# On the Hierarchy of Finite-Word Hyperlanguages

Hadar Frenkel
Department of Computer Science and
Artificial Intelligence, Bar-Ilan University
Ramat Gan, Israel
hadar.frenkel@biu.ac.il

Sarai Sheinvald
Faculty of Computer Science,
Technion – Israel Institute of Technology
Haifa, Israel
surke@technion.ac.il

***Abstract*—*Hyperproperties* lift conventional properties, which reason about traces, to properties that reason about sets of traces. Their formal-language counterpart are *hyperlanguages*, which are sets of languages. Several formalisms for hyperlanguages have been defined and studied: *hyperautomata* and *context-free hypergrammars* lift the notions of regular and context-free languages to *regular hyperlanguages* and *context-free hyperlanguages*, respectively. We unify and generalize these notions, and define general *hypermodels*, for both synchronous and asynchronous semantics. We then proceed to study the Chomsky hierarchy for hyperlanguages. In this setting, we study the relations within each semantic separately, as well as between the semantics. We show that both synchronous and asynchronous semantics follows the standard Chomsky hierarchy, separately. We further study the relations between the different levels of the hierarchy over the two semantics, and show them to be incomparable.**

## I. Introduction

Hyperproperties [16] generalize trace properties to system properties, and arise naturally in many applications, such as information-flow security [26], epistemic properties [20], [11], and causal analysis [17]. Due to the wide applicability of hyperproperties, and to their significant role in proving the correctness of programs, many formalisms have been proposed in recent years to express and to reason about them.

For reactive systems and infinite computations, the prominent logic to reason about hyperproperties is HyperLTL [15], that extends Linear Temporal Logic (LTL) [32] with quantification over traces, to reason about hyperproperties. Many logics extend HyperLTL to express a variety of hyperproperties [19], [34], [6]. One of the most studied extension types is to express *asynchronous* hyperproperties [1], [2], [4], [12], [27], [28], [29], [30], which arise naturally in many applications, particularly, in the context of verification of concurrent systems. The expressiveness of the different formalisms for infinite words has been extensively studied in recent years [18], [23], [13], [8], [24], focusing on temporal hyperlogics.

In this work, we focus on finite-word hyperproperties. These are becoming more and more relevant, with applications in runtime verification and monitoring [7], and with increasingly growing interest and applications in robotics [36] and planning [5]. In particular, finite-word versions of HyperLTL are used to express and verify planning problems [33], [36], [3], motivating us to formally explore the theory of finite-word hyperlanguages.

The main computational model used to verify temporal (hyper)properties is finite-state automata, which are used to both model the system and to algorithmically reason about the specification. Specifically, the vast majority of existing algorithms for verification of temporal hyperlogics are via automata reductions [22], [27], [2]. However, these reductions use standard ($\omega$)-automata, which do not capture the relational notion of hyperproperties.

In [10], [21] the authors presented hyperautomata – a synchronous formalism to express regular hyperlanguages, and studied their closure and decidability properties. A similar analysis was done in [25] for synchronous context-free hyperlanguages. Logical formalisms were also suggested, such as [9], which studies asynchronous temporal hyperlogic for finite traces.

Yet, while the expressive power of infinite-word hyperproperties is widely explored, and while the expressive power of finite-word formalisms such as finite automata, context-free grammars, etc., is well studied and defined by the Chomsky hierarchy [14], no such analysis exists for finite-word hyperlanguages. In addition, while temporal asynchronous hyperproperties over infinite words are very well studied, no general asynchronous computational model was suggested for finite-word hyperlanguages,[1] and the correspondence between synchronous and asynchronous computation models for finite-word hyperlanguages was, to the best of our knowledge, not studied.

In this paper, we address this gap and study the hierarchy of synchronous and asynchronous computational models for finite-word hyperlanguages. We show that each semantic – the synchronous and asynchronous – follows the Chomsky hierarchy. We further demonstrate that there is no clear correspondence between synchronous and asynchronous models at different levels of the hierarchy.

To properly analyze the correspondence between different models, our first contribution is the unification and generalization of previous notions of synchronous hyperautomata and hypergrammars, via the definition of general *hypermodels*, both for the synchronous and asynchronous semantics (Section III).

For synchronous hypermodels, we show that they follow the Chomsky hierarchy (Theorem V.2), with a strict containment at each level of the hierarchy. We also demonstrate that

[1] In [25] the authors define context-free hypergrammars, including asynchronous grammars. Similar models for asynchronous non-deterministic automata or for more general grammars were not defined.

https://doi.org/

| | | | | | |
|---|---|---|---|---|---|
| syncNFH | $\subsetneq$ | syncCFHG | $\subsetneq$ | syncCSHG | Theorem V.2 |
| $\not\cap$ | | $\not\cap$ | | | Theorem VI.1 |
| asyncNFH | $\subsetneq$ | asyncCFHG | $\subsetneq$ | asyncCSHG | Theorem VI.3, Theorem VI.5 |
| | | $\mathbb{Q}^k$asyncNFH | $\subsetneq$ | $\mathbb{Q}^{k+1}$asyncNFH | Corollary VI.4 |
| | | asyncNFH | $\not\subseteq$ | syncCFHG | Theorem VI.6, Theorem VI.1 |
| | | asyncNFH | $\not\supseteq$ | syncCFHG | Theorem VI.6, Theorem VI.3 |
| | | asyncCFHG | $\not\supseteq$ | syncCSHG | Theorem VI.7 |

TABLE I
SUMMARY OF OUR RESULTS

synchronous hypermodels are much more expressive than the respective non-hyper counterparts (Theorem V.1).

We establish a strict hierarchy also for asynchronous hypermodels: we show that asynchronous context-free hypergrammars are strictly more expressive than asynchronous hyperautomata (Theorem VI.3), and that asynchronous context-sensitive hypergrammars are strictly more expressive than asynchronous context-free hypergrammars (Theorem VI.5). Regarding the correspondence between the different semantics, we show that asynchronous hyperautomata and hypergrammars are strictly more expressive than synchronous hyperautomata and hypergrammars, respectively (Theorem VI.1). However, the correspondence between the asynchronous and the synchronous hypermodels at different levels of the hierarchy is not clear: Theorem VI.6 shows that there are hyperlanguages that are expressible via asynchronous hyperautomata and not via synchronous context-free hypergrammars; but we show that the opposite also holds, and some languages are expressible via synchronous context-free hypergrammars and not asynchronous hyperautomata. Thus, the stronger expressive power of context-free grammars does not suffice to overcome the stronger asynchronous behavior, but also vice versa.

Our results are summarized in Table I.

## II. Preliminaries

*Words and Languages:* Let $\Sigma$ be a finite alphabet. A *word* over $\Sigma$ is a finite sequence $\sigma_1 \cdots \sigma_n$ where $\sigma_i \in \Sigma$ for every $1 \leq i \leq n$. We denote $w[i] = \sigma_i$ and denote the length $n$ of $w$ by $|w|$. The *reversed word* of $w$ is $w^R = \sigma_n \sigma_{n-1} \dots \sigma_1$. The empty word is denoted by $\epsilon$. A *language* $L$ over $\Sigma$ is a set of words over $\Sigma$, and $\Sigma^*$ is the language of all words over $\Sigma$.

A *singleton language* is a language that consists of exactly one word, that is $L = \{w\}$ for $w \in \Sigma^*$.

*Nondeterministic Finite-word Automata:* A *nondeterministic finite-word automaton* (NFA) is a tuple $\langle \Sigma, Q, q_0, \delta, F \rangle$ where $Q$ is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is a set of *accepting states*, and $\delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*. A *run* of $A$ on a word $w = \sigma_1 \dots \sigma_n$ over $\Sigma$ is a sequence $r = r_0, r_1, ..., r_n$ of states such that: $(r_i, \sigma_{i+1}, r_{i+1}) \in \delta$ for every $0 \leq i < n$; and $r_0 = q_0$. We say that $r$ is *accepting* if $r_n \in F$. Otherwise, it is *rejecting*. The *language* of $A$ is $\mathcal{L}(A) = \{w \in \Sigma^* \mid \text{there exists an accepting run of } A \text{ on } w\}$. A language $\mathcal{L}$ is *regular* if there exists an NFA $A$ such that $\mathcal{L}(A) = \mathcal{L}$.

*Context-Free Grammars:* A *Context-Free Grammar* (CFG) over $\Sigma$ is a tuple $G = \langle \Sigma, V, S, P \rangle$ where $V$ is a finite set of variables, $S \in V$ is an initial variable, and $P \subseteq V \times (V \cup \Sigma)^*$ is a set of *derivation rules*. Given $v \in V, \alpha \in (V \cup \Sigma)^*$, we denote $v \to \alpha$ for $(v, \alpha) \in P$.

For $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$ and $v \in V$, we denote $\beta v \gamma \Rightarrow \beta \alpha \gamma$ if $v \to \alpha$. We denote $\beta \overset{*}{\Rightarrow} \gamma$ if there exists a sequence $\beta_1, ..., \beta_k$ such that $\beta_1 = \beta, \beta_k = \gamma$, and $\beta_i \Rightarrow \beta_{i+1}$ for every $1 \leq i < k$. The *Language* of $G$ is $\mathcal{L}(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$. A language $\mathcal{L}$ is *context-free* if there exists a CFG $G$ such that $\mathcal{L}(G) = \mathcal{L}$.

*Context-Sensitive Grammars:* A *Context-Sensitive Grammar* (CSG) over $\Sigma$ is a tuple $G = \langle \Sigma, V, S, P \rangle$ where $V$ and $S$ are as in CFG. The difference is in the derivation rules: as its name suggest, these depend not only on a variable, but on its surrounding symbols. Formally, a derivation rule in $P$ is one of the two forms:

- $S \to \epsilon$, or
- $\alpha A \beta \to \alpha \gamma \beta$ where: $\alpha, \beta \in (\Sigma \cup V)^*, A \in V$, and $\gamma \in (\Sigma \cup V)^+ \setminus \{S\}$. That is, only $S$ may derive $\epsilon$, and there is no derivation rule that can derive $S$.

The *Language* of $G$ is $\mathcal{L}(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$. A language $\mathcal{L}$ is *context-sensitive* if there exists a CSG $G$ such that $\mathcal{L}(G) = \mathcal{L}$.

*The Chomsky Hierarchy [14]:* Chomsky has famously shown that the families of regular, context-free, and context-sensitive languages form a chain of strict containment. That is, in terms of expressive power, we have:

$$\text{NFA} \subsetneq \text{CFG} \subsetneq \text{CSG}$$

*Hyperlanguages:* A *hyperlanguage* $\mathfrak{L}$ over $\Sigma$ is a set of languages over $\Sigma$, that is, $\mathfrak{L} \subseteq 2^{\Sigma^*}$. A hyperlanguage $\mathfrak{L}$ is a *singletons hyperlanguage* if it consists of singletons, i.e., languages of the type $\{w\}$ for a word $w$.

## III. Hypermodels

We now define *hypermodels*, which are computational models that are designed to recognize hyperlanguages. Hypermodels unify and generalize the earlier notions of *hyperautomata* introduced in [10], and *context-free hypergrammars* introduced in [25].

For an alphabet $\Sigma$ and $\# \notin \Sigma$, we denote $\Sigma \cup \{\#\}$ by $\Sigma_\#$. For a word $w \in \Sigma^*$, we define the *padding* of $w$ to be $\mathsf{pad}(w) = h^{-1}(w)$, where $h$ is the homomorphism

$$h(\tau) = \begin{cases} \tau & \tau \in \Sigma \\ \epsilon & \tau = \# \end{cases}$$

That is, $\mathsf{pad}(w)$ is the language of all words obtained from $w$ by adding occurrences of $\#$ freely within and around $w$.

Given a $k$-tuple $z = \langle w_1, ..., w_k \rangle$ of words over $\Sigma$, we define $\mathsf{pad}(z)$ to be the set of $k$-tuples of the form $\langle u_1, ..., u_k \rangle$, where $u_i \in \mathsf{pad}(w_i)$ for every $1 \leq i \leq k$, and where $|u_i| = |u_j|$ for every $1 \leq i < j \leq k$. That is, a $k$-tuple in $\mathsf{pad}(z)$ pads all words in $z$ in a way that obtains words of equal length. We abuse the notation and refer to a $k$-tuple in $\mathsf{pad}(z)$ of words of length $n$ as a word $\rho$ of length $n$ over $\Sigma_\#^k$ (which is the set of $k$-tuples of letters from $\Sigma_\#$), whose $i$'th letter is $\langle u_1[i], \ldots, u_k[i] \rangle$, for $1 \leq i \leq |\rho|$.

A tuple $\langle u_1, ..., u_k \rangle$ in $\mathsf{pad}(z)$ is *synced* if $\forall 1 \leq i \leq k : u_i \in \Sigma^* \cdot \{\#\}^*$. That is, in a synced tuple, the letter $\#$ is only allowed at the end of a word.

**Example III.1.** For the tuple of words $z = \langle aab, bb \rangle$, and for $\rho_1 = \langle a\#ab\#, b\#\#b\# \rangle, \rho_2 = \langle aab\#, bb\#\# \rangle$, we have $\rho_1, \rho_2 \in \mathsf{pad}(z)$. Further, $\rho_2$ is synced whereas $\rho_1$ is not.

We represent $\rho_2$ as

$$\langle a, b \rangle \langle a, b \rangle \langle b, \# \rangle \langle \#, \# \rangle$$

which is a word over $\{a, b, \#\}^2$.

We are now ready to define hypermodels.

**Definition III.1.** A *hypermodel* over $\Sigma$ is a tuple $\mathcal{M} = \langle M, \Sigma, X, \alpha, \mathsf{sync\text{-}type} \rangle$ where $M$ is a specific instance of a finite-word computational model over $\Sigma_\#^{|X|}$ (e.g., an NFA, or a CFG), which we refer to as the *underlying model of* $\mathcal{M}$, and where

- $X = \{x_1, \ldots, x_k\}$ is a finite set of *word variables*.
- $\alpha = \mathbb{Q}_1 x_1 \mathbb{Q}_2 x_2 \cdots \mathbb{Q}_k x_k$ is a *quantification condition*, where $\mathbb{Q}_i \in \{\forall, \exists\}$.
- $\mathsf{sync\text{-}type}$ is either $\mathsf{sync}$ (synchronous) or $\mathsf{async}$ (asynchronous).

When $\mathsf{sync\text{-}type} = \mathsf{sync}$, we require that the underlying model $M$ only recognizes synced word-tuples. For $\mathsf{sync\text{-}type} = \mathsf{async}$, there is no such limitation.

Let $L$ be a language over $\Sigma$, and let $\nu : X \to L$ be an assignment, that assigns each variable from $X$ a word in $L$. We sometimes refer to $\nu$ as the tuple $\langle \nu(x_1), \ldots, \nu(x_k) \rangle$. We denote by $\nu[x \mapsto w]$ the assignment that agrees with $\nu$ on all variables $x' \neq x$, and that assigns $x$ with $w$.

We define the hyperlanguage $\mathfrak{L}(\mathcal{M})$ of $\mathcal{M}$ inductively, based on $\nu$, using the relation $\vdash_{\nu,\alpha}$ and with respect to the language $\mathcal{L}(M)$ of the underlying model $M$.

- If $\alpha = \epsilon$ then $L \vdash_{\nu,\alpha} \mathcal{M}$ iff there exists some $z$ in $\mathsf{pad}(\nu)$ such that $z \in \mathcal{L}(M)$ (recall that we refer to $z$ as a word over $\Sigma_\#^k$).
- If $\alpha = \exists x \alpha'$ then $L \vdash_{\nu,\alpha} \mathcal{M}$ iff there exist $w_1 \in L$ such that $L \vdash_{\nu[x \mapsto w_1],\alpha'} \mathcal{M}$.

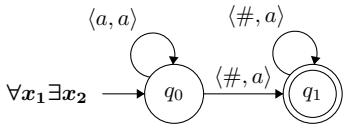


Fig. 1. A syncNFH $\mathcal{A}$ for the Hyperlanguage $\{L \subseteq \{a\}^* \mid L \text{ is infinite}\}$

- If $\alpha = \forall x \alpha'$ then $L \vdash_{\nu,\alpha} \mathcal{M}$ iff for every $w_1 \in L$ we have $L \vdash_{\nu[x \mapsto w_1],\alpha'} \mathcal{M}$. An important point to notice is that in case that $\alpha$ begins with $\forall$, membership holds vacuously with the empty language. As usual with first-order semantics, we restrict the definition to nonempty languages.

Since $\alpha$ quantifies all variables in $X$, the relation $\vdash$ is independent of the specific assignment $\nu$. Therefore, we denote $L \in \mathfrak{L}(\mathcal{M})$ iff $L \vdash_\alpha \mathcal{M}$. The Hyperlanguage of $\mathcal{M}$ is then $\mathfrak{L}(\mathcal{M}) = \{L \subseteq \Sigma^* \mid L \vdash_\alpha \mathcal{M}\}$.

Hypermodels can therefore be thought of as a first-order formalism in prenex-normal-form, where the inner formalism is a computational model rather than a formula.

The different types of hypermodels are distinguished by the type of their underlying models, and the type of semantics that they use. The classes for NFA are *nondeterministic finite-word hyperautomata* (NFH), denoted syncNFH and asyncNFH; the classes for CFG are *context-free hypergrammars*, denoted syncCFHG and asyncCFHG; and the classes for CSG are *context-sensitive hypergrammars*, denoted syncCSHG and asyncCSHG.

For all hypermodels, we assume no transitions or derivation rules with $\langle \#, \ldots, \# \rangle$, that is, with only $\#$'s (when needed and allowed by the model, we can simply use $\epsilon$ instead).

For NFH, guaranteeing a synchronized underlying NFA amounts to a rather simple construction. For CFHGs this is not as simple, and structural conditions over a CFHG, which guarantee synchronization, were studied in [25]. Similar constructions for CSHGs have yet to be defined. This work focuses on the hierarchy and expressiveness of the different formalisms. Hence, we leave such structural definitions for future work, and treat syncCSHGs as per their definition, without relying on such structures.

**Example III.2.** We present some examples for the different types of hypermodels.

- sync**NFH**: Figure 1 presents a syncNFH $\mathcal{A}$ that accepts a language $\mathcal{L}$ over $\{a\}$ iff for every $w \in \mathcal{L}$, there exists a longer word $w \cdot a^n \in \mathcal{L}$ for some $n > 0$. This holds iff $\mathcal{L}$ is infinite. Therefore, $\mathfrak{L}(\mathcal{A})$ is the set of all infinite languages over {a}.
- async**NFH**: Figure 4 presents an asyncNFH for the singletons hyperlanguage $\{\{w\$w\} \mid w \in \{a, b\}^*\}$, which we discuss in Theorem VI.1.
- sync**CFHG**: The following syncCFHG $\mathcal{G}_1$ accepts the hyperlanguage $\{\mathcal{L} \mid \mathcal{L} \subseteq \{a^m b^n | n > m\}\}$. $\mathcal{G}_1 = \langle G_1, \{a, b\}, \{x\}, \forall x, \mathsf{sync} \rangle$ where $G_1 =$

$\langle \{a,b\}_{\#}, \{V_0, V_1\}, V_0, P\rangle$ with derivation rules:

$$V_0 \to \langle a\rangle V_0 \langle b\rangle \mid \langle b\rangle V_1$$
$$V_1 \to \langle b\rangle V_1 \mid \epsilon$$

- asyncCFHG: The following asyncCFHG $\mathcal{G}_2$ accepts the hyperlanguage $\{\mathcal{L} \subseteq \{a^n b^n | n \in \mathbb{N}\} \mid \mathcal{L} \text{ is infinite}\}$, that is, all infinite languages that only contain words of the form $a^n b^n$. $\mathcal{G}_2 = \langle G_2, \{a,b\}, \{x_1, x_2\}, \forall x_1 \exists x_2, \mathsf{async}\rangle$ where $G_2 = \langle \{a,b\}_{\#}, \{V_0, V_1\}, V_0, P\rangle$ with the following derivation rules:

$$V_0 \to \langle a,a\rangle V_0 \langle b,b\rangle \mid \langle \#,a\rangle V_1 \langle \#,b\rangle$$
$$V_1 \to \langle \#,a\rangle V_1 \langle \#,b\rangle \mid \epsilon$$

  That is, for every word $a^n b^n$ (first element) there exists a word of the same form, but that continues reading more letters on the second component.

## IV. Hyperlanguages Made of Singletons

Before diving into the relations between the different types of hypermodels, we note that much of our analysis relies on singletons hyperlanguages. These turn out to be useful, as they allow lifting well-known results for (standard) finite-word models to the setting of hyperlanguages.

Recall that a singletons hyperlanguage is of the type $\{\{w\} \mid w \in L\}$ for some language $L$.

Notice that when we restrict the hyperlanguage of a hypermodel to singletons, the quantification condition no longer "matters". Indeed, a singleton $\{w\}$ satisfies an $\exists$-condition iff it satisfies a $\forall$-condition. We formalize this in the following.

**Lemma IV.1.** *Let $\mathcal{M} = \langle M, \Sigma, X, \alpha, s\rangle$ and let $\mathcal{M}' = \langle M, \Sigma, X, \alpha', s\rangle$ be hypermodels. Then, for every $w \in \Sigma^*$, we have $\{w\} \in \mathfrak{L}(\mathcal{M})$ iff $\{w\} \in \mathfrak{L}(\mathcal{M}')$. That is, both hypermodels agree on singletons languages.*

Next, we show that given a hypermodel $\mathcal{M}$, we can construct a matching hypermodel which restricts $\mathfrak{L}(\mathcal{M})$ to its singletons.

**Theorem IV.2.** *Let $\mathcal{M} \in \{$async*NFH*, async*CFHG*, async*CSHG*$\}$. There exists $\mathcal{M}_{\mathsf{sngl}}$ such that $\mathfrak{L}(\mathcal{M}_{\mathsf{sngl}}) = \{\{w\} \mid \{w\} \in \mathfrak{L}(\mathcal{M})\}$.*

*Proof.* The idea for all hypermodels is to use two extra variables, universally quantified, that must be equally assigned, and allow every relevant word assignment. By the semantics, this can only be satisfied by a singleton.

We now describe the construction formally for the different types of hypermodels, beginning with asyncNFH.

Let $\mathcal{A} = \langle A, \Sigma, \{x_1, ..., x_k\}, \alpha, \mathsf{async}\rangle$ be an asyncNFH, and let $\delta$ be the transition relation of $A$. We construct the following asyncNFH:

$$\mathcal{A}_{\mathsf{sngl}} = \langle A', \Sigma, \{x_1, ..., x_k, x_{k+1}, x_{k+2}\}, \alpha \forall x_{k+1} \forall x_{k+2}, \mathsf{async}\rangle$$

For every transition $\langle q, \langle \sigma_1, \ldots, \sigma_k\rangle, p\rangle \in \delta$, we add the transition $\langle q, \langle \sigma_1, \ldots, \sigma_k, \sigma, \sigma\rangle, p\rangle$ to $\delta'$, for every $\sigma \in \Sigma_{\#}$.

According to the semantics of asyncNFH, for a language $L$ to be accepted by $\mathcal{A}_{\mathsf{sngl}}$, every two words in $L$ must be assigned to $x_{k+1}, x_{k+2}$ in some accepting run. By the construction, these words must always be identical. Therefore, there cannot be two different words in $L$, and so it is a singleton $\{w\}$.

Every singleton $\{w\} \in \mathfrak{L}(\mathcal{A})$ is accepted via some run of $A$ that assigns $w$ to all variables. Using the same run in $A'$ while assigning $w$ to $x_{k+1}, x_{k+2}$ as well, implies that $\{w\} \in \mathfrak{L}(\mathcal{A}_{\mathsf{sngl}})$. In the other direction, if $\{w\} \in \mathfrak{L}(\mathcal{A}_{\mathsf{sngl}})$, then restricting an accepting run of $A'$ that assigns $w$ to all variables to the first $k$ variables induces an accepting run in $A$ leading to $\{w\} \in \mathfrak{L}(\mathcal{A})$.

We therefore have that $\mathfrak{L}(\mathcal{A}_{\mathsf{sngl}}) = \{\{w\} \mid \{w\} \in \mathfrak{L}(\mathcal{A})\}$. We note that this construction is in fact the intersection of $\mathcal{A}$ with a single-state syncNFH that accepts all singletons over $\Sigma$. We can show that asyncNFH are closed under intersection, but since this is beyond our scope, we provide a direct construction.

We now turn to asyncCFHG. The idea is similar to the construction for asyncNFH, with some attention to the fact that intersection cannot be simulated on CFGs. However, in this case this is a minor hurdle; instead of allowing every word assignment to the extra variables, we allow an assignment that is identical to the assignment to one of the original variables. This ensures that the word is already in a language $L$ accepted by the hypergrammar, and so the extra variables indeed serve as a restriction to all relevant singletons.

Formally, let $\mathcal{G} = \langle G, \Sigma, \{x_1, \ldots, x_k\}, \alpha, \mathsf{async}\rangle$ be a hypergrammar, where $G = \langle \Sigma_{\#}^k, V, S, P\rangle$ is its underlying CFG.

We construct an asyncCFHG $\mathcal{G}_{\mathsf{sngl}}$ as follows.

$$\mathcal{G}_{\mathsf{sngl}} = \langle G', \Sigma, \{x_1, \ldots, x_k, x_{k+1}, x_{k+2}\}, \alpha \forall x_{k+1} \forall x_{k+2}, \mathsf{async}\rangle$$

where $G' = \langle \Sigma_{\#}^{k+2}, V, S, P'\rangle$, and where $P'$ is constructed as follows. For every rule $B \to \beta \in P$, we add the rule $B \to \beta'$ to $P'$, where $\beta'$ is obtained from $\beta$ by replacing every letter-tuple $\langle \sigma_1, \ldots, \sigma_k\rangle$ that occurs in $\beta$ by $\langle \sigma_1, \ldots \sigma_k, \sigma_k, \sigma_k\rangle$ . Hence, $x_{k+1}$ and $x_{k+2}$ behave exactly like $x_k$. For example, suppose $k = 2$ and variables $B, C$. Then for a rule $B \to \langle \#,a\rangle\langle b,a\rangle C \langle \#,b\rangle$, we add the rule $B \to \langle \#,a,a,a\rangle\langle b,a,a,a\rangle C \langle \#,b,b,b\rangle$.

As with the construction for asyncNFH, we have that $\mathcal{G}_{\mathsf{sngl}}$ can only derive singletons. Further, a singleton $\{w\}$ is derived by $G$ iff $w$ is assigned to all variables in a matching word-tuple derived by $G$, iff $w$ is assigned to all variables in a matching word-tuple derived by $G'$. We therefore have that $\mathfrak{L}(\mathcal{G}_{\mathsf{sngl}}) = \{\{w\} \mid \{w\} \in \mathfrak{L}(\mathcal{G})\}$.

Using a similar construction and arguments, we obtain a similar result for asyncCSHG as well. Notice that the construction for asyncCSHG requires replacing the letters on the left-hand side of each rule on top of on the right-hand side, which is consistent with the derivation rules and desired result. □

Theorem IV.2 addresses asynchronized hypermodels. In the special case of synchronized hypermodels, restricting the

hyperlanguage to singletons is even simpler: One must only restrict the letter-tuples to *homogeneous* letters.

**Definition IV.1.** A letter-tuple in $\Sigma_{\#}^{k}$ is called *homogeneous* if it is of the type $\langle \sigma, ..., \sigma \rangle$, for $\sigma \in \Sigma_{\#}$.

Indeed, synchronized models only consider synchronized word-tuples, and a singleton language produces only word-tuples in which the same word is assigned to all variables.

We now formalize this intuition.

**Theorem IV.3.** *Let* $\mathcal{M} \in \{\mathsf{sync}NFH, \mathsf{sync}CFHG, \mathsf{sync}CSHG\}$. *We can construct* $\mathcal{M}_{\mathsf{sngl}}$ *such that* $\mathfrak{L}(\mathcal{M}_{\mathsf{sngl}}) = \{\{w\} \mid \{w\} \in \mathfrak{L}(\mathcal{M})\}$, *by restricting the underlying model of* $M$ *to homogeneous letter-tuples.*

*Proof.* We begin with syncNFH. Let $\mathcal{A} = \langle A, \Sigma, \{x_1, \ldots, x_k\}, \alpha, \mathsf{sync} \rangle$ be a syncNFH, and let $\mathcal{A}_{\mathsf{sngl}} = \langle A', \Sigma, \{x_1, \ldots, x_k\}, \alpha, \mathsf{sync} \rangle$, where $A'$ is obtained from $A$ by removing all transitions that are not homogeneous. We claim that $\mathfrak{L}(\mathcal{A}_{\mathsf{sngl}}) = \{\{w\} \mid \{w\} \in \mathfrak{L}(\mathcal{A})\}$.

First, we notice that $A'$ only accepts synchronized word-tuples of the form $\langle w\#^m, \ldots, w\#^m \rangle$ for $m \in \mathbb{N}$. According to the semantics, $\mathcal{A}_{\mathsf{sngl}}$ can then only accept singleton languages. Next, since all runs of $A$ on homogeneous word-tuples remain in $A'$, it is easy to see that a singleton $\{w\}$ is in $\mathfrak{L}(\mathcal{A})$ iff it is in $\mathfrak{L}(\mathcal{A}_{\mathsf{sngl}})$.

Now, let $\mathcal{G} = \langle G, \Sigma, \{x_1, \ldots, x_k\}, \alpha, \mathsf{sync} \rangle$ be a syncCFHG, and let $P$ be the set of derivation rules of $G$. We construct a syncCFHG $\mathcal{G}_{\mathsf{sngl}} = \langle G', \Sigma, \{x_1, \ldots, x_k\}, \alpha, \mathsf{sync} \rangle$ by restricting the rules in $P$ to those that use only homogeneous letters. As with syncNFH, the languages in $\mathfrak{L}(\mathcal{G}_{\mathsf{sngl}})$ are only singletons. Also, as with syncNFH, a singleton $\{w\}$ is derived by $\mathcal{G}$ iff it is derived by $\mathcal{G}_{\mathsf{sngl}}$.

Similar construction and results also hold for syncCSHG. □

We now show that given a language $\mathcal{L}$ for which there exists some finite-word model $M$, that is $\mathcal{L}(M) = \mathcal{L}$, there exists a synced hypermodel whose hyperlanguage is the set of all singletons in $\mathcal{L}$.

**Lemma IV.4.** *Let* $\mathcal{L}$ *over* $\Sigma$ *be a* {*regular, context-free, context-sensitive*} *language. Then there exists a synced hypermodel* $\mathcal{M} \in \{\mathsf{sync}NFH, \mathsf{sync}CFHG, \mathsf{sync}CSHG\}$, *respectively, such that* $\mathfrak{L}(\mathcal{M}) = \{\{w\} \mid w \in \mathcal{L}\}$.

*Proof.* Let $M$ be a model over $\Sigma$ such that $\mathcal{L}(M) = \mathcal{L}$. We construct a hypermodel $\mathcal{M} = \langle M', \Sigma, \{x_1, x_2\}, \forall x_1 \forall x_2, \mathsf{sync} \rangle$ where $M'$ is obtained from $M$ by replacing every $\sigma \in \Sigma$ with $\langle \sigma, \sigma \rangle$.

As argued in the proof of Theorem IV.3, $\mathfrak{L}(\mathcal{M})$ includes only singletons, since all letter-tuples are homogeneous. Further, a (either run or derivation) of $w$ in $M$ exactly matches that of $\langle w, w \rangle$ in $M'$. It is then easy to see that $w \in \mathcal{L}$ iff $\{w\} \in \mathfrak{L}(\mathcal{M})$. □

As a result of Theorem IV.3 and Lemma IV.4, we can conclude the following.

**Lemma IV.5.** *Let* $\mathcal{L}$ *be a language.*

1) $\mathcal{L}$ *is regular iff there exists a* sync*NFH* $\mathcal{A}$ *such that* $\mathcal{L} = \{w \mid \{w\} \in \mathfrak{L}(\mathcal{A})\}$.
2) $\mathcal{L}$ *is context-free iff there exists a* sync*CFHG* $\mathcal{G}$ *such that* $\mathcal{L} = \{w \mid \{w\} \in \mathfrak{L}(\mathcal{G})\}$.
3) $\mathcal{L}$ *is context-sensitive iff there exists a* sync*CSHG* $\mathcal{G}$ *such that* $\mathcal{L} = \{w \mid \{w\} \in \mathfrak{L}(\mathcal{G})\}$.

*Proof.* The direction: if $\mathcal{L}$ is {regular, context-free, context-sensitive} then there is a corresponding hypermodel $\mathcal{M} \in$ {syncNFH, syncCFHG, syncCSHG} such that $\mathcal{L} = \{w \mid \{w\} \in \mathfrak{L}(\mathcal{M})\}$ is given in Lemma IV.4.

For the second direction, we first provide some intuition: Since the models are synchronized, the only way to recognize a singleton is by using only homogeneous letters. Non-homogeneous letters may also appear in the underlying model, but they play no role in the hyperlanguage. Therefore, we can restrict the underlying model to homogeneous letters only without changing its hyperlanguage. This is equivalent to restricting the underlying model to the standard alphabet (i.e.,every $\langle \sigma, \sigma, ..., \sigma \rangle$ is in fact a $\sigma$). Doing so, we get an instance of the underlying model whose language is exactly the words in the singletons language: for syncNFH, we get an NFA – hence $\mathcal{L}$ is regular, for syncCFHG we get a CFG, etc.

We now provide the formal proof: Let $\mathcal{L} \subseteq \Sigma^*$, and assume there is a synced hypermodel, e.g., a syncNFH $\mathcal{A}$, such that $\mathfrak{L}(\mathcal{A}) = \{\{w\} \mid w \in \mathcal{L}\}$. We show that $\mathcal{L}$ is regular. The proof for syncCFHG and syncCSHG is similar. Note that, due to Lemma IV.1, the quantification condition in $\mathcal{A}$ does not matter, since its hyperlanguage is that of singletons. Denote $\mathcal{A} = (A, \Sigma, X, \alpha, \mathsf{sync})$, and $A$, the underlying NFA, as $A = (\Sigma_{\#}^{|X|}, Q, q_0, \delta, F)$. Every $\{w\} \in \mathfrak{L}(\mathcal{A})$ is recognized by assigning $w$ to all variables, and vice versa: Every run of $A$ on $(w, ...., w)$ induces $\{w\} \in \mathfrak{L}(\mathcal{A})$. Since $\mathcal{A}$ is synced, $\{w\}$ may only be recognized by runs in which all transitions in $\delta$ are homogeneous. Therefore, we can construct an NFA $A'$ over $\Sigma$ as follows: $A' = (\Sigma, Q, q_0, \delta', F)$ where $\delta' = \{(q, \sigma, q') | (q, \langle \sigma, \ldots, \sigma \rangle, q') \in \delta\} \cup \{(q, \epsilon, q') | (q, \langle \#, ...\# \rangle, q') \in \delta\}$ (we use epsilon transitions here but these are not necessary and can be eliminated using the standard techniques). The language $\mathcal{L}(A')$ is regular, since it is accepted by an NFA. All we need to show is that $\mathcal{L} = \mathcal{L}(A')$. This follows directly from the construction of $A'$: $\{w\}$ is accepted by $\mathcal{A}$ iff there is a run of $A$ on the vector $\langle w, \ldots, w \rangle$ (due to synchronicity and homogeneity), iff there is a run of $A'$ on $w$. Therefore, the second direction of Lemma IV.5 follows. □

We point out that Theorem V.1 shows that syncNFH and syncCFHG are in fact capable of exhibiting non-context-free behavior, as we show in Section V.

We further remark that Lemma IV.5 does not hold for the

asynchronous models, as we demonstrate in Section VI.[2]

## V. Synchronous Hypermodels

In this section we discuss the expressive power of synchronous hypermodels. One might (rightfully) expect the expressive power of a hypermodel to be restricted by the expressive power of its underlying model. As it turns out, this is less restrictive than anticipated.

It is easy to see that using only $\exists$ quantifiers, which merely require the existence of a (finite) set of words in a language that is recognized by a syncNFH $\mathcal{A}$, leads to the existence of non-regular languages in $\mathfrak{L}(\mathcal{A})$. However, we can construct a syncNFH $\mathcal{A}$ for which $\mathfrak{L}(\mathcal{A})$ consists *only* of non-regular (and even non-context-free) languages. Thus, despite their underlying model, syncNFH are capable of "non-regular" behaviors, in this sense. The same also holds for syncCFHG, which are capable of displaying non-context-free behaviors.[3]

**Theorem V.1.** *There exists a* syncNFH $\mathcal{A}$ *such that* $\mathfrak{L}(\mathcal{A}) = \{L\}$ *for a non-context-free language* $L$*.*

*Proof.* We construct a syncNFH $\mathcal{A}$ over $\{a, b, c\}$ whose hyperlanguage is $\mathfrak{L}(\mathcal{A}) = \{L_{abc}\}$ where $L_{abc} = \{a^n b^n c^n \mid n \in \mathbb{N}\}$. It is well-known that $L_{abc}$ is not context-free.

$\mathcal{A}$ is formed by intersecting two syncNFH: $\mathcal{A}_1$, expressing languages that subsume $L_{abc}$; and $\mathcal{A}_2$, expressing languages that are subsumed by $L_{abc}$. Then, the intersection $\mathfrak{L}(\mathcal{A}_1) \cap \mathfrak{L}(\mathcal{A}_2)$ is $\{L_{abc}\}$, and since syncNFH are closed under intersection [10], we are done.

We now elaborate on the construction of $\mathcal{A}_1$ and $\mathcal{A}_2$.

Intuitively, we define $\mathcal{A}_1$ using the condition $\exists\epsilon \forall a^i b^j c^k \exists a^{i+1} b^{j+1} c^{k+1}$. This means that $\mathcal{A}_1$ requires the existence of the empty word, and for every word of the form $a^n b^n c^n$, it requires the existence of the "next" word $a^{n+1} b^{n+1} c^{n+1}$. This way, we can inductively generate all words $a^{n+1} b^{n+1} c^{n+1}$. We define $\mathcal{A}_1$ as follows, where $A_1$ is according to Figure 2.

$$\mathcal{A}_1 := \langle A_1, \{a, b, c\}, \{x_1, x_2, x_3\}, \exists x_1 \forall x_2 \exists x_3, \mathsf{sync} \rangle$$

The syncNFH $\mathcal{A}_2$ expresses the condition $\forall a^i b^j c^k \exists a^{i-1} b^{j-1} c^{k-1}$, and is defined as follows, where $A_2$ is given in Figure 3.

$$\mathcal{A}_2 := \langle A_2, \{a, b, c\}, \{x_1, x_2\}, \forall x_1 \exists x_2, \mathsf{sync} \rangle$$

Note that $\mathcal{A}_1$ does not capture *all* languages that subsume $L_{abc}$, but only those that are closed under the addition of a single $a$, $b$, and $c$ to words of the form $a^i b^j c^k$. For example, such a language may contain $abbc$, leading to the forced existence of $aabbbcc$.

Similarly, $\mathcal{A}_2$ captures languages that are subsumed by $L_{abc}$ and are closed under the removal of a single $a$,$b$, and $c$ from words of the correct form. Still, we have $\mathfrak{L}(\mathcal{A}_1) \cap \mathfrak{L}(\mathcal{A}_2) = \{L_{abc}\}$ as needed. Therefore, $\mathfrak{L}(\mathcal{A}) = \{L_{abc}\}$, as required. □

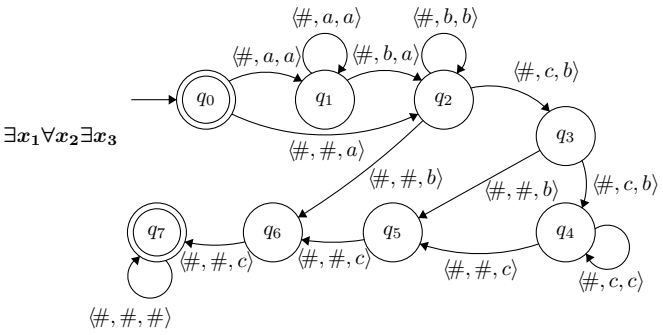


Fig. 2. The syncNFH $\mathcal{A}_1$ for the proof of Theorem V.1: For every word $a^i b^j c^k$ (and, in particular, for words of the form $a^n b^n c^n$), there exists $a^{i+1} b^{j+1} c^{k+1}$, by shifting the word assigned to $x_3$ by three positions, one for each added letter. The transitions $q_0 \to q_2, q_2 \to q_6, q_3 \to q_5$ take care of the special cases $x_2 = \epsilon$ or $x_2 = abc$.

**Remark V.1.** Note that usually, when one talks about hyperproperties that are recognized by synchronous ($\omega$) automata, they are called *($\omega$)-regular hyperproperties*. While this captures the fact that they correspond to hyperautomata, or hyperlogics like HyperQPTL [34] (the $\omega$-regular extension of HyperLTL, similar to the $\omega$-regular extension of QPTL [35] over LTL [32]), this can be misleading, as the properties they express are far beyond regular properties.

### A. The Hierarchy of Synchronous Hypermodels

We use the results in Section IV, and rely our arguments on the restriction of a given hypermodel to its set of singletons to analyze the hierarchy of synchronous hypermodels. We next show that these classes follow the standard Chomsky hierarchy, dictated by their underlying models.

**Theorem V.2.** *The classes of synchronized hypermodels follow the standard Chomsky hierarchy. That is,*

$$\text{sync}\mathit{NFH} \subsetneq \text{sync}\mathit{CFHG} \subsetneq \text{sync}\mathit{CSHG}$$

*Proof.* First, due to the (standard) hierarchy of their underlying models (e.g., every NFA has an equivalent CFG), we have

$$\mathsf{syncNFH} \subseteq \mathsf{syncCFHG} \subseteq \mathsf{syncCSHG}$$

Also due to the standard hierarchy, there exist $\mathcal{L}_1$ which is context-free and not regular, and $\mathcal{L}_2$ which is context-sensitive and not context-free. According to Lemma IV.5, we can construct a syncCFHG for the hyperlanguage $\mathfrak{L}_1 = \{\{w\} \mid w \in \mathcal{L}_1\}$. According to Lemma IV.5, there exists no syncNFH for $\mathfrak{L}_1$. For similar reasons, we can construct a syncCSHG for $\mathfrak{L}_2 = \{\{w\} \mid w \in \mathcal{L}_2\}$, for which there exists no syncCFHG. Therefore, we have strict containment. □

[2]One direction does hold: for every {regular, context-free, context-sensitive} language $\mathcal{L}$ there exists {asyncNFH, asyncCFHG, asyncCSHG} that accepts exactly the singletons that are words of $\mathcal{L}$, since any synchronous model is in particular also asynchronous. In Section VI, we show that the opposite direction does not hold.

[3]For infinite words, Finkbeiner and Zimmermann [23] showed that there exists an HyperLTL sentence with no $\omega$-regular models. While their final result relies on words of the form $w \cdot \emptyset^\omega$, that is, words that represent finite words, the proof uses general HyperLTL formulas, and cannot be directly applied here. Therefore, we provide a direct proof.

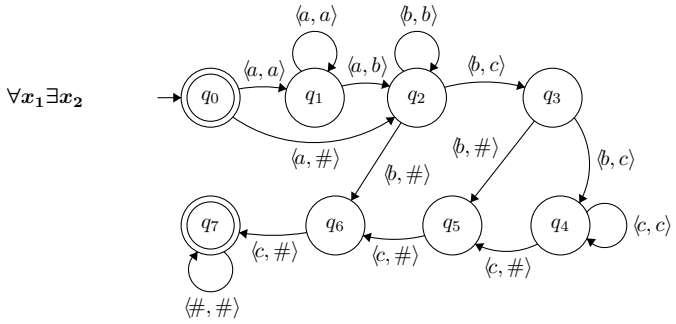


Fig. 3. The syncNFH $\mathcal{A}_2$ for the proof of Theorem V.1: For every word $a^i b^j c^k$ there exists $a^{i-1} b^{j-1} c^{k-1}$. The transitions $q_0 \to q_2, q_2 \to q_6$, $q_3 \to q_5$ take care of the special cases $x_1 = abc, x_2 = \epsilon$, or $x_1 = aabbcc, x_2 = abc$.

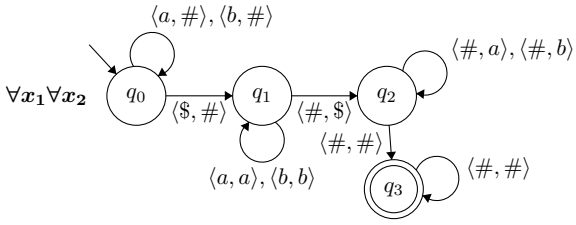


Fig. 4. An asyncNFH $\mathcal{A}$ for $\{\{w\$w\} \mid w \in \{a, b\}^*\}$: Intuitively, $x_1$ reads the word $w$, while $x_2$ stays at the beginning of the word (state $q_0$). Then, when $x_1$ reaches \$, both variables advance at the same time, comparing the two occurrences of $w$, letter by letter (state $q_1$). State $q_2$ is symmetrical to $q_0$, with $x_2$ reading the second $w$ after $x_1$ is done.

**Remark V.2.** The use of Lemma IV.5 (and Lemma IV.4) implies that even when restricting the "stronger" synchronized hypermodel in the Chomsky hierarchy to only universally quantified variables, it still can recognize languages that are not recognized by hypermodels that are lower in the hierarchy, regardless of their number of variables or quantification condition.

## VI. Asynchronous Hypermodels

In Section V we have settled the Chomsky hierarchy for synchronous hypermodels, and we now turn to do the same for the asynchronous ones. First, we show that asynchronous hypermodels are strictly more expressive than synchronous ones (Section VI-A). In Section VI-B we establish the Chomsky hierarchy for asynchronous hypermodels. Last, in Section VI-C we further study the relations between the levels in both semantics, and show that adding asynchronous behavior renders asyncNFH and syncCFHG incomparable.

### A. The Power of Asynchronicity

We first show that asynchronous models are more expressive than synchronous ones, as we prove that asyncCFHG and asyncNFH are strictly more expressive than syncCFHG and syncNFH, respectively.

**Theorem VI.1.** *1)* sync*NFH* $\subsetneq$ async*NFH*.
*2)* sync*CFHG* $\subsetneq$ async*CFHG*.

*Proof.* We clearly have syncNFH $\subseteq$ asyncNFH and syncCFHG $\subseteq$ asyncCFHG.

For strict containment, we use the hyperlanguage $\mathfrak{L}_{ww} = \{\{w\$w\} \mid w \in \{a, b\}^*\}$, and show that it is asyncNFH-recognizable (and therefore asyncCFHG-recognizable) but not syncCFHG-recognizable (and therefore not syncNFH-recognizable).

We present the following asyncNFH $\mathcal{A}$ for $\mathfrak{L}_{ww}$, where $A$ is given in Figure 4.

$$\mathcal{A} = \langle A, \{a, b, \$\}, \{x_1, x_2\}, \forall x_1 \forall x_2, \mathsf{async}\rangle$$

Essentially, $\mathcal{A}$ advances to the middle of the input word with one variable, while the second reads only # letters, keeping it at the start of the word. Then, once the first variable reads \$, both copies advance as long as they both read the same input letter $a$ or $b$. This makes sure that the same word is read before and after the \$ letter. Once the second variable reaches \$, the run accepts.

Now, the language $\{w\$w \mid w \in \{a, b\}^*\}$ is not context-free, and so according to Lemma IV.5, $\mathfrak{L}_{ww}$ is not syncCFHG-recognizable. □

Not only do the asynchronous hypermodels subsume their synchronous counterparts, we now show that it is not possible to procedurally produce an equivalent syncNFH given an asyncNFH, or conclude that no such syncNFH exists. The same holds for asyncCFHG and syncCFHG.

**Theorem VI.2.** *There is no algorithmic procedure that computes the following: given an* async*NFH (resp.* async*CFHG), return an equivalent* sync*NFH (resp.* sync*CFHG), or return that no such hypermodel exists.*

*Proof.* We use Post's Correspondence Problem (PCP), which is known to be undecidable. A PCP instance is a set of pairs of the form $[u_1, v_1], \ldots, [u_n, v_n]$ where $u_i, v_i \in \{a, b\}^*$ for every $1 \leq i \leq n$. The problem is to decide whether there exists a sequence of indices $i_1 \cdots i_m \in [1, n]$, such that $u_{i_1} u_{i_2} \cdots u_{i_m} = v_{i_1} v_{i_2} \cdots v_{i_m}$. For example, consider the instance $T_1 = \{[a, baa]_1, [ab, aa]_2, [bba, bb]_3\}$. Then, a solution to the PCP is the sequence $3, 2, 3, 1$, since $u_3 u_2 u_3 u_1 = bba \cdot ab \cdot bba \cdot a$ and $v_3 v_2 v_3 v_1 = bb \cdot aa \cdot bb \cdot baa$. The word that corresponds to this solution is $bbaabbbaa$. We abuse the notation and refer to words corresponding to PCP solutions, as solutions as well.

Let $T = \{[u_1, v_1], \ldots, [u_n, v_n]\}$ be a PCP instance.

For every pair $[u_i, v_i] \in T$ we define the words $A_i, B_i \in \Sigma_{\#}^*$ obtained from $u_i, v_i$ by padding the shorter of $u_i, v_i$ with #-symbols so that $A_i, B_i$ are of equal length. For example, for $T_1$ above, and the pair $[a, baa]_1$, we create the pair $[a\#\#, baa]_1$.

Next, we define an asyncCFHG $\mathcal{G}_T = \langle G_T, \{a, b\}, \{x_1, x_2\}, \forall x_1 \forall x_2, \mathsf{async}\rangle$, where $G_T = \langle \{a, b\}_{\#}, \{V_0\}, V_0, P\rangle$, and where $P$ is as follows.

$$P := V_0 \to \langle A_1, B_1\rangle V_0 \mid \cdots \mid \langle A_n, B_n\rangle V_0 \mid \langle A_1, B_1\rangle \mid \cdots \mid \langle A_n, B_n\rangle$$

We claim that a word $w$ is a solution to the PCP instance $T$ if and only if $\{w\} \in \mathfrak{L}(\mathcal{G}_T)$. Indeed, the sequence

$$\begin{aligned} V_0 \Rightarrow \quad & \langle A_{i_1}, B_{i_1}\rangle V_0 \Rightarrow \langle A_{i_1}, B_{i_1}\rangle\langle A_{i_2}, B_{i_2}\rangle V_0 \\ \Rightarrow \quad & \cdots \Rightarrow \langle A_{i_1}, B_{i_1}\rangle \cdots \langle A_{i_m}, B_{i_m}\rangle \end{aligned}$$

corresponds exactly to the assignment $x_1 \mapsto u_{i_1} a_{i_2} \cdots u_{i_m}$ and $x_2 \mapsto v_{i_1} b_{i_2} \cdots v_{i_m}$. Thus, $\{w\}$ can be derived by $G_T$ iff $w = u_{i_1} v_{i_2} \cdots v_{i_m} = v_{i_1} v_{i_2} \cdots v_{i_m}$, i.e., $w$ is a solution to $T$, and the claim follows.

Let $\mathcal{G}_{\mathsf{sngl}}$ be the asyncCFHG obtained from $\mathcal{G}_T$ by restricting it to its set of singletons, as per Theorem IV.2, and by replacing all quantifiers by $\forall$, which does not alter the result, as per Lemma IV.1.

As discussed above, $\mathfrak{L}(\mathcal{G}_{\mathsf{sngl}})$ is the set of all singletons languages that represent a solution to $T$.

Now, assume by way of contradiction the existence of a computational procedure $\mathcal{P}$ as described.

Assume first that $\mathcal{P}(\mathcal{G}_{\mathsf{sngl}}) = \mathit{false}$. Since there does exist a syncCFHG for the empty hyperlanguage (simply with an empty underlying CFG), we can deduce that $\mathfrak{L}(\mathcal{G}_{\mathsf{sngl}})$ is nonempty, hence $T$ has a solution.

Now, assume that $\mathcal{P}(\mathcal{G}_{\mathsf{sngl}}) = \mathcal{G}$, where $\mathcal{G}$ is an equivalent syncCFHG. Since $\mathcal{G}$ only accepts singletons, we can replace all quantifiers in its quantification condition with universal quantifiers without affecting its hyperlanguage, due to Lemma IV.1.

Since $\mathcal{G}$ is now universally quantified, it can be tested for emptiness [25]. We then have that $\mathcal{G}$ is nonempty iff $T$ has a solution. We conclude that $\mathcal{P}$ offers a decision procedure for PCP, a contradiction.

Notice that since $\mathcal{G}_T$ is right-linear, an asyncNFH for $\mathfrak{L}(\mathcal{G}_T)$ can be constructed accordingly. Since universality-quantified syncNFH can be tested for emptiness [10], we can construct a similar proof for syncNFH. □

### B. Chomsky Hierarchy for Asynchronous Hypermodels

We continue our analysis at the lowest level, namely NFH.

In *multi-head one-way finite automata*, a fixed number of heads independently read a single input word left-to-right, under a finite control. At the beginning of the run, all heads are positioned on the first letter. At every step, depending on the current state and the vector of letters read by the heads, every head either stays put, or moves one place to the right.

When the same word $w$ is assigned to all variables, the underlying NFA of an asyncNFH essentially behaves as a multi-head automaton, where the variables act as the heads. Initially, all variables are positioned at the beginning of their copy of $w$. When the underlying automaton reads a letter-tuple $\tau$, a $\Sigma$-letter in $\tau[i]$ is equivalent to the variable $x_i$ advancing one position on $w$, and $\tau[i] = \#$ is equivalent to $x_i$ staying in place. Therefore, the behavior of such multi-head automata matches that of asyncNFH for singletons, that is, all variables are equally assigned.

In [37], Yao and Rivest show that multi-head one-way finite automata are more expressive as their number of heads grows.

We now use a variation of their proof, tailored to a language that separates regular languages from context-free languages, to show that one-wayness is restrictive enough to separate between asyncNFH and asyncCFHG.

**Theorem VI.3.** async*NFH* $\subsetneq$ async*CFHG*.

*Proof.* We obviously have asyncNFH $\subseteq$ asyncCFHG. We show that the hyperlanguage

$$\mathfrak{L}_{pal} = \{\{xx^R\} \mid x \in \{a,b\}^*\}$$

is asyncCFHG-recognizable, but not asyncNFH-recognizable.

Since $\{xx^R \mid x \in \{a,b\}^*\}$ is context-free, we can construct an asyncCFHG (in fact, a syncCFHG) that recognizes $\mathfrak{L}_{pal}$, according to Lemma IV.4

For strict containment, we rely on observations in the proof in [37]. We mostly outline (albeit in detail) these observations, and refer the reader to [37] for the full details.

Consider the following set of languages for all $n \in \mathbb{N}$.

$$\mathcal{L}_n = \{w_1\$ \cdots \$w_n\$w_n^R\$ \cdots \$w_1^R\}$$

where $w_i \in \{a,b\}^*$ for every $1 \leq i \leq n$. That is, a word in $\mathcal{L}_n$ consists of a sequence of $n$ words (separated by \$-delimeters), followed by the reversed word of this sequence. Consider an asyncNFH $\mathcal{A}$ with $k$ variables, where $n > \binom{k}{2}$. We show that $\mathcal{A}$ cannot recognize $\{\{w\} | w \in \mathcal{L}_n\}$.

Assume by way of contradiction otherwise, and consider a run of the underlying NFA $A$ of $\mathcal{A}$ on a word-tuple in $\mathsf{pad}(\langle w, ..., w\rangle)$ for some $w \in \mathcal{L}_n$.

A *configuration* of the run is a tuple $c = \langle s, p_1, \ldots, p_k\rangle$ where $s$ is the current state, and $p_i$ is the position of $x_i$ within $w$. The *type* of $c$ is a tuple $\langle t_1, \ldots t_k\rangle$, where $1 \leq t_i \leq 2n$ marks the index $j$ of the subword $w_j$ (or the index $2n+1-j$ of $w_j^R$) on which $x_i$ is currently placed (including the following delimeter \$).

Given a word $w \in \mathcal{L}_n$, an accepting run $r$ of $A$ on a word-tuple $\langle w, \ldots, w\rangle$ induces a sequence $C$ of configurations. We obtain from $C$ a subsequence $D = d_1, \ldots d_l$ of configurations by taking $d_1 = C_1$, and $d_{i+1}$ to be the first following configuration in $C$ whose type is different from $d_{i+1}$. We call $D$ a *pattern* of $w$. Thus, $D$ records the sequence of configurations within $C$ whenever (at least) one of the variables moves to the next subword. Notice that $l$ is bounded by $k(2n-1)+1$, and does not depend on the length of $w$. Therefore, the number of different patterns is finite and does not depend on the lengths of the words in $\mathfrak{L}_n$, but rather on $n, k$, and the number of states in $A$.

Now, take a sublanguage $\mathcal{L}_n^m$ of $\mathcal{L}_n$ in which the subwords $w_i$ (and $w_i^R$) are all of the same (long) length $m$, and associate each $w \in \mathcal{L}_n^m$ with some accepting run on it. By the pigeonhole principle, there exists a large enough set of words $\mathcal{L} \subseteq \mathcal{L}_n^m$ that all share the same pattern $D$.

Consider a pair of variables $x$ and $x'$, and two indices $i < j$. Since $w$ is read from left to right, there cannot be both a step in $r$ in which $x$ is positioned on $w_i$ and $x'$ is positioned on $w_i^R$, and another step in $r$ in which $x$ is positioned on $w_j$ and $x'$ is positioned on $w_j^R$. This, since $x'$ cannot move backwards from $x_i^R$ to $x_j^R$.

Therefore, since $n > \binom{k}{2}$, for every $w \in \mathcal{L}_n^m$ there exists some index $i$ for which, throughout the entire accepting run on $w$, either there is no variable positioned on $w_i$, or there is no variable positioned on $w_i^R$. Notice that $i$ is dictated by $D$, and therefore, all words in $\mathcal{L}$ share such an index $i$.

Assuming again that $m$ is large enough, there exist two different words $w$ and $u$ in $\mathcal{L}$ for which $w_j = u_j$ (and $w_j^R = u_j^R$) for every $j \neq i$, while $w_i \neq u_i$ (and $w_i^R \neq u_i^R$).

Since $w$ and $u$ share the pattern $D$, we can use $D$ to carefully construct a sequence of configurations that matches an accepting run on a word $z$ that is obtained from $w$ by replacing $w_i^R$ with $u_i^R$. This is made possible by the fact that the transitions never depend on letters of both $w_i$ and $w_i^R$ at the same time. Since $z \notin \mathcal{L}_n$, we reach a contradiction to the existence of $\mathcal{A}$.

Thus far, we have shown that for every $k$, there exists a singletons hyperlanguage that cannot be recognized by an asyncNFH with $k$ variables.

Now, notice that when we drop the delimeter \$ from the definition of $\mathfrak{L}_n$, we in fact obtain $\mathfrak{L}_{pal}$. Then, for every $k$, there exist $n$ and $m$ such that we can segment words of the type $ww^R$ of length $2n \cdot m$ into $2n$ subwords of length $m$, and follow the proof idea above to show that this singletons hyperlanguage cannot be recognized by an asyncNFH with $k$ variables. Therefore, for every $k$, there exists no asyncNFH with $k$ variables that recognizes $\mathfrak{L}_{pal}$. □

Theorem VI.3 shows a hyperlanguage $\mathfrak{L}_n$ that cannot be recognized by an asyncNFH that does not have enough variables. Obviously, it is possible to construct an asyncNFH with enough variables to cover all $n$ subwords in a word in $\mathfrak{L}_n$ that recognizes $\mathfrak{L}_n$. This implies an inner hierarchy of asyncNFH, formed by the number of variables.

**Corollary VI.4.** *For every $k$, there exist $k' > k$ and a hyperlanguage that is recognizable by an* async*NFH with $k'$ variables, but cannot be recognized by an* async*NFH with $k$ variables.*

We continue climbing up the hierarchy and show that we can separate between asyncCFHG and asyncCSHG.

**Theorem VI.5.** async*CFHG* $\subsetneq$ async*CSHG*

*Proof.* Clearly, we have asyncCFHG $\subseteq$ asyncCSHG. We prove that the singletons hyperlanguage $\mathfrak{L}=\{\{a^{i^2}\} \mid i \in \mathbb{N}\}$ is asyncCSHG-recgonizable, but not asyncCFHG-recognizable.

For the former, the language $\mathcal{L} = \{a^{i^2} \mid n \in \mathbb{N}\}$ is known to be context-sensitive. We can therefore construct an asyncCSHG (in fact, a syncCSHG) for $\mathfrak{L}$ according to Lemma IV.4.

For the latter, our proof relies on Parikh's Theorem, which we first briefly explain and state.

Given a word $w$ over $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$, its *Parikh image* is a vector $\psi(w) = \langle m_1, \ldots m_n \rangle$, which counts the number of occurrences of $\sigma_i$ in $w$, for every $1 \leq i \leq n$. The Parikh image of a language $L$ is then $\psi(L) = \{\psi(w) \mid w \in L\}$.

For $n \in \mathbb{N}$, a set $S \subseteq \mathbb{N}^n$ is *semi-linear* if it is a finite union of sets of the form

$$\{\alpha_0 + n_1\alpha_1 + \cdots + n_m\alpha_m \mid \forall i : n_i \geq 0, \alpha_i \in \mathbb{N}^n\}$$

Parikh's Theorem [31] states that for a context-free language $L$, it holds that $\psi(L)$ is semi-linear. We now turn to our proof. We first claim that for every asyncCFHG $\mathcal{G} = \langle G, \Sigma, \{x_1, \ldots x_k\}, \alpha, \mathsf{async}\rangle$, we can construct an equivalent asyncCFHG $\mathcal{G}' = \langle G', \Sigma, \{x_1, \ldots x_k\}, \alpha, \mathsf{async}\rangle$, where $G'$ only uses letter-tuples in $\Sigma_{\#}^k$ that contain a single letter $\sigma \in \Sigma$ (and the rest are $\#$-letters) in its set of derivation rules.

We do this as follows. Let $\tau$ be a letter-tuple in which $\Sigma$-letters $\sigma_1 \ldots, \sigma_t$ occur in positions $i_1, \ldots i_t$, and the rest are $\#$-letters. We define a sequence $f(\tau) = \tau_1 \ldots \tau_t$, where $\tau_j$ is a letter-tuple in which $\tau_j[i_j] = \sigma_j$, and $\tau_j[i] = \#$ for every other index $i \neq i_j$. If $\tau$ only contains $\#$-letters, then we define $f(\tau)$ as the empty word.

For example, for $\tau = \langle \#, a, \#, b, a\rangle$ we have

$$f(\tau) = \langle \#, a, \#, \#, \#\rangle\langle \#, \#, \#, b, \#\rangle\langle \#, \#, \#, \#, a\rangle$$

Clearly, for every tuple $z = \langle w_1, \ldots, w_k\rangle$ of words over $\Sigma$, and for every word $\tau_1\tau_2 \ldots \tau_d$ over $\Sigma_{\#}^k$, we have $\tau_1\tau_2 \ldots \tau_d \in \mathsf{pad}(z)$ iff $f(\tau_1) \cdots f(\tau_d) \in \mathsf{pad}(z)$. In other words, applying $f$ does not change the words that are padded along every index.

The CFG $G'$ is then obtained from $G$ by replacing every letter $\tau \in \Sigma_{\#}^k$ that occurs in a rule in $G$, with $f(\tau)$. As argued above, and according to the semantics, we have $\mathfrak{L}(\mathcal{G}) = \mathfrak{L}(\mathcal{G}')$.

Another general observation is that a singleton $\{w\}$ is in $\mathfrak{L}(\mathcal{G})$ iff $G$ derives a word that is in $\mathsf{pad}(\langle w, \ldots, w\rangle)$. Indeed, as argued in Lemma IV.1, a single word is agnostic to the identity of the quantifiers. Therefore, if the same word $w$ is assigned to all variables in a way that is recognized by the underlying model $G$, then $\{w\} \in \mathfrak{L}(\mathcal{G})$ regardless of its quantification condition. In the other direction, for a singleton $\{w\}$ to be accepted, $w$ (in some padded form) must be assigned to all variables in a tuple that is recognized by $G$.

Now, assume by way of contradiction an asyncCFHG $\mathcal{G} = \langle G, \{a\}, \{x_1, \ldots, x_k\}, \alpha, \mathsf{async}\rangle$ for $\mathfrak{L}$. Following the discussion above, we can assume that the rules of $G$ only use letters that contain a single $a$ each. We denote by $\tau_i$ the letter in $\{a\}_{\#}^k$ in which $\tau[i] = a$, and $\tau[j] = \#$ for every other index $j \neq i$. We can then define the alphabet of $G$ to be $T = \{\tau_1, \ldots, \tau_k\}$.

Notice that while the alphabet of $\mathcal{G}$ is unary yet the alphabet of $G$ is not, due to the nature of $T$, the Parikh image $\psi(u)$ of a $u$ over $T$ uniquely determines the word-tuple over $a$ which $u$ pads: indeed, $\psi(u) = \langle i_1, \ldots, i_k\rangle$ iff $u \in \mathsf{pad}(\langle a^{i_1}, \ldots, a^{i_k}\rangle)$. As discussed above, we therefore have $\{a^i\} \in \mathfrak{L}(\mathcal{G})$ iff there exists $u \in \mathsf{pad}(\langle a^i, \ldots, a^i\rangle)$ such that $\psi(u) = \langle i, \ldots, i\rangle$, and $u \in L(G)$.

Now, according to our assumption that $\mathfrak{L}(\mathcal{G}) = \mathfrak{L}$, we have that $u \in L(G)$ for a word $u$ for which $\psi(u) = \langle j, \ldots, j\rangle$ iff $j = i^2$ for some $i \in \mathbb{N}$.

Of course, $G$ may also derive other words, that do not induce languages in $\mathfrak{L}$ and whose Parikh image is not of this form.

Denote $U = \{\langle i, \ldots, i\rangle \mid i \in \mathbb{N}\}$. It holds that $U$ is semi-linear, as it is spanned by the basis $\langle 1, \ldots, 1\rangle$. Denote $U^2 = \{\langle i^2, \ldots, i^2\rangle \mid i \in \mathbb{N}\}$. It holds that $U^2$ is not semi-linear. According to our assumption on $\mathfrak{L}(\mathcal{G})$, we have that $\psi(L(G)) \cap U = U^2$. However, $\psi(L(G))$ is semi-linear, and $U$ is semi-linear. Since semi-linear sets are closed under intersection, it follows that $U^2$ is semi-linear, a contradiction. □

### C. Synchronous vs. Asynchronous Hypermodels

Finally, we compare between different levels of the two semantics, and show that the stronger underlying model on the one hand and the stronger asynchronous semantics on the other render asyncNFH and syncCFHG incomparable.

**Theorem VI.6.** async*NFH are incomparable with* sync*CFHG.*

*Proof.* For the first direction, the proof of Theorem VI.1 serves to show that syncCFHG do not subsume asyncNFH, as the proof presents a language that is asyncNFH-recognizable but not syncCFHG-recognizable.

For the second direction, we use the proof of Theorem VI.3, which shows that we can construct a syncCFHG for $\mathfrak{L}_{pal}$ for which an asyncNFH does not exist, and we are done. □

**Theorem VI.7.** async*CFHG do not subsume* sync*CSHG.*

*Proof.* According to Lemma IV.4, we can construct a syncCSHG for the singletons hyperlanguage $\{\{a^{n^2}\} \mid n \in \mathbb{N}\}$, for which there exists no asyncCFHG. □

We remark that we leave the relations between syncCSHG and asyncCSHG for future work. Indeed, the nature of languages that are not context-sensitive is more elaborate, and does not naturally induce singletons examples, as with the formalisms that are lower in the hierarchy.

## VII. Conclusions and Future Work

We have provided a unified definition of computational hypermodels for finite-word hyperproperties, both for the synchronous and asynchronous semantics, and studied the expressiveness of the different classes of hyperlanguages. We showed that while synchronous hypermodels are much more expressive than their corresponding non-hyper counterparts, they still follow the Chomsky hierarchy, with strict containment at each level.

For asynchronous hypermodels, we show that they are strictly more expressive than the synchronous ones, which aligns with previous results regarding infinite words, in which classic synchronous hyperlogics like HyperLTL cannot express general asynchronous hyperproperties [12].

We have also established a similar result for the asynchronous models, as for the synchronous ones, and showed that despite their ability to express languages that go beyond the expressive power of their underlying models, and beyond their synchronous counterparts, the asynchronous hypermodels too maintain the Chomsky hierarchy.

We have further showed that there is no clear correspondence between asynchronous and synchronous models at different levels of the Chomsky hierarchy; the stronger underlying models on the one hand are not enough to overcome the stronger notion of asynchronicity on the other, as we have shown that asyncNFH are incomparable with syncCFHG.

For asyncNFH, we have established a hierarchy with respect to their number of quantifiers, and showed that as the number of variable groes, so does the expressive power of asyncNFH. We leave similar results for syncNFH, as well as for the hypergrammar models, for future work.

One may wonder whether techniques that work for the classical models can be lifted to hypermodels. As it turns out, this is not the case. Pumping arguments, for example, fail for two reasons. First, for languages that are not singletons, a hyperlanguage is defined by many different runs of the underlying model rather than a single (possibly pumpable) one. Second, for the asynchronous models, pumping affects every word in the word vector differently, due to the different padding of the different word vectors along the cycle. Moreover, as we have shown, hypermodels are capable of displaying much stronger expressive power than what their underlying models seemingly enable.

The hardness of the exact mapping of the full hierarchy is demonstrated by the sequence of works studying asynchronous logics over infinite words [12], [8], [13].

This paper focuses on the lower levels of the Chomsky hierarchy, and addresses finite automata and grammars. The full hierarchy requires the study of hyper-Turing machines, which we believe to be a fascinating direction on its own. We can define and study the hyper-versions of the classes of recursive and recursively-enumerable languages. It may be that hyper-Turing machines are capable of displaying non-recursive behaviors, just as the hypermodels in the lower levels are capable of going beyond the expressive power of their underlying models. We can further define matching complexity classes – the hyper-versions of P, NP, etc., and study their behaviors. We leave this exploration for future work.

We believe that the results presented here will help understand the power and expressiveness of the synchronous and asynchronous models and their interplay, and paves the way to painting a complete picture of these hypermodels.

## Acknowledgments

This work was supported in part by the Israel Science Foundation (ISF grant No. 655/25).

## References

[1] Ezio Bartocci, Thomas A. Henzinger, Dejan Nickovic, and Ana Oliveira da Costa. Hypernode automata. In Guillermo A. Pérez and Jean-François Raskin, editors, *CONCUR 2023*, volume 279 of *LIPIcs*, pages 21:1–21:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.CONCUR.2023.21.

[2] Jan Baumeister, Norine Coenen, Borzoo Bonakdarpour, Bernd Finkbeiner, and César Sánchez. A temporal logic for asynchronous hyperproperties. In Alexandra Silva and K. Rustan M. Leino, editors, *CAV 2021, Part I*, volume 12759 of *LNCS*, pages 694–717. Springer, 2021. doi:10.1007/978-3-030-81685-8_33.

[3] R. Beutner and B. Finkbeiner. On conformant planning and model-checking of exists forall hyperproperties. In *the European Conference on Artificial Intelligence (ECAI)*, 2025.

[4] Raven Beutner and Bernd Finkbeiner. AutoHyper: Explicit-state model checking for HyperLTL. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2023*, volume 13993. Springer, 2023. doi:10.1007/978-3-031-30823-9_8.

[5] Raven Beutner and Bernd Finkbeiner. Non-deterministic planning for hyperproperty verification. In Sara Bernardini and Christian Muise, editors, *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2024, Banff, Alberta, Canada, June 1-6, 2024*, pages 25–30. AAAI Press, 2024. URL: https://doi.org/10.1609/icaps.v34i1.31457, doi:10.1609/ICAPS.V34I1.31457.

[6] Raven Beutner, Bernd Finkbeiner, Hadar Frenkel, and Niklas Metzger. Second-order hyperproperties. In Constantin Enea and Akash Lal, editors, *CAV 2023, Part II*, volume 13965 of *LNCS*, pages 309–332. Springer, 2023. doi:10.1007/978-3-031-37703-7_15.

[7] Raven Beutner, Bernd Finkbeiner, Hadar Frenkel, and Niklas Metzger. Monitoring second-order hyperproperties. In Mehdi Dastani, Jaime Simão Sichman, Natasha Alechina, and Virginia Dignum, editors, *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024*, pages 180–188. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2024. URL: https://dl.acm.org/doi/10.5555/3635637.3662865, doi:10.5555/3635637.3662865.

[8] Alberto Bombardelli, Laura Bozzelli, César Sánchez, and Stefano Tonetta. Unifying asynchronous logics for hyperproperties. In Siddharth Barman and Slawomir Lasota, editors, *44th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2024, December 16-18, 2024, Gandhinagar, Gujarat, India*, volume 323 of *LIPIcs*, pages 14:1–14:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2024.14, doi:10.4230/LIPICS.FSTTCS.2024.14.

[9] Alberto Bombardelli, Laura Bozzelli, César Sánchez, and Stefano Tonetta. (asynchronous) temporal logics for hyperproperties on finite traces. In Gidon Ernst and Kristin Yvonne Rozier, editors, *Model Checking Software - 31st International Symposium, SPIN 2025, Hamilton, ON, Canada, May 7-8, 2025, Proceedings*, volume 15945 of *Lecture Notes in Computer Science*, pages 25–43. Springer, 2025. doi:10.1007/978-3-032-06847-7_2.

[10] Borzoo Bonakdarpour and Sarai Sheinvald. Finite-word hyperlanguages. *Inf. Comput.*, 295(Part A):104944, 2023. URL: https://doi.org/10.1016/j.ic.2022.104944, doi:10.1016/J.IC.2022.104944.

[11] Laura Bozzelli, Bastien Maubert, and Sophie Pinchinat. Unifying hyper and epistemic temporal logics. In Andrew M. Pitts, editor, *Foundations of Software Science and Computation Structures - 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9034 of *Lecture Notes in Computer Science*, pages 167–182. Springer, 2015. doi:10.1007/978-3-662-46678-0_11.

[12] Laura Bozzelli, Adriano Peron, and César Sánchez. Asynchronous extensions of HyperLTL. In *LICS 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470583.

[13] Laura Bozzelli, Adriano Peron, and César Sánchez. Expressiveness and decidability of temporal logics for asynchronous hyperproperties. In Bartek Klin, Slawomir Lasota, and Anca Muscholl, editors, *CONCUR 2022*, volume 243 of *LIPIcs*, pages 27:1–27:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CONCUR.2022.27.

[14] Noam Chomsky. Three models for the description of language. *IRE Trans. Inf. Theory*, 2(3):113–124, 1956. doi:10.1109/TIT.1956.1056813.

[15] Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. Temporal logics for hyperproperties. In Martín Abadi and Steve Kremer, editors, *Principles of Security and Trust - Third International Conference, POST 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, volume 8414 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2014. doi:10.1007/978-3-642-54792-8_15.

[16] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *J. Comput. Secur.*, 18(6):1157–1210, 2010. doi:10.3233/JCS-2009-0393.

[17] Norine Coenen, Bernd Finkbeiner, Hadar Frenkel, Christopher Hahn, Niklas Metzger, and Julian Siber. Temporal causality in reactive systems. In Ahmed Bouajjani, Lukás Holík, and Zhilin Wu, editors, *Automated Technology for Verification and Analysis - 20th International Symposium, ATVA 2022, Virtual Event, October 25-28, 2022, Proceedings*, volume 13505 of *Lecture Notes in Computer Science*, pages 208–224. Springer, 2022. doi:10.1007/978-3-031-19992-9_13.

[18] Norine Coenen, Bernd Finkbeiner, Christopher Hahn, and Jana Hofmann. The hierarchy of hyperlogics. In *Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019*. IEEE, 2019. doi:10.1109/LICS.2019.8785713.

[19] Norine Coenen, Bernd Finkbeiner, Jana Hofmann, and Julia J. Tillman. Smart contract synthesis modulo hyperproperties. In *36th IEEE Computer Security Foundations Symposium, CSF 2023, Dubrovnik, Croatia, July 10-14, 2023*, pages 276–291. IEEE, 2023. doi:10.1109/CSF57540.2023.00006.

[20] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995. doi:10.7551/mitpress/5803.001.0001.

[21] Bernd Finkbeiner, Lennart Haas, and Hazem Torfah. Canonical representations of k-safety hyperproperties. In *32nd IEEE Computer Security Foundations Symposium, CSF 2019, Hoboken, NJ, USA, June 25-28, 2019*, pages 17–31. IEEE, 2019. doi:10.1109/CSF.2019.00009.

[22] Bernd Finkbeiner, Markus N. Rabe, and César Sánchez. Algorithms for Model Checking HyperLTL and HyperCTL*. In Daniel Kroening and Corina S. Pasareanu, editors, *CAV 2015, Part I*, volume 9206 of *LNCS*, pages 30–48. Springer, 2015. doi:10.1007/978-3-319-21690-4_3.

[23] Bernd Finkbeiner and Martin Zimmermann. The first-order logic of hyperproperties. In *34th Symposium on Theoretical Aspects of Computer Science, STACS*, 2017. doi:10.4230/LIPIcs.STACS.2017.30.

[24] Hadar Frenkel, Gaëtan Regaud, and Martin Zimmermann. The complexity of second-order hyperltl. *Logical Methods in Computer Science*, Volume 22, Issue 1, Mar 2026. URL: https://lmcs.episciences.org/16039, doi:10.46298/lmcs-22(1:26)2026.

[25] Hadar Frenkel and Sarai Sheinvald. Realizable and context-free hyperlanguages. In Pierre Ganty and Dario Della Monica, editors, *Proceedings of the 13th International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2022, Madrid, Spain, September 21-23, 2022*, volume 370 of *EPTCS*, pages 114–130, 2022. doi:10.4204/EPTCS.370.8.

[26] Joseph A. Goguen and José Meseguer. Security policies and security models. In *1982 IEEE Symposium on Security and Privacy, Oakland, CA, USA, April 26-28, 1982*, pages 11–20. IEEE Computer Society, 1982. doi:10.1109/SP.1982.10014.

[27] Jens Oliver Gutsfeld, Markus Müller-Olm, and Christoph Ohrem. Automata and fixpoints for asynchronous hyperproperties. *Proc. ACM Program. Lang.*, 5(POPL):1–29, 2021. doi:10.1145/3434319.

[28] Juha Kontinen, Max Sandström, and Jonni Virtema. Set semantics for asynchronous TeamLTL: Expressivity and complexity. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *MFCS 2023*, volume 272 of *LIPIcs*, pages 60:1–60:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.MFCS.2023.60.

[29] Juha Kontinen, Max Sandström, and Jonni Virtema. A remark on the expressivity of asynchronous TeamLTL and HyperLTL. In Arne Meier and Magdalena Ortiz, editors, *FoIKS 2024*, volume 14589 of *LNCS*, pages 275–286. Springer, 2024. doi:10.1007/978-3-031-56940-1_15.

[30] Andreas Krebs, Arne Meier, Jonni Virtema, and Martin Zimmermann. Team semantics for the specification and verification of hyperproperties. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *MFCS 2018*, volume 117 of *LIPIcs*, pages 10:1–10:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.10.

[31] Rohit J. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, October 1966. doi:10.1145/321356.321364.

[32] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.32.

[33] Francesco Pontiggia, Filip Macák, Roman Andriushchenko, Michele Chiari, and Milan Ceska. Decentralized planning using probabilis-

tic hyperproperties. In Sanmay Das, Ann Nowé, and Yevgeniy Vorobeychik, editors, *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*, pages 1688–1697. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025. URL: https://dl.acm.org/doi/10.5555/3709347.3743804, doi:10.5555/3709347.3743804.

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

[35] Aravinda Prasad Sistla. *Theoretical issues in the design and verification of distributed systems*. PhD thesis, Harvard University, 1983.

[36] Yu Wang, Siddhartha Nalluri, and Miroslav Pajic. Hyperproperties for robotics: Planning via hyperltl. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pages 8462–8468. IEEE, 2020. doi:10.1109/ICRA40945.2020.9196874.

[37] Andrew C. Yao and Ronald L. Rivest. k + 1 heads are better than k. *J. ACM*, 25(2):337–340, April 1978. doi:10.1145/322063.322076.