, and decrement requirement, respectively, and are defined as follows.

$$\psi_=(\delta, c) \stackrel{\text{def}}{=} [c \neq c(\delta) \vee op(\delta) = (\mathsf{zero}, c)] \longrightarrow \mathsf{X}[(c[x_1] \wedge c[x_2]) \,\mathsf{U}\, (\neg c[x_1] \wedge \neg c[x_2])]$$

$$\psi_{inc}(\delta, c) \stackrel{\text{def}}{=} op(\delta) = (\mathsf{inc}, c) \longrightarrow \mathsf{X}[(c[x_1] \wedge c[x_2]) \,\mathsf{U}\, (\neg c[x_1] \wedge c[x_2] \wedge \mathsf{X} \neg c[x_2])]$$

$$\psi_{dec}(\delta, c) \stackrel{\text{def}}{=} op(\delta) = (\mathsf{dec}, c) \longrightarrow \mathsf{X}[(c[x_1] \wedge c[x_2]) \,\mathsf{U}\, (c[x_1] \wedge \neg c[x_2] \wedge \mathsf{X} \neg c[x_1])]$$

This finishes the proof. □

B. Fragment of HyperLTL$_C$ with decidable model checking

By Theorem IV.1, model checking HyperLTL$_C$ is undecidable even for formulas where F is the unique temporal modality occurring in the scope of a non-global context operator. This justifies the investigation of the fragment, we call *bounded HyperLTL$_C$*, consisting of the HyperLTL$_C$ formulas where the unique temporal modality occurring in a non-global context is the next modality X. For instance, for each $k \geq 0$, the formula $\langle \{x_1\} \rangle \mathsf{X}^k (\langle \{x_1, x_2\} \rangle \mathsf{G}(p[x_1] \leftrightarrow p[x_2]))$ is bounded while the formula $\langle \{x_1\} \rangle \mathsf{F} (\langle \{x_1, x_2\} \rangle \mathsf{G}(p[x_1] \leftrightarrow p[x_2]))$ is not. Note that bounded HyperLTL$_C$ subsumes HyperLTL and is able to express a restricted form of asynchronicity by allowing to compare traces at different timestamps whose distances are bounded (a bound is given by the nesting depth of next modalities in the formula). As an example, the after-initialization synchronization requirement described after the definition of HyperLTL$_C$ can be expressed by assuming that the lengths of the initialization phases differ at most a given integer k. We conjecture that bounded HyperLTL$_C$ is not more expressive than HyperLTL. However, as a consequence of Theorem IV.2 below, for a fixed quantifier alternation depth, bounded HyperLTL$_C$ is at least singly exponentially more succinct than HyperLTL.

We show that model checking against bounded HyperLTL$_C$ is decidable by a polynomial-time translation of bounded HyperLTL$_C$ quantifier-free formulas ψ into equivalent $(|\psi| + 1)$-synchronous Büchi AAWA.

Proposition IV.2. *Given a HyperLTL$_C$ quantifier-free formula ψ with trace variables $x_1, \ldots, x_n$, one can build in polynomial time a Büchi nAAWA $\mathcal{A}_\psi$ such that $\mathcal{L}(\mathcal{A}_\psi)$ is the set of n-tuples $(\pi_1, \ldots, \pi_n)$ of traces so that $(\{x_1 \mapsto (\pi_1, 0), \ldots, x_1 \mapsto (\pi_n, 0)\}, \{x_1, \ldots, x_n\}) \models \psi$. Moreover, $\mathcal{A}_\psi$ is $(|\psi| + 1)$-synchronous if ψ is in the bounded fragment of HyperLTL$_C$.*

Proof. By exploiting the release modality R (the dual of the until modality), we can assume without loss of generality that ψ is in negation normal form, so negation is applied only to relativized atomic propositions. The construction of the Büchi nAAWA $\mathcal{A}_\psi$ is a generalization of the standard translation of LTL formulas into equivalent standard Büchi alternating word automata. In particular, the automaton $\mathcal{A}_\psi$ keeps track in its state of the sub-formula of ψ currently processed, of the current context C, and of a counter modulo the cardinality $|C|$ of C. This counter is used for recording the directions associated to the variables in C for which a move of one position to the right has already been done in the current phase of $|C|$-steps. By construction, whenever the automaton is in a state associated with a sub-formula θ of ψ, then $\mathcal{A}_\psi$ can move only to states associated with θ or with strict sub-formulas of θ. In particular, each path in a run of $\mathcal{A}_\psi$ can be factorized into a finite number $\nu_1, \ldots, \nu_k$ of contiguous segments (with ν_k possibly infinite) such that for each $i \in [1, k]$, segment ν_i is associated with a sub-formula θ_i of ψ and a context C_i occurring in ψ, and the following holds, where the *offset* of a position vector $\wp = (j_1, \ldots, j_n)$ in $\mathbb{N}^n$ is the maximum over the differences between pairs of components, i.e. $\max(\{j_\ell - j_{\ell'} \mid \ell, \ell' \in [1, n]\})$:

- there is some occurrence of θ_i in ψ which is in the scope of the context modality $\langle C_i \rangle$;
- if $i < k$, then θ_{i+1} is a strict sub-formula of θ_i;
- if either C_i is global or the root modality of θ_i is not in $\{\mathsf{U}, \mathsf{R}\}$, then the offset at each node along the segment ν_i and at the first node of ν_{i+1} if $i < k$ is at most the offset at the beginning of ν_i plus one.

Hence, if ψ is in the bounded fragment of HyperLTL$_C$, the offset at each node of a run is at most $|\psi| + 1$, i.e. $\mathcal{A}_\psi$ is $(|\psi| + 1)$-synchronous and the result follows. □

By exploiting Propositions II.1 and IV.2, we deduce that for a fixed quantifier alternation depth d, model checking against bounded HyperLTL$_C$ is $(d+1)$-EXPSPACE-complete, hence singly exponentially harder than model checking against HyperLTL. However, for a fixed formula, the complexity of the problem is the same as for HyperLTL.

Theorem IV.2. *Let $d \in \mathbb{N}$. The (fair) model checking problem against bounded HyperLTL$_C$ sentences of quantifier alternation depth d is $(d+1)$-EXPSPACE-complete, and for a fixed formula, it is $(d-1)$-EXPSPACE-complete for $d > 0$ and NLOGSPACE-complete otherwise.*

Proof. The upper bounds directly follow from Propositions II.1 and IV.2, while since bounded HyperLTL$_C$ subsumes HyperLTL, the lower bound for a fixed formula of alternation

depth d is inherited from the known one for HyperLTL [13]. Finally, for $(d+1)$-EXPSPACE-hardness, we adapt the reduction given in [13] for showing that for all integer constants $c > 1$ and $c' \geq 1$, model checking against HyperLTL sentences φ with quantifier alternation depth d requires space at least $\Omega(\mathsf{Tower}_c(d, |\varphi|^{c'}))$. Here, for simplicity, we assume that $c = 2$ and $c' = 1$. The reduction in [13] for model checking HyperLTL is based on building, for each $n > 1$, an HyperLTL formula of size polynomial in n, with quantifier alternation depth d over a singleton set $\mathsf{AP} = \{p\}$ of atomic propositions. This formula is of the form $\psi_d(x, y)$ for two free trace variables x and y such that for all traces π_x and π_y (over AP), $\{x \mapsto (\pi_x, 0), y \mapsto (\pi_y, 0)\} \models \psi_d(x, y)$ *if and only if* p occurs exactly once in π_x (resp., π_y) and p occurs on π_y exactly $g(d+1, n)$ positions after p occurs on π_x, where

- $g(0, n) = \mathsf{Tower}_2(0, n) = n$;
- $g(d+1, n) = g(d, n) * \mathsf{Tower}_2(d+1, n)$.

The construction is given by induction on d, and the formula $\psi_0(x, y)$ for the base case $d = 0$ and a fixed $n > 1$ do not use universal quantifiers (note that $\psi_0(x, y)$ requires that $p[y]$ occurs exactly $n * 2^n$ positions after $p[x]$ occurs). Thus, since bounded HyperLTL$_C$ subsumes HyperLTL and $g(2, n) = n * 2^n * 2^{2^n}$, in order to show that model checking against bounded HyperLTL$_C$ formulas φ with quantifier alternation depth d requires space at least $\Omega(\mathsf{Tower}_2(d+1, |\varphi|))$, it suffices to show the following result.

Claim. Let $\mathsf{AP} = \{p\}$ and $n > 1$. One construct in time polynomial in n a bounded HyperLTL$_C$ formula $\psi(x, y)$ with two free variables x and y and not containing universal quantifiers (hence, the quantifier alternation depth is 0) such that for all traces π_x and π_y, $\{x \mapsto (\pi_x, 0), y \mapsto (\pi_y, 0)\} \models \psi(x, y)$ iff

- p occurs exactly once on π_x (resp., π_y);
- for each $i \geq$, $p \in \pi_x(i)$ iff $p \in \pi_y(i + n * 2^n * 2^{2^n})$.

□

V. CONCLUSIONS

We have introduced in this paper two extensions of HyperLTL to express asynchronous hyperproperties: HyperLTL$_S$ and HyperLTL$_C$. Even though the model-checking problems of these logics are in general undecidable, we have identified for each of them a decidable fragment that subsumes HyperLTL and allows to express asynchronous properties of interest.

We plan to extend our work in many directions. First, we intend to settle the question concerning the comparison of the expressive power of HyperLTL$_S$ and HyperLTL$_C$. Second, we aim to understand the decidability border of model checking syntactical fragments of the framework resulting by combining HyperLTL$_S$ and HyperLTL$_C$. In particular, the decidability status of model checking against the fragment obtained by merging simple HyperLTL$_S$ and bounded HyperLTL$_S$ is open. Finally, other goals regard the extensions of the considered logic to the branching-time setting and the investigation of first-order and monadic second-order logics for the specification of asynchronous hyperproperties in the linear-time and branching-time settings.

REFERENCES

[1] E. Clarke and E. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Proc. of LP'81*, ser. LNCS, vol. 131. Springer, 1981, pp. 52–71.

[2] J. Queille and J. Sifakis, "Specification and verification of concurrent programs in Cesar," in *SP'81*, ser. LNCS, vol. 137. Springer, 1981, pp. 337–351.

[3] A. Pnueli, "The temporal logic of programs," in *Proc. 18th FOCS*. IEEE Computer Society, 1977, pp. 46–57.

[4] E. Emerson and J. Halpern, ""Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic," *J. ACM*, vol. 33, no. 1, pp. 151–178, 1986.

[5] M. Clarkson and F. Schneider, "Hyperproperties," *Journal of Computer Security*, vol. 18, no. 6, pp. 1157–1210, 2010.

[6] J. Goguen and J. Meseguer, "Security policies and security models," in *IEEE Symposium on Security and privacy*, vol. 12, 1982.

[7] J. McLean, "A general theory of composition for a class of "possibilistic" properties," *IEEE Trans. Software Eng.*, vol. 22, no. 1, pp. 53–67, 1996.

[8] S. Zdancewic and A. Myers, "Observational determinism for concurrent program security," in *Proc. 16th IEEE CSFW-16*. IEEE Computer Society, 2003, pp. 29–43.

[9] B. Finkbeiner, M. N. Rabe, and C. Sánchez, "Algorithms for model checking HyperLTL and HyperCTL*," in *Proc. 27th CAV Part I*, ser. LNCS, vol. 9206. Springer, 2015, pp. 30–48.

[10] R. Dimitrova, B. Finkbeiner, M. Kovács, M. Rabe, and H. Seidl, "Model checking information flow in reactive systems," in *Proc. 13th VMCAI*, ser. LNCS 7148. Springer, 2012, pp. 169–185.

[11] M. Clarkson, B. Finkbeiner, M. Koleini, K. Micinski, M. Rabe, and C. Sánchez, "Temporal logics for hyperproperties," in *Proc. 3rd POST*, ser. LNCS, vol. 8414. Springer, 2014, pp. 265–284.

[12] L. Bozzelli, B. Maubert, and S. Pinchinat, "Unifying Hyper and Epistemic Temporal Logics," in *Proc. 18th FoSSaCS*, ser. LNCS, vol. 9034. Springer, 2015, pp. 167–182.

[13] M. Rabe, "A temporal logic approach to information-flow control," Ph.D. dissertation, Saarland University, 2016.

[14] B. Finkbeiner and C. Hahn, "Deciding hyperproperties," in *Proc. 27th CONCUR*, ser. LIPIcs, vol. 59. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, pp. 13:1–13:14.

[15] N. Coenen, B. Finkbeiner, C. Hahn, and J. Hofmann, "The hierarchy of hyperlogics," in *Proc. 34th LICS*. IEEE, 2019, pp. 1–13.

[16] J. Gutsfeld, M. Müller-Olm, and C. Ohrem, "Propositional dynamic logic for hyperproperties," in *Proc. 31st CONCUR*, ser. LIPIcs 171. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 50:1–50:22.

[17] A. Sistla, M. Vardi, and P. Wolper, "The complementation problem for Büchi automata with appplications to temporal logic," *Theoretical Computer Science*, vol. 49, pp. 217–237, 1987.

[18] M. Fischer and R. Ladner, "Propositional dynamic logic of regular programs," *J. Comput. Syst. Sci.*, vol. 18, no. 2, pp. 194–211, 1979.

[19] B. Finkbeiner, "Temporal hyperproperties," *Bull. EATCS*, vol. 123, 2017.

[20] J. Gutsfeld, M. Müller-Olm, and C. Ohrem, "Automata and fixpoints for asynchronous hyperproperties," *Proc. ACM Program. Lang.*, vol. 4, no. POPL, 2021.

[21] J. Baumeister, N. Coenen, B. Bonakdarpour, B. Finkbeiner, and C. Sánchez, "A temporal logic for asynchronous hyperproperties," in *Proc. of 33rd CAV'21*, ser. LNCS, vol. 12759. Springer, 2021.

[22] B. Finkbeiner and M. Zimmermann, "The first-order logic of hyperproperties," in *Proc. 34th STACS*, ser. LIPIcs, vol. 66. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 30:1–30:14.

[23] L. Bozzelli, A. Peron, and C. Sánchez, "Asynchronous extensions of HyperLTL," *CoRR*, vol. abs/2104.12886, 2021. [Online]. Available: http://arxiv.org/abs/2104.12886

[24] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[25] M. Y. Vardi and P. Wolper, "Reasoning about infinite computations," *Inf. Comput.*, vol. 115, no. 1, pp. 1–37, 1994.

[26] O. Kupferman, N. Piterman, and M. Y. Vardi, "From Liveness to Promptness," *Formal Methods in System Design*, vol. 34, no. 2, pp. 83–103, 2009.

[27] M. L. Minsky, *Computation: Finite and Infinite Machines*, ser. Automatic Computation. Prentice-Hall, Inc., 1967.