Finite-Word Hyperlanguages

Borzoo Bonakdarpour[1] [0000−0003−1800−5419] and
Sarai Sheinvald[2] [0000−0002−0524−7390]

[1] Department of Computer Science and Engineering, Michigan State University, USA
[2] Department of Software Engineering, ORT Braude College, Israel

Abstract. *Formal languages* are in the core of models of computation and their behavior. A rich family of models for many classes of languages have been widely studied. *Hyperproperties* lift conventional trace-based languages from a set of execution traces to a set of sets of executions. Hyperproperties have been shown to be a powerful formalism for expressing and reasoning about information-flow security policies and important properties of cyber-physical systems. Although there is an extensive body of work on formal-language representation of trace properties, we currently lack such a general characterization for hyperproperties. We introduce *hyperlanguages* over finite words and models for expressing them. Essentially, these models express multiple words by using assignments to quantified *word variables*. Relying on the standard models for regular languages, we propose *hyperregular expressions* and *finite-word hyperautomata (NFH)*, for modeling the class of *regular hyperlanguages*. We demonstrate the ability of regular hyperlanguages to express hyperproperties for finite traces. We explore the closure properties and the complexity of the fundamental decision problems such as nonemptiness, universality, membership, and containment for various fragments of NFH.

1 Introduction

Formal languages along with the models that express them are in the core of modeling, specification, and verification of computing systems. Execution traces are formally described as words, and various families of automata are used for modeling systems of different types. *Regular languages* are a classic formalism for finite traces and when the traces are infinite, *ω-regular languages* are used.

There are well-known connections between specification logics and formal languages. For example, LTL [15] formulas can be translated to ω-regular expressions, and CTL* [8] formulas can be expressed using tree automata. Accordingly, many verification techniques that exploit these relations have been developed. For instance, in the automata-theoretic approach to verification [17, 18], the model-checking problem is reduced to checking the nonemptiness of the product automaton of the model and the complement of the specification.

Hyperproperties [6] generalize the traditional trace properties [2] to *system properties*, i.e., a set of sets of traces. A hyperproperty prescribes how the 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 [6] and important properties of cyber-physical systems [19] such

as sensitivity and robustness, as well as consistency conditions in distributed computing such as linearizability [4]. While different types of logics have been suggested for expressing hyperproperties, their formal-language counterparts and the models that express them are currently missing.

In this paper, we establish a formal-language theoretical framework for *hyperlanguages*, that are sets of sets of words, which we term *hyperwords*. Our framework is based on an underlying standard automata model for formal languages, augmented with quantified *word variables* that are assigned words from a set of words in the hyperlanguage. This formalism is in line with logics for hyperproperties (e.g., HyperLTL [5] and HyperPCTL [1]). These logics express the behavior of infinite trace systems. However, a basic formal model for expressing general hyperproperties for finite words has not been defined yet. Hyperlanguages based on finite words have many practical applications. For instance, path planning objectives for robotic systems often stipulate the *existence* of one or more *finite* paths that stand out from *all* other paths.

To begin with the basics, we focus this paper on a regular type of hyperlanguages of sets consisting of finite words, which we call *regular hyperlanguages*. The models we introduce and study are based on the standard models for regular languages, namely regular expressions and finite-word automata. We explain the idea with two examples.

Example 1. Consider the following *hyperregular expression* (HRE) over the alphabet $\{a\}$.

$$r_1 = \forall x.\exists y.\underbrace{\Big(\{a_x, a_y\}^*\{\#_x, a_y\}^*\Big)}_{\hat{r}_1}$$

The HRE r_1 uses two word variables x and y, which are assigned words from a hyperword. The HRE r_1 contains an underlying regular expression $\hat{r}_1$, whose alphabet is $(\{a\} \cup \{\#\})^{\{x,y\}}$, and whose (regular) language describes different word assignments to x and y, where $\#$ is used for padding at the end if the words assigned to x and y are of different lengths. In a word in the language of $\hat{r}_1$, the i'th letter describes both i'th letters in the words assigned to x and y. For example, the word $\{a_x, a_y\}\{a_x, a_y\}\{\#_x, a_y\}$ describes the assignment $x \mapsto aa, y \mapsto aaa$. The regular expression $\hat{r}_1$ requires that the word assigned to y be longer than the word assigned to x. The *quantification condition* $\forall x.\exists y$ of r_1 requires that for every word in a hyperword S in the hyperlanguage of r_1, there exists a longer word in S. This holds iff S contains infinitely many words. Therefore, the hyperlanguage of r_1 is the set of all infinite hyperwords over $\{a\}$. □

Example 2. Robotics applications are often concerned with finding the shortest path that reaches a goal g, starting from an initial location i. The shortest path requirement can be expressed by the following HRE over an alphabet Σ:

$$r_2 = \exists x.\forall y.\{i_x, i_y\}\{\bar{g}_x, \bar{g}_y\}^*\Big(\{g_x, \bar{g}_y\} \mid \{g_x, g_y\}\Big)\{\#_x, \$_y\}^*$$

where $\bar{g} \in \Sigma - \{g\}$ and $\$ \in \Sigma$. That is, there exists a path x that is shorter than any other path y in reaching g. □

Although there is an ongoing line of research on model-checking hyperproperties [11, 3, 7], the work on finite-trace hyperproperties is limited to [9], where the au-

Property		Result
Closure	Complementation, Union, Intersection (Theorem 1)	
Nonemptiness	$\forall\exists\exists$	Undecidable (Theorem 2)
	$\exists^* \,/\, \forall^*$	NL-complete (Theorem 2)
	$\exists^*\forall^*$	PSPACE-complete (Theorem 2)
Universality	$\exists\forall\forall$	Undecidable (Theorem 3)
	$\exists^* \,/\, \forall^*$	PSPACE-complete (Theorem 3)
	$\forall^*\exists^*$	EXPSPACE (Theorem 3)
Finite membership	NFH	PSPACE (Theorem 4)
	$O(\log(k))$ $\forall$	NP-complete (Theorem 4)
Regular membership	Decidable (Theorem 5)	
Containment	NFH	Undecidable (Theorem 6)
	$\exists^* \subseteq \forall^* \,/\, \forall^* \subseteq \exists^*$	PSPACE-complete (Theorem 7)
	$\exists^*\forall^* \subseteq \forall^*\exists^*$	EXPSPACE (Theorem 7)

Table 1. Summary of results on properties of hyperregular languages.

thors construct a finite-word representation for the class of regular k-safety hyperproperties. We make the following contributions:

- Introduce regular hyperlanguages and HREs, and demonstrate the ability of HREs to express important information-flow security policies such as different variations of *noninterference* [13] and *observational determinism* [20].
- Present *nondeterministic finite-word hyperautomata* (NFH), an automata-based model for expressing regular hyperlanguages.
- Conduct a comprehensive study of the properties of regular hyperlanguages (see Table 1). We show that regular hyperlanguages are *closed* under union, intersection, and complementation. We further prove that the *nonemptiness* problem is in general undecidable for NFH. However, for the alternation-free fragments (which only allow one type of quantifier), as well as for the $\exists\forall$ fragment (in which the quantification condition is limited to a sequence of $\exists$ quantifiers followed by a sequence of $\forall$ quantifiers), nonemptiness is decidable. We also study the *universality*, *membership* and *containment* problems. These results are aligned with the complexity of HyperLTL model checking for tree-shaped and general Kripke structures [3]. This shows that the complexity results in [3] mainly stem from the nature of quantification over finite words and depend on neither the full power of the temporal operators nor the infinite nature of HyperLTL semantics.

2 Preliminaries

An *alphabet* is a nonempty finite set Σ of *letters*. A *word* over Σ is a finite sequence of letters from Σ. The *empty word* is denoted by ϵ, and the set of all words is denoted by Σ^*. A *language* is a subset of Σ^*. We assume that the reader is familiar with the syntax and semantics of regular expressions (RE). We use the standard notations $\{\cdot, |, *\}$ for concatenation, union, and Kleene star, respectively, and denote the language of an RE r by $\mathcal{L}(r)$. A language L is *regular* if there exists an RE r such that $\mathcal{L}(r) = L$.

Definition 1. *A* nondeterministic finite-word automaton *(NFA) is a tuple* $A = \langle \Sigma, Q, Q_0, \delta, F \rangle$*, where* Σ *is an alphabet,* Q *is a nonempty finite set of* states*,* $Q_0 \subseteq Q$ *is a set of* initial states*,* $F \subseteq Q$ *is a set of* accepting states*, and* $\delta \subseteq Q \times \Sigma \times Q$ *is a* transition relation*.*

Given a word $w = \sigma_1 \sigma_2 \cdots \sigma_n$ over Σ, a *run of* A *on* w is a sequence of states $(q_0, q_1, \ldots q_n)$, such that $q_0 \in Q_0$, and for every $0 < i \leq n$, it holds that $(q_{i-1}, \sigma_i, q_i) \in \delta$. The run is *accepting* if $q_n \in F$. We say that A *accepts* w if there exists an accepting run of A on w. The *language* of A, denoted $\mathcal{L}(A)$, is the set of all words that A accepts. It holds that a language L is regular iff there exists an NFA A such that $\mathcal{L}(A) = L$.

3 Hyperregular Expressions

Definition 2. *A* hyperword over Σ *is a set of words over* Σ *and a* hyperlanguage *over* Σ *is a set of hyperwords over* Σ*.*

Before formally defining hyperregular expressions, we explain the idea behind them. A *hyperregular expression* (HRE) over Σ uses a set of *word variables* $X = \{x_1, x_2, \ldots, x_k\}$. When expressing a hyperword S, these variables are assigned words from S. An HRE r is composed of a *quantification condition* α over X, and an underlying RE $\hat{r}$, which represents word assignments to X. An HRE r defines a hyperlanguage $\mathfrak{L}(r)$. The condition α defines the assignments that should be in $\mathcal{L}(\hat{r})$. For example, $\alpha = \exists x_1 . \forall x_2$ requires that there exists a word $w_1 \in S$ (assigned to x_1), such that for every word $w_2 \in S$ (assigned to x_2), the word that represents the assignment $x_1 \mapsto w_1, x_2 \mapsto w_2$, is in $\mathcal{L}(\hat{r})$. The hyperword S is in $\mathfrak{L}(r)$ iff S meets these conditions.

We represent an assignment $v : X \to S$ 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 σ_j. For example, the assignment $v(x_1) = aa$ and $v(x_2) = abb$ is represented by the word assignment $\boldsymbol{w}_v = \{a_{x_1}, a_{x_2}\}\{a_{x_1}, b_{x_2}\}\{\#_{x_1}, b_{x_2}\}$.

Definition 3. *A* hyperregular expression *is a tuple* $r = \langle X, \Sigma, \alpha, \hat{r} \rangle$*, where* $\alpha = \mathbb{Q}_1 x_1 \cdots \mathbb{Q}_k x_k$*, where* $\mathbb{Q}_i \in \{\exists, \forall\}$ *for every* $i \in [1, k]$*, and where* $\hat{r}$ *is an RE over* $\hat{\Sigma} = (\Sigma \cup \{\#\})^X$*.*

Let S be a hyperword and let $v : X \to S$ be an assignment of the word variables of r to words in S. We denote by $v[x \mapsto w]$ the assignment obtained from v by assigning the word $w \in S$ to $x \in X$. We represent v by $\boldsymbol{w}_v$. We now define the membership condition of a hyperword S in the hyperlanguage of r. We first define a relation $\vdash$ for S, $\hat{r}$, a quantification condition α, and an assignment $v : X \to S$, as follows.

- For $\alpha = \epsilon$, define $S \vdash_v (\alpha, \hat{r})$ if $\boldsymbol{w}_v \in \mathcal{L}(\hat{r})$.
- For $\alpha = \exists x . \alpha'$, define $S \vdash_v (\alpha, \hat{r})$ if there exists $w \in S$ s.t. $S \vdash_{v[x \mapsto w]} (\alpha', \hat{r})$.
- For $\alpha = \forall x . \alpha'$, define $S \vdash_v (\alpha, \hat{r})$ if $S \vdash_{v[x \mapsto w]} (\alpha', \hat{r})$ for every $w \in S$.[3]

[3] In case that α begins with $\forall$, membership holds vacuously with an empty hyperword. We restrict the discussion to nonempty hyperwords.

Since all variables are under the scope of α, membership is independent of v, and so if $S \vdash (\alpha, \hat{r})$, we denote $S \in \mathfrak{L}(r)$. The hyperlanguage of r is $\mathfrak{L}(r) = \{S \mid S \in \mathfrak{L}(r)\}$.

Definition 4. *We call a hyperlanguage $\mathfrak{L}$ a* regular hyperlanguage *if there exists an HRE r such that $\mathfrak{L}(r) = \mathfrak{L}$.*

Application of HRE in Information-flow Security

Noninterference [13] requires high-secret commands be removable without affecting observations of users holding low clearances:

$$\varphi_{\mathsf{ni}} = \forall x.\exists y\{l_x, l\lambda_y\}^*,$$

where l denotes a low state and $l\lambda$ denotes a low state such that all high commands are replaced by a dummy value λ.

Observational determinism [20] requires that if two executions of a system start with low-security-equivalent events, they should remain low equivalent:

$$\varphi_{\mathsf{od}} = \forall x.\forall y.\Big(\{l_x, l_y\}^+ \mid \{\bar{l}_x, \bar{l}_y\}\{\$_x, \$_y\}^* \mid \{l_x, \bar{l}_y\}\{\$_x, \$_y\}^* \mid \{\bar{l}_x, l_y\}\{\$_x, \$_y\}^*\Big)$$

where l denotes a low event, $\bar{l} \in \Sigma \setminus \{l\}$, and $\$ \in \Sigma$. We note that similar policies such as *Boudol and Castellani's noninterference* [12] can be formulated in the same fashion. [4]

Generalized noninterference (GNI) [14] allows nondeterminism in the low-observable behavior, but requires that low-security outputs may not be altered by the injection of high-security inputs:

$$\varphi_{\mathsf{gni}} = \forall x.\forall y.\exists z.\Big(\{h_x, l_y, hl_z\} \mid \{\bar{h}_x, l_y, \bar{h}l_z\} \mid \{h_x, \bar{l}_y, h\bar{l}_z\} \mid \{\bar{h}_x, \bar{l}_y, \bar{h}\bar{l}_z\}\Big)^*$$

where h denotes the high-security input, l denotes the low-security output, $\bar{l} \in \Sigma \setminus \{l\}$, and $\bar{h} \in \Sigma \setminus \{h\}$.

Declassification [16] relaxes noninterference by allowing leaking information when necessary. Some programs must reveal secret information to fulfill functional requirements. For example, a password checker must reveal whether the entered password is correct or not:

$$\varphi_{\mathsf{dc}} = \forall x.\forall y.\{li_x, li_y\}\{pw_x, pw_y\}\{lo_x, lo_y\}^+$$

where li denotes low-input state, pw denotes that the password is correct, and lo denotes low-output states. We note that for brevity, φ_{dc} does not include behaviors where the first two events are not low or, in the second event, the password is not valid.

Termination-sensitive noninterference requires that for two executions that start from low-observable states, information leaks are not permitted by the termination behavior of the program (here, l denotes a low state and $\$ \in \Sigma$):

$$\varphi_{\mathsf{tsni}} = \forall x.\forall y.\Big(\{l_x, l_y\}\{\$_x, \$_y\}^*\{l_x, l_y\} \mid \{\bar{l}_x, \bar{l}_y\}\{\$_x, \$_y\}^* \mid \{l_x, \bar{l}_y\}\{\$_x, \$_y\}^* \mid \{\bar{l}_x, l_y\}\{\$_x, \$_y\}^*\Big)$$

[4] This policy states that every two executions that start from bisimilar states (in terms of memory low-observability), should remain bisimilarly low-observable.

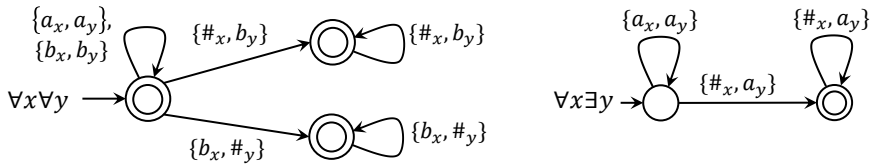

Fig. 1. The NFH $\mathcal{A}_1$ (left) and $\mathcal{A}_2$ (right).

4 Nondeterminsitic Finite-Word Hyperautomata

We now present a model for regular hyperlanguages, namely *finite-word hyperautomata*. A hyperautomaton is composed of a set X of word variables, a quantification condition, and an underlying finite-word automaton that accepts representations of assignments to X.

Definition 5. *A* nondeterministic finite-word hyperautomaton *(NFH) is a tuple* $\mathcal{A} = \langle \Sigma, X, Q, Q_0, F, \delta, \alpha \rangle$*, where* Σ, X *and* α *are as in Definition 3, and where* $\langle \hat{\Sigma}, Q, Q_0, F, \delta \rangle$ *forms an underlying NFA over* $\hat{\Sigma} = (\Sigma \cup \{\#\})^X$.

The acceptance condition for NFH, as for HRE, is defined with respect to a hyperword S, the NFH $\mathcal{A}$, the quantification condition α, and an assignment $v : X \to S$. For the base case of $\alpha = \epsilon$, we define $S \vdash_v (\alpha, \mathcal{A})$ if $\hat{\mathcal{A}}$ accepts $\boldsymbol{w}_v$. The cases where α is of the type $\exists x.\alpha'$ and $\forall x.\alpha'$ are defined similarly as for HRE, and if $S \vdash (\alpha, \mathcal{A})$, we say that $\mathcal{A}$ *accepts* S.

Definition 6. *Let* $\mathcal{A}$ *be an NFH. The* hyperlanguage *of* $\mathcal{A}$*, denoted* $\mathfrak{L}(\mathcal{A})$*, is the set of all hyperwords that* $\mathcal{A}$ *accepts.*

Example 3. Consider the NFH $\mathcal{A}_1$ in Figure 1 (left), whose alphabet is $\Sigma = \{a, b\}$, over two word variables x and y. The NFH $\mathcal{A}_1$ contains an underlying standard NFA $\hat{\mathcal{A}}_1$. For two words w_1, w_2 that are assigned to x and y, respectively, $\hat{\mathcal{A}}_1$ requires that (1) w_1, w_2 agree on their a (and, consequently, on their b) positions, and (2) once one of the words has ended (denoted by $\#$), the other must only contain b letters. Since the quantification condition of $\mathcal{A}_1$ is $\forall x_1.\forall x_2$, in a hyperword S that is accepted by $\mathcal{A}_1$, every two words agree on their a positions. As a result, all the words in S must agree on their a positions. The hyperlanguage of $\mathcal{A}_1$ is then all hyperwords in which all words agree on their a positions.

Example 4. The NFH $\mathcal{A}_2$ of Figure 1 (right) depicts the translation of the HRE of Example 1 to an NFH.

Since regular expressions are equivalent to NFA, we can translate the underlying regular expression $\hat{r}$ of an HRE r to an equivalent NFA, and vice versa – translate the underlying NFA $\hat{\mathcal{A}}$ of an NFA $\mathcal{A}$ to a regular expression. It is then easy to see that every HRE has an equivalent NFH over the same set of variables with the same quantification condition.

We consider several fragments of NFH, which limit the structure of the quantification condition α. $\text{HRE}_\forall$ is the fragment in which α contains only $\forall$ quantifiers, and similarly, in $\text{HRE}_\exists$, α contains only $\exists$ quantifiers. In the fragment $\text{HRE}_{\exists\forall}$, α is of the form $\exists x_1 \cdots \exists x_i \forall x_{i+1} \cdots \forall x_k$.

5 Properties of Regular Hyperlanguages

5.1 Closure Properties

We now consider closure properties of regular hyperlanguages. We show, via constructions on NFH, that regular hyperlanguages are closed under all the Boolean operations.

Theorem 1. *Regular hyperlanguages are closed under union, intersection, and complementation.*

proof sketch. Complementing an NFH $\mathcal{A}$ amounts to dualizing its quantification condition by replacing $\exists$ with $\forall$ and vice versa, and complementing $\hat{\mathcal{A}}$ via the standard construction for NFA. Since complementing $\hat{\mathcal{A}}$ is exponential in its state space, $\overline{\mathcal{A}}$ is exponential in the size of $\mathcal{A}$.

Now, let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two NFH over Σ, with the variables X and Y, respectively. The NFH $\mathcal{A}_\cap$ for $\mathfrak{L}(\mathcal{A}_1) \cap \mathfrak{L}(\mathcal{A}_2)$ is based on the product construction of $\hat{\mathcal{A}}_1$ and $\hat{\mathcal{A}}_2$. The quantification condition of $\mathcal{A}_\cap$ is $\alpha_1 \cdot \alpha_2$. The underlying NFA $\hat{\mathcal{A}}_\cap$ advances simultaneously on both $\mathcal{A}_1$ and $\mathcal{A}_2$: when $\hat{\mathcal{A}}_1$ and $\hat{\mathcal{A}}_2$ run on word assignments $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, respectively, $\hat{\mathcal{A}}_\cap$ runs on a word assignment $\boldsymbol{w}_1 \cup \boldsymbol{w}_2$, which represents both assignments $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, and accepts only if both $\hat{\mathcal{A}}_1$ and $\hat{\mathcal{A}}_2$ accept. To run on both assignments simultaneously, every letter in $\mathcal{A}_\cap$ is of the type $f_1 \cup f_2$, where $f_1 : X \to (\Sigma \cup \{\#\})$ is a letter in $\hat{\Sigma}_1$, and $f_2 : Y \to (\Sigma \cup \{\#\})$ is a letter in $\hat{\Sigma}_2$. This construction is polynomial in the sizes of $\mathcal{A}_1$ and $\mathcal{A}_2$.

Similarly, the NFH $\mathcal{A}_\cup$ for $\mathfrak{L}(\mathcal{A}_1) \cup \mathfrak{L}(\mathcal{A}_2)$ is based on the union construction of $\hat{\mathcal{A}}_1$ and $\hat{\mathcal{A}}_2$. The quantification condition of $\mathcal{A}_\cup$ is again $\alpha_1 \cdot \alpha_2$. The underlying NFA $\hat{\mathcal{A}}_\cup$ advances either on $\mathcal{A}_1$ or $\mathcal{A}_2$. For every word word assignemt $\boldsymbol{w}$ read by $\hat{\mathcal{A}}_1$, the NFH $\hat{\mathcal{A}}_\cup$ reads $\boldsymbol{w} \cup \boldsymbol{w}'$, for every $\boldsymbol{w}' \in \hat{\Sigma}_2^*$, and dually, for every word $\boldsymbol{w}$ read by $\hat{\mathcal{A}}_2$, the NFH $\hat{\mathcal{A}}_\cup$ reads $\boldsymbol{w}' \cup \boldsymbol{w}$, for every $\boldsymbol{w}' \in \hat{\Sigma}_1^*$. The state space of $\mathcal{A}_\cup$ is linear in the state spaces of $\mathcal{A}_1, \mathcal{A}_2$. However, the size of the alphabet of $\mathcal{A}_\cup$ may be exponentially larger than that of $\mathcal{A}_1$ and $\mathcal{A}_2$. □

5.2 Decision Procedures

We now turn to study several decision problems for the various fragments of NFH. Throughout this section, $\mathcal{A}$ is an NFH $\langle \Sigma, X, Q, Q_0, \delta, F, \alpha \rangle$, where $X = \{x_1, \ldots x_k\}$.

Nonemptiness. The *nonemptiness problem* is to decide, given an NFH $\mathcal{A}$, whether $\mathfrak{L}(\mathcal{A}) = \emptyset$. In [10], a reduction from the *Post correspondence problem* is used for proving the undecidability of HyperLTL satisfiability. A roughly similar reduction shows that the nonemptiness problem for NFH is, in general, undecidable. However, nonemptiness is decidable for the fragments we consider, with varying complexities.

For the alternation-free fragments, we show that a simple reachability test on their underlying automata suffices to verify nonemptiness.

For NFH$_{\exists\forall}$, we show that the problem is decidable, by checking the nonemptiness of an exponentially larger equi-empty NFA. To summarize, we have the following.

Theorem 2. *The nonemptiness problem for*

1. *NFH$_\exists$ and NFH$_\forall$ is* NL*-complete,*
2. *NFH$_{\exists\forall}$ is* PSPACE*-complete, and*
3. *NFH is undecidable.*

proof sketch. **NFH$_\forall$ and NFH$_\exists$.** The lower bound follows from the NL-hardness of NFA nonemptiness. For the upper bounds, an NFH$_\exists$ $\mathcal{A}_\exists$ is nonempty iff $\hat{\mathcal{A}}_\exists$ accepts some word assignment $\boldsymbol{w}_v$. Indeed, any hyperword that contains the words in v is accepted by $\mathcal{A}_\exists$. We can therefore run a restricted reachability test on $\hat{\mathcal{A}}_\exists$, that considers only consecutive transitions in which for every $x \in X$, a letter σ_x never follows $\#_x$, which guarantees a run on a legal word assignment.

We can show that an NFH$_\forall$ $\mathcal{A}_\forall$ is nonempty iff $\mathcal{A}_\forall$ accepts a hyperword of size 1. Accordingly, $\mathcal{A}_\forall$ is nonempty iff $\hat{\mathcal{A}}$ accepts a word that represents an assignment that assigns all variables the same word. We thus restrict the transitions of $\hat{\mathcal{A}}_\forall$ to fixed functions, and check the nonemptiness of the restricted NFA.

NFH$_{\exists\forall}$. We begin with a PSPACE upper bound. Let $\mathcal{A}$ be an NFH$_{\exists\forall}$ with m existential quantifiers, and let $S \in \mathfrak{L}(\mathcal{A})$. Then, there exist $w_1, \ldots, w_m \in S$, such that for every assignment $v : X \to S$ in which $v(x_i) = w_i$ for every $1 \leq i \leq m$, we have that $\hat{\mathcal{A}}$ accepts $\boldsymbol{w}_v$. In particular, $\hat{\mathcal{A}}$ accepts every assignment that agrees with v on $x_1, \ldots x_m$, and assigns only words from $\{w_1, \ldots, w_m\}$. Therefore, $\hat{\mathcal{A}}$ accepts the hyperword $\{w_1, \ldots, w_m\}$. That is, $\mathcal{A}$ is nonempty iff it accepts a hyperword of size at most m. We can construct an NFA A based on $\hat{\mathcal{A}}$ that is nonempty iff $\hat{\mathcal{A}}$ accepts all appropriate assignments of a hyperword of size m. The size of A is exponential in the size of $\hat{\mathcal{A}}$, and the result follows from the NL upper bound for NFA nonemptiness.

Next, we prove the lower-bound for NFH$_{\exists\forall}$ by a reduction from a polynomial version of the *corridor tiling problem*, defined as follows. We are given a finite set T of tiles, two relations $V \subseteq T \times T$ and $H \subseteq T \times T$, an initial tile t_0, a final tile t_f, and a bound $n > 0$. We have to decide whether there is some $m > 0$ and a tiling of a $n \times m$-grid such that (1) The tile t_0 is in the bottom left corner and the tile t_f is in the top right corner, (2) Every pair of horizontal neighbors is in H, and (3) Every pair of vertical neighbors is in V. When n is given in unary notation, the problem is known to be PSPACE-complete.

Given an instance C of the tiling problem, we construct an NFH$_{\exists\forall}$ $\mathcal{A}$ that is nonempty iff C has a solution. We encode a solution to C as a word $w_{sol} = w_1 \cdot w_2 \cdot w_m\$$ over $\Sigma = T \cup \{1, 2, \ldots n, \$\}$, where the word w_i, of the form $1 \cdot t_{1,i} \cdot 2 \cdot t_{2,i}, \ldots n \cdot t_{n,i}$, describes the contents of row i. To check that w_{sol} indeed encodes a solution, we need to make sure that: (1) w_1 begins with t_0 and w_m ends with $t_f\$$, (2) Every w_i is of the correct form, (3) Within every w_i, it holds that $(t_{j,i}, t_{j+1,i}) \in H$, and (4) For w_i, w_{i+1}, it holds that $(t_{j,i}, t_{j,i+1}) \in V$ for every $j \in [1, n]$.

Verifying conditions $(1) - (3)$ above is easy via an NFA of size $O(n|H|)$. The main obstacle is condition (4). We describe an NFH$_{\exists\forall}$ $\mathcal{A} = \langle T \cup \{0, 1, \ldots n, \$\}, \{y_1, y_2, y_3, x_1,$

$\ldots x_{\log(n)}\}, Q, \{q_0\}, \delta, F, \alpha = \exists y_1 \exists y_2 \exists y_3 \forall x_1 \ldots \forall x_{\log(n)}\rangle$ that is nonempty iff there exists a word that satisfies conditions $(1)-(4)$. The NFH $\mathcal{A}$ only proceeds on letters whose assignments to y_1, y_1, y_3 is $r, 0, 1$, respectively, where $r \in T \cup \{1, \ldots n, \$\}$. Then $\mathcal{A}$ requires the existence of the words $0^{|w_{sol}|}$ and $1^{|w_{sol}|}$ (the 0 word and 1 word, henceforth). $\mathcal{A}$ makes sure that the word assigned to y_1 matches a correct solution w.r.t. conditions $(1)-(3)$ above. Now, we need to make sure that for every position j in a row, the tile in position j in the next row matches the current one w.r.t. V. We can use a state q_j to remember the tile in position j, and compare it to the tile in the next occurrence of j. To avoid checking all positions simultaneously (which would require exponentially many states), we use $\log(n)$ copies of the 0 and 1 words to encode j. The $\log(n)$ $\forall$ conditions make sure that every position within $1-n$ is checked.

We limit the checks to words in which $x_1, \ldots x_{\log(n)}$ are the 0 or 1 words, by having $\hat{\mathcal{A}}$ accept every word in which some x variable is not assigned 0 or 1. This accepts all cases in which the word assigned to y_1 is also assigned to one of the x variables.

To check that $x_1, \ldots x_{\log(n)}$ are the 0 or 1 words, $\hat{\mathcal{A}}$ checks that the letter assignments to these variables remain constant throughout the run. In these cases, upon reading the first letter, $\hat{\mathcal{A}}$ remembers the value j that is encoded by the assignments to $x_1, \ldots x_{\log(n)}$ in a state, and makes sure that throughout the run, the tile that occurs in the assignment to y_1 in position j in the current row matches the tile in position j in the next row.

We construct a similar reduction for the case that the number of $\forall$ quantifiers is fixed: instead of encoding the position by $\log(n)$ bits, we can directly specify the position by a word of the form j^*, for every $j \in [1, n]$, and we construct a matching $\text{NFH}_{\exists\forall}$ over $O(n)$ variables under $\exists$, and a single variable under $\forall$. □

Universality. The *universality problem* is to decide whether a given NFH $\mathcal{A}$ accepts every hyperword over Σ. Notice that $\mathcal{A}$ is universal iff $\overline{\mathcal{A}}$ is empty. Since complementing an NFH involves an exponential blow-up, we conclude the following from the results in Section 5.2, combined with the PSPACE lower bound for the universality of NFA.

Theorem 3. *The universality problem for*

1. *NFH is undecidable,*
2. *$\text{NFH}_\exists$ and $\text{NFH}_\forall$ is PSPACE-complete, and*
3. *$\text{NFH}_{\forall\exists}$ is in EXPSPACE.*

Membership. We turn to study the membership problem for NFH: given an NFH $\mathcal{A}$ and a hyperword S, is $S \in \mathfrak{L}(\mathcal{A})$? When S is finite, so is the set of assignments from X to S, and so the problem is decidable. We call this case the *finite membership problem*.

Theorem 4.
- *The finite membership problem for NFH is in PSPACE.*
- *The finite membership problem for NFH with $O(\log(k))$ $\forall$ quantifiers is NP-complete.*

Proof. We can decide the membership of a finite hyperword S in $\mathfrak{L}(\mathcal{A})$ by iterating over all relevant assignments from X to S, and for every such assignment v, checking

on-the-fly whether $\boldsymbol{w}_v \in \mathcal{L}(\hat{\mathcal{A}})$. The space size of this algorithm is polynomial in k and logarithmic in $|\mathcal{A}|$ and in $|S|$.

When the number of $\forall$ quantifiers in $\mathcal{A}$ is $|O(\log(k))|$, we can iterate over all assignments to the $\forall$ variables in polynomial time, while guessing assignments to the variables under $\exists$. Thus, membership in this case is in NP.

We show NP-hardness for this case by a reduction from the Hamiltonian cycle problem. Given a graph $G = \langle V, E\rangle$ where $V = \{v_1, \ldots, v_n\}$ and $|E| = m$, we construct an $\text{NFH}_\exists$ $\mathcal{A}$ over $\{0, 1\}$ with n states, n variables, δ of size m, and a hyperword S of size n, as follows. $S = \{w_1, \ldots, w_n\}$, where $w_i = 0^{i-1} \cdot 1 \cdot 0^{n-i}$. The structure of $\hat{\mathcal{A}}$ is identical to that of G, and we set $Q_0 = F = \{v_1\}$. For every $(v_i, v_j) \in E$, we have $(v_i, f_i, v_j) \in \delta$, where $f_i(x_i) = 1$ and $f_i(x_j) = 0$ for every $x_j \neq x_i$. Intuitively, the i'th letter in an accepting run of $\hat{\mathcal{A}}$ marks traversing v_i. Assigning w_j to x_i means that the j'th step of the run traverses v_i. Since the words in w make sure that every $v \in V$ is traversed exactly once, and are all of length n, we have that $\mathcal{A}$ accepts S iff there exists some ordering of the words in S that matches a Hamiltonian cycle in G.
Note: For a hyperword of size ≥ 2, the size of δ must be exponential in the number k' of $\forall$ quantifiers, to account for all the assignments to these variables. Thus, if $k = O(k')$, an algorithm that uses a space of size k is in fact logarithmic in the size of $\mathcal{A}$. □

When S is infinite, it may still be finitely represented, allowing for algorithmic membership testing. We now address the problem of deciding whether a regular language $\mathcal{L}$ (given as an NFA) is accepted by an NFH. We call this *the regular membership problem for NFH*. We show that this problem is decidable for the entire class of NFH.

Theorem 5. *The regular membership problem for NFH is decidable.*

Proof. Let $A = \langle \Sigma, P, P_0, \rho, F\rangle$ be an NFA with n states. We first extend the alphabet of A to $\Sigma \cup \{\#\}$, and extend its language to $\mathcal{L}(A) \cdot \{\#\}^*$. We describe a recursive procedure (iterating over α) for deciding whether $\mathcal{L}(A) \in \mathfrak{L}(\mathcal{A})$.

For the case that $k = 1$, if $\alpha = \exists x_1$, then $\mathcal{L}(A) \in \mathfrak{L}(\mathcal{A})$ iff $\mathcal{L}(A) \cap \mathcal{L}(\hat{\mathcal{A}}) \neq \emptyset$. Otherwise, if $\alpha = \forall x_1$, then $\mathcal{L}(A) \in \mathfrak{L}(\mathcal{A})$ iff $\mathcal{L}(A) \notin \mathfrak{L}(\overline{\mathcal{A}})$, where $\overline{\mathcal{A}}$ is the NFH for $\overline{\mathfrak{L}(\mathcal{A})}$. The quantification condition for $\overline{\mathcal{A}}$ is $\exists x_1$, conforming to the base case.

For $k > 1$, we construct a sequence of NFH $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$. If $\mathbb{Q}_1 = \exists$ then we set $\mathcal{A}_1 = \mathcal{A}$, and otherwise we set $\mathcal{A}_1 = \overline{\mathcal{A}}$. Let $\mathcal{A}_i = \langle \Sigma, \{x_i, \ldots x_k\}, Q_i, Q_i^0, \delta_i, \mathcal{F}_i, \alpha_i\rangle$. If α_i starts with $\exists$, then we construct $\mathcal{A}_{i+1}$ as follows.

The variables of $\mathcal{A}_{i+1}$ are $\{x_{i+1}, \ldots, x_k\}$, and $\alpha_{i+1} = \mathbb{Q}_{i+1}x_{i+1} \cdots \mathbb{Q}_k x_k$, for $\alpha_i = \mathbb{Q}_i x_i \cdots \mathbb{Q}_k x_k$. The set of states of $\mathcal{A}_{i+1}$ is $Q_i \times P$, and the set of initial states is $Q_i^0 \times P_0$. The set of accepting states is $\mathcal{F}_i \times F$. For every $(q \xrightarrow{f} q') \in \delta_i$ and every $(p \xrightarrow{f(x_i)} p') \in \rho$, we have $((q, p) \xrightarrow{f \setminus \{\sigma_{i_{x_i}}\}} (q', p')) \in \delta_{i+1}$. Then, $\hat{\mathcal{A}}_{i+1}$ accepts a word assignment $\boldsymbol{w}_v$ iff there exists a word $u \in \mathcal{L}(A)$, such that $\hat{\mathcal{A}}_i$ accepts $\boldsymbol{w}_{v \cup \{x_i \mapsto u\}}$.

Let $v : \{x_i, \ldots, x_k\} \to \mathcal{L}(A)$. Then $\mathcal{L}(A) \vdash_v (\alpha_i, \mathcal{A}_i)$ iff there exists $w \in \mathcal{L}(A)$ such that $\mathcal{L}(A) \vdash_{v[x_i \mapsto w]} (\alpha_{i+1}, \mathcal{A}_i)$. For an assignment $v' : \{x_{i+1}, \ldots, x_k\} \to \mathcal{L}(A)$, it holds that $\boldsymbol{w}_{v'}$ is accepted by $\hat{\mathcal{A}}_{i+1}$ iff there exists $w \in \mathcal{L}(A)$ such that $\boldsymbol{w}_v \in \mathcal{L}(\hat{\mathcal{A}}_i)$, where $v = v' \cup \{x_i \mapsto w\}$. Therefore, we have that $\mathcal{L}(A) \vdash_{v[x_i \to w]} (\alpha_i, \mathcal{A}_i)$ iff $\mathcal{L}(A) \vdash_{v'} (\alpha_{i+1}, \mathcal{A}_{i+1})$, that is, $\mathcal{L}(A) \in \mathfrak{L}(\mathcal{A}_i)$ iff $\mathcal{L}(A) \in \mathfrak{L}(\mathcal{A}_{i+1})$.

If α_i starts with $\forall$, then we have that $\mathcal{L}(A) \in \mathfrak{L}(\mathcal{A}_i)$ iff $\mathcal{L}(A) \notin \overline{\mathfrak{L}(\mathcal{A}_i)}$. We construct $\overline{\mathcal{A}_i}$ for $\overline{\mathfrak{L}(\mathcal{A}_i)}$ as described in Theorem 1. The quantification condition of $\overline{\mathcal{A}_i}$ begins with $\exists x_i$. We then construct $\mathcal{A}_{i+1}$ w.r.t. $\overline{\mathcal{A}_i}$, and check for non-membership.

Every $\forall$ quantifier requires complementation, which is exponential in n.Therefore, in the worst case, the complexity of this algorithm is $O(2^{2^{\cdots^{|Q||A|}}})$, where the tower is of height k. If the number of $\forall$ quantifiers is fixed, then the complexity is $O(|Q||A|^k)$. □

Containment. The *containment problem* is to decide, given NFH $\mathcal{A}_1$ and $\mathcal{A}_2$, whether $\mathfrak{L}(\mathcal{A}_1) \subseteq \mathfrak{L}(\mathcal{A}_2)$. Since we can reduce the nonemptiness problem to the containment problem, we have the following as a result of Theorem 2.

Theorem 6. *The containment problem for NFH is undecidable.*

However, the containment problem is decidable for various fragments of NFH.

Theorem 7. *The containment problem of NFH$_\exists$ $\subseteq$ NFH$_\forall$ and NFH$_\forall$ $\subseteq$ NFH$_\exists$ is PSPACE-complete. The containment problem of NFH$_{\exists\forall}$ $\subseteq$ NFH$_{\forall\exists}$ is in EXPSPACE*

Proof. A lower bound for all cases follows from the PSPACE-hardness of the containment problem for NFA. For the upper bound, for two NFH $\mathcal{A}_1$ and $\mathcal{A}_2$, we have that $\mathfrak{L}(\mathcal{A}_1) \subseteq \mathfrak{L}(\mathcal{A}_2)$ iff $\mathfrak{L}(\mathcal{A}_1) \cap \overline{\mathfrak{L}(\mathcal{A}_2)} = \emptyset$. We can compute an NFH $\mathcal{A} = \mathcal{A}_1 \cap \overline{\mathcal{A}_2}$ (Theorem 1), and check its nonemptiness. Complementing $\mathcal{A}_2$ is exponential in its number of states, and the intersection construction is polynomial.

If $\mathcal{A}_1 \in$ NFH$_\exists$ and $\mathcal{A}_2 \in$ NFH$_\forall$ or vice versa, then $\mathcal{A}$ is an NFH$_\exists$ or NFH$_\forall$, respectively, whose nonemptiness can be decided in space that is logarithmic in $|\mathcal{A}|$.

The quantification condition of an NFH for the intersection may be any interleaving of the quantification conditions of the two intersected NFH. (Theorem 1). Therefore, for the rest of the fragments, we can construct the intersection such that $\mathcal{A}$ is an NFH$_{\exists\forall}$. The exponential blow-up in complementing $\mathcal{A}_2$, along with The PSPACE upper bound of Theorem 2 gives an EXPSPACE upper bound for the rest of the cases. □

6 Discussion and Future Work

We have introduced and studied *hyperlanguages* and a framework for their modeling, focusing on the basic class of regular hyperlanguages, modeled by HRE and NFH. We have shown that regular hyperlanguages are closed under set operations and are capable of expressing important hyperproperties for information-flow security policies over finite traces. We have also investigated fundamental decision procedures for various fragments of NFH. Some gaps, such as the precise lower bound for the universality and containment problems for NFH$_{\exists\forall}$, are left open.

Since our framework does not limit the type of underlying model, it can be lifted to handle hyperwords consisting of infinite words, with an underlying model designed for such languages, such as *nondeterministic Büchi automata*, which model ω-regular languages. Just as Büchi automata can express LTL, such a model can express the entire logic of HyperLTL [5].

As future work, we plan on studying non-regular hyperlanguages (e.g., context-free), and object hyperlanguages (e.g., trees). Another direction is designing learning algorithms for hyperlanguages, by exploiting known canonical forms for the underlying models, and basing on existing learning algorithms for them. The main challenge would be handling learning sets and a mechanism for learning word variables and quantifiers.

References

1. Ábrahám, E., Bonakdarpour, B.: HyperPCTL: A temporal logic for probabilistic hyperproperties. In: QEST. pp. 20–35 (2018)
2. Alpern, B., Schneider, F.: Defining liveness. Information Processing Letters pp. 181–185 (1985)
3. B. Bonakdarpour, B., Finkbeiner, B.: The complexity of monitoring hyperproperties. In: CSF. pp. 162–174 (2018)
4. Bonakdarpour, B., Sánchez, C., Schneider, G.: Monitoring hyperproperties by combining static analysis and runtime verification. In: ISoLA. pp. 8–27 (2018)
5. Clarkson, M., Finkbeiner, B., Koleini, M., Micinski, K., Rabe, M., Sánchez, C.: Temporal logics for hyperproperties. In: POST. pp. 265–284 (2014)
6. Clarkson, M., Schneider, F.: Hyperproperties. Journal of Computer Security pp. 1157–1210 (2010)
7. Coenen, N., Finkbeiner, B., C. Sánchez, C., Tentrup, L.: Verifying hyperliveness. In: CAV. pp. 121–139 (2019)
8. Emerson, E.A., Halpern, J.: "sometimes" and "not never" revisited: on branching versus linear time temporal logic. Journal of the ACM pp. 151–178 (1986)
9. Finkbeiner, B., Haas, L., Torfah, H.: Canonical representations of k-safety hyperproperties. In: CSF 2019. pp. 17–31 (2019)
10. Finkbeiner, B., Hahn, C.: Deciding hyperproperties. In: CONCUR. pp. 13:1–13:14 (2016)
11. Finkbeiner, B., Rabe, M., Sánchez, C.: Algorithms for model checking HyperLTL and HyperCTL*. In: CAV. pp. 30–48 (2015)
12. G. Boudol, G., Castellani, I.: Noninterference for concurrent programs and thread. In: TCS 2002. pp. 109–130 (2002)
13. Goguen, J., Meseguer, J.: Security policies and security models. In: IEEE Symp. on Security and Privacy. pp. 11–20 (1982)
14. McCullough, D.: Noninterference and the composability of security properties. In: Proceedings of the 1988 IEEE Symposium on Security and Privacy. pp. 177–186 (1988)
15. Pnueli, A.: The temporal logic of programs. In: FOCS. pp. 46–57 (1977)
16. Sabelfeld, A., Sands, D.: Probabilistic noninterference for multi-threaded programs. In: CSFW. pp. 200–214 (2000)
17. Vardi, M., Wolper, P.: Automata theoretic techniques for modal logic of programs. Journal of Computer and System Sciences pp. 183–221 (1986)
18. Vardi, M., Wolper, P.: Reasoning about infinite computations. Information and Computation pp. 1–37 (1994)
19. Wang, Y., Zarei, M., Bonakdarpour, B., Pajic, M.: Statistical verification of hyperproperties for cyber-physical systems. ACM Transactions on Embedded Computing systems (TECS) pp. 92:1–92:23 (2019)
20. Zdancewic, S., Myers, A.: Observational determinism for concurrent program security. In: CSFW. p. 29 (2003)

Appendix

A Proofs

We present several terms and notations which we use throughout the following proofs. Recall that we represent an assignment $v : X \to S$ as a word assignment $\boldsymbol{w}_v$. Conversely, a word $\boldsymbol{w}$ over $(\Sigma \cup \{\#\})^X$ represents an assignment $v_{\boldsymbol{w}} : X \to \Sigma^*$, where $v_{\boldsymbol{w}}(x_i)$ is formed by concatenating the letters of Σ that are assigned to x_i in the letters of $\boldsymbol{w}$ We denote the set of all such words $\{v_{\boldsymbol{w}}(x_1), \ldots, v_{\boldsymbol{w}}(x_k)\}$ by $S(\boldsymbol{w})$. Since we only allow padding at the end of a word, if a padding occurs in the middle of $\boldsymbol{w}$, then $\boldsymbol{w}$ does not represent a legal assignment. Notice that this occurs iff $\boldsymbol{w}$ contains two consecutive letters $\boldsymbol{w}_i\boldsymbol{w}_{i+1}$ such that $\boldsymbol{w}_i(x) = \#$ and $\boldsymbol{w}_{i+1}(x) \neq \#$ for some $x \in X$. We call $\boldsymbol{w}$ *legal* if $v_{\boldsymbol{w}}$ represents a legal assignment from X to Σ^*.

Consider a function $g : A \to B$ where A, B are some sets. The *range* of g, denoted $\mathsf{range}(g)$ is the set $\{g(a) | a \in A\}$.

A *sequence* of g is a function $g' : A \to B$ such that $\mathsf{range}(g') \subseteq \mathsf{range}(g)$. A *permutation* of g is a function $g' : A \to B$ such that $\mathsf{range}(g') = \mathsf{range}(g)$. We extend the notions of sequences and permutations to word assignments. Let $\boldsymbol{w}$ be a word over $\hat{\Sigma}$. A sequence of $\boldsymbol{w}$ is a word $\boldsymbol{w}'$ such that $S(w') \subseteq S(w)$, and a permutation of $\boldsymbol{w}$ is a word $\boldsymbol{w}'$ such that $S(w') = S(w)$.

Theorem 1

Proof. **Complementation.** Let $\mathcal{A}$ be an NFH. The NFA $\hat{\mathcal{A}}$ can be complemented with respect to its language over $\hat{\Sigma}$ to an NFA $\overline{\hat{\mathcal{A}}}$. Then for every assignment $v : X \to S$, it holds that $\hat{\mathcal{A}}$ accepts $\boldsymbol{w}_v$ iff $\overline{\hat{\mathcal{A}}}$ does not accept $\boldsymbol{w}_v$. Let $\overline{\alpha}$ be the quantification condition obtained from α by replacing every $\exists$ with $\forall$ and vice versa. We can prove by induction on α that $\overline{\mathcal{A}}$, the NFH whose underlying NFA is $\overline{\hat{\mathcal{A}}}$, and whose quantification condition is $\overline{\alpha}$, accepts $\overline{\mathfrak{L}(\mathcal{A})}$. The size of $\overline{\mathcal{A}}$ is exponential in $|Q|$, due to the complementation construction for $\hat{\mathcal{A}}$.

Now, let $\mathcal{A}_1 = \langle \Sigma, X, Q, Q_0, \delta_1, F_1, \alpha_1 \rangle$ and $\mathcal{A}_2 = \langle \Sigma, Y, P, P_0, \delta_2, F_2, \alpha_2 \rangle$ be two NFH with $|X| = k$ and $|Y| = k'$ variables, respectively.

Union. We construct an NFH $\mathcal{A}_\cup = \langle \Sigma, X \cup Y, Q \cup P \cup \{p_1, p_2\}, Q_0 \cup P_0, \delta, F_1 \cup F_2 \cup \{p_1, p_2\}, \alpha \rangle$, where $\alpha = \alpha_1\alpha_2$ (that is, we concatenate the two quantification conditions), and where δ is defined as follows.

- For every $(q_1 \xrightarrow{f} q_2) \in \delta_1$ we set $(q_1 \xrightarrow{f \cup g} q_2) \in \delta$ for every $g \in (\Sigma \cup \{\#\})^Y$.
- For every $(q_1 \xrightarrow{f} q_2) \in \delta_2$ we set $(q_1 \xrightarrow{f \cup g} q_2) \in \delta$ for every $g \in (\Sigma \cup \{\#\})^X$.
- For every $q \in F_1$, we set $(q \xrightarrow{\{\#\}^X \cup g} p_1), (p_1 \xrightarrow{\{\#\}^X \cup g} p_1) \in \delta$ for every $g \in (\Sigma \cup \{\#\})^Y$.
- For every $q \in F_2$, we set $(q \xrightarrow{g \cup \{\#\}^Y} p_2), (p_2 \xrightarrow{g \cup \{\#\}^Y} p_2) \in \delta$ for every $g \in (\Sigma \cup \{\#\})^X$.

Let S be a hyperword. For every $v : (X \cup Y) \to S$, it holds that if $\boldsymbol{w}_{v|_X} \in \mathcal{L}(\hat{\mathcal{A}}_1)$, then $\boldsymbol{w}_v \in \mathcal{L}(\hat{\mathcal{A}}_\cup)$. Indeed, according to our construction, every word assigned to the Y variables is accepted in the $\mathcal{A}_1$ component of the construction, and so it satisfies both types of quantifiers. A similar argument holds for $v|_Y$ and $\mathcal{A}_2$.

Also, according to our construction, for every $v : (X \cup Y) \to S$, if $\boldsymbol{w}_v \in \mathcal{L}(\hat{\mathcal{A}}_\cup)$, then either $\boldsymbol{w}_{v|_X} \in \mathcal{L}(\hat{\mathcal{A}}_1)$, or $\boldsymbol{w}_{v|_Y} \in \mathcal{L}(\hat{\mathcal{A}}_2)$. As a conclusion, we have that $\mathfrak{L}(\mathcal{A}_\cup) = \mathfrak{L}(\mathcal{A}_1) \cup \mathfrak{L}(\mathcal{A}_2)$.

The state space of $\mathcal{A}_\cup$ is linear in the state spaces of $\mathcal{A}_1, \mathcal{A}_2$. However, the size of the alphabet of $\mathcal{A}_\cup$ may be exponentially larger than that of $\mathcal{A}_1$ and $\mathcal{A}_2$, since we augment each letter with all functions from Y to $\Sigma \cup \{\#\}$ (in $\mathcal{A}_1$) and from X to $\Sigma \cup \{\#\}$ (in $\mathcal{A}_2$).

Intersection. The proof follows the closure of regular hyperlanguages under union and complementation. However, we also offer a direct translation, which avoids the need to complement. We construct an NFH $\mathcal{A}_\cap = \langle \Sigma, X \cup Y, (Q \cup \{q\}) \times (P \cup \{p\}), (Q_0 \times P_0), \delta, (F_1 \cup \{q\}) \times (F_2 \cup \{p\}), \alpha_1\alpha_2 \rangle$, where δ is defined as follows.

- For every $(q_1 \xrightarrow{f} q_2) \in \delta_1$ and every $(p_1 \xrightarrow{g} p_2) \in \delta_2$, we have

$$\Big((q_1, p_1) \xrightarrow{f \cup g} (q_2, p_2)\Big) \in \delta$$

- For every $q_1 \in F_1, (p_1 \xrightarrow{g} p_2) \in \delta_2$ we have

$$\Big((q_1, p_1) \xrightarrow{\{\#\}^X \cup g} (q, p_2)\Big), \Big((q, p_1) \xrightarrow{\{\#\}^k \cup g} (q, p_2)\Big) \in \delta$$

- For every $(q_1 \xrightarrow{f} q_2) \in \delta_1$ and $p_1 \in F_2$, we have

$$\Big((q_1, p_1) \xrightarrow{f \cup \{\#\}^Y} (q_2, p)\Big), \Big((q_1, p) \xrightarrow{f \cup \{\#\}^Y} (q_2, p)\Big) \in \delta$$

Intuitively, the role of q, p is to keep reading $\{\#\}^X$ and $\{\#\}^Y$ after the word read by $\hat{\mathcal{A}}_1$ or $\hat{\mathcal{A}}_2$, respectively, has ended.

The NFH $\hat{\mathcal{A}}_\cap$ simultaneously reads two word assignments that are read along $\hat{\mathcal{A}}_1$ and $\hat{\mathcal{A}}_2$, respectively, and accepts iff both word assignments are accepted. The correctness follows from the fact that for $v : (X \cup Y) \to S$, we have that $\boldsymbol{w}_v$ is accepted by $\hat{\mathcal{A}}$ iff $\boldsymbol{w}_{v|_X}$ and $\boldsymbol{w}_{v|_Y}$ are accepted by $\hat{\mathcal{A}}_1$ and $\hat{\mathcal{A}}_2$, respectively. This construction is polynomial in the sizes of $\mathcal{A}_1$ and $\mathcal{A}_2$.

□

Theorem 2

Proof. **NFH$_\exists$ and NFH$_\forall$.** The lower bound for both fragments follows from the NL-hardness of the nonemptiness problem for NFA.

We turn to the upper bound, and begin with NFH$_\exists$. Let $\mathcal{A}_\exists$ be an NFH$_\exists$. We claim that $\mathcal{A}_\exists$ is nonempty iff $\hat{\mathcal{A}}_\exists$ accepts some legal word $\boldsymbol{w}$. The first direction is trivial. For

the second direction, let $\boldsymbol{w} \in \mathcal{L}(\hat{\mathcal{A}}_\exists)$. By assigning $v(x_i) = v_{\boldsymbol{w}}(x_i)$ for every $x_i \in X$, we get $\boldsymbol{w}_v = \boldsymbol{w}$, and according to the semantics of $\exists$, we have that $\mathcal{A}_\exists$ accepts $S(\boldsymbol{w})$. To check whether $\hat{\mathcal{A}}_\exists$ accepts a legal word, we can run a reachability check on-the-fly, while advancing from a letter σ to the next letter σ' only if σ' assigns $\#$ to all variables for which σ assigns $\#$. While each transition $T = q \xrightarrow{f} p$ in $\hat{\mathcal{A}}$ is of size k, we can encode T as a set of size k of encodings of transitions of type $q \xrightarrow{(x_i,\sigma_i)} p$ with a binary encoding of p, q, σ_i, as well as i, t, where t marks the index of T within the set of transitions of $\hat{\mathcal{A}}$. Therefore, the reachability test can be performed within space that is logarithmic in the size of $\mathcal{A}_\exists$.

Now, let $\mathcal{A}_\forall$ be an NFH$_\forall$ over X. We claim that $\mathcal{A}_\forall$ is nonempty iff $\mathcal{A}_\forall$ accepts a hyperword of size 1. For the first direction, let $S \in \mathfrak{L}(\mathcal{A}_\forall)$. Then, by the semantics of $\forall$, we have that for every assignment $v : X \to S$, it holds that $\boldsymbol{w}_v \in \mathcal{L}(\hat{\mathcal{A}}_\forall)$. Let $u \in S$, and let $v_u(x_i) = u$ for every $x_i \in X$. Then, in particular, $\boldsymbol{w}_{v_u} \in \mathcal{L}(\hat{\mathcal{A}}_\forall)$. Then for every assignment $v : X \to \{u\}$ (which consists of the single assignment v_u), it holds that $\hat{\mathcal{A}}_\forall$ accepts $\boldsymbol{w}_v$, and therefore $\mathcal{A}_\forall$ accepts $\{u\}$. The second direction is trivial.

To check whether $\mathcal{A}_\forall$ accepts a hyperword of size 1, we restrict the reachability test on $\hat{\mathcal{A}}_\forall$ to letters over $\hat{\Sigma}$ that represent fixed functions.

NFH$_{\exists\forall}$. We prove a PSPACE upper bound. Let m be the number of $\exists$ quantifiers in α, and let $S \in \mathfrak{L}(\mathcal{A})$. Then, according to the semantics of the quantifiers, there exist $w_1, \ldots w_m \in S$, such that for every assignment $v : X \to S$ in which $v(x_i) = w_i$ for every $1 \leq i \leq m$, it holds that $\hat{\mathcal{A}}$ accepts $\boldsymbol{w}_v$. Let $v : X \to S$ be such an assignment. Then, $\hat{\mathcal{A}}$ accepts $\boldsymbol{w}_{v'}$ for every sequence v' of v that agrees with v on its assignments to $x_1, \ldots, x_m$, and in particular, for such sequences whose range is $\{w_1, \ldots, w_m\}$. Therefore, by the semantics of the quantifiers, we have that $\{w_1, \ldots, w_m\}$ is in $\mathfrak{L}(\mathcal{A})$. The second direction is trivial.

We call $\boldsymbol{w}_{v'}$ as described above a *witness to the nonemptiness of* $\mathcal{A}$. We construct an NFA A based on $\hat{\mathcal{A}}$ that is nonempty iff $\hat{\mathcal{A}}$ accepts a witness to the nonemptiness of $\mathcal{A}$.

Let Γ be the set of all functions of the type $\zeta : [1, k] \to [1, m]$ such that $\zeta(i) = i$ for every $i \in [1, m]$, and such that $\mathsf{range}(\zeta) = [1, m]$. For a letter assignment $f = \{\sigma_{1_{x_1}}, \ldots \sigma_{k_{x_k}}\}$, we denote by f_ζ the letter assignment $\{\sigma_{\zeta(1)_{x_1}}, \ldots, \sigma_{\zeta(k)_{x_k}}\}$.

For every function $\zeta \in \Gamma$, we construct an NFA $A_\zeta = \langle \hat{\Sigma}, Q, Q_0, \delta_\zeta, F \rangle$, where for every $q \xrightarrow{g} q'$ in δ, we have $q \xrightarrow{f} q'$ in δ_ζ, for every f that occurs in $\hat{\mathcal{A}}$ for which $f_\zeta = g$. Intuitively, for every run of A_ζ on a word $\boldsymbol{w}$ there exists a similar run of $\hat{\mathcal{A}}$ on the sequence of $\boldsymbol{w}$ that matches ζ. Therefore, $\hat{\mathcal{A}}$ accepts a witness $\boldsymbol{w}$ to the nonemptiness of $\mathcal{A}$ iff $\boldsymbol{w} \in \mathcal{L}(A_\zeta)$ for every $\zeta \in \Gamma$.

We define $A = \bigcap_{\zeta \in \Gamma} A_\zeta$. Then $\hat{\mathcal{A}}$ accepts a witness to the nonemptiness of $\mathcal{A}$ iff A is nonempty.

Since $|\Gamma| = m^{k-m}$, the state space of A is of size $O(n^{m^{k-m}})$, where $n = |Q|$, and its alphabet is of size $|\hat{\Sigma}|$. Notice that for $\mathcal{A}$ to be nonempty, δ must be of size at least $|(\Sigma \cup \#)|^{(k-m)}$, to account for all the sequences of letters in the words assigned to the variables under $\forall$ quantifiers (otherwise, we can immediately return "empty"). Therefore, $|\hat{\mathcal{A}}|$ is $O(n \cdot |\Sigma|^k)$. We then have that the size of A is $O(|\hat{\mathcal{A}}|^k)$. If the number

$k-m$ of $\forall$ quantifiers is fixed, then m^{k-m} is polynomial in k. However, now $|\hat{\mathcal{A}}|$ may be polynomial in n, k, and $|\Sigma|$, and so in this case as well, the size of A is $O(|\hat{A}|^k)$.

Since the nonemptiness problem for NFA is NL-complete, the problem for NFH$_{\exists\forall}$ can be decided in space of size that is polynomial in $|\hat{\mathcal{A}}|$.

General NFH. We mimic the proof idea in [10], which uses a reduction from the *Post correspondence problem (PCP)*, which is known to be undecidable. A PCP instance is a collection C of dominoes of the form:

$$\left\{ \left[\frac{u_1}{v_1}\right], \left[\frac{u_2}{v_2}\right], \ldots, \left[\frac{u_k}{v_k}\right] \right\}$$

where for all $i \in [1,k]$, we have $v_i, u_i \in \{a,b\}^*$. The problem is to decide whether there exists a finite sequence of the dominoes of the form

$$\left[\frac{u_{i_1}}{v_{i_1}}\right]\left[\frac{u_{i_2}}{v_{i_2}}\right] \cdots \left[\frac{u_{i_m}}{v_{i_m}}\right]$$

where each index $i_j \in [1,k]$, such that the upper and lower finite strings of the dominoes are equal, i.e.,

$$u_{i_1} u_{i_2} \cdots u_{i_m} = v_{i_1} v_{i_2} \cdots v_{i_m}$$

For example, if the set of dominoes is

$$C_{\mathsf{exmp}} = \left\{ \left[\frac{ab}{b}\right], \left[\frac{ba}{a}\right], \left[\frac{a}{aba}\right] \right\}$$

Then, a possible solution is the following sequence of dominoes from C_{exmp}:

$$\mathsf{sol} = \left[\frac{a}{aba}\right]\left[\frac{ba}{a}\right]\left[\frac{ab}{b}\right].$$

Given an instance C of PCP, we encode a solution as a word w_{sol} over the following alphabet:

$$\Sigma = \left\{ \frac{\sigma}{\sigma'} \mid \sigma, \sigma' \in \{a, b, \dot{a}, \dot{b}, \$\} \right\}.$$

Intuitively, $\dot{\sigma}$ marks the beginning of a new domino, and \$ marks the end of a sequence of the upper or lower parts of the dominoes sequence.

We note that w_{sol} encodes a legal solution iff the following conditions are met:

1. For every $\frac{\sigma}{\sigma'}$ that occurs in w_{sol}, it holds that σ, σ' represent the same domino letter (both a or both b, either dotted or undotted).
2. The number of dotted letters in the upper part of w_{sol} is equal to the number of dotted letters in the lower part of w_{sol}.
3. w_{sol} starts with two dotted letters, and the word u_i between the i'th and $i+1$'th dotted letters in the upper part of w_{sol}, and the word v_i between the corresponding dotted letters in the lower part of w_{sol} are such that $\left[\frac{u_i}{v_i}\right] \in C$, for every i.

We call a word that represents the removal of the first k dominoes from w_{sol} a *partial solution*, denoted by $w_{sol,k}$. Note that the upper and lower parts of $w_{sol,k}$ are not necessarily of equal lengths (in terms of a and b sequences), since the upper and lower parts of a domino may be of different lengths, and so we use letter \$ to pad the end of the encoding in the shorter of the two parts.

We construct an NFH $\mathcal{A}$, which, intuitively, expresses the following ideas: (1) There exists an encoding w_{sol} of a solution to C, and (2) For every $w_{sol,k} \neq \epsilon$ in a hyperword S accepted by $\mathcal{A}$, the word $w_{sol,k+1}$ is also in S.

$\mathfrak{L}(\mathcal{A})$ is then the set of all hyperwords that contain an encoded solution w_{sol}, as well as all its suffixes obtained by removing a prefix of dominoes from w_{sol}. This ensures that w_{sol} indeed encodes a legal solution. For example, a matching hyperword S (for the solution sol discussed earlier) that is accepted by $\mathcal{A}$ is:

$$S = \{w_{sol} = \frac{\dot{a}}{\dot{a}}\frac{\dot{b}}{\bar{b}}\frac{a}{\bar{a}}\frac{\dot{a}}{\bar{\dot{a}}}\frac{b}{\bar{\dot{b}}}, w_{sol,1} = \frac{\dot{b}}{\dot{a}}\frac{a}{\bar{\dot{b}}}\frac{\dot{a}}{\bar{\$}}\frac{b}{\bar{\$}}, w_{sol,2} = \frac{\dot{a}}{\bar{\dot{b}}}\frac{b}{\bar{\$}}, w_{sol,3} = \epsilon\}$$

Thus, the quantification condition of $\mathcal{A}$ is $\alpha = \forall x_1 \exists x_2 \exists x_3$, where x_1 is to be assigned a potential partial solution $w_{sol,k}$, and x_2 is to be assigned $w_{sol,k+1}$, and x_3 is to be assigned w_{sol}.

During a run on a hyperword S and an assignment $v : \{x_1, x_2, x_3\} \to S$, the NFH $\mathcal{A}$ checks that the upper and lower letters of w_{sol} all match. In addition, $\mathcal{A}$ checks that the first domino of $v(x_1)$ is indeed in C, and that $v(x_2)$ is obtained from $v(x_1)$ by removing the first tile. $\mathcal{A}$ performs the latter task by checking that the upper and lower parts of $v(x_2)$ are the upper and lower parts of $v(x_1)$ that have been "shifted" back appropriately. That is, if the first tile in $v(x_2)$ is the encoding of $[\frac{w_i}{v_i}]$, then $\mathcal{A}$ uses states to remember, at each point, the last $|w_i|$ letters of the upper part of $v(x_2)$ and the last $|v_i|$ letters of the lower part of $v(x_2)$, and verifies, at each point, that the next letter in $v(x_1)$ matches the matching letter remembered by the state. □