# Realizable and Context-Free Hyperlanguages

Hadar Frenkel
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
hadar.frenkel@cispa.de

Sarai Sheinvald
Department of Software Engineering, Braude College of Engineering, Karmiel, Israel
sarai@braude.ac.il

*Hyperproperties* lift conventional trace-based languages from a set of execution traces to a set of sets of executions. From a formal-language perspective, these are sets of sets of words, namely *hyperlanguages*. *Hyperautomata* are based on classical automata models that are lifted to handle hyperlanguages. *Finite hyperautomata* (NFH) have been suggested to express regular hyperproperties. We study the *realizability* problem for regular hyperlanguages: given a set of languages, can it be precisely described by an NFH? We show that the problem is complex already for singleton hyperlanguages. We then go beyond regular hyperlanguages, and study *context-free* hyperlanguages. We show that the natural extension to context-free hypergrammars is highly undecidable. We then suggest a refined model, namely *synchronous hypergrammars*, which enables describing interesting non-regular hyperproperties, while retaining many decidable properties of context-free grammars.

## 1 Introduction

*Hyperproperties* [10] generalize traditional trace properties [1] to *system properties*, i.e., from sets of traces to sets of sets of traces. A hyperproperty dictates how a system should behave in its entirety and not just based on its individual executions. Hyperproperties have been shown to be a powerful tool for expressing and reasoning about information-flow security policies [10] and important properties of cyber-physical systems [23] such as *sensitivity* and *robustness*, as well as consistency conditions in distributed computing such as *linearizability* [4]. Different types of logics, such as HyperLTL, HyperCTL$^*$ [9], HyperQPTL [19] and HyperQCTL$^*$ [11] have been suggested for expressing hyperproperties.

In the *automata-theoretic approach* both the system and the specification are modeled as automata whose language is the sets of execution traces of the system, and the set of executions that satisfy the specification [21, 20]. Then, problems such as model-checking [8] ("Does the system satisfy the property?") and satisfiability ("Is there a system that satisfies the property?") are reduced to decision problems for automata, such as containment ("Is the language of an automaton $A$ contained in the language of automaton $B$?") and nonemptiness ("Is there a word that the automaton accepts?"). Finite-word and $\omega$-regular automata are used for modeling trace specifications [22]. *Hyperautomata*, introduced in [5], generalize word automata to automata that run on sets of words. Just as hyperproperties describe a system in its entirety, the *hyperlanguages* of hyperautomata describe the language in its entirety.

The work in [5] focuses on *nondeterministic finite-word hyperautomata* (NFH), that are able to model *regular hyperlanguages*. An NFH $\mathcal{A}$ uses *word variables* that are assigned words from a language $\mathcal{L}$, as well as a *quantification condition* over the variables, which describes the existential ($\exists$) and global ($\forall$) requirements from $\mathcal{L}$. The *underlying NFA* of $\mathcal{A}$ runs on the set of words assigned to the word variables, all at the same time. The hyperlanguage of $\mathcal{A}$ is then the set of all languages that satisfy the

To appear in EPTCS.

| Type of language | Quantification Type | Realizability |
|---|---|---|
| Finite | $\exists^*$, $\forall^*$ | unrealizable (T.5) |
|  | $\forall\exists$ | polynomial (T.6) |
| Infinite | $\exists^*$, $\forall^*$, $\exists^*\forall^*$ | unrealizable (T.5) |
| Ordered | $\exists\forall\exists$ | polynomial (T. 8) |
| partially Ordered | $\exists^*\forall\exists^*$ | exponential (T.10) |
| Prefix-Closed Regular | $\exists\forall\exists^*$ | polynomial (T.11) |
| Regular | $\exists^*\forall\exists^*$ | doubly exponential (T.12)[1] |

Table 1: Summary of realizability results for singleton hyperlanguages.

quantification condition. The decidability of the different decision problems for NFH heavily depends on the quantification condition. For example, nonemptiness of NFH is decidable for the conditions $\forall^*$ (a sequence of $\forall$-quantifiers), $\exists^*$ and $\exists^*\forall^*$, but is undecidable for $\forall\exists$. In [15] NFH are used to specify *multi-properties*, which express the behaviour of several models that run in parallel.

A natural problem for a model $M$ for languages is *realizability*: given a language $\mathcal{L}$, can it be described by $M$? For finite-word automata, for example, the answer relies on the number of equivalence classes of the Myhill-Nerode relation for $\mathcal{L}$. For hyperlanguages, we ask whether we can formulate a hyperproperty that precisely describes a given set of languages. We study this problem for NFH: given a set $\mathfrak{L}$ of languages, can we construct an NFH whose hyperlanguage is $\mathfrak{L}$? We can ask the question generally, or for specific quantification conditions. We focus on a simple case of this problem, where $\mathfrak{L}$ consists of a single language, (that is, $\mathfrak{L} = \{\mathcal{L}\}$ for some language $\mathcal{L}$), which turns out to be non-trivial. In [12], the authors present automata constructions for safety regular hyperproperties. As such, they are strictly restricted only to $\forall^*$-conditions.

We show that for the simplest quantification conditions, $\exists^*$ and $\forall^*$, no singleton hyperlanguage is realizable, and that single alternation does not suffice for a singleton hyperlanguage consisting of an infinite language. We show that when $\mathcal{L}$ is finite, then $\{\mathcal{L}\}$ is realizable with a $\forall\exists$ quantification condition.

We then define *ordered* languages. These are languages that can be enumerated by a function $f$ that can be described by an automaton that reads pairs of words: a word $w$, and $f(w)$. We show that for an ordered language $\mathcal{L}$, the hyperlanguage $\{\mathcal{L}\}$ is realizable with a quantification condition of $\exists\forall\exists$. We then generalize this notion to *partially ordered* languages, which are enumerated by a relation rather than a function. We show that for this case, $\{\mathcal{L}\}$ is realizable with a quantification condition of $\exists^*\forall\exists^*$.

Finally, we use ordered languages to realize singleton hyperlanguages consisting of regular languages: we show that when $\mathcal{L}$ is a prefix-closed regular language, then it is partially ordered, and that an NFH construction for $\{\mathcal{L}\}$ is polynomial in the size of a finite automaton for $\mathcal{L}$. We then show that every regular language is partially ordered. Therefore, when $\mathcal{L}$ is regular, then $\{\mathcal{L}\}$ is realizable with a quantification condition of $\exists^*\forall\exists^*$. The summary of our results is listed in Table 1.

In the second part of the paper, we go beyond regular hyperlanguages, and study *context-free hyperlanguages*. To model this class, we generalize context-free grammars (CFG) to *context-free hypergrammars* (CFHG), similarly to the generalization of finite-word automata to NFH: we use an *underlying CFG* that derives the sets of words that are assigned to the word variables in the CFHG $G$. The quantification condition of $G$ defines the existential and global requirements from these assignments.

The motivation for context-free hyperlanguages is clear: they allow expressing more interesting hyperproperties. As a simple example, consider a robot which we want to return to its charging area before its battery is empty. This can be easily expressed with a CFHG with a $\forall$-condition, as we demonstrate in

[1]See remark 2.

| Problem | | syncCFHGs | General CFHGs |
|---|---|---|---|
| Emptiness | $\exists^*$ | polynomial (T.15) | polynomial (T.15) |
| | $\forall^*$, $\exists\forall^*$ | polynomial (T.23) | undecidable (T.18) |
| | $\exists^*\forall^*$ | undecidable (T.25) | undecidable (T.18) |
| Finite Membership | $*$ | exponential (R.4) | exponential (T.17) |
| Regular Membership | $\exists^*$ | exponential (R.5) | exponential (T.16) |
| | $\forall^*$ | undecidable (T.24) | undecidable (T.24) |

Table 2: Summary of decidability results for synchronous and general CFHGs.

Section 4, Example 3. Note that since the underlying property – *charging time is larger than action time* – is non-regular, NFH cannot capture this specification. Extending this example, using an $\exists\forall$-condition we can express the property that all such executions of the robot are bounded, so that the robot cannot charge and act unboundedly.

Some aspects regarding context-free languages in the context of model-checking have been studied. In [13, 18] the authors explore model-checking of HyperLTL properties with respect to context-free models. There, the systems are context-free, but not the specifications. The work of [6] studies the verification of non-regular temporal properties, and [14] studies the synthesis problem of context-free specifications. These do not handle context-free *hyperproperties*.

While most natural decision problems are decidable for regular languages, this is not the case for CFG. For example, the universality ("Does the CFG derive all possible words?") and containment problems for context-free languages are undecidable. The same therefore holds also for CFHG. However, the nonemptiness and membership ("Does the CFG derive the word $w$"?) problems are decidable for CFG.

We study the various decision problems for CFHG. Specifically, We explore the nonemptiness problem ("Is there a language that the CFHG derives?"); and the membership problem ("Does the CFHG derive the language $\mathcal{L}$"?). These problems correspond to the satisfiability and model-checking problems, respecitively. We show that for general CFHG, most of these problems soon become undecidable (see Table 2). Some of the undecidability results are inherent to CFG. Some, however, are due to the *asynchronous* nature of CFHG: when the underlying CFG of a CFHG derives a set of words that are assigned to the word variables, it does not necessarily do so synchronously. For example, in one derivation step one word in the set may be added 2 letters, and another 1 letter. NFH read one letter at a time from every word in the set, and are hence naturally synchronous. In [3], the authors study asynchronous hyperLTL, which suffers from the same phenomenon.

We therefore define *synchronous context-free hyperlanguages*, which require synchronous reading of the set of words assigned to the word variables. We also define *synchronous CFHG* (syncCFHG), a fragment of CFHG in which the structure of the underlying CFG is limited in a way that ensures synchronous behavior. We prove that syncCFHG precisely captures the class of synchronous context-free hyperlanguages. Further, we show that some of the undecidable problems for CFHG, such as the nonemptiness problem for the $\forall^*$- and $\exists\forall^*$-fragments, become decidable for syncCFHG.

## 2 Preliminaries

**Hyperautomata** We assume that the reader is familiar with the definitions of deterministic finite automata (DFA) and non-deterministic finite automata (NFA).

**Definition 1.** *Let $\Sigma$ be an alphabet. A* hyperlanguage $\mathfrak{L}$ over $\Sigma$ *is a set of languages over $\Sigma$, that is, $\mathfrak{L} \in 2^{2^{\Sigma^*}}$. A* nondeterministic finite-word hyperautomaton *(NFH) is a tuple $\mathcal{A} = \langle \Sigma, X, Q, Q_0, F, \delta, \alpha \rangle$,*

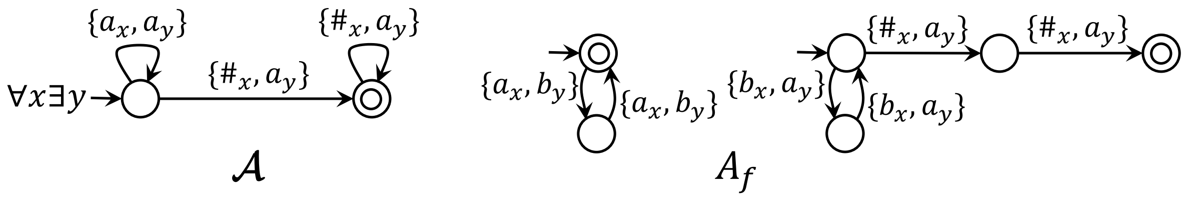

Figure 1: The NFH $\mathcal{A}$ (left), whose hyperlanguage is the set of infinite languages over $\{a\}$, and the NFA $A_f$ that computes $f$ (right).

*where $X$ is a finite set of* word variables, *and* $\alpha = \mathbb{Q}_1 x_1 \cdots \mathbb{Q}_k x_k$ *is a* quantification condition, *where* $\mathbb{Q}_i \in \{\exists, \forall\}$ *for every* $i \in [1,k]$, *and* $\langle \hat{\Sigma}, Q, Q_0, F, \delta \rangle$ *forms an underlying NFA over* $\hat{\Sigma} = (\Sigma \cup \{\#\})^X$.

Let $\mathcal{L}$ be a language. We represent an assignment $v : X \to \mathcal{L}$ as a *word assignment* $\boldsymbol{w}_v$, which is a word over the alphabet $(\Sigma \cup \{\#\})^X$ (that is, assignments from $X$ to $(\Sigma \cup \{\#\})^*$), where the $i$'th letter of $\boldsymbol{w}_v$ represents the $k$ $i$'th letters of the words $v(x_1), \ldots, v(x_k)$ (in case that the words are not of equal length, we "pad" the end of the shorter words with #-symbols). We represent these $k$ $i$'th letters as an assignment denoted $\{\sigma_{1x_1}, \sigma_{2x_2}, \ldots, \sigma_{kx_k}\}$, where $x_j$ is assigned $\sigma_j$. For example, the assignment $v(x_1) = aa$ and $v(x_2) = abb$ is represented by the word assignment $\boldsymbol{w}_v = \{a_{x_1}, a_{x_2}\}\{a_{x_1}, b_{x_2}\}\{\#_{x_1}, b_{x_2}\}$.

The acceptance condition for NFH is defined with respect to a language $\mathcal{L}$, the underlying NFA $\hat{\mathcal{A}}$, the quantification condition $\alpha$, and an assignment $v : X \to \mathcal{L}$.

- For $\alpha = \varepsilon$, define $\mathcal{L} \vdash_v (\alpha, \hat{\mathcal{A}})$ if $\boldsymbol{w}_v \in \mathcal{L}(\hat{\mathcal{A}})$.
- For $\alpha = \exists x.\alpha'$, define $\mathcal{L} \vdash_v (\alpha, \hat{\mathcal{A}})$ if there exists $w \in \mathcal{L}$ s.t. $\mathcal{L} \vdash_{v[x \mapsto w]} (\alpha', \hat{\mathcal{A}})$.
- For $\alpha = \forall x.\alpha'$, define $\mathcal{L} \vdash_v (\alpha, \hat{\mathcal{A}})$ if $\mathcal{L} \vdash_{v[x \mapsto w]} (\alpha', \hat{\mathcal{A}})$ for every $w \in \mathcal{L}$ .[2]

When $\alpha$ includes all of $X$, then membership is independent of the assignment, and we say that $\mathcal{A}$ *accepts* $\mathcal{L}$, and denote $\mathcal{L} \in \mathfrak{L}(\mathcal{A})$.

**Definition 2.** *Let $\mathcal{A}$ be an NFH. The* hyperlanguage *of $\mathcal{A}$, denoted $\mathfrak{L}(\mathcal{A})$, is the set of all languages that $\mathcal{A}$ accepts. When the quantification condition $\alpha$ of an NFH $\mathcal{A}$ is $\mathbb{Q}_1 x_1 . \mathbb{Q} 2 x_2 \cdots \mathbb{Q}_k x_k$, we denote $\mathcal{A}$ as being a $\mathbb{Q}_1 \mathbb{Q}_2 \ldots \mathbb{Q}_k$-NFH (or, sometimes, as an $\alpha$-NFH).*

*Example* 1. Consider the NFH $\mathcal{A}$ depicted in Figure 1, over the alphabet $\{a\}$. The quantification condition $\forall x. \exists y$ requires that in a language $\mathcal{L}$ accepted by $\mathcal{A}$, for every word $u_1$ that is assigned to $x$, there exists a word $u_2$ that is assigned to $y$ such that the joint run of $u_1, u_2$ is accepted by the underlying NFA $\hat{\mathcal{A}}$ of $\mathcal{A}$. The NFA $\hat{\mathcal{A}}$ requires that the word assigned to $y$ is longer than the word assigned to $x$: once the word assigned to $x$ ends (and the padding # begins), the word assigned to $y$ must still read at least one more $a$. Therefore, $\mathcal{A}$ requests that for every word in $\mathcal{L}$, there exists a longer word in $\mathcal{L}$. This holds iff $\mathcal{L}$ is infinite. Therefore, the hyperlanguage of $\mathcal{A}$ is the set of all infinite languages over $\{a\}$.

## Context-Free Grammars

**Definition 3.** *A* context-free grammar *(CFG) is a tuple $G = \langle \Sigma, V, V_0, P \rangle$, where $\Sigma$ is an alphabet, $V$ is a set of* grammar variables, *$V_0 \in V$ is an* initial variable, *and $P \subseteq V \times (V \cup \Sigma)^*$ is a set of* grammar rules.

We say that $w$ is a *terminal word* if $w \in \Sigma^*$. Let $v \in V$ and $\alpha, \beta \in (V \cup \Sigma)^*$.

[2]In case that $\alpha$ begins with $\forall$, membership holds vacuously with the empty language. We restrict the discussion to satisfaction by nonempty languages.

- We say that $v$ *derives* $\alpha$ if $(v,\alpha) \in P$. We then denote $v \to \alpha$.
- We denote $\alpha \Rightarrow \beta$ if there exist $v \in V$ and $\alpha_1, \alpha_2, \beta' \in (V \cup \Sigma)^*$ such that $v \to \beta'$, $\alpha = \alpha_1 v \alpha_2$ and $\beta = \alpha_1 \beta' \alpha_2$.
- We say that $\alpha$ *derives* $\beta$, or that $\beta$ is *derived* by $\alpha$, if there exists $n \in \mathbf{N}$ and $\alpha_1, \dots \alpha_n \in (V \cup \Sigma)^*$ such that $\alpha_1 = \alpha$, $\alpha_n = \beta$ and $\forall 1 \le i < n: \ \alpha_i \Rightarrow \alpha_{i+1}$. We then denote $\alpha \Rightarrow^* \beta$.

The *language* of a CFG $G$ is the set of all terminal words that are derived by the initial variable. That is, $\mathcal{L}(G) = \{w \in \Sigma^* \mid V_0 \Rightarrow^* w\}$.

## 3 Realizability of Regular Hyperlanguages

Every NFH $\mathcal{A}$ defines a set of languages $\mathfrak{L}$. In the *realizability problem* for NFH, we are given a hyperlanguage $\mathfrak{L}$, and ask whether there exists an NFH $\mathcal{A}$ such that $\mathfrak{L}(\mathcal{A}) = \mathfrak{L}$. The answer may depend on the quantification condition that we allow using. In this section we study the realizability problem for singleton hyperlanguages, which turns out to be non-trivial. We show that while for finite languages we can construct a $\forall\exists$-NFH, such a condition cannot suffice for infinite languages. Further, we show that a general regular language requires a complex construction and quantification condition.

**Definition 4.** *Let $\mathfrak{L}$ be a hyperlanguage. For a sequence of quantifiers $\alpha = \mathbb{Q}_1 \dots \mathbb{Q}_k$, we say that $\mathfrak{L}$ is* $\alpha$-realizable *if there exists an NFH $\mathcal{A}$ with a quantification condition $\mathbb{Q}_1 x_1 \dots \mathbb{Q}_k x_k$ such that $\mathfrak{L}(\mathcal{A}) = \mathfrak{L}$.*

We first define some operations and notations on the underlying NFA of NFH we use in our proofs.

For a word $w$, the NFA $A_w$ is an NFA for $\{w\}$.

Let $A_1, A_2$ be NFA, and let $A_1 \uparrow^{\#} = \langle \Sigma, Q, q_0, \delta_1, F_1 \rangle$ and $A_2 \uparrow^{\#} = \langle \Sigma, P, p_0, \delta_2, F_2 \rangle$, where $A_i \uparrow^{\#}$ is an NFA for the language $\mathcal{L}(A_i) \cdot \#^*$. We define the *composition* $A_1 \otimes A_2$ of $A_1$ and $A_2$ to be an NFA over $\Sigma_2 = (\Sigma \cup \{\#\})^{\{x,y\}}$, defined as $A_1 \otimes A_2 = \langle \Sigma_2, Q \times P, (q_0, p_0), \delta, F_1 \times F_2 \rangle$, where for every $(q, \sigma, q') \in \delta_1$, $(p, \tau, p') \in \delta_2$, we have $((q,p), \{\sigma_x, \tau_y\}, (q', p')) \in \delta$. That is, $A_1 \otimes A_2$ is the composition of $A_1$ and $A_2$, which follows both automata simultaneously on two words (adding padding by # when necessary). A word assignment $\{x \mapsto w_1, y \mapsto w_2\}$ is accepted by $A_1 \otimes A_2$ iff $w_1 \in \mathcal{L}(A_1)$ and $w_2 \in \mathcal{L}(A_2)$ (excluding the #-padding).

We also define $A_1 \oplus A_2$ of $A_1$ and $A_2$ in a similar way, but the transitions are restricted to equally labeled letters. That is, for every $(q, \sigma, q') \in \delta_1$, $(p, \sigma, p') \in \delta_2$, we have $((q,p), \{\sigma_x, \sigma_y\}, (q', p')) \in \delta$. A run of the restricted composition then describes the run of $A_1$ and $A_2$ on the same word.

We generalize the definition of both types of compositions to a sequence of $k$ NFA $A_1, A_2, \dots A_k$, forming NFA $\bigotimes_{i=1}^{k} A_i$ and $\bigoplus_{=1}^{k} A_i$ over $(\Sigma \cup \#)^{\{x_1, x_2, \dots x_k\}}$, in the natural way. When all NFA are equal to $A$, we denote this composition by $A^{\otimes k}$ (or $A^{\oplus k}$). When we want to explicitly name the variables $x_1, x_2, \dots x_k$ in the compositions, we denote $\bigotimes_{i=1}^{k} A_i[x_1, \dots x_k]$ (or $\bigoplus_{i=1}^{k} A_i[x_1, \dots x_k]$).

We also generalize the notion of composition to NFA over $(\Sigma \cup \{\#\})^X$ in the natural way. That is, for NFA $A_1$ and $A_2$ with sets of variables $X$ and $Y$, respectively, the NFA $A_1 \otimes A_2$ is over $X \cup Y$, and follows both NFA simultaneously on both assignments (if $X \cap Y \neq \emptyset$, we rename the variables).

We study the realizability problem for the case of singleton hyperlanguages, that is, hyperlanguages of the type $\{\mathcal{L}\}$. We begin with a few observations on unrealizability of this problem, and show that general singleton hyperlanguages cannot be realized using simple quantification conditions.

### 3.1 Unrealizability

For the homogeneous quantification conditions, we have that a $\forall$-NFH $\mathcal{A}$ accepts a language $\mathcal{L}$ iff $\mathcal{A}$ accepts every $\mathcal{L}' \subseteq \mathcal{L}$. Therefore, a hyperlanguage $\{\mathcal{L}\}$ is not $\forall$-realizable for every $\mathcal{L}$ that is not a singleton. The same holds for every $\forall^*$-NFH.

An $\exists$-NFH $\mathcal{A}$ accepts a language $\mathcal{L}$ iff $\mathcal{L}$ contains some word that is accepted by $\hat{\mathcal{A}}$. Thus if $\mathcal{A}$ is nonempty, its hyperlanguage is infinite, and clearly not a singleton. The same holds for every $\exists^*$-NFH.

Now, consider an $\exists^k\forall^m$-NFH $\mathcal{A}$. As shown in [5], $\mathcal{A}$ is nonempty iff it accepts a language whose size is at most $k$. Therefore, $\{\mathcal{L}\}$ is not $\exists^k\forall^m$-realizable for every $\mathcal{L}$ such that $|\mathcal{L}| > k$.

As we show in Theorem 6, if $\mathcal{L}$ is finite then $\{\mathcal{L}\}$ is $\forall\exists$-realizable. We now show that if $\mathcal{L}$ is infinite, then $\{\mathcal{L}\}$ is not $\forall\exists$-realizable. Assume otherwise by contradiction, and let $\mathcal{A}$ be a $\forall x.\exists y$-NFH that accepts $\{\mathcal{L}\}$. Then for every $w \in \mathcal{L}$ there exists $u \in \mathcal{L}$ such that $\boldsymbol{w}_{[x\mapsto w][y\mapsto u]} \in \mathcal{L}(\hat{\mathcal{A}})$. Let $w_1$ be some word in $\mathcal{L}$. We construct an infinite sequence $w_1, w_2, \ldots$ of words in $\mathcal{L}$, as follows. For every $w_i$, let $w_{i+1}$ be a word in $\mathcal{L}$ such that $\boldsymbol{w}_{[x\mapsto w_i][y\mapsto w_{i+1}]} \in \mathcal{L}(\hat{\mathcal{A}})$. If $w_i = w_j$ for some $i < j$, then the language $\{w_i, w_{i+1}, \ldots, w_j\}$ is accepted by $\mathcal{A}$, and so $\mathcal{L}$ is not the only language that $\mathcal{A}$ accepts. Otherwise, all words in the sequence are distinct. Then, the language $\mathcal{L}_i = \{w_i, w_{i+1}, \ldots\}$ is accepted by $\mathcal{A}$ for every $i > 1$, and $\mathcal{L}_i \subset \mathcal{L}$. In both cases, $\mathcal{L}$ is not the only language that $\mathcal{A}$ accepts, and so $\{\mathcal{L}\}$ is not $\forall\exists$-realizable.

To conclude, we have the following.

**Theorem 5.** *If $\mathcal{L}$ contains more than one word, then $\{\mathcal{L}\}$ is not $\forall^*$-realizable and not $\exists^*$-realizable. If $\mathcal{L}$ contains more than $k$ words then $\{\mathcal{L}\}$ is not $\exists^k\forall^*$-realizable. If $\mathcal{L}$ is infinite then $\{\mathcal{L}\}$ is not $\exists^*\forall^*$-realizable and not $\forall\exists$-realizable.*

For positive realizability results, we first consider a simple case of a hyperlanguage consisting of a single finite language.

**Theorem 6.** *Let $\mathcal{L}$ be a finite language. Then $\{\mathcal{L}\}$ is $\forall\exists$-realizable.*

*Proof.* Let $\mathfrak{L} = \{w_1, w_2, \ldots w_k\}$. We construct a $\forall\exists$-NFH $\mathcal{A}$ for $\mathcal{L}$, whose underlying NFA $\hat{\mathcal{A}}$ is the union of all NFA $A_{w_i} \otimes A_{w_{i+1 (\text{mod } k)}}$. Let $\mathcal{L}' \in \mathfrak{L}(\mathcal{A})$. Since $\hat{\mathcal{A}}$ can only accept words in $\mathcal{L}$, we have that $\mathcal{L}' \subseteq \mathcal{L}$. Since $\mathcal{A}$ requires, for every $w_i \in \mathcal{L}'$, the existence of $w_{i+1(\text{mod } k)}$, and since $\mathcal{L}' \neq \emptyset$, we have that by induction, $w_i \in \mathcal{L}'$ implies $w_{i+j(\text{mod } k)} \in \mathcal{L}'$ for every $1 \leq j \leq k$. Therefore, $\mathcal{L} \subseteq \mathcal{L}'$. □

## 3.2 Realizability of Ordered Languages

Since every language is countable, we can always order its words. We show that for an ordering of a language $\mathcal{L}$ that is *regular*, that is, can be computed by an NFA, $\{\mathcal{L}\}$ can be realized by an $\exists\forall\exists$-NFH.

**Definition 7.** *Let $\mathcal{L}$ be a language. We say that a function $f : \mathcal{L} \to \mathcal{L}$ is $\mathcal{L}$-*regular *if there exists an NFA $A_f$ over $(\Sigma \cup \{\#\})^{\{x,y\}}$ such that for every $w \in \mathcal{L}$, it holds that $f(w) = u$ iff $\boldsymbol{w}_{[x\mapsto w][y\mapsto u]} \in \mathcal{L}(A_f)$. We then say that $A_f$* computes *$f$.*

*We say that a language $\mathcal{L}$ is* ordered *if the words in $\mathcal{L}$ can be arranged in a sequence $w_1, w_2, \ldots$ such that there exists an $\mathcal{L}$-regular function such that $f(w_i) = w_{i+1}$ for every $i \geq 0$.*

*Example* 2. Consider the language $\{a^{2i}, b^{2i} | i \in \mathbb{N}\}$, and a function $\forall i \in \mathbb{N} : f(a^{2i}) = b^{2i}, f(b^{2i}) = a^{2i+2}$, which matches the sequence $\varepsilon, a^2, b^2, a^4, b^4, \ldots$. The function $f$ can be computed by the NFA $A_f$ depicted in Figure 1, which has two components: one that reads $a^{2i}$ on $x$ and $b^{2i}$ on $y$, and one that reads $b^{2i}$ on $x$ and $a^{2i+2}$ on $y$.

**Theorem 8.** *Let $\mathcal{L}$ be an ordered language. Then $\{\mathcal{L}\}$ is $\exists\forall\exists$-realizable.*

*Proof.* Let $\mathcal{L} = \{w_1, w_2, \ldots\}$ be an ordered language via a regular function $f$, and let $A_f$ be an NFA that computes $f$. We construct an $\exists x_1 \forall x_2 \exists x_3$-NFH $\mathcal{A}$ for $\{\mathcal{L}\}$ by setting its underlying NFA to be $\hat{\mathcal{A}} = A_{w_1} \otimes A_f[x_1, x_2, x_3]$. Intuitively, $\mathcal{A}$ creates a "chain-reaction": $A_{w_1}$ requires the existence of $w_1$, and $A_f$ requires the existence of $w_{i+1}$ for every $w_i$. By the definition of $f$, only words in $\mathcal{L}$ may be assigned to $x_2$. Therefore, $\mathfrak{L}(\mathcal{A}) = \{\mathcal{L}\}$. □

We now generalize the definition of ordered languages, by allowing several minimal words instead of one, and allowing each word to have several successors. The computation of such a language then matches a *relation* over the words, rather than a function.

**Definition 9.** *We say that a language $\mathcal{L}$ is $m,k$-ordered, if there exists a relation $R \subseteq \mathcal{L} \times \mathcal{L}$ such that:*

- *There exist exactly $m$ words $w \in \mathcal{L}$ such that $(u,w) \notin R$ for every $u \neq w \in \mathcal{L}$ (that is, there are $m$ minimal words).*
- *$R \subseteq S$ for a total order $S$ of $\mathcal{L}$ with a minimal element.*
- *For every $w \in \mathcal{L}$ there exist $1 \leq i \leq k$ successor words: words $u$ such that $(w,u) \in R$.*
- *There exists an NFA $A_R$ over $(\Sigma \cup \{\#\})^{\{x,y\}}$ such that for every $u,v \in \mathcal{L}$, it holds that $R(u,v)$ iff $\mathbf{w}_{[x \mapsto u][y \mapsto v]} \in \mathcal{L}(A_R)$.*

*We then say that $A_R$* computes *$\mathcal{L}$. We call $\mathcal{L}$* partially ordered *if there exist $m,k$ such that $\mathcal{L}$ is $m,k$-ordered.*

**Theorem 10.** *Let $\mathcal{L}$ be $m,k$-ordered. Then $\{\mathcal{L}\}$ is $\exists^m\forall\exists^k$-realizable.*

*Proof.* Let $A_R = \langle (\Sigma \cup \{\#\})^{\{x,y\}}, Q, q_0, \delta, F \rangle$ be a DFA that computes $\mathcal{L}$. We construct an $\exists^m\forall\exists^k$-NFH $\mathcal{A}$ for $\mathcal{L}$, as follows. The quantification condition of $\mathcal{A}$ is $\exists x_1 \cdots \exists x_m \forall z \exists y_1 \cdots \exists y_k$. The $x$-variables are to be assigned $u_1, \ldots u_m$, the $m$ minimal words of $R$. We set $A_U = \bigotimes_{i=1}^{m} A_{u_i}[x_1, \ldots x_m]$.

The underlying NFA $\hat{\mathcal{A}}$ of $\mathcal{A}$ comprises of an NFA $A_i$ for every $1 \leq i \leq k$. Let $\mathcal{L}_i$ be the set of words in $\mathcal{L}$ that have exactly $i$ successors. Intuitively, $A_i$ requires, for every word $w \in \mathcal{L}_i$ that is assigned to $z$, the existence of the $i$ successors of $w$.

To do so, we construct an NFA $B_i$ over $z, y_1, \ldots y_i$ that accepts $\mathbf{w}_{[z \mapsto w][y_1 \mapsto w_1] \ldots [y_i \mapsto w_i]}$, for every word $w$ and its successors $w_1, \ldots w_i$. The construction of $B_i$ requires that: (1) all assignment to $y$-variables are successors of the assignment to $z$, by basing $B_i$ on a composition of $A_R$, and (2) $y_1, \ldots y_i$ are all assigned different words. This is done by keeping track of the pairs of assignments to $y$-variables that at some point read different letters. The run may accept only once all pairs are listed. We finally set $A_i = A_U \otimes B_i$, and require $y_{i+1} \ldots y_k$ to be equally assigned to $z$.

Since $R \subseteq S$ and $R$ has minimal elements, for every word $w \in \mathcal{L}$ there exists a sequence $w_0, w_1, \ldots w_t$ such that $w_t = w$, and $w_0$ is minimal, and $R(w_j, w_{j+1})$ for every $j \in [0, t-1]$. The NFH $\mathcal{A}$ then requires $w_0$ (via $A_u$), and for every $w_j$, requires the existence of all of its successors, according to their number, and in particular, the existence of $w_{j+1}$ (via $A_i$). Therefore, $\mathcal{A}$ requires the existence of $w_t$. On the other hand, by construction, every word that is assigned to $z$ is in $\mathcal{L}$. Therefore, $\mathfrak{L}(\mathcal{A}) = \{\mathcal{L}\}$.

The size of $A_U$ is linear in $|U|$, and the size of $A_i$ is exponential in $k$ and in $|A|$. □

## 3.3 Realizability of Regular Languages

We now show that every regular language $\mathcal{L}$ is partially ordered. To present the idea more clearly, we begin with a simpler case of prefix-closed regular languages, and then proceed to general regular languages. A prefix-closed regular language $\mathcal{L}$ has a DFA $A$ in which every accepting state is only reachable by accepting states. We use the structure of $A$ to define a relation that partially orders $\mathcal{L}$.

**Theorem 11.** *Let $\mathcal{L}$ be a non-empty prefix-closed regular language. Then $\mathcal{L}$ is partially ordered.*

*Proof.* Let $A = \langle \Sigma, Q, q_0, \delta, F \rangle$ be a DFA for $\mathcal{L}$. Then for every $p,q \in Q$, if $q \in F$ and $q$ is reachable from $p$, then $p \in F$. Let $k$ be the maximal number of transitions from a state $q \in F$ to its neighboring accepting states. We show that $\mathcal{L}$ is $1,k$-ordered. We define a relation $R$ as follows. Since $\mathcal{L}$ is prefix-closed, we have that $\varepsilon \in \mathcal{L}$. We set it to be the minimal element in $R$.

Let $q \in Q$, and let $\{(q,\sigma_1,p_1),\ldots(q,\sigma_m,p_m)\}$ be the set of transitions in $\delta$ from $q$ to accepting states. For every word $w \in \mathcal{L}$ that reaches $q$, we set $(w,w\sigma_1),\ldots(w,w\sigma_m) \in R$. For every word $w \in \mathcal{L}$ that reaches a state $q$ from which there are no transitions to accepting states, we set $(w,w) \in R$.

It holds that the number of successors for every $w \in \mathcal{L}$ is between 1 and $k$. Further, $R \subseteq S$ for the length-lexicographic order $S$ of $\mathcal{L}$.

We construct an NFA $A_R$ for $R$ by replacing every transition labeled $\sigma$ with $\{\sigma_x,\sigma_y\}$ and adding a state $p'$, which is the only accepting state. For every $q \in F$, we add a transition $(q,\{\#_x,\sigma_y\},p')$ for every $(q,\sigma,p) \in \delta$ such that $p \in F$. If $q$ has no transitions to accepting states in $A$, then we add $(q,\{\#_x,\#_y\},p')$. $A_R$ then runs on word assignments $\boldsymbol{w}_{[x\mapsto w][y\mapsto u]}$ such that $u = w\sigma$ for some $\sigma$, such that $w,u \in \mathcal{L}$, or $\boldsymbol{w}_{[x\mapsto w][y\mapsto w]}$ if $w$ cannot be extended to a longer word in $\mathcal{L}$. Therefore, $A_R$ computes $\mathcal{L}$. □

*Remark* 1. The construction in the proof of Theorem 10 is exponential, due to the composition of several automata. In the case of prefix-closed languages, the successors of a word $w \in \mathcal{L}$ are all of the type $w\sigma$. Therefore, it suffices to extend every transition in $A$ to $\{\sigma_{x_1},\sigma_{x_2},\ldots\sigma_{x_k}\}$, and to add a transition from every $q \in F$ to a new accepting state with all letters leading from $q$ to an accepting state. Composed with a single-state DFA for $\varepsilon$, we get an $\exists\forall\exists^k$-NFH for $\{\mathcal{L}\}$, whose size is polynomial in $|A|$.

We now turn to prove the realizability of $\{\mathcal{L}\}$ for every regular language $\mathcal{L}$. The proof relies on a similar technique to that of Theorem 11: a relation that computes $\{\mathcal{L}\}$ requires, for every word $w \in \mathcal{L}$, the existence of a longer word $w' \in \mathcal{L}$. Here, $w'$ is not simply the extension of $w$ by a single letter, but a pumping of $w$ by a single cycle in a DFA for $\mathcal{L}$.

**Theorem 12.** *Let $\mathcal{L}$ be a regular language. Then $\{\mathcal{L}\}$ is partially ordered.*

*Proof.* Let $A = \langle\Sigma,Q,q_0,\delta,F\rangle$ be a DFA for $\mathcal{L}$. We mark by $P$ the set of words that reach accepting states from $q_0$ along a simple path. For a state $q \in Q$, we mark by $C_q$ the set of words that reach $q$ from $q$ along a simple cycle. Note that $P$ and $C_q$ are finite for every $q \in Q$. Let $n = |P|$, and let $m = \Sigma_{q\in Q}|C_q|$. We show that $\mathcal{L}$ is $n,m$-ordered, by defining an appropriate relation $R$.

The set of minimal words in $R$ is $P$. The successors of a word $w \in \mathcal{L}$ are $w$ itself (that is, $R$ is reflexive), and every possible pumping of $w$ by a single simple cycle that precedes all other cycles within the run of $A$ on $w$. That is, for a state $q$ that is reached by a prefix $u$ of $w$ along a simple path, and for a word $c$ read along a simple cycle from $q$ to itself, the word $ucv$ is a successor of $w$ in $R$, where $w = uv$.

To see that the only minimal words in $R$ are $P$, let $w = \sigma_1\sigma_2\cdots\sigma_k \in \mathcal{L}$, and let $r = (q_0,q_1,\ldots q_k)$ be the accepting run of $A$ on $w$. If all states in $r$ are unique, then $w \in P$. Otherwise, we set $w_t = w$, and repeatedly remove simple cycles from $r$: let $j$ be a minimal index for which there exists $j' > j$ such that $q_j = q_{j'}$ and such that $q_{j+1},\ldots q_{j'}$ are unique. We define $w_{i-1} = w_1\cdots w_j w_{j'+1}\cdots w_k$. We repeat this process until we reach a run in which all states are unique, which matches a word $w_0 \in P$. The sequence of words $w_t,w_{t-1},\ldots w_0$ we obtain is such that $(w_i,w_{i+1}) \in R$ for every $i \in [0,t-1]$.

It is easy to see that $R \subseteq S$ for the length-lexicographical order $S$ of $\mathcal{L}$. Additionally, every $w \in \mathcal{L}$ has between 1 and $m$ successors. We now construct an NFA $A_R$ for $R$.

$A_R$ is the union of several components, described next. Let $A_q$ be the DFA obtained from $A$ by setting its only accepting state to be $q$. For every $p \in Q$ and for every $c \in C_p$, we construct an NFA $B_{c,q}$, which pumps a word read along a run that reaches $q$ and traverses $p$, by $c$. The NFA $B_{c,q}$ comprises two copies $A_1$,$A_2$ of $A$, where the copy $q_2$ of $q$ in $A_2$ is the only accepting state. The word $c$ is read between $A_1$ to $A_2$, from $p_1$ and $p_2$.

We construct an NFA $A_{c,q}$ by composing $B_{c,q}$ and $A_q$, and making sure that $B_{c,q}$ reads the same word as $A_q$, pumped by $c$. That is, if $A_q$ reads a word $uv$, where $u$ reaches $p$, then $B_{c,q}$ reads $ucv$. To this end, while $B_{c,q}$ is in $A_1$, the DFA $A_q$ and $B_{c,q}$ both advance on the same letters. When $B_{c,q}$ leaves $A_1$ to read $c$

followed by the suffix $v$ in $A_2$, the composition remembers, via states, the previous (up to) $|c|$ letters read by $A_q$, to make sure that once $B_{c,q}$ finishes reading $uc$, it reads the same suffix $v$ as $A_q$ did. The NFA $A_R$ is then the union of $A_{c,q}$ for every $q \in Q, c \in \bigcup_{p\in Q} C_p$. To accept the reflexive pairs as well, we union all the components with an additional component $A \oplus A$.

The size of every $A_{c,q}$ is exponential in $c$, due to the need to remember the previous $c$ letters. There are exponentially many simple paths and cycles in $A$. Therefore, we have that the size of $A_R$ is exponential in $|A|$. Combined with the exponential blow-up involved in the proof of Theorem 10, we have that an NFH for $\{\mathcal{L}\}$ is doubly-exponential in the $|A|$. □

*Remark* 2. Using automatic structures [17] and relying on the length-lexicographical order $S$, one can prove the existence of an $\exists\forall\exists$-NFH $\mathcal{A}$ for $\{\mathcal{L}\}$, which is smaller and simpler than the one we present in Theorem 12. Indeed, one can phrase the direct successor relation in $\mathcal{L}$ with respect to $S$ using the First Order Logic (FOL) formula $\varphi(x,y) = \mathcal{L}(x) \wedge \mathcal{L}(y) \wedge S(x,y) \wedge \forall(z).(z \neq y) \rightarrow (\neg(S(x,z) \wedge S(z,y)))$. Since $S$ is NFA-realizable, and since every relation expressible by FOL over an automatic structure is regular [17], we have that $\varphi$ is NFA-realizable. We can then construct $\mathcal{A}$, requiring the existence of a minimal word in $\mathcal{L}$ with respect to $S$, together with the requirement of the existence of a successor for every $w \in \mathcal{L}$.

While this construction is polynomial, it does not directly rely on the structure of $A$. Since in this paper we wish to lay the ground for richer realizable fragments, in which relying on the underlying graph structures may be useful, we present it here.

## 4 Context-Free Hypergrammars

We now go beyond regular hyperlanguages, and define and study *context-free hyperlanguages*. We begin with a natural definition for context-free hypergrammars (CFHG), based on the definition of NFH, and then identify a more decidable fragment of CFHG, namely *synchronized CFHG*.

**Definition 13.** *A* context-free hypergrammar *(CFHG) is a tuple $\langle \Sigma, X, V, V_0, P, \alpha \rangle$, where $X$ and $\alpha$ are as in NFH, and where $\hat{G} = \langle \hat{\Sigma}, V, V_0, P \rangle$ is a CFG over the alphabet $\hat{\Sigma} = (\Sigma \cup \{\#\})^X$.*

Definition 1 defines word assignments for NFH, where the #-symbol may only appear at the end of a word. This is naturally enforced by the nature of the underlying NFA. For the most general case of hypergrammars, we consider words in which # can appear anywhere in the word. In Section 4.1 we allow # to occur only at the end of the word. For a word $w \in \Sigma^*$ we define the *set of words* $w\uparrow_\#$ to be the set of all words that are obtained from $w$ by adding #-symbols in arbitrary locations in $w$. For $w \in (\Sigma \cup \{\#\})^*$, we define the *word* $w\downarrow_\#$ to be the word obtained from $w$ by removing all occurrences of #.

The acceptance condition for CFHG is defined with respect to a language $\mathcal{L}$, the underlying CFG $\hat{G}$, the quantification condition $\alpha$, and an assignment $u : X \rightarrow \mathcal{L}$.

1. For $\alpha = \varepsilon$, define $\mathcal{L} \vdash_v (\alpha, \hat{G})$ if $\boldsymbol{w}_u \in \mathcal{L}(\hat{G})$.
2. For $\alpha = \exists x.\alpha'$, define $\mathcal{L} \vdash_u (\alpha, \hat{G})$ if there exist $w \in \mathcal{L}$ and $w_\# \in w\uparrow_\#$ s.t. $\mathcal{L} \vdash_{u[x\mapsto w_\#]} (\alpha', \hat{G})$.
3. For $\alpha = \forall x.\alpha'$, define $\mathcal{L} \vdash_u (\alpha, \hat{G})$ if for every $w \in \mathcal{L}$ there exists $w_\# \in w\uparrow_\#$ s.t. $\mathcal{L} \vdash_{u[x\mapsto w_\#]} (\alpha', \hat{G})$.

When $\alpha$ includes all of $X$, we say that $G$ *derives* $\mathcal{L}$ (or that $G$ accepts $\mathcal{L}$), and denote $\mathcal{L} \in \mathfrak{L}(G)$.

**Definition 14.** *Let $G$ be a CFHG. The* hyperlanguage *of $G$, denoted $\mathfrak{L}(G)$, is the set of all languages that $G$ derives. We denote $G$ as being a $\mathbb{Q}_1\mathbb{Q}_2\ldots\mathbb{Q}_k$-CFHG similarly as with NFH.*

Henceforth we assume that (1) the underlying grammar $\hat{G}$ does not contain variables and rules that derive no terminal words (these can be removed); and (2) there are no rules of the form $v \to \varepsilon$ except for possibly $V_0 \to \varepsilon$. Every CFG can be converted to a CFG that satisfies these conditions [16].

*Example* 3. Consider the robot scenario described in Section 1, and the $\forall x$-CFHG $G_1$ with the rules

$$\begin{aligned} P_1 := V_0 &\to \{c_x\}V_0\{a_x\} \mid \{c_x\}V_1 \\ V_1 &\to \{c_x\}V_1 \mid \{c_x\} \end{aligned}$$

The letters $a$ and $c$ correspond to *action* and *charge*, respectively. Then, $\mathfrak{L}(G_1)$ is the set of all languages in which the robot has enough battery to act.

Consider now the CFHG $G_2 = \langle\{a,c\},\{x_1,x_2\},\{V_0,V_1\},V_0,P_2,\exists x_1\forall x_2\rangle$ where

$$\begin{aligned} P_2 := V_0 &\to \{c_{x_1},c_{x_2}\}V_0\{a_{x_1},a_{x_2}\} \mid \{c_{x_1},c_{x_2}\}V_1\{a_{x_1},\#_{x_2}\} \mid \{c_{x_1},c_{x_2}\}V_1\{a_{x_1},a_{x_2}\} \\ V_1 &\to \{c_{x_1},\#_{x_2}\}V_1\{a_{x_1},\#_{x_2}\} \mid \{c_{x_1},\#_{x_2}\} \mid \{c_{x_1},c_{x_2}\} \end{aligned}$$

We now require that the robot only has one additional unit of charging (unlike in $G_1$). In addition, we require an upper bound (assigned to $x_1$) on the charging and action times. All other words in the language (assigned to $x_2$) correspond to shorter computations.

We now study the nonemptiness and membership problems for CFHGs. When regarding a CFHG as a specification, these correspond to the model-checking and satisfiability problems.

**Theorem 15.** *The nonemptiness problem for $\exists^*$-CFHG is in* P.

*Proof.* According to the semantics of the $\exists$-requirement, an $\exists^*$-CFHG $G$ derives a language $\mathcal{L}$ if $\hat{G}$ accepts a word assignment that corresponds to words in $\mathcal{L}$. Therefore, it is easy to see that $G$ is nonempty iff $\hat{G}$ is nonempty. Since the nonemptiness of CFG is in P [16], we are done. □

**Theorem 16.** *The membership problem for a regular language in an $\exists^*$-CFHG is in* EXPTIME.

*Proof.* Let $A = \langle\Sigma,Q,Q_0,\delta,F\rangle$ be an NFA and let $G$ be a CFHG with $\alpha = \exists x_1 \cdots \exists x_k$. In order to check whether $\mathcal{L}(A) \in \mathfrak{L}(G)$, we need to check whether there exists a subset of $\mathcal{L}(A)$ of size $k$ or less, that can be accepted as a word assignment by $\hat{G}$. Since $\hat{G}$ derives words over $\Sigma \cup \{\#\}$, we first construct the NFA $A \uparrow \#$, that accepts all #-paddings of words in $\mathcal{L}(A)$. We can do so easily by adding a self-loop labeled # to every state in $A$. We then compute $(A \uparrow \#)^{\otimes k}$ to allow different paddings for different words (an exponential construction), intersect the resulting automaton with $\hat{G}$ and test the intersection for nonemptiness. Context-free languages are closed under intersection with regular languages via a polynomial construction. In addition, if the grammar is given in Chomsky Normal Form [7], then checking the emptiness of the intersection is polynomial in the sizes of the grammar and automaton. As the conversation to Chomsky normal form is also polynomial, we get that the entire procedure is exponential, due to the size of $(A \uparrow \#)^{\otimes k}$. □

**Theorem 17.** *The membership problem for a finite language in a CFHG is in* EXPTIME.

*Proof.* Let $\mathcal{L}$ be a finite language and let $G$ be a CFHG with variables $\{x_1,\ldots x_k\}$. Since $\mathcal{L}$ is finite, we can construct every assignment of words in $\mathcal{L}$ to the variables in $G$, and check if it is accepted by $\hat{G}$. Similarly to the proof of Theorem 16, to do so, we use an NFA $A_w$ whose language is the set $w\uparrow_\#$, for every $w \in \mathcal{L}$. For an assignment $v = [x_1 \mapsto w_1]\ldots[x_k \mapsto w_k]$, we construct $\bigotimes_{i=1}^k A_{w_i}$, and check the nonemptiness of its intersection with $\hat{G}$. As in Theorem 16, this procedure is exponential in the length of the words in $\mathcal{L}$ and in $|G|$. Since we can finitely enumerate all assignments, we can check whether the quantification condition $\alpha$ of $G$ is satisfied. Enumerating all assignments amounts to traversing the decision tree dictated by $\alpha$, which is exponential in $|\alpha|$. Therefore, the entire procedure can be done in exponential time in $|G|$ and $\mathcal{L}$. $\square$

**Theorem 18.** *The emptiness problem for $\forall^*$-CFHG and $\exists\forall$-CFHG is undecidable.*

*Proof.* We show reductions from the Post correspondence problem (PCP).A PCP instance is a set of pairs of the form $[a_1,b_1],\dots,[a_n,b_n]$ where $a_i,b_i \in \{a,b\}^*$. The problem is then to decide whether there exists a sequence of indices $i_1\cdots i_m,\ i_j \in [1,n]$, such that $a_{i_1}a_{i_2}\cdots a_{i_m} = b_{i_1}b_{i_2}\cdots b_{i_m}$. For example, consider the instance $\{[a,baa]_1,[ab,aa]_2,[bba,bb]_3\}$. Then, a solution to the PCP is the sequence $3,2,3,1$ since $a_3a_2a_3a_1 = bba\cdot ab\cdot bba\cdot a$ and $b_3b_2b_3b_1 = bb\cdot aa\cdot bb\cdot baa$.

Let $T = \{[a_1,b_1],\dots,[a_n,b_n]\}$ be a PCP instance. Let $G = \langle\{a,b\},\{x_1,x_2\},\{V_0\},V_0,P,\forall x_1\forall x_2\rangle$ be a $\forall^*$-CFHG defined as follows. For every pair $[a_i,b_i] \in T$ we define the words $A_i,B_i \in (\Sigma\cup\{\#\})^*$ obtained from $a_i,b_i$ by padding the shorter of $a_i,b_i$ with #-symbols so that $A_i,B_i$ are of equal length. We define $P$ as follows.

$$P := V_0 \to \{A_{1_{x_1}},B_{1_{x_2}}\}V_0 \mid \cdots \mid \{A_{n_{x_1}},B_{n_{x_2}}\}V_0 \mid \{A_{1_{x_1}},B_{1_{x_2}}\} \mid \cdots \mid \{A_{n_{x_1}},B_{n_{x_2}}\}$$

For a language $\mathcal{L} \in \mathfrak{L}(G)$, it must hold that $\boldsymbol{w}_{[x_1\mapsto u][x_2\mapsto v]} \in \mathcal{L}(\hat{G})$ for every $u,v \in \mathcal{L}$, due to the $\forall\forall$-condition. Let $u \in \mathcal{L}$. Then, in particular, $\boldsymbol{w}_{[x_1\mapsto u][x_2\mapsto u]} \in \mathcal{L}(\hat{G})$. Notice that in this case, $u$ is a solution to $T$. In the other direction, a solution to $T$ induces a word $u = a_{i_1}a_{i_2}\cdots a_{i_m}$ such that $\{u\} \in \mathfrak{L}(G)$. The same reduction holds also for the case of $\exists\forall$, since according to the $\forall$ requirement, one of the word assignments must assign the same word to both variables. $\square$

Note that the proof of Theorem 18 compares between two words in order to simulate PCP. For a single $\forall$-quantifier, the nonemptiness problem is equivalent to that of CFG, and is therefore in P.

The underlying CFG we use in the proof of Theorem 18 is linear, and so the result follows also to asynchronous NFH, that allow #-symbols arbitrarily. This is in line with the results in [3], which shows that the model-checking problem for asynchronous hyperLTL is undecidable.

## 4.1 Synchronous Hypergrammars

As we show in Section 4, the asynchronicity of general CFHG leads to undecidability of most decision problems for them, already for simple quantification conditions. We now introduce *ranked CFHG*, a fragment of CFHG that ensures synchronous behavior. We then prove that ranked CFHG capture exactly the set of *synchronous hyperlanguages*. Intuitively, synchronous hyperlanguages are derived from grammars in which # only appears at the end of the word, similarly to NFH (we say that such a word assignment is *synchronous*). Since CFHG may use non-linear rules, in order to characterize the grammar rules that derive synchronous hyperlanguages, we need to reason about structural properties of the grammar. To this end, we define a *rank* for each variable $v$, which, intuitively, corresponds to word variables for which $v$ derives #-symbols.

*Remark* 3. Before we turn to the definition of ranks of variables and ranked grammars we note on the difference between a definition of grammars which their hyperlanguages are synchronous, as we do in the rest of this section; and the problem of, given some hypergrammar $G$, finding the hyperlanguage $\mathfrak{L}(G_s) \subseteq \mathfrak{L}(G)$ that corresponds to the synchronous sub-hyperlanguage of $G$. Assume that $G$ is over $\Sigma$ and has $k$ quantifiers. Then, the latter can be done by constructing an NFA $A_s$ over $(\Sigma\cup\{\#\})^k$ that

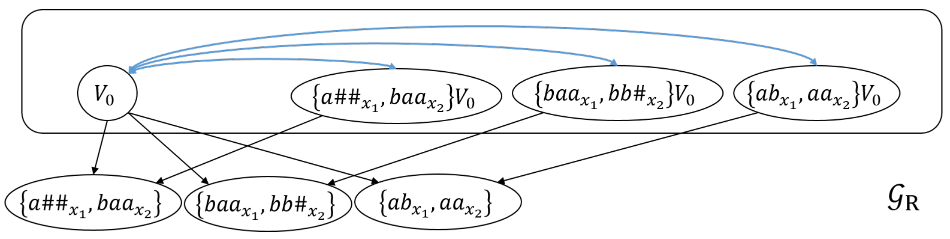
$V_0$
$\{a\#\#_{x_1}, baa_{x_2}\}V_0$
$\{baa_{x_1}, bb\#_{x_2}\}V_0$
$\{ab_{x_1}, aa_{x_2}\}V_0$
$\{a\#\#_{x_1}, baa_{x_2}\}$
$\{baa_{x_1}, bb\#_{x_2}\}$
$\{ab_{x_1}, aa_{x_2}\}$
$\mathcal{G}_R$

Figure 2: The MSSC graph $\mathcal{G}_R$ for the grammar and PCP instance of Example 4 and the Proof of Theorem 18. Blue edges are bidirectional, and the rectangle represents an MSCC.

accepts all words in which # appears only at the end of words. The intersection of $A_s$ and $G$ results in the grammar $G_s$, whose language is a subset of that of $G$. We approach a different problem, namely defining a fragment of grammars that accept exactly the class of synchronous hyperlanguages.

In order to define the ranks of variables, we use the *rule graph* $\mathcal{G}$, defined as follows. The set of vertices of $\mathcal{G}$ is $V \cup W$, where $W = \{\gamma \in (\hat{\Sigma} \cup V)^* \mid \exists v \in V. v \to \gamma \in P\}$ is the set of sequences appearing on the right side of one of the grammar rules. The set of edges $E$ of $\mathcal{G}$ is $E = E_L \cup E_R$ where

$$E_L = \{\langle v,w\rangle \mid v \to w \in P\} \cup \{\langle w,v\rangle \mid w = v\gamma\}$$
$$E_R = \{\langle v,w\rangle \mid v \to w \in P\} \cup \{\langle w,v\rangle \mid w = \gamma v\}$$

We partition $\mathcal{G}$ into maximal strongly connected components (MSCCs) with respect to each type of edges ($E_L$ and $E_R$), resulting in two directed a-cyclic graphs $\mathcal{G}_L$ and $\mathcal{G}_R$. The vertices of $\mathcal{G}_d$ for $d \in \{L,R\}$ are the MSCCs according to $E_d$, and there is an edge $C_1^d \to C_2^d$ iff there exist $u,u' \in (V \cup W)$ such that $u \in C_1^d, u' \in C_2^d$ and $\langle u,u'\rangle \in E_d$. Note that every terminal word is a singleton MSCC in both graphs.

*Example* 4. Figure 2 presents $\mathcal{G}_R$ for $G$ of the proof of Theorem 18, and the PCP instance $\{[a,baa]_1, [ab,aa]_2, [bba,bb]_3\}$, with the concrete derivation rules:

$$V_0 \to \{a\#\#_{x_1}, baa_{x_2}\}V_0 \mid \{ab_{x_1}, aa_{x_2}\}V_0 \mid \{baa_{x_1}, bb\#_{x_2}\}V_0 \mid$$
$$\{a\#\#_{x_1}, baa_{x_2}\} \mid \{ab_{x_1}, aa_{x_2}\} \mid \{baa_{x_1}, bb\#_{x_2}\}$$

We now define the *left ranks* and *right ranks* of synchronous words, variables and sequences.

1. **Ranks of terminal synchronous words**. The rank of a letter $\hat{\sigma} = \{\sigma_{1_{x_1}}, \ldots \sigma_{n_{x_n}}\} \in \hat{\Sigma}$ is $t(\hat{\sigma}) = \{x_i \mid \sigma_{i_{x_i}} = \#\}$. The left rank of $\hat{w}$ is $\mathbf{L}(\hat{w}) = t(\hat{\sigma}_1)$, and its right rank is $\mathbf{R}(\hat{w}) = t(\hat{\sigma}_n)$, where $\hat{\sigma}_1$ and $\hat{\sigma}_n$ are the first and last letters of $\hat{w}$, respectively.

2. **Inductive definition for variables and sequences**. Let $d \in \{L,R\}$, and let $C_1^d \to C_2^d$ in $\mathcal{G}_d$ such that $\mathbf{d}(u')$ is defined for every $u' \in C_2^d$ and $\mathbf{d} \in \{\mathbf{L},\mathbf{R}\}$. Let $\gamma \in (\hat{\Sigma} \cup V)^*$, $\sigma \in \hat{\Sigma}$, and $v \in V$.
   - For $u \in C_1^d \in \mathcal{G}_d$ such that $u = \sigma\gamma$ we define $\mathbf{L}(u) = l(u) = \mathbf{L}(\sigma)$.
   - For $u \in C_1^d \in \mathcal{G}_d$ such that $u = \gamma\sigma$ we define $\mathbf{R}(u) = r(u) = \mathbf{R}(\sigma)$.
   - For $u \in C_1^L \in \mathcal{G}_L$ such that $u = v\gamma$ we define $l(u) = \bigcup_{C_1^L \to C_2^L} \bigcap_{u' \in C_2^L} \mathbf{L}(u')$.
   - For $u \in C_1^R \in \mathcal{G}_R$ such that $u = \gamma v$ we define $r(u) = \bigcup_{C_1^R \to C_2^R} \bigcup_{u' \in C_2^R} \mathbf{R}(u')$.

   Now, for each $u = v\gamma \in C_1^L$ we define $\mathbf{L}(u) = \bigcap_{u' \in C_1^L} l(u')$, and for each $u = \gamma v \in C_1^R$ we define $\mathbf{R}(u) = \bigcup_{u' \in C_1^R} r(u')$.

Note that this process is guaranteed to terminate, since we traverse both graphs in reverse topological order. Therefore, at the end of the process, $\mathbf{L}(u)$ and $\mathbf{R}(u)$ are defined for every $u \in V \cup W$.

We define *ranked CFGs* to be CFGs in which for every rule $v \to \gamma_1 \cdots \gamma_n$ for $\gamma_i \in (\hat{\Sigma} \cup V)$, it holds that $\mathbf{R}(\gamma_i) \subseteq \mathbf{L}(\gamma_{i+1})$. Intuitively, this means that $\gamma_i$ may not produce # to its right, if $\gamma_{i+1}$ can produce $\sigma \neq \#$ to its left, leading to unsynchronous derivation. A CFHG $G$ is *ranked* if $\hat{G}$ is ranked.

*Example* 5. Consider $G$ of Example 4 and $\mathcal{G}_R$ of Figure 2. The graph $\mathcal{G}_L$ is similar to $\mathcal{G}_R$, with no edges back to $V_0$, (and thus without the rectangle MSCC). We compute some of the ranks for $G$:

$$\mathbf{L}(\{a\#\#_{x_1}, baa_{x_2}\}) = \mathbf{L}(\{baa_{x_1}, bb\#_{x_2}\}) = \mathbf{L}(\{ab_{x_1}, aa_{x_2}\}) = \emptyset \quad \mathbf{L}(V_0) = \emptyset$$
$$\mathbf{R}(\{a\#\#_{x_1}, baa_{x_2}\}) = \{x_1\} \quad \mathbf{R}(\{baa_{x_1}, bb\#_{x_2}\}) = \{x_2\} \quad \mathbf{R}(\{ab_{x_1}, aa_{x_2}\}) = \emptyset \quad \mathbf{R}(V_0) = \{x_1, x_2\}$$

$G$ is not ranked, since for the rule $V_0 \to \{a\#\#_{x_1}, baa_{x_2}\}V_0$, it holds that $\mathbf{R}(\{a\#\#_{x_1}, baa_{x_2}\}) \not\subseteq \mathbf{L}(V_0)$.

*Example* 6. The following CFHG $G_r = \langle \{a,b\}, \{x_1,x_2\}, \{V_0,V_1\}, V_0, P, \forall x_1 \exists x_2 \rangle$ is ranked, where $P$ is:

$$\begin{aligned} P := {} & V_0 \to V_1 V_2 \\ & V_1 \to \{a_{x_1}, a_{x_2}\} V_1 \{b_{x_1}, b_{x_2}\} \mid \{ab_{x_1}, ab_{x_2}\} \\ & V_2 \to V_2 \{\#_{x_1}, b_{x_2}\} \mid \{\#_{x_1}, b_{x_2}\} \end{aligned}$$

$G_r$ accepts all languages in which for every word of the type $a^n b^n$ there exits a word with more $b$'s, that is, there exists $a^n b^m$ for $m > n$.

The ranks of $G_r$, as shown below, demonstrate that $G_r$ is indeed ranked.

$$\mathbf{L}(\{\#_{x_1}, b_{x_2}\}) = \mathbf{R}(\{\#_{x_1}, b_{x_2}\}) = \{x_1\} \quad \mathbf{L}(\{ab_{x_1}, ab_{x_2}\}) = \mathbf{R}(\{ab_{x_1}, ab_{x_2}\}) = \emptyset$$
$$\mathbf{L}(\{a_{x_1}, a_{x_2}\}V_1\{b_{x_1}, b_{x_2}\}) = \mathbf{R}(\{a_{x_1}, a_{x_2}\}V_1\{b_{x_1}, b_{x_2}\}) = \mathbf{R}(V_1) = \mathbf{L}(V_1) = \emptyset$$
$$\mathbf{R}(V_2\{\#_{x_1}, b_{x_2}\}) = \mathbf{L}(V_2\{\#_{x_1}, b_{x_2}\}) = \mathbf{R}(V_2) = \mathbf{L}(V_2) = \{x_1\}$$
$$\mathbf{R}(V_0) = \emptyset \quad \mathbf{L}(V_0) = \{x_1\}$$

**Definition 19.** *$\mathfrak{L}$ is a* synchronous context-free hyperlanguage *if there exists a CFHG $G$ for $\mathfrak{L}$ in which $\hat{G}$ only derives synchronous word assignments.*

**Theorem 20.** *A hyperlanguage $\mathfrak{L}$ is derived by a ranked CFHG iff $\mathfrak{L}$ is synchronous context-free.*

In order to prove Theorem 20, we use the following claims.

**Claim 21.** *Let $G = \langle \Sigma, X, V, V_0, P, \alpha \rangle$ be a ranked CFHG. Then, for every word $\gamma = \gamma_1 \cdots \gamma_n \in (\hat{\Sigma} \cup V)^*$, if there exists $v \in V$ such that $v \Rightarrow^* \gamma$, then $\mathbf{R}(\gamma_i) \subseteq \mathbf{L}(\gamma_{i+1})$ for all $i \in [1, n-1]$.*

**Claim 22.** *Let $G = \langle \Sigma, X, V, V_0, P, \alpha \rangle$ be a (possibly not ranked) CFHG with $|X| = k$, and let $v \in V$.*

1. *For every $j \in [1,k] \setminus \mathbf{L}(v)$ there exists $w \in \hat{\Sigma}^*$ such that $v \Rightarrow^* w$ and $j \notin \mathbf{L}(w)$.*
2. *For every $j \in \mathbf{R}(v)$ there exists $w \in \hat{\Sigma}^*$ such that $v \Rightarrow^* w$ and $j \in \mathbf{R}(w)$.*

*Proof of Theorem 20.* Let $\mathfrak{L}$ be a context-free language that is accepted by a ranked grammar $G$. According to Claim 21, for every word $w = w_1 \cdots w_n \in \hat{\Sigma}^*$ such that $V_0 \Rightarrow^* w$, it holds that $\mathbf{R}(w_i) \subseteq \mathbf{L}(w_{i+1})$ for $i \in [1, n-1]$. That is, # is allowed to only appear at the end of words, and so $\hat{G}$ only derives synchronous word assignments.

For the other direction, let $\mathfrak{L}$ be a synchronous context-free hyperlanguage, and let $G$ be a CFHG for $\mathfrak{L}$ that only derives synchronous word assignments. Assume by way of contradiction that $G$ is not ranked. Then, there exists some rule $v \to \gamma_1 \cdots \gamma_n \in P$ where $\gamma_i \in (\hat{\Sigma} \cup V)$ such that $\mathbf{R}(\gamma_i) \not\subseteq \mathbf{L}(\gamma_{i+1})$ for some

$i \in [1,n]$. Recall that we assume that all rules are reachable and that every variable can derive a terminal word. Consider a derivation sequence $V_0 \Rightarrow^* \beta v \beta' \Rightarrow \beta \gamma_1 \cdots \gamma_n \beta'$. Then, there exist $w, w_i, w_{i+1}, w' \in \hat{\Sigma}^*$ such that $\gamma_i \Rightarrow^* w_i$, $\gamma_{i+1} \Rightarrow^* w_{i+1}$ and $V_0 \Rightarrow^* w w_i w_{i+1} w'$; and due to claim 22, for some $j \in \mathbf{R}(\gamma_i) \setminus \mathbf{L}(\gamma_{i+1})$, it holds that $j \in \mathbf{R}(w_i) \setminus \mathbf{L}(w_{i+1})$. Hence, $w_i$ ends with # in some location which is followed by a letter in $w_{i+1}$, and so the word assignment $w w_i w_{i+1} w'$ is not synchronous, a contradiction. $\square$

We therefore term ranked grammars *syncCFHG*. Given a CFHG $G$, deciding whether it is ranked amounts to constructing the graph $\mathcal{G}$ and traversing the topological sorting of its MSCC graph in reverse order in order to compute all ranks, and finally checking that all grammar rules of $G$ comply to the rank rules. All these steps can be computed in polynomial time. We now show that syncCFHG is more decidabile than CFHG.

**Theorem 23.** *The nonemptiness problem for $\forall^*$-syncCFHG and $\exists\forall^*$-syncCFHG is in P.*

*Proof.* Let $G$ be a syncCFHG. Since universal quantification is closed under subsets, it holds that if $\mathcal{L} \in \mathfrak{L}(G)$, then $\mathcal{L}' \in \mathfrak{L}(G)$ for every $\mathcal{L}' \subseteq \mathcal{L}$. Therefore, it suffices to check whether there exists a singleton $\mathcal{L}$ such that $\mathcal{L} \in \mathfrak{L}(G)$. Therefore, we consider only word assignments of the form $\boldsymbol{w} = \boldsymbol{w}_{[x_1 \mapsto w] \cdots [x_k \mapsto w]}$ for some $w \in (\Sigma \cup \{\#\})^*$. Notice that $\boldsymbol{w}$ has a single representation, since # may not appear arbitrarily. We construct a syncCFHG $G'$ by restricting $\hat{G}$ to the alphabet $\bigcup_{\sigma \in \Sigma} \{\sigma\}^X$, that is, all variables are assigned the same letter. All rules over other alphabet letters are eliminated. Since elimination of rules cannot induce asynchronization, $G'$ is synchronous.

Now, for a singleton language $\{w\}$, we have $\{w\} \in \mathfrak{L}(G)$ iff $\{w\} \in \mathfrak{L}(G')$. Therefore, it suffices to check the nonemptiness of $\mathfrak{L}(G')$, which amounts to checking the nonemptiness of $\hat{G}'$.

The proof holds also for the case of $\exists\forall^*$-syncCFHG. Indeed, an $\exists\forall^*$-syncCFHG $G$ is nonempty iff it derives a singleton hyperlanguage. This, since in a language derived by $G$, a word $w$ that is assigned to the variable under $\exists$ must also be assigned to all variables under $\forall$ in one of the word assignments derived by $\hat{G}$, which in turn fulfills the requirements for deriving $\{w\}$. Since $G$ is synchronous, it suffices to restrict the alphabet to homogeneous letters and check for nonemptiness, as with $\forall^*$. $\square$

In the *regular membership problem*, we ask whether a regular language $\mathcal{L}$ can be derived by a synchCFHG $G$. This problem is decidable for NFH [5]. For $\mathcal{L} = \Sigma^*$ and a $\forall$-CFHG $G$, the question amounts to checking the universality of $\hat{G}$, which is undecidable [2]. Therefore, we have the following.

**Theorem 24.** *The regular membership problem for $\forall^*$-syncCFHG grammars is undecidable.*

*Remark* 4. Membership of a finite language $\mathcal{L}$ in a CFHG with any quantification condition is decidable already for general CFHG (Theorem 17), with exponential complexity. For syncCFHG, we can reduce the complexity by checking membership of word assignments instead. This, since we only need to consider synchronous words, which have a single representation. Since checking membership is polynomial in the size of the word (for a grammar of fixed size) [16], every such test is then polynomial. Since we may still need to traverse all possible word assignment, the complexity is exponential in the length of the quantification condition, but is polynomial in $|G|$ and the size of the words in $\mathcal{L}$.

*Remark* 5. For regular languages and $\exists^*$-CFHG, synchronization does not avoid the composition of automata, and we use a construction similar to the one of Theorem 16.

We now show that synchronicity does not suffice for deciding nonemptiness of $\exists^*\forall^*$-syncCFHG.

**Theorem 25.** *The nonemptiness problem for $\exists^*\forall^*$-syncCFHG is undecidable.*

*Proof.* We reduce from PCP. Let $T = \{[a_1,b_1],\dots,[a_n,b_n]\}$ be a PCP instance over $\{a,b\}$, and let

$$G = \langle \{a,b,c\} \cup [1,n], \{x_1,x_2,x_3\}, \{V_0,V_1,V_2\}, V_0, P, \exists x_1 \exists x_2 \forall x_3 \rangle$$

be a CFHG where $P$ is defined as follows.

$$\begin{aligned} &P := V_0 \to V_1 \mid V_2 \\ &V_1 \to \{a_{i_{x_1}}, {c^{|a_i|}}_{x_2}, a_{i_{x_3}}\}V_1\{i_{x_1}, c_{x_2}, i_{x_3}\} \mid \{a_{i_{x_1}}, {c^{|a_i|}}_{x_2}, a_{i_{x_3}}\}\{i_{x_1}, c_{x_2}, i_{x_3}\} \;\; \forall i \in [1,n] \\ &V_2 \to \{b_{i_{x_1}}, {c^{|b_i|}}_{x_2}, {c^{|b_i|}}_{x_3}\}V_2\{i_{x_1}, c_{x_2}, c_{x_3}\} \mid \{b_{i_{x_1}}, {c^{|b_i|}}_{x_2}, {c^{|b_i|}}_{x_3}\}\{i_{x_1}, c_{x_2}, c_{x_3}\} \;\; \forall i \in [1,n] \end{aligned}$$

Since none of the rules include the #-symbol, $G$ is indeed a syncCFHG. Now, if there exists $\mathcal{L} \in \mathfrak{L}(G)$, then there exist $w \in \{a,b\}^* \cdot [1,n]^*$ and $w_c \in \{c\}^*$ both in $\mathcal{L}$, such that for every $w' \in \mathcal{L}$, we have $V_0 \Rightarrow^* \boldsymbol{w}_{[x_1 \mapsto w][x_1 \mapsto w_c][x_3 \mapsto w']}$. In particular, for $w' = w$, we have $V_0 \Rightarrow^* \boldsymbol{w}_{[x_1 \mapsto w][x_1 \mapsto w_c][x_3 \mapsto w]}$. Since $V_2$ only derives words of the form $c^k$ in $x_3$, the derivation of $\boldsymbol{w}_{[x_1 \mapsto w][x_2 \mapsto w_c][x_3 \mapsto w]}$ is of the form $V_0 \Rightarrow V_1 \Rightarrow^* \boldsymbol{w}_{[x_1 \mapsto w][x_2 \mapsto w_c][x_3 \mapsto w]}$. In addition, all words in $\{c\}^*$ can only be assigned to $x_3$ if derived from $V_2$, thus we have $V_0 \Rightarrow V_2 \Rightarrow^* \boldsymbol{w}_{[x_1 \mapsto w][x_2 \mapsto w_c][x_3 \mapsto w_c]}$. Denote $w = w_1 w_2$ where $w_1 \in \{a,b\}^*, w_2 \in [1,n]$. Then, $w$ encodes a solution to $T$, where $w_1$ is the string obtained from $a_i$ (and $b_i$), and $w_2$ is the sequence of indices.

For the other direction, a solution to $T$ encoded by a string $w_1$ and sequence of indices $w_2$ corresponds to the language $\{w_1 w_2, c^{|w_1 w_2|}\}$ that is accepted by $G$. □

The nonemptiness problem for $\forall^*\exists^*$-NFH is undecidable [5]. Therefore, this is also the case for syncCFHG, and for general CFHG.

## 5 Discussion and Future Work

We have studied the realizability problem for regular hyperlanguages, focusing on the case of singleton hyperlanguages. We have shown that simple quantification conditions cannot realize this case. We have defined ordered and partially-ordered languages, for which we can construct hyperautomata that enumerate the language by order. We have shown that all regular languages are partially ordered. Since regular hyperlanguages are closed under union [5], the result extends to a finite hyperlanguage containing regular languages. Naturally, there are richer cases one can consider. For an infinite hyperlanguage $\mathfrak{L}$, some characterization on the elements of $\mathfrak{L}$ would need to be defined in order to explore its realizability. We plan on pursuing this direction as future work. Another related direction is finding techniques for proving unrealizability for certain quantification conditions, for various types of hyperlanguages.

In the second part of the paper we have studied the natural extension of context-free grammars to handle context-free hyperlanguages. Here, we have shown that beyond the inherent undecidability of some decision problems for hypergrammars, some undecidability properties stem from the asynchronous nature of these hypergrammars. We have then defined a synchronous fragment of context-free hyperlanguages, and defined a fragment of context-free grammars which exactly captures this fragment. The result retains some of the decidability properties of context-free grammars. As a future direction, we plan to study the realizability problem for CFHG and syncCFHG. Due to the limited closure properties of CFG, this is expected to be more challenging than for NFH. Another possible future direction is studying the entire Chomsky hierarchy for hyperlanguages, and finding fragments of the extensions to hyperlanguages that conserve the properties of these models for standard languages.

**Acknowledgements.** We thank the anonymous reviewer for suggesting the elegant construction mentioned in Remark 2.

## References

[1] B. Alpern & F.B. Schneider (1985): *Defining Liveness*. *Information Processing Letters*, pp. 181–185, doi:10.1016/0020-0190(85)90056-0.

[2] Brenda S. Baker & Ronald V. Book (1974): *Reversal-Bounded Multipushdown Machines*. *J. Comput. Syst. Sci.* 8(3), pp. 315–332, doi:10.1016/S0022-0000(74)80027-9.

[3] Jan Baumeister, Norine Coenen, Borzoo Bonakdarpour, Bernd Finkbeiner & César Sánchez (2021): *A Temporal Logic for Asynchronous Hyperproperties*. In Alexandra Silva & K. Rustan M. Leino, editors: *Computer Aided Verification - 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part I*, *Lecture Notes in Computer Science* 12759, Springer, pp. 694–717, doi:10.1007/978-3-030-81685-8_33.

[4] Borzoo Bonakdarpour, César Sánchez & Gerardo Schneider (2018): *Monitoring Hyperproperties by Combining Static Analysis and Runtime Verification*. In Tiziana Margaria & Bernhard Steffen, editors: *Leveraging Applications of Formal Methods, Verification and Validation. Verification - 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, Proceedings, Part II*, *Lecture Notes in Computer Science* 11245, Springer, pp. 8–27, doi:10.1007/978-3-030-03421-4_2.

[5] Borzoo Bonakdarpour & Sarai Sheinvald (2021): *Finite-Word Hyperlanguages*. In Alberto Leporati, Carlos Martín-Vide, Dana Shapira & Claudio Zandron, editors: *Language and Automata Theory and Applications - 15th International Conference, LATA 2021, Milan, Italy, March 1-5, 2021, Proceedings*, *Lecture Notes in Computer Science* 12638, Springer, pp. 173–186, doi:10.1007/978-3-030-68195-1_17.

[6] Ahmed Bouajjani, Rachid Echahed & Riadh Robbana (1994): *Verification of Nonregular Temporal Properties for Context-Free Processes*. In Bengt Jonsson & Joachim Parrow, editors: *CONCUR '94, Concurrency Theory, 5th International Conference, Uppsala, Sweden, August 22-25, 1994, Proceedings*, *Lecture Notes in Computer Science* 836, Springer, pp. 81–97, doi:10.1007/978-3-540-48654-1_8.

[7] Noam Chomsky (1959): *On Certain Formal Properties of Grammars*. *Inf. Control.* 2(2), pp. 137–167, doi:10.1016/S0019-9958(59)90362-6.

[8] Edmund M. Clarke, Orna Grumberg, Daniel Kroening, Doron A. Peled & Helmut Veith (2018): *Model checking, 2nd Edition*. MIT Press. Available at `https://mitpress.mit.edu/books/model-checking-second-edition`.

[9] Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe & César Sánchez (2014): *Temporal Logics for Hyperproperties*. In Martín Abadi & 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*, *Lecture Notes in Computer Science* 8414, Springer, pp. 265–284, doi:10.1007/978-3-642-54792-8_15.

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

[11] Norine Coenen, Bernd Finkbeiner, Christopher Hahn & Jana Hofmann (2019): *The Hierarchy of Hyperlogics*. In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, IEEE, pp. 1–13, doi:10.1109/LICS.2019.8785713.

[12] Bernd Finkbeiner, Lennart Haas & Hazem Torfah (2019): *Canonical Representations of k-Safety Hyperproperties*. In: *32nd IEEE Computer Security Foundations Symposium, CSF 2019, Hoboken, NJ, USA, June 25-28, 2019*, IEEE, pp. 17–31, doi:10.1109/CSF.2019.00009.

[13] Bernd Finkbeiner & Martin Zimmermann (2017): *The First-Order Logic of Hyperproperties*. In Heribert Vollmer & Brigitte Vallée, editors: *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, *LIPIcs* 66, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 30:1–30:14, doi:10.4230/LIPIcs.STACS.2017.30.

[14] Wladimir Fridman & Bernd Puchala (2014): *Distributed Synthesis for Regular and Contextfree Specifications*. *Acta Informatica* 51(3-4), pp. 221–260, doi:10.1007/s00236-014-0194-x.

[15] Ohad Goudsmid, Orna Grumberg & Sarai Sheinvald (2021): *Compositional Model Checking for Multi-properties*. In Fritz Henglein, Sharon Shoham & Yakir Vizel, editors: *Verification, Model Checking, and Abstract Interpretation - 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17-19, 2021, Proceedings*, *Lecture Notes in Computer Science* 12597, Springer, pp. 55–80, doi:10.1007/978-3-030-67067-2_4.

[16] John E. Hopcroft, Rajeev Motwani & Jeffrey D. Ullman (2001): *Introduction to Automata Theory, Languages, and Computation, 2nd Edition*. Addison-Wesley series in computer science, Addison-Wesley-Longman.

[17] Bakhadyr Khoussainov & Anil Nerode (1994): *Automatic Presentations of Structures*. In Daniel Leivant, editor: *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13-16 October 1994*, *Lecture Notes in Computer Science* 960, Springer, pp. 367–392, doi:10.1007/3-540-60178-3_93.

[18] Adrien Pommellet & Tayssir Touili (2018): *Model-Checking HyperLTL for Pushdown Systems*. In María-del-Mar Gallardo & Pedro Merino, editors: *Model Checking Software - 25th International Symposium, SPIN 2018, Malaga, Spain, June 20-22, 2018, Proceedings*, *Lecture Notes in Computer Science* 10869, Springer, pp. 133–152, doi:10.1007/978-3-319-94111-0_8.

[19] Markus N. Rabe (2016): *A Temporal Logic Approach to Information-flow Control*. Ph.D. thesis, Saarland University. Available at `http://scidok.sulb.uni-saarland.de/volltexte/2016/6387/`.

[20] Moshe Y. Vardi (1995): *An Automata-Theoretic Approach to Linear Temporal Logic*. In Faron Moller & Graham M. Birtwistle, editors: *Logics for Concurrency - Structure versus Automata (8th Banff Higher Order Workshop, Banff, Canada, August 27 - September 3, 1995, Proceedings)*, *Lecture Notes in Computer Science* 1043, Springer, pp. 238–266, doi:10.1007/3-540-60915-6_6.

[21] Moshe Y. Vardi & Pierre Wolper (1986): *An Automata-Theoretic Approach to Automatic Program Verification (Preliminary Report)*. In: *Proceedings of the Symposium on Logic in Computer Science (LICS '86), Cambridge, Massachusetts, USA, June 16-18, 1986*, IEEE Computer Society, pp. 332–344.

[22] Moshe Y. Vardi & Pierre Wolper (1994): *Reasoning About Infinite Computations*. *Inf. Comput.* 115(1), pp. 1–37, doi:10.1006/inco.1994.1092.

[23] Yu Wang, Mojtaba Zarei, Borzoo Bonakdarpour & Miroslav Pajic (2019): *Statistical Verification of Hyperproperties for Cyber-Physical Systems*. *ACM Trans. Embed. Comput. Syst.* 18(5s), pp. 92:1–92:23, doi:10.1145/3358232.

## A Appendix

### A.1 Proof of Theorem 10

**Theorem 10 (restated)** Let $\mathcal{L}$ be $m,k$-ordered. Then $\{\mathcal{L}\}$ is $\exists^m\forall\exists^k$-realizable.

*Proof.* We present some additional constructions used in our proof. Let $A$ be an NFA over $(\Sigma\cup\{\#\})^X$, and let $x_i\in X$. We define $A\downarrow x_i$ to be the NFA over $\Sigma\cup\{\#\}$ obtained from $A$ by projecting only the letters assigned to $x_i$. That is, for every transition $(q,\{\sigma_{1_{x_1}},\dots\sigma_{k_{x_k}}\},p)$ in $A$, there is a transition $(q,\sigma_i,p)$ in $A\downarrow x_i$.

Let $A_1=\langle(\Sigma\cup\{\#\})^X,Q,q_0,\delta_1,F_1\rangle$ and $A_2\uparrow^{\#}=\langle\Sigma,P,p_0,\delta_2,F_2\rangle$ We denote by $A_1\cap_x A_2$ the NFA over $(\Sigma\cup\{\#\})^X$ that is formed by the product construction of $A_1$ and $A_2$ with respect to $x$. That is, for every $(q,S\cup\{\sigma_x\},q')\in\delta_1$ and $(p,\sigma,p')\in\delta_2$, we have $((q,p),S\cup\{\sigma_x\},(q',p'))$. The set of accepting states is $F_1\times F_2$.

For a sequence of variables $(x_1,\dots,x_k)$ over $X$, we define $A[(x_1,\dots x_k)\rightarrow(y_1,\dots,y_k)]$ to be the NFA obtained from $A$ by renaming every $x_i$ in the sequence by $y_i$, that is, by replacing every occurrence of a letter $\sigma_{x_i}$ in $A$ with $\sigma_{y_i}$.

Let $A_R=\langle(\Sigma\cup\{\#\})^{\{x,y\}},Q,q_0,\delta,F\rangle$ be a DFA that computes $\mathcal{L}$. We construct an $\exists^m\forall\exists^k$-NFH $\mathcal{A}$ for $\mathcal{L}$, as follows. The quantification condition of $\mathcal{A}$ is $\exists x_1\cdots\exists x_m\forall z\exists y_1\cdots\exists y_k$. The $x$-variables are to be assigned $u_1,\dots u_m$, the $m$ minimal words of $R$. To this end, for every $u\in\{u_1,\dots u_m\}$, let $A_u$ be a DFA whose language is $\{u_i\}$, and we set $A_U=\bigotimes_{i=1}^m A_{u_i}[x_1,\dots x_m]$.

The underlying NFA $\hat{\mathcal{A}}$ of $\mathcal{A}$ comprises an NFA $A_i$ For every $1\le i\le k$. Let $\mathcal{L}_i$ be the set of words in $\mathcal{L}$ that have exactly $i$ successors. Intuitively, $A_i$ requires, for every word $w\in\mathcal{L}_i$ that is assigned to $z$, the existence of the $i$ successors of $w$.

We first construct an NFA $A_{R,i}$ whose language is $\{w|w\in\mathcal{L}_j,j\le i\}$. That is, the set of words that have at most $i$ successors. Let $S_i=\{\{i_1,i_2\}|i_{1,2}\in[1,i+1],i_1\ne i_2\}$. To compute $A_{R,i}$, we construct an NFA $B_{R,i}$ which, intuitively, follows the composition of size $i+1$ of $A_R$, while keeping track of the number of different words that are the successors of a single word $w$. Once it is found that there are more than $i$ different words that are matched with $w$, the run may accept. Then we use the complementation of $B_{R,i}$ to construct $A_{R,i}$.

The states of $B_{R,i}$ are of the type $\langle\langle q_1,\dots q_{i+1}\rangle,S\rangle$, where $S\subseteq S_i$. The initial state of $B_{R,i}$ is $\langle\langle q_0\rangle^k,\emptyset\rangle$. For a state $s=\{\langle q_1,\dots q_{i+1}\rangle,S\}$ and $\sigma\in\Sigma$, we add a transition $\langle s,\{\sigma_x,\sigma_{1_{x_1}},\dots\sigma_{i_{x_i}}\},s'\rangle$, where $s'=\langle\langle p_1,p_2,\dots p_{i+1}\rangle,S'\rangle$, such that for every $j\in[1,i+1]$, it holds that $\langle q_j,\{\sigma_x,\sigma_{j_y}\},p_j\rangle\in\delta$ (of $A_R$), and where $S'=S\cup\{\{i_1,i_2\}|\sigma_{i_1}\ne\sigma_{i_2}\}$. We pad by # the ends of words assigned to any of the variables when needed. Notice that a state with $S=S_i$ is reachable only via a word assignment in which $x$ is assigned a word $w$, and $x_1,\dots x_{i+1}$ are all assigned different words $u_1,\dots u_{i+1}$ such that $(w,u_i)\in R$ for every $j\in[1,i+1]$. The set of accepting states of $B_{R,i}$ is $\{\langle\langle q_1,\dots q_{i+1}\rangle,S_i\rangle|q_j\in F\ \forall j\in[1,i]\}$. Therefore, $B_{R,i}\downarrow x$ is $\{w|w\in\mathcal{L}_j,j>i\}$. Finally, we set $A_{R,i}=\overline{B_{R,i}\downarrow x}\cap A_R\downarrow x$, which accepts all words that have at most $i$ successors in $R$.

Using $A_{R,i}$ we proceed to construct $A_i$. Recall that $B_{R,i-1}$ represents the set of words in $\mathcal{L}_j$ for $j\ge i$, each along with all combinations of $i$ of their successors. The accepting runs are those in which all $i$ successors are different. Hence, we can use $B_i=B_{R,i-1}\cap_x A_{R,i}[(x,x_1,\dots x_i)\rightarrow(z,y_1,\dots y_i)]$ to represent exactly the words in $\mathcal{L}_i$ (assigned to $z$), along with all their successors (assigned to $y_1,\dots y_i$. For $i=1$, we use $A_{R,1}\cap_x A_R$: in this case there is a single successor).

Finally, we set $A_i=A_U\otimes B_i$, and assign $y_{i+1},\dots y_k$ equally to $z$ in each transition. We then have that in a word assignment that is accepted by $A_i$, the variables $x_1,\dots x_m$ are assigned the $m$ minimal words, $z$

is assigned some word $w$ in $\mathcal{L}_i$, the variables $y_1,\dots y_i$ are assigned all $i$ successors of $w$, and the extra $y$ variables are assigned equally to $z$.

Since $R \subseteq S$ and $R$ has minimal elements, for every word $w \in \mathcal{L}$ there exists a sequence $w_0, w_1, \dots w_t$ such that $w_t = w$, and $w_0$ is minimal, and $R(w_i, w_{i+1})$ for every $i \in [0, t-1]$. The NFH $\mathcal{A}$ requires $w_0$, and for every $w_i$, requires the existence of all of its successors, according to their number, and in particular, the existence of $w_{i+1}$. Therefore, $\mathcal{A}$ requires the existence of $w_t$. On the other hand, by construction, every word that is assigned to $z$ is in $\mathcal{L}$. Therefore, $\mathfrak{L}(\mathcal{A}) = \{\mathcal{L}\}$. □

## A.2 Proofs for Ranked Grammars

**Claim 21 (restated)** Let $G = \langle \Sigma, X, V, V_0, P, \alpha \rangle$ be a ranked CFHG. Then, for every word $\alpha = \alpha_1 \cdots \alpha_n \in (\hat{\Sigma} \cup V)^*$, if there exists $v \in V$ such that $v \Rightarrow^* \alpha$, then $\mathbf{R}(\alpha_i) \subseteq \mathbf{L}(\alpha_{i+1})$ for all $i \in [1, n-1]$.[3]

*Proof.* Assume $v \Rightarrow^m \alpha$. We prove the claim by induction on $m$.

*Base case*. For $m = 1$ the claim is straightforward from the definition of ranked grammars.

*Induction step.* Assume that $v \Rightarrow^{m+1} \alpha$, thus $v \Rightarrow^m \beta \Rightarrow^1 \alpha$ for some $\beta \in (\hat{\Sigma} \cup V)^*$. Denote $\beta = \beta_1 \cdots \beta_k$, $\beta_i \in (\hat{\Sigma} \cup V)$. Then, according to the induction hypothesis, it holds that $\mathbf{R}(\beta_i) \subseteq \mathbf{L}(\beta_{i+1})$ for every $i \in [1, k-1]$. Since $G$ is context free, we can split $\alpha$ to $k$ segments $\alpha^1 \cdots \alpha^k$ such that $\alpha^i \in (\hat{\Sigma} \cup V)^*$ and $\beta_i \Rightarrow \alpha^i$. Then, for every $i$, either $\beta_i \in \hat{\Sigma}$ and $\beta_i = \alpha^i$, or $\beta_i \in V$ and $\beta_i \to \alpha^i \in P$. In both cases it holds that inside each fragment $\alpha^i$ we have for every $j \in [1, |\alpha^i| - 1]$: $\mathbf{R}(\alpha^i_j) \subseteq \mathbf{L}(\alpha^i_{j+1})$, since $G$ is ranked. Denote by $\alpha^i_T$ the last element of each sequence $\alpha^i$. We are then left to show that for every $i \in [1, k-1]$: $\mathbf{R}(\alpha^i_T) \subseteq \mathbf{L}(\alpha^{i+1}_1)$, that is, that the claim holds for the letters connecting between the different segments. In the following, we show that

$$(\star) \quad \mathbf{R}(\alpha^i) \subseteq \mathbf{R}(\beta_i) \text{ and } \mathbf{L}(\beta_i) \subseteq \mathbf{L}(\alpha^i)$$

Given $(\star)$ we conclude our proof, since we have $\mathbf{R}(\alpha^i) \subseteq^{\star} \mathbf{R}(\beta_i) \subseteq^{i.h.} \mathbf{L}(\beta_{i+1}) \subseteq^{\star} \mathbf{L}(\alpha^{i+1})$ and thus $\mathbf{R}(\alpha^i_T) = \mathbf{R}(\alpha^i) \subseteq \mathbf{L}(\alpha^{i+1}) = \mathbf{L}(\alpha^{i+1}_1)$.

We are now left to prove $(\star)$. Note that we are only interested in the case where $\beta_i$ is a variable, since for letters we have equality and thus containment. For ease of read we change notations, and prove the following: for $v \to \alpha \in P$ it holds that $\mathbf{R}(\alpha) \subseteq \mathbf{R}(v)$ and $\mathbf{L}(v) \subseteq \mathbf{L}(\alpha)$.

Since $v \to \alpha$, it is either the case that $v \in C^d_1$, $\alpha \in C^d_2$ such that $C^d_1 \to C^d_2$, or that both $v$ and $\alpha$ are in the same MSCC $C^d_1$. We consider the different cases.

For $\mathcal{G}_L$:

- If $v \in C^L_1$, $\alpha \in C^L_2$, then

$$\mathbf{L}(v) =^{\text{def.}} \bigcap_{u \in C^L_1} l(u) \subseteq^{v \in C^L_1} l(v) =^{\text{def.}} \bigcup_{C^L_1 \to C^L_2} \bigcap_{u' \in C^L_2} \mathbf{L}(u') \subseteq^{\alpha \in C^L_2} \mathbf{L}(\alpha)$$

- If $v, \alpha \in C^L_1$ then, if $\alpha$ starts with a variable then $\mathbf{L}(v) =^{\text{def.}} \bigcap_{u \in C^L_1} l(u) =^{\text{def.}} \mathbf{L}(\alpha)$. If $\alpha$ starts with a letter $\sigma$ then $\mathbf{L}(v) =^{\text{def.}} \bigcap_{u \in C^L_1} l(u) \subseteq^{\alpha \in C^L_1} l(\alpha) =^{\text{def.}} \mathbf{L}(\alpha)$.

Similarly, for $\mathcal{G}_R$:

[3] recall that we assume that $G$ only contain rules that can derive a terminal word, and in particular, it does not contain non-reachable rules.

- If $v \in C_1^R$, $\alpha \in C_2^R$, then

$$\mathbf{R}(\alpha) \subseteq^{\alpha \in C_2^R} \bigcup_{C_1^R \to C_2^R} \bigcup_{u' \in C_2^R} \mathbf{R}(u') =^{\text{def.}} r(v) \subseteq^{v \in C_1^R} \bigcup_{u \in C_1^R} r(u) =^{\text{def.}} \mathbf{R}(v)$$

- If $v, \alpha \in C_1^R$, if $\alpha$ ends with a variable then $\mathbf{R}(v) = \mathbf{R}(\alpha)$, and if it ends with a letter then $\mathbf{R}(\alpha) =^{\text{def.}} r(\alpha) \subseteq^{\alpha \in C_1^R} \bigcup_{u \in C_1^R} r(u) =^{\text{def.}} \mathbf{R}(v)$.

This concludes the proof of ($\star$) and the proof of the claim. □

**Claim 22 (restated)** Let $G = \langle \Sigma, X, V, V_0, P, \alpha \rangle$ be a (possibly not ranked) CFHG with $|X| = k$, and let $v \in V$.

1. For every $j \in [1,k] \setminus \mathbf{L}(v)$ there exists $w \in \hat{\Sigma}^*$ such that $v \Rightarrow^* w$ and $j \notin \mathbf{L}(w)$.
2. For every $j \in \mathbf{R}(v)$ there exists $w \in \hat{\Sigma}^*$ such that $v \Rightarrow^* w$ and $j \in \mathbf{R}(w)$.

*Proof.* We prove both parts of the claim by induction of the location of $v$ in the respective MSCC graph $\mathcal{G}_d$.

*Base case.* Assume that $v$ only derives terminal words. Then, $v$ is a singleton $C_v$ and all MSCCs $C_2$ such that $C_v \to C_2$ are also singletons as they are terminal words. This applies for both $d = L$ and $d = R$. Therefore,

1. $\mathbf{L}(v) = l(v) = \bigcap_{v \to w} \mathbf{L}(w)$. Thus for $j \in [1,k] \setminus \mathbf{L}(v)$, for at least one word $w$ such that $v \to w$, it holds that $j \notin \mathbf{L}(w)$.
2. $\mathbf{R}(v) = r(v) = \bigcup_{v \to w} \mathbf{R}(w)$, thus for every $j \in \mathbf{R}(v)$ there exists $w$ such that $v \to w$ and $j \in \mathbf{R}(w)$.

*Induction step.* Let $v \in V$ such that $v \in C_1^d$ for some MSCC $C_1^d \in \mathcal{G}_d$.

1. $\mathbf{L}(v) = \bigcap_{u \in C_1^L} l(u)$. Then if $j \notin \mathbf{L}(v)$, there exists $u \in C_1^L$ such that $j \notin l(u)$.
   - If $u = \sigma\alpha$ for $\sigma \in \hat{\Sigma}$ then $l(u) = \mathbf{L}(\sigma)$, and since we assume all rules can lead to terminal words, we have that $v \Rightarrow^* u \Rightarrow^* \sigma w'$ and $j \notin \mathbf{L}(\sigma w')$ for some $w' \in \hat{\Sigma}^*$.
   - If $u \neq \sigma\alpha$ then $u$ begins with a variable (as we assume no epsilon rules), and thus $l(u) = \bigcup_{C_1^L \to C_2^L} \bigcap_{u' \in C_2^L} \mathbf{L}(u')$. Therefore, there exists $u' \in C_2^L$ with $j \notin \mathbf{L}(u')$. Since we look at MSCCs in $\mathcal{G}_L$, it holds that there exists a finite derivation sequence from $u$ to $u'$ such that $u \Rightarrow^* u'\alpha$, as there are edges in $\mathcal{G}_L$ from a sequence $w$ to a variable $v'$ only if $v'$ appears at the left-most part of the sequence. According to the induction hypothesis, there exists $w \in \hat{\Sigma}^*$ such that $u' \Rightarrow^* w$ and $j \notin \mathbf{L}(w)$. All together, we have that there exists $ww'$ such that $v \Rightarrow^* u\beta \Rightarrow^* u'\alpha\beta \Rightarrow^* ww'$ and $j \notin \mathbf{L}(ww')$.
2. $\mathbf{R}(u) = \bigcup_{u \in C_1^R} r(u)$. Therefore, if $j \in \mathbf{R}(v)$, there exists $u \in C_1^R$ such that $j \in r(u)$.
   - If $u = \alpha\sigma$ then $r(u) = \mathbf{R}(\sigma)$ and as above, there exists $w'$ such that $v \Rightarrow^* w'\sigma$ and $j \in \mathbf{R}(w'\sigma)$.
   - Otherwise, $u$ ends with a variable and $r(u) = \bigcup_{C_1^R \to C_2^R} \bigcup_{u' \in C_2^R} \mathbf{R}(u')$. Therefore, there exists $u' \in C_2^R$ with $j \in \mathbf{R}(u')$. Again, since we look at MSCCs in $\mathcal{G}_R$, this implies a finite derivation $u \Rightarrow \alpha u'$. From the induction hypothesis, there exists $w \in \hat{\Sigma}^*$ such that $u' \Rightarrow^* w$ and $j \in \mathbf{R}(w)$, and thus there exists $w'$ such that $v \Rightarrow w'w$ and $j \in \mathbf{R}(w'w)$.

□