Complexity of Universality and Related Problems for Partially Ordered NFAs

Markus Krötzsch[a,1], Tomáš Masopust[a,b,1,*], Michaël Thomazo[c]

[a] *Institute of Theoretical Computer Science and Center of Advancing Electronics Dresden (cfaed), TU Dresden, Germany*
[b] *Institute of Mathematics, Czech Academy of Sciences, Žižkova 22, 616 62 Brno, Czechia*
[c] *Inria, France*

Abstract

Partially ordered nondeterministic finite automata (poNFAs) are NFAs whose transition relation induces a partial order on states, that is, for which cycles occur only in the form of self-loops on a single state. A poNFA is universal if it accepts all words over its input alphabet. Deciding universality is PSpace-complete for poNFAs, and we show that this remains true even when restricting to a fixed alphabet. This is nontrivial since standard encodings of alphabet symbols in, e.g., binary can turn self-loops into longer cycles. A lower coNP-complete complexity bound can be obtained if we require that all self-loops in the poNFA are deterministic, in the sense that the symbol read in the loop cannot occur in any other transition from that state. We find that such restricted poNFAs (rpoNFAs) characterize the class of $\mathcal{R}$-trivial languages, and we establish the complexity of deciding if the language of an NFA is $\mathcal{R}$-trivial. Nevertheless, the limitation to fixed alphabets turns out to be essential even in the restricted case: deciding universality of rpoNFAs with unbounded alphabets is PSpace-complete. Based on a close relation between universality and the problems of inclusion and equivalence, we also obtain the complexity results for these two problems. Finally, we show that the languages of rpoNFAs are definable by deterministic (one-unambiguous) regular expressions, which makes them interesting in schema languages for XML data.

Keywords: Automata, Nondeterminism, Partial order, Universality, Inclusion, Equivalence
2010 MSC: 68Q45, 68Q17, 68Q25, 03D05

1. Introduction

The universality problem asks if a given automaton (or grammar) accepts (or generates) all possible words over its alphabet. In typical cases, deciding universality is more difficult than deciding the word problem. For example, universality is undecidable for context-free grammars [3] and PSpace-complete for nondeterministic finite automata (NFAs) [29]. The study of universality (and its complement, emptiness) has a long tradition in formal languages, with many applications across computer science, e.g., in the context of formal knowledge representation and database theory [4, 10, 38]. Recent studies investigate the problem for specific types of automata or grammars, e.g., for prefixes or factors of regular languages [32].

In this paper, we are interested in the universality problem for *partially ordered NFAs* (poNFAs) and special cases thereof. An NFA is partially ordered if its transition relation induces a partial order on states: the only cycles allowed are self-loops on a single state. Partially ordered NFAs define a natural class of languages that has been shown to coincide with level $\frac{3}{2}$ of the Straubing-Thérien hierarchy [35] and with Alphabetical Pattern Constraint (APC) languages, a subclass of regular languages effectively closed under permutation rewriting [6]. Deciding whether an automaton recognizes an APC language (and hence whether it can be recognized by a poNFA) is PSpace-complete for NFAs and NL-complete for DFAs [6].

[*] Corresponding author
Email addresses: `markus.kroetzsch@tu-dresden.de` (Markus Krötzsch), `masopust@math.cas.cz` (Tomáš Masopust), `michael.thomazo@inria.fr` (Michaël Thomazo)

[1] This work was supported by the German Research Foundation (DFG) in Emmy Noether grant KR 4381/1-1 (DIAMOND).

Preprint submitted to Elsevier *June 26, 2017*

	Unary alphabet	Fixed alphabet	Arbitrary alphabet
DFA	L-comp. [21]	NL-comp. [21]	NL-comp. [21]
rpoNFA	NL-comp. (Cor. 25)	coNP-comp. (Cor. 24)	PSpace-comp. (Thm. 28)
poNFA	NL-comp. (Thm. 4)	PSpace-comp. (Thm. 3)	PSpace-comp. [1]
NFA	coNP-comp. [39]	PSpace-comp. [1]	PSpace-comp. [1]

Table 1: Complexity of deciding universality

Restricting to partially ordered deterministic finite automata (poDFAs), we can capture further classes of interest: two-way poDFAs characterize languages whose syntactic monoid belongs to the variety **DA** [35], introduced by Schützenberger [34]; poDFAs characterize $\mathcal{R}$-trivial languages [8]; and confluent poDFAs characterize level 1 of the Straubing-Thérien hierarchy, also known as $\mathcal{J}$-trivial languages or piecewise testable languages [37]. Other relevant classes of partially ordered automata include partially ordered Büchi automata [24] and two-way poDFAs with look-around [25].

The first result on the complexity of universality for poNFAs is readily obtained. It is well known that universality of regular expressions is PSpace-complete [1, Lemma 10.2], and it is easy to verify that the regular expressions used in the proof can be expressed in poNFAs:

Corollary 1 (Lemma 10.2 [1])**.** *The universality problem for poNFAs is* PSpace*-complete.*

A closer look at the proof reveals that the underlying encoding requires an alphabet of size linear in the input: PSpace-hardness is not established for alphabets of bounded size. Usually, one could simply encode alphabet symbols σ by sequences $\sigma_1 \cdots \sigma_n$ of symbols from a smaller alphabet, say $\{0, 1\}$. However, doing this requires self-loops $q \xrightarrow{\sigma} q$ to be replaced by nontrivial cycles $q \xrightarrow{\sigma_1} \cdots \xrightarrow{\sigma_n} q$, which are not permitted in poNFAs.

We settle this open problem by showing that PSpace-hardness is retained even for binary alphabets. This negative result leads us to ask if there is a natural subclass of poNFAs for which universality does become simpler. We consider *restricted* poNFAs (rpoNFAs), which require self-loops to be deterministic in the sense that the automaton contains no transition as in Figure 1, which we call *nondeterministic self-loops* in the rest of the paper. Large parts of the former hardness proof hinge on transitions of this form, which, speaking intuitively, allow the automaton to navigate to an arbitrary position in the input (using the loop) and, thereafter, continue checking an arbitrary pattern. Indeed, we find that the universality becomes coNP-complete for rpoNFAs with a fixed alphabet.

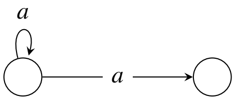

Figure 1: Nondeterministic self-loops – the forbidden pattern of rpoNFAs

However, this reduction of complexity is not preserved for unrestricted alphabets. We use a novel construction of rpoNFAs that characterize certain exponentially long words to show that universality is PSpace-complete even for rpoNFAs if the alphabet may grow polynomially. Our complexity results are summarized in Table 1.

As a by-product, we show that rpoNFAs provide another characterization of $\mathcal{R}$-trivial languages introduced and studied by Brzozowski and Fich [8], and we establish the complexity of detecting $\mathcal{R}$-triviality and k-$\mathcal{R}$-triviality for rpoNFAs.

From the practical point of view, the problems of inclusion and equivalence of two languages, which are closely related to universality, are of interest, e.g., in optimization. Indeed, universality can be expressed either as the inclusion $\Sigma^* \subseteq L$ or as the equivalence $\Sigma^* = L$. Although equivalence can be seen as two inclusions, the complexity of inclusion does not play the role of a lower bound. For instance, for two deterministic context-free languages inclusion is undecidable [14], whereas equivalence is decidable [36]. However, the complexity of universality gives a lower bound on the complexity of both inclusion and equivalence, and we show that, for the partially ordered NFAs studied in this paper, the complexities of inclusion and equivalence coincide with the complexity of universality.

This paper is a full version of the work [23] presented at the 41st International Symposium on Mathematical Foundations of Computer Science.

2. Preliminaries and Definitions

We assume that the reader is familiar with automata theory [1]. The cardinality of a set A is denoted by $|A|$ and the power set of A by 2^A. An *alphabet* Σ is a finite nonempty set. A *word* over Σ is any element of the free monoid Σ^*, the *empty word* is denoted by ε. A *language* over Σ is a subset of Σ^*. For a language L over Σ, let $\overline{L} = \Sigma^* \setminus L$ denote its complement.

A *subword* of w is a word u such that $w = w_1 u w_2$, for some words w_1, w_2; u is a *prefix* of w if $w_1 = \varepsilon$ and it is a *suffix* of w if $w_2 = \varepsilon$.

A *nondeterministic finite automaton* (NFA) is a quintuple $\mathcal{A} = (Q, \Sigma, \cdot, I, F)$, where Q is a finite nonempty set of states, Σ is an input alphabet, $I \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of accepting states, and $\cdot \colon Q \times \Sigma \to 2^Q$ is the transition function that can be extended to the domain $2^Q \times \Sigma^*$ by induction. The language *accepted* by $\mathcal{A}$ is the set $L(\mathcal{A}) = \{w \in \Sigma^* \mid I \cdot w \cap F \neq \emptyset\}$. We often omit $\cdot$ and write simply Iw instead of $I \cdot w$. The NFA $\mathcal{A}$ is *complete* if for every state q and every letter a in Σ, the set $q \cdot a$ is nonempty. It is *deterministic* (DFA) if $|I| = 1$ and $|q \cdot a| = 1$ for every state q in Q and every letter a in Σ.

A *path* π from a state q_0 to a state q_n under a word $a_1 a_2 \cdots a_n$, for some $n \geq 0$, is a sequence of states and input symbols $q_0 a_1 q_1 a_2 \cdots q_{n-1} a_n q_n$ such that $q_{i+1} \in q_i \cdot a_{i+1}$, for $i = 0, 1, \ldots, n-1$. Path π is *accepting* if $q_0 \in I$ and $q_n \in F$. A path is *simple* if all the states are pairwise distinct.

A *deterministic Turing machine* (DTM) is a tuple $M = \langle Q, T, I, \gamma, \sqcup, q_o, q_f \rangle$, where Q is the finite state set, T is the tape alphabet, $I \subseteq T$ is the input alphabet, $\sqcup \in T \setminus I$ is the blank symbol, q_o is the initial state, q_f is the accepting state, and γ is the transition function mapping $Q \times T$ to $Q \times T \times \{L, R, S\}$, see Aho et al. [1] for details.

The *universality problem* asks, given an automaton $\mathcal{A}$ over Σ, whether $L(\mathcal{A}) = \Sigma^*$. The *inclusion problem* asks, given two automata $\mathcal{A}$ and $\mathcal{B}$ over a common alphabet, whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$, and the *equivalence problem* asks whether $L(\mathcal{A}) = L(\mathcal{B})$.

3. Partially Ordered NFAs

In this section, we introduce poNFAs, recall their characterization in terms of the Straubing-Thérien hierarchy, and show that universality remains PSPACE-complete even when restricting to binary alphabets. Merely the case of unary alphabets turns out to be simpler.

Definition 2. Let $\mathcal{A}$ be an NFA. A state q is *reachable* from a state p, written $p \leq q$, if there exists a word $w \in \Sigma^*$ such that $q \in p \cdot w$. We write $p < q$ if $p \leq q$ and $p \neq q$. $\mathcal{A}$ is a *partially ordered NFA* (*poNFA*) if $\leq$ is a partial order.

The expressive power of poNFAs can be characterized by the *Straubing-Thérien (ST) hierarchy* [40, 42]. For an alphabet Σ, level 0 of this hierarchy is defined as $\mathscr{L}(0) = \{\emptyset, \Sigma^*\}$. For integers $n \geq 0$, the levels $\mathscr{L}(n + \frac{1}{2})$ and $\mathscr{L}(n+1)$ are as follows:

- $\mathscr{L}(n + \frac{1}{2})$ consists of all finite unions of languages $L_0 a_1 L_1 a_2 \cdots a_k L_k$, with $k \geq 0$, $L_0, \ldots, L_k \in \mathscr{L}(n)$, and $a_1, \ldots, a_k \in \Sigma$;
- $\mathscr{L}(n+1)$ consists of all finite Boolean combinations of languages from level $\mathscr{L}(n + \frac{1}{2})$.

Note that the levels of the hierarchy contain only *star-free* languages by definition. It is known that the hierarchy does not collapse on any level [9], but the problem of deciding if a language belongs to some level k is largely open for $k > \frac{7}{2}$ [2, 30, 31]. The ST hierarchy further has close relations to the *dot-depth hierarchy* [9, 11, 41] and to complexity theory [43].

Interestingly, the languages recognized by poNFAs are exactly the languages on level $\frac{3}{2}$ of the Straubing-Thérien hierarchy [35]. Since the hierarchy is proper, this means that poNFAs can only recognize a strict subset of star-free regular languages. In spite of this rather low expressive power, the universality problem of poNFAs has the same worst-case complexity as for general NFAs, even when restricting to a fixed alphabet with only a few letters.

Theorem 3. *For every alphabet Σ with $|\Sigma| \geq 2$, the universality problem for poNFAs over Σ is* PSPACE*-complete.*

Proof. Membership follows from the fact that universality is in PSpace for NFAs [15].

To show hardness, we modify the construction of Aho et al. [1, Section 10.6] to work on a two-letter alphabet. Consider a polynomial p and a p-space-bounded DTM $M = \langle Q, T, I, \gamma, \sqcup, q_o, q_f \rangle$. Without loss of generality, we assume that $q_o \neq q_f$. We define an encoding of runs of M as a word over a given alphabet. For any input $x \in I^*$, we construct, in polynomial time, a regular expression R_x that represents all words that do *not* encode an accepting run of M on x. Therefore, R_x matches all words if and only if M does not accept x. The claim then follows by showing that R_x can be encoded by a poNFA.

A configuration of M on an input x consists of a current state $q \in Q$, the position $1 \leq \ell \leq p(|x|)$ of the read/write head, and the current tape contents $\theta_1, \ldots, \theta_{p(|x|)}$ with $\theta_i \in T$. We represent it by a sequence

$$\langle \theta_1, \varepsilon \rangle \cdots \langle \theta_{\ell-1}, \varepsilon \rangle \langle \theta_\ell, q \rangle \langle \theta_{\ell+1}, \varepsilon \rangle \cdots \langle \theta_{p(|x|)}, \varepsilon \rangle$$

of symbols from $T \times (Q \cup \{\varepsilon\})$. We denote $T \times (Q \cup \{\varepsilon\})$ by Δ. A potential run of M on x is represented by word $\# w_1 \# w_2 \# \cdots \# w_m \#$, where $w_i \in \Delta^{p(|x|)}$ and $\# \notin \Delta$ is a fresh separator symbol. One can construct a regular expression recognizing all words over $\Delta \cup \{\#\}$ that do not correctly encode a run of M at all, or that encode a run that is not accepting [1].

We encode symbols of $\Delta \cup \{\#\}$ using the fixed alphabet $\Sigma = \{0, 1\}$. For each $\delta \in \Delta \cup \{\#\}$, let $\hat{\delta}_1 \cdots \hat{\delta}_K \in \{0, 1\}^K$ be the unique binary encoding of length $K = \lceil \log_2 (|\Delta \cup \{\#\}|) \rceil$. We define $\mathrm{enc}(\delta)$ to be the binary sequence

$$\mathrm{enc}(\delta) = 001\hat{\delta}_1 1 \hat{\delta}_2 1 \cdots \hat{\delta}_K 1$$

of length $L = 2K + 3$. We extend enc to words and sets of symbols as usual: $\mathrm{enc}(\delta_1 \cdots \delta_m) = \mathrm{enc}(\delta_1) \cdots \mathrm{enc}(\delta_m)$ and $\mathrm{enc}(\Delta') = \{\mathrm{enc}(\delta) \mid \delta \in \Delta'\}$. Importantly, any word of the form $\mathrm{enc}(\delta_1 \cdots \delta_m)$ contains 00 only at positions that are multiples of L, marking the start of one encoded symbol.

We now construct the regular expression R_x that matches all words of Σ^* that do not represent an accepting computation of M on x. We proceed in four steps:

(A) We detect all words that contain words from Σ^* that are not of the form $\mathrm{enc}(\delta)$;

(B) We detect all words that do not start with the initial configuration;

(C) We detect all words that do not encode a valid run since they violate a transition rule; and

(D) We detect all words that encode non-accepting runs, or runs that end prematurely.

For (A), note that a word $w \in \Sigma^*$ that is not of the form $\mathrm{enc}(v)$ for any word $v \in (\Delta \cup \{\#\})^*$ must either (A.1) start with 1 or 01; (A.2) end with 0; (A.3) contain a word $00\Sigma^{L-2}$ that is not in $\mathrm{enc}(\Delta \cup \{\#\})$; (A.4) contain a word from $\mathrm{enc}(\Delta \cup \{\#\})\{1, 01\}$; or (A.5) end in a word $00\Sigma^M$ with $M < L - 2$. Using E to abbreviate $\mathrm{enc}(\Delta \cup \{\#\})$ and $\bar{E}$ to abbreviate $00\Sigma^{L-2} \setminus E$ (both sets of polynomially many binary sequences), we can express (A.1)–(A.5) in the regular expression

$$(1\Sigma^* + 01\Sigma^*) + (\Sigma^* 0) + \left(\Sigma^* \bar{E} \Sigma^*\right) + (\Sigma^* E (1 + 01) \Sigma^*) + \left(\Sigma^* 00 (\Sigma + \Sigma^2 + \cdots + \Sigma^{L-3})\right) \tag{1}$$

where we use finite sets $\{e_1, \ldots, e_m\}$ to denote regular expressions $(e_1 + \cdots + e_m)$, as usual. All sets in (1) are polynomial in size, so that the overall expression is polynomial. The expression (1) can be captured by a poNFA since the only cycles required arise when translating Σ^*; they can be expressed as self-loops. All other repetitions of the form Σ^i in (1) can be expanded to polynomial-length sequences without cycles.

For (B), we want to detect all words that do not start with the word

$$w = \mathrm{enc}(\# \langle x_1, q_0 \rangle \langle x_2, \varepsilon \rangle \cdots \langle x_{|x|}, \varepsilon \rangle \langle \sqcup, \varepsilon \rangle \cdots \langle \sqcup, \varepsilon \rangle \#) = \mathrm{enc}(v_0 v_1 \cdots v_{p(x)+1})$$

of length $(p(|x|) + 2)L$. This happens if (B.1) the word is shorter than $(p(|x|) + 2)L$, or (B.2), starting at position jL for $0 \leq j \leq p(|x|) + 1$, there is a word from the polynomial set $\Sigma^L \setminus \{\mathrm{enc}(v_j)\}$, which we abbreviate by $\bar{E}_j$. We can capture (B.1) and (B.2) in the regular expression

$$\left(\varepsilon + \Sigma + \Sigma^2 + \cdots + \Sigma^{L(p(|x|)+2)-1}\right) + \sum_{0 \leq j \leq p(|x|)+1} (\Sigma^{jL} \cdot \bar{E}_j \cdot \Sigma^*) \tag{2}$$

The empty expression ε is used for readability; it can easily be expressed in the NFA encoding. As before, it is easy to see that this expression is polynomial and does not require any nontrivial cycles when encoded in an NFA. Note that we ensure that the surrounding # in the initial configuration are present.

For (C), we need to check for incorrect transitions. Consider again the encoding $\#w_1\#\cdots\#w_m\#$ of a sequence of configurations with a word over $\Delta \cup \{\#\}$, where we can assume that w_1 encodes the initial configuration according to (A) and (B). In an encoding of a valid run, the symbol at any position $j \geq p(|x|) + 2$ is uniquely determined by the symbols at positions $j - p(|x|) - 2$, $j - p(|x|) - 1$, and $j - p(|x|)$, corresponding to the cell and its left and right neighbor in the previous configuration. Given symbols $\delta_\ell, \delta, \delta_r \in \Delta \cup \{\#\}$, we can therefore define $f(\delta_\ell, \delta, \delta_r) \in \Delta \cup \{\#\}$ to be the symbol required in the next configuration. The case where $\delta_\ell = \#$ or $\delta_r = \#$ corresponds to transitions applied at the left and right edge of the tape, respectively; for the case that $\delta = \#$, we define $f(\delta_\ell, \delta, \delta_r) = \#$, ensuring that the separator # is always present in successor configurations as well. We can then check for invalid transitions using the regular expression

$$\sum_{\delta_\ell,\delta,\delta_r \in \Delta\cup\{\#\}} \Sigma^* \cdot \mathrm{enc}(\delta_\ell\delta\delta_r) \cdot \Sigma^{L(p(|x|)-1)} \cdot \mathrm{enc}(\overline{f}(\delta_\ell, \delta, \delta_r)) \cdot \Sigma^* \tag{3}$$

where $\overline{f}(\delta_\ell, \delta, \delta_r) = \Delta \cup \{\#\} \setminus \{f(\delta_\ell, \delta, \delta_r)\}$. Polynomiality and poNFA-expressibility are again immediate. Note that expression (3) only detects wrong transitions if a (long enough) next configuration exists. The case that the run stops prematurely is covered next.

Finally, for (D) we detect all words that either (D.1) end in a configuration that is incomplete (too short) or (D.2) end in a configuration that is not in the final state q_f. Abbreviating $\mathrm{enc}(T \times (Q \setminus \{q_f\}))$ as $\bar{E}_f$, and using similar ideas as above, we obtain

$$\Big(\Sigma^* \,\mathrm{enc}(\#)(\Sigma^L + \cdots + \Sigma^{p(|x|)L})\Big) + \Big(\Sigma^* \bar{E}_f(\varepsilon + \Sigma^L + \cdots + \Sigma^{(p(|x|)-1)L})\,\mathrm{enc}(\#)\Big) \tag{4}$$

and this can again be expressed as a polynomial poNFA.

The expressions (1)–(4) together then detect all non-accepting or wrongly encoded runs of M. In particular, if we start from the correct initial configuration ((2) does not match), then for (3) not to match, all complete future configurations must have exactly one state and be delimited by encodings of #. Expressing the regular expressions as a single poNFA of polynomial size, we have thus reduced the word problem of polynomially space-bounded Turing machines to the universality problem of poNFAs. □

Ellul et al. [13, Section 5] give an example of a regular expression over a 5-letter alphabet such that the shortest non-accepted word is of exponential length, and which can also be encoded as a poNFA. Our previous proof shows such an example for an alphabet of two letters, if we use a Turing machine that runs for exponentially many steps before accepting. Note, however, that this property alone would not imply Theorem 3.

Unary Alphabet

Reducing the size of the alphabet to one leads to a reduction in complexity. This is expected, since the universality problem for NFAs over a unary alphabet is merely coNP-complete [39]. For poNFAs, the situation is even simpler:

Theorem 4. *The universality problem for poNFAs over a unary alphabet is* NL*-complete. It can be checked in linear time.*

Proof. Let $\mathcal{A}$ be a poNFA over the alphabet $\{a\}$, and let n be the number of states in $\mathcal{A}$. Language $L(\mathcal{A})$ is infinite if and only if a word of length n is accepted by $\mathcal{A}$. If a^n is accepted, then there must be a simple path from an initial state to an accepting state via a state with a self-loop. Therefore, all words of length n or more are accepted. It remains to check that $\varepsilon, a, \ldots, a^n$ are accepted, which amounts to n acceptance checks that can be realized in nondeterministic logarithmic space. Notice that, using linear space, these checks altogether can be done in linear time. Hardness can be shown by reducing the NL-complete DAG-reachability problem [21]. Let G be a directed acyclic graph, and let s and t be two nodes of G. We define a poNFA $\mathcal{A}$ as follows. With each node of G, we associate a state in $\mathcal{A}$. Whenever there is an edge from i to j in G, we add an a-transition from i to j in $\mathcal{A}$. We add a self-loop labeled by a to t. The initial state of $\mathcal{A}$ is state s, all states are final. Then $\mathcal{A}$ is universal if and only if there is a path from s to t in G. □

4. Restricted Partially Ordered NFAs

We now introduce restricted poNFAs, which are distinguished by deterministic self-loops. We relate them to the known class of $\mathcal{R}$-trivial languages, and we establish complexity results for deciding if a language falls into this class.

Definition 5. A *restricted partially ordered NFA (rpoNFA)* is a poNFA such that, for every state q and symbol a, if $q \in q \cdot a$ then $q \cdot a = \{q\}$.

We will show below that rpoNFAs characterize $\mathcal{R}$-trivial languages [8]. To introduce this class of languages, we first require some auxiliary definitions. A word $v = a_1 a_2 \cdots a_n$ is a *subsequence* of a word w, denoted by $v \preccurlyeq w$, if $w \in \Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots \Sigma^* a_n \Sigma^*$. For $k \geq 0$, we write $\mathrm{sub}_k(v) = \{u \in \Sigma^* \mid u \preccurlyeq v, |u| \leq k\}$ for the set of all subsequences of v of length up to k. Two words w_1, w_2 are *$\sim_k$-equivalent*, written $w_1 \sim_k w_2$, if $\mathrm{sub}_k(w_1) = \mathrm{sub}_k(w_2)$. Then $\sim_k$ is a congruence (for concatenation) of finite index (i.e., with finitely many equivalence classes) [37]. $\mathcal{R}$-trivial languages are defined by defining a related congruence $\sim_k^{\mathcal{R}}$ that considers subsequences of prefixes:

Definition 6. Let $x, y \in \Sigma^*$ and $k \geq 0$. Then $x \sim_k^{\mathcal{R}} y$ if and only if

- for each prefix u of x, there exists a prefix v of y such that $u \sim_k v$, and
- for each prefix v of y, there exists a prefix u of x such that $u \sim_k v$.

A regular language is *k-$\mathcal{R}$-trivial* if it is a union of $\sim_k^{\mathcal{R}}$ classes, and it is *$\mathcal{R}$-trivial* if it is k-$\mathcal{R}$-trivial for some $k \geq 0$.

It is known that $x \sim_k^{\mathcal{R}} y$ implies $x \sim_k y$ and (if $k \geq 1$) $x \sim_{k-1}^{\mathcal{R}} y$ [8]. Therefore, every k-$\mathcal{R}$-trivial language is also $(k+1)$-$\mathcal{R}$-trivial. Moreover, it has been shown that a language L is $\mathcal{R}$-trivial if and only if the minimal DFA recognizing L is partially ordered [8]. We can lift this result to characterize the expressive power of rpoNFAs.

Theorem 7. *A regular language is $\mathcal{R}$-trivial if and only if it is accepted by an rpoNFA.*

Proof. Brzozowski and Fich [8] have shown that every $\mathcal{R}$-trivial language is accepted by a partially ordered DFA. As a partially ordered DFA is an rpoNFA, this concludes this direction.

To prove the other direction, notice that every rpoNFA can be decomposed into a finite number of DFAs. More specifically, let $\mathcal{A}$ over Σ be an rpoNFA. For a state q, let $\Sigma_q = \{a \in \Sigma \mid q \in q \cdot a\}$ be the set of all symbols that appear in self-loops in state q. Let $q_1 a_1 q_2 a_2 \cdots q_n a_n q_{n+1}$ be a simple accepting path in $\mathcal{A}$. Then it defines an expression $\Sigma_{q_1}^* a_1 \Sigma_{q_2}^* a_2 \cdots \Sigma_{q_n}^* a_n \Sigma_{q_{n+1}}^*$ with the property $a_i \notin \Sigma_{q_i}$ for $1 \leq i \leq n$. Since every NFA has only finitely many simple paths, the proof now follows from the results of Brzozowski and Fich [8], who have shown that a language is $\mathcal{R}$-trivial if and only if it is a finite union of $\mathcal{R}$-expressions, i.e., expressions of the form $\Sigma_1^* a_1 \Sigma_2^* a_2 \cdots \Sigma_m^* a_m \Sigma_{m+1}^*$, for some $m \geq 0$, where $a_i \notin \Sigma_i$ for $1 \leq i \leq m$. □

This characterization in terms of automata with forbidden patterns can be compared to results of Glaßer and Schmitz, who use DFAs with a forbidden pattern to obtain a characterization of level $\frac{3}{2}$ of the dot-depth hierarchy [16, 33].

We can further relate the *depth* of rpoNFAs to k-$\mathcal{R}$-trivial languages. Recall that the depth of an rpoNFA $\mathcal{A}$, denoted by $\mathrm{depth}(\mathcal{A})$, is the number of input symbols on a longest simple path of $\mathcal{A}$ that starts in an initial state.

Theorem 8. *The language recognized by a complete rpoNFA $\mathcal{A}$ is $\mathrm{depth}(\mathcal{A})$-$\mathcal{R}$-trivial.*

The proof of Theorem 8 follows from Lemmas 9 and 12 proved below.

Let p be a state of an NFA $\mathcal{A} = (Q, \Sigma, \cdot, I, F)$. The *sub-automaton* of $\mathcal{A}$ induced by state p is the automaton $\mathcal{A}_p = (\mathrm{reach}(p), \Sigma, \cdot_p, \{p\}, F \cap \mathrm{reach}(p))$ with state p being the sole initial state and with only those states of $\mathcal{A}$ that are reachable from p; formally, $\mathrm{reach}(p)$ denotes the set of all states reachable from state p in $\mathcal{A}$ and $\cdot_p$ is the restriction of $\cdot$ to $\mathrm{reach}(p) \times \Sigma$.

The following lemma is clear.

Lemma 9. *Let $\mathcal{A}$ be an rpoNFA with I denoting the set of initial states. Then the language $L(\mathcal{A}) = \bigcup_{i \in I} L(\mathcal{A}_i)$, where every sub-automaton $\mathcal{A}_i$ is an rpoNFA.*

Thus, it is sufficient to prove the theorem for rpoNFAs with a single initial state. Indeed, if $\mathcal{A}_i$ is of depth k_i, then its language is k_i-$\mathcal{R}$-trivial by Lemma 12. Since every k-$\mathcal{R}$-trivial language is also $(k+1)$-$\mathcal{R}$-trivial, the union of $L(\mathcal{A}_i)$ is $\max\{k_i \mid i \in I\}$-$\mathcal{R}$-trivial.

We need the following two lemmas first. For a word w, we denote by $\operatorname{alph}(w)$ the set of all letters occurring in w.

Lemma 10 ([22]). *Let $\ell \geq 1$, and let $x, y \in \Sigma^*$ be such that $x \sim_\ell y$. Let $x = x'ax''$ and $y = y'ay''$ such that $a \notin \operatorname{alph}(x'y')$. Then $x'' \sim_{\ell-1} y''$.*

Lemma 11. *Let $\ell \geq 1$, and let $x, y \in \Sigma^*$ be such that $x \sim_\ell^{\mathcal{R}} y$. Let $x = x'ax''$ and $y = y'ay''$ such that $a \notin \operatorname{alph}(x'y')$. Then $x'' \sim_{\ell-1}^{\mathcal{R}} y''$.*

Proof. Let u'' be a prefix of x''. Consider the prefix $u = x'au''$ of x. Since $x \sim_\ell^{\mathcal{R}} y$, there exists a prefix v of y such that $u \sim_\ell v$. Then $\ell \geq 1$ implies that letter a appears in v. Thus, we can write $v = y'av''$. By Lemma 10, $u'' \sim_{\ell-1} v''$. Thus, for any prefix u'' of x'', there exists a prefix v'' of y'' such that $u'' \sim_{\ell-1} v''$. Similarly the other way round, and therefore $x'' \sim_{\ell-1}^{\mathcal{R}} y''$. □

Lemma 12. *Let $\mathcal{A}$ be a complete rpoNFA with a single initial state and depth k. Then the language $L(\mathcal{A})$ is k-$\mathcal{R}$-trivial.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \cdot, i, F)$. If the depth of $\mathcal{A}$ is 0, then $L(\mathcal{A})$ is either $\emptyset$ or Σ^*, which are both 0-$\mathcal{R}$-trivial by definition. Thus, assume that the depth of $\mathcal{A}$ is $\ell \geq 1$ and that the claim holds for rpoNFAs of depth less than ℓ. Let $u, v \in \Sigma^*$ be such that $u \sim_\ell^{\mathcal{R}} v$. We prove that u is accepted by $\mathcal{A}$ if and only if v is accepted by $\mathcal{A}$.

Assume that the word u is accepted by $\mathcal{A}$ and fix an accepting path of u in $\mathcal{A}$. Let $\Sigma_i = \{a \in \Sigma \mid i \in i \cdot a\}$ denote the set of all letters under which there is a self-loop in state i. If $\operatorname{alph}(u) \subseteq \Sigma_i$, then the definition of rpoNFA $\mathcal{A}$ implies that $i \in F$. Since $\ell \geq 1$ implies that $\operatorname{alph}(u) = \operatorname{alph}(v)$, we have that v is also accepted in state i.

If $\operatorname{alph}(u) \nsubseteq \Sigma_i$, then

$$u = u'au'' \quad \text{and} \quad v = v'bv''$$

where $u', v' \in \Sigma_i^*$, $a, b \in \Sigma \setminus \Sigma_i$, and $u'', v'' \in \Sigma^*$. Let $p \in i \cdot a$ be a state on the fixed accepting path of u, and let $\mathcal{A}_p$ be the sub-automaton of $\mathcal{A}$ induced by state p. Notice that $\mathcal{A}_p$ is a complete rpoNFA of depth at most $\ell - 1$, and that $\mathcal{A}_p$ accepts u''.

If $a \neq b$, then $u = u'au_0bu_1$ and $v = v'bv_0av_1$, where the depicted a and b are the first occurrences of those letters from the left, that is, $b \notin \operatorname{alph}(u'au_0) \cup \operatorname{alph}(v')$ and $a \notin \operatorname{alph}(u') \cup \operatorname{alph}(v'bv_0)$. If $\ell = 1$, let $z = u'a$ be a prefix of u. Since $u \sim_1^{\mathcal{R}} v$, there exists a prefix t of v such that $z \sim_1 t$. Because $a \in \operatorname{alph}(z)$, we also have that $a \in \operatorname{alph}(t)$, which implies that $t = v'bv_0at'$, for some t' being a prefix of v_1. But then $b \in \operatorname{alph}(t) \setminus \operatorname{alph}(z)$, which is a contradiction with $z \sim_1 t$. If $\ell \geq 2$, let $z = u'au_0b$ be a prefix of u. Since $u \sim_\ell^{\mathcal{R}} v$, there exists a prefix t of v such that $z \sim_\ell t$. Because $a, b \in \operatorname{alph}(z)$, we also have that $a, b \in \operatorname{alph}(t)$, which implies that $t = v'bv_0at'$, for some t' being a prefix of v_1. But then $ba \in \operatorname{sub}_\ell(t) \setminus \operatorname{sub}_\ell(z)$, which is a contradiction with $z \sim_\ell t$. Thus, $u \sim_\ell^{\mathcal{R}} v$ implies that $a = b$.

If $a = b$, Lemma 11 implies that $u'' \sim_{\ell-1}^{\mathcal{R}} v''$. By the induction hypothesis, u'' is accepted by $\mathcal{A}_p$ if and only if v'' is accepted by $\mathcal{A}_p$. Hence, $v = v'av''$ is accepted by $\mathcal{A}$, which was to be shown. □

Proof (of Theorem 8). By Lemma 9 and the definition of k-$\mathcal{R}$-triviality, the language recognized by the rpoNFA $\mathcal{A}$ is $\operatorname{depth}(\mathcal{A})$-$\mathcal{R}$-trivial if the language recognized by each $\mathcal{A}_i$ is $\operatorname{depth}(\mathcal{A})$-$\mathcal{R}$-trivial. Since the depth of every $\mathcal{A}_i$ is at most the depth of $\mathcal{A}$, Lemma 12 concludes the proof. □

Similar relationships have been studied for $\mathcal{J}$-trivial languages [22, 27], but we are not aware of any such investigation for $\mathcal{R}$-trivial languages.

Finally, we may ask how difficult it is to decide whether a given NFA $\mathcal{A}$ accepts a language that is $\mathcal{R}$-trivial or k-$\mathcal{R}$-trivial for a specific $k \geq 0$. For most levels of the ST hierarchy, it is not even known if this problem is decidable, and when it is, exact complexity bounds are often missing [31]. The main exception are $\mathcal{J}$-trivial languages – level 1 of the hierarchy – which have recently attracted some attention, motivated by applications in algebra and XML databases [17, 22, 28].

To the best of our knowledge, the following complexity results for recognizing (k-)$\mathcal{R}$-trivial languages had not been obtained previously.

Theorem 13. *Given an NFA $\mathcal{A}$, it is* PSPACE*-complete to decide if the language accepted by $\mathcal{A}$ is $\mathcal{R}$-trivial.*

PROOF. The hardness follows from Theorem 3.1 in Hunt III and Rosenkrantz [20]. To decide whether the language $L(\mathcal{A})$ is $\mathcal{R}$-trivial means to check whether its equivalent (minimal) DFA is partially ordered. The non-partial-order of the DFA can be checked in PSPACE by nondeterministically guessing two reachable subsets of states and verifying that they are inequivalent and reachable from each other. This shows that $\mathcal{R}$-triviality is PSPACE-complete. □

To prove a similar claim for k-$\mathcal{R}$-triviality, we use some results from the literature.

Lemma 14 ([8])**.** *Every congruence class of $\sim_k^{\mathcal{R}}$ contains a unique element of minimal length. If $a_1, a_2, \ldots, a_n \in \Sigma$, then $a_1 a_2 \cdots a_n$ is minimal if and only if $\mathrm{sub}_k(\varepsilon) \subsetneq \mathrm{sub}_k(a_1) \subsetneq \mathrm{sub}_k(a_1 a_2) \subsetneq \cdots \subsetneq \mathrm{sub}_k(a_1 a_2 \cdots a_n)$.*

The maximal length of such a word has also been studied [28].

Lemma 15 ([28])**.** *Let Σ be an alphabet of cardinality $|\Sigma| \geq 1$, and let $k \geq 1$. The length of a longest word w such that $\mathrm{sub}_k(w) = \{v \in \Sigma^* \mid |v| \leq k\}$, and, for any two distinct prefixes w_1 and w_2 of w, $\mathrm{sub}_k(w_1) \neq \mathrm{sub}_k(w_2)$, is exactly $\binom{k+|\Sigma|}{k} - 1$.*

Lemmas 14 and 15 provide the main ingredients for showing membership in PSPACE.

Theorem 16. *Given an NFA $\mathcal{A}$ and $k \geq 0$, it is* PSPACE*-complete to decide if the language accepted by $\mathcal{A}$ is k-$\mathcal{R}$-trivial.*

PROOF. Again, the hardness follows from Theorem 3.1 in Hunt III and Rosenkrantz [20].

To prove the membership, let $\mathcal{A}$ be an NFA over an n-letter alphabet Σ. By definition, every k-$\mathcal{R}$-trivial language is a finite union of $\sim_k^{\mathcal{R}}$-classes. By Lemmas 14 and 15, every class has a unique shortest representative of length at most $\binom{k+n}{k} - 1$. Since k is a constant, this number is polynomial, $O(n^k)$. If $L(\mathcal{A})$ is not k-$\mathcal{R}$-trivial, then there exists a class $C_w = w/_{\sim_k^{\mathcal{R}}}$, where w is the unique shortest representative, such that $C_w \cap L(\mathcal{A}) \neq \emptyset$ and $C_w \cap \overline{L(\mathcal{A})} \neq \emptyset$. The nondeterministic algorithm can guess w and build the minimal DFA accepting the class C_w as described below. Having this, the intersections can be checked in PSPACE. (The non-emptiness of the intersection with a complemented NFA can be verified, for instance, by the on-the-fly determinization of the NFA and reverting the status of the reached state, or by building and alternating finite automaton and checking non-emptiness [18]).

We construct the minimal incomplete DFA $\mathcal{D}_w$ recognizing only the word w. It consists of $|w| + 1$ states labeled by prefixes of w so that the initial state is labeled with $[\varepsilon]$ and the only accepting state is labeled with $[w]$. The transitions are defined so that if $w = uau'$, then $[u] \cdot a = [ua]$. Now, for every prefix v of w and every letter b such that $\mathrm{sub}_k(v) = \mathrm{sub}_k(vb)$, we add the self-loop $[v] \cdot b = [v]$ to $\mathcal{D}_w$. Notice that for $w = uau'$, $\mathrm{sub}_k(u) \neq \mathrm{sub}_k(ua)$ by the properties of the unique shortest representative, c.f. Lemma 14, and therefore the construction produces a DFA. We make it complete by adding a sink state, if needed. Denote the obtained DFA by $\mathcal{D}$. We claim that $L(\mathcal{D}) = C_w$.

Claim 17. *$L(\mathcal{D}) \subseteq C_w$.*

PROOF. Let $w' \in L(\mathcal{D})$. We show that $w' \sim_k^{\mathcal{R}} w$. To do this, let $w = a_1 a_2 \cdots a_n$. Then, by the structure of $\mathcal{D}$, $w' = u_0 a_1 u_1 a_2 u_2 \cdots u_{n-1} a_n u_n$, for some words u_i that are read in self-loops of states $[a_1 a_2 \cdots a_i]$, for $0 \leq i \leq n$.

By definition of $\sim_k^{\mathcal{R}}$, we need to show that for each prefix u of w', there exists a prefix v of w such that $u \sim_k v$, and that for each prefix v of w, there exists a prefix u of w' such that $u \sim_k v$. We prove by induction on i, $0 \leq i \leq n$, that $u_0 a_1 u_1 a_2 u_2 \cdots a_i u_i' \sim_k a_1 a_2 \cdots a_i$, where u_i' is any prefix of u_i.

For $i = 0$, we show that $\mathrm{sub}_k(u_0') = \mathrm{sub}_k(\varepsilon)$ for any prefix u_0' of u_0. Since $[\varepsilon] \cdot u_0' = [\varepsilon]$ in $\mathcal{D}$, we have that $\mathrm{sub}_k(u_0') = \mathrm{sub}_k(\varepsilon)$. Indeed, if $u_0' = b_1 b_2 \cdots b_m$, then, by the construction of $\mathcal{D}$, $\varepsilon \sim_k b_j$, for $1 \leq j \leq m$. Since $\sim_k$ is a congruence, $\varepsilon \sim_k b_1 b_2 \cdots b_m = u_0'$.

Assume that it holds for $i-1$ and consider the prefixes $u_0 a_1 u_1 \cdots u_{i-1} a_i u_i'$ and $a_1 \cdots a_{i-1} a_i$, where u_i' is a prefix of u_i. By the induction hypothesis, $u_0 a_1 u_1 \cdots u_{i-1} \sim_k a_1 \cdots a_{i-1}$, and by the congruence property of $\sim_k$, we obtain that $u_0 a_1 u_1 \cdots u_{i-1} a_i \sim_k a_1 \cdots a_{i-1} a_i$. Let $u = u_0 a_1 u_1 \cdots u_{i-1} a_i$, $v = a_1 \cdots a_{i-1} a_i$, and $u_i' = c_1 c_2 \cdots c_s$. By the construction of $\mathcal{D}$, the state $[v]$ has self-loops under all letters c_j, which means that $v \sim_k vc_j$, for $1 \leq j \leq s$. It implies that $v \sim_k vu_i'$, because $vc_{j+1} \cdots c_s \sim_k vc_j c_{j+1} \cdots c_s$ using $v \sim_k vc_j$ and the property that $\sim_k$ is a congruence. Since $u \sim_k v$ implies that $vu_i' \sim_k uu_i'$, we have that $v \sim_k vu_i' \sim_k uu_i'$, which was to be shown.

Claim 18. $C_w \subsetneq L(\mathcal{D})$.

Proof. Let $w' \in \Sigma^*$ be such that $w' \sim_k^{\mathcal{R}} w$. We show that w' is accepted by $\mathcal{D}$. For the sake of contradiction, assume that w' does not belong to $L(\mathcal{D})$. Let w'_1 denote the longest prefix of w' that can be read by $\mathcal{D}$, that is, $w'_1 = u_0 a_1 u_1 \cdots a_i u_i$, where u_i's correspond to words read in self-loops and a_i to letters of w. Let $w_1 = a_1 a_2 \cdots a_i$ denote the corresponding prefix of w. Then $w = w_1 w_2$ and $w'_1 w_2$ is accepted by $\mathcal{D}$. By Claim 17, $w \sim_k^{\mathcal{R}} w'_1 w_2$; namely, $w'_1 = u_0 a_1 u_1 \cdots a_i u_i \sim_k a_1 a_2 \cdots a_i = w_1$. Since w' is not accepted by $\mathcal{D}$, there is a letter b such that $w'_1 b$ is a prefix of w' and it leads $\mathcal{D}$ to the sink state. Thus, $\mathrm{sub}_k(w_1) = \mathrm{sub}_k(w'_1) \subsetneq \mathrm{sub}_k(w'_1 b)$ by the construction of $\mathcal{D}$. Moreover, $w' \sim_k^{\mathcal{R}} w$ implies that there is a prefix v of w such that $w'_1 b \sim_k v$. Since $\mathrm{sub}_k(w_1) \subsetneq \mathrm{sub}_k(w'_1 b)$, there must be a letter a such that $v = w_1 a y$, for some word y. Notice that $a \neq b$. Since w is the unique minimal representative, $\mathrm{sub}_k(w_1) \subsetneq \mathrm{sub}_k(w_1 a)$, and hence there exists x such that $x \in \mathrm{sub}_k(w_1 a)$ and $x \notin \mathrm{sub}_k(w_1) = \mathrm{sub}_k(w'_1)$. Since $a \neq b$, $x \notin \mathrm{sub}_k(w'_1 b)$, which is a contradiction with $w'_1 b \sim_k v$.

This completes the proof of Theorem 16. □

In both previous theorems, hardness is shown by reduction from the universality problem for NFAs [1, 29]. Hence it holds even for binary alphabets. For a unary alphabet, we can obtain the following result.

Theorem 19. *Given an NFA $\mathcal{A}$ over a unary alphabet, the problems of deciding if the language accepted by $\mathcal{A}$ is $\mathcal{R}$-trivial, or k-$\mathcal{R}$-trivial for a given $k \geq 0$, are both* coNP*-complete.*

Proof. To show that $\mathcal{R}$-triviality for NFAs over a unary alphabet $\{a\}$ is in coNP, we show that non-$\mathcal{R}$-triviality is in NP. It requires to check that the corresponding DFA is not partially ordered, which is if and only if there are $0 \leq \ell_1 < \ell_2 < \ell_3 \leq 2^n$, where n is the number of states, such that $I \cdot a^{\ell_1} = I \cdot a^{\ell_3} \neq I \cdot a^{\ell_2}$, where I is the set of initial states, and one of these sets is accepting and the other is not (otherwise they are equivalent). Note that the numbers can be guessed in binary. The matrix multiplication (fast exponentiation) can then be used to compute resulting sets of those transitions in polynomial time. Thus, we can check in coNP whether the language of an NFA is $\mathcal{R}$-trivial.

To show that k-$\mathcal{R}$-triviality is in coNP, we first check in coNP, given an NFA $\mathcal{A}$, whether the language $L(\mathcal{A})$ is $\mathcal{R}$-trivial. If so, then it is 2^n-$\mathcal{R}$-trivial by Theorem 8, since the depth of the minimal DFA is bounded by 2^n, where n is the number of states of $\mathcal{A}$. To show that $L(\mathcal{A})$ is not k-$\mathcal{R}$-trivial, we need to find two $\sim_k^{\mathcal{R}}$-equivalent words such that exactly one of them belongs to $L(\mathcal{A})$. Since every class defined by a^ℓ, for $\ell < k$, is a singleton, we need to find $k < \ell \leq 2^n$ such that $a^k \sim_k a^\ell$ and only one of them belongs to $L(\mathcal{A})$. Since $a^k \sim_k a^\ell$ holds for every $\ell > k$, this can be done in nondeterministic polynomial time by guessing ℓ in binary and using the matrix multiplication to compare the states reachable by a^k and a^ℓ and verifying that one is accepting and the other is not.

To show that both problems are coNP-hard, we use the construction of [39] that we recall here showing that universality is coNP-hard for unary NFAs. Let φ be a formula in 3CNF with n distinct variables, and let C_k be the set of literals in the k-th conjunct, $1 \leq k \leq m$. The assignment to the variables can be represented as a binary vector of length n. Let $p_1, p_2, \ldots, p_n$ be the first n prime numbers. For a natural number z congruent with 0 or 1 modulo p_i, for every $1 \leq i \leq n$, we say that z satisfies φ if the assignment $(z \bmod p_1, z \bmod p_2, \ldots, z \bmod p_n)$ satisfies φ. Let

$$E_0 = \bigcup_{k=1}^{n} \bigcup_{j=2}^{p_k - 1} 0^j \cdot (0^{p_k})^*$$

that is, $L(E_0) = \{0^z \mid \exists k \leq n, z \not\equiv 0 \bmod p_k \text{ and } z \not\equiv 1 \bmod p_k\}$ is the set of natural numbers that do not encode an assignment to the variables. For each conjunct C_k, we construct an expression E_k such that if $0^z \in L(E_k)$ and z is an assignment, then z does not assign the value 1 to any literal in C_k. For example, if $C_k = \{x_r, \neg x_s, x_t\}$, for $1 \leq r, s, t \leq n$ and r, s, t distinct, let z_k be the unique integer such that $0 \leq z_k < p_r p_s p_t$, $z_k \equiv 0 \bmod p_r$, $z_k \equiv 1 \bmod p_s$, and $z_k \equiv 0 \bmod p_t$. Then

$$E_k = 0^{z_k} \cdot (0^{p_r p_s p_t})^* \,.$$

Now, φ is satisfiable if and only if there exists z such that z encodes an assignment to φ and $0^z \notin L(E_k)$ for all $1 \leq k \leq m$, which is if and only if $L(E_0 \cup \bigcup_{k=1}^{m} E_k) \neq 0^*$. This shows that universality is coNP-hard for NFAs over a unary alphabet. Let $p_n^\# = \Pi_{i=1}^{n} p_i$. If z encodes an assignment of φ, then, for any natural number c, $z + c \cdot p_n^\#$ also encodes an assignment of φ. Indeed, if $z \equiv x_i \bmod p_i$, then $z + c \cdot p_n^\# \equiv x_i \bmod p_i$, for every $1 \le i \le n$. This shows that if $0^z \notin L(E_k)$ for all k, then $0^z(0^{p_n^\#})^* \cap L(E_0 \cup \bigcup_{k=1}^m E_k) = \emptyset$. Since both languages are infinite, the minimal DFA recognizing the language $L(E_0 \cup \bigcup_{k=1}^m E_k)$ must have a nontrivial cycle. Therefore, if the language is universal, then it is k-$\mathcal{R}$-trivial for any $k \ge 0$, and if it is non-universal, then it is not $\mathcal{R}$-trivial. This proves coNP-hardness of k-$\mathcal{R}$-triviality for every $k \ge 0$. □

We now briefly discuss the complexity of the problem if the language is given as a poNFA rather than an NFA.

Theorem 20. *Given a poNFA $\mathcal{A}$, the problems of deciding whether the language accepted by $\mathcal{A}$ is $\mathcal{R}$-trivial, or k-$\mathcal{R}$-trivial for a given $k \ge 0$, are both* PSpace*-complete.*

Proof. The membership in PSpace follows from Theorems 13 and 16. PSpace-hardness can be shown by a slight modification of the proof of Theorem 3. Let M be a DTM and x be an input. We construct a binary regular expression R_x from M and x as in the proof of Theorem 3 with the modification as if M had a self-loop in the accepting state q_f. That is, if $\#w_1\#\cdots\#w_m\#$ is the unique accepting computation of M on x, we consider all words of the form $\#w_1\#\cdots\#w_m\#(w_m\#)^*$ as correct encodings of the accepting computation of M on x. The binary regular expression R_x and its corresponding binary poNFA $\mathcal{A}_x$ are then constructed as in the proof of Theorem 3. If M does not accept x, then $L(\mathcal{A}_x) = \{0,1\}^*$, which is a k-$\mathcal{R}$-trivial language for every $k \ge 0$. If M accepts x, then $L(\mathcal{A}_x) = \{0,1\}^* \setminus \mathrm{enc}(\#w_1\#\cdots\#w_m\#(w_m\#)^*)$. Since $|\,\mathrm{enc}(w_m\#)| \ge 2$, the sequence of prefixes

$$\Big(\mathrm{enc}(\#w_1\#\cdots\#w_m\#(w_m\#)^i), \mathrm{enc}(\#w_1\#\cdots\#w_m\#(w_m\#)^i w_m)\Big)_{i=0}^{\infty}$$

is infinite and alternates between non-accepted and accepted words of $\mathcal{A}_x$. Consequently, the minimal DFA equivalent to $\mathcal{A}_x$ must have a nontrivial cycle, which means that the language $L(\mathcal{A}_x) = \{0,1\}^* \setminus \mathrm{enc}(\#w_1\#\cdots\#w_m\#(w_m\#)^*)$ is not $\mathcal{R}$-trivial [8]. Therefore, the language of a binary poNFA $\mathcal{A}_x$ is (k-)$\mathcal{R}$-trivial if and only if M does not accept x. □

Notice that we have used a binary alphabet. For unary languages, we now show that the class of languages of unary poNFAs and unary $\mathcal{R}$-trivial languages coincide.

Theorem 21. *The classes of unary poNFA languages and unary $\mathcal{R}$-trivial languages coincide.*

Proof. Since every $\mathcal{R}$-trivial language is a poNFA language (see, for example, Theorem 7), we only need to prove that unary poNFA languages are $\mathcal{R}$-trivial. Assume for the contrary that there is a poNFA language L over the alphabet $\{a\}$ that is not $\mathcal{R}$-trivial. Then the minimal DFA for L is not partially ordered, and hence it has a non-trivial cycle. In other words, there are $k \ge 0$ and $\ell \ge 2$ such that for every $m \ge 0$, $a^{k+m\ell} \in L$ and $a^{k+m\ell+1} \notin L$. However, if a unary poNFA accepts an infinite language, then there is an integer n such that the poNFA accepts all words of length longer than n (cf. the proof of Theorem 4). This contradicts the existence of k and ℓ. □

5. Deciding Universality of rpoNFAs

In this section, we return to the universality problem for the case of rpoNFAs. We first show that we can indeed obtain the hoped-for reduction in complexity when using a fixed alphabet. For the general case, however, we can recover the same PSpace lower bound as for poNFAs, albeit with a more involved proof. Even for fixed alphabets, we can get a coNP lower bound:

Lemma 22. *The universality problem of rpoNFAs is* coNP*-hard even when restricting to alphabets with two letters.*

Proof. The first part of the proof is adapted from [19]. We use a reduction from the complement of CNF satisfiability. Let $U = \{x_1, x_2, \ldots, x_n\}$ be a set of variables and $\varphi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_m$ be a formula in CNF, where every φ_i is a disjunction of literals. Without loss of generality, we may assume that no clause φ_i contains both x and $\neg x$. Let $\neg\varphi$ be the negation of φ obtained by the de Morgan's laws. Then $\neg\varphi = \neg\varphi_1 \vee \neg\varphi_2 \vee \cdots \vee \neg\varphi_m$ is in DNF. For every $i = 1, \ldots, m$, define $\beta_i = \beta_{i,1}\beta_{i,2}\cdots\beta_{i,n}$, where

$$\beta_{i,j} = \begin{cases} 0+1 & \text{if } x_j \text{ and } \neg x_j \text{ do not appear in } \neg\varphi_i \\ 0 & \text{if } \neg x_j \text{ appears in } \neg\varphi_i \\ 1 & \text{if } x_j \text{ appears in } \neg\varphi_i \end{cases}$$

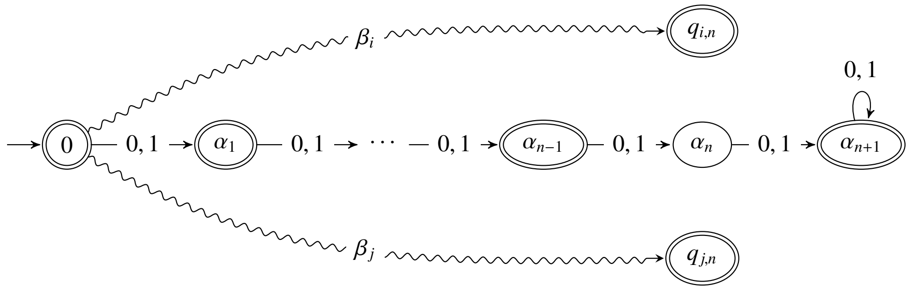

Figure 2: The rpoNFA $\mathcal{M}$ from the proof of Lemma 22

for $j = 1, 2, \dots, n$. Let $\beta = \bigcup_{i=1}^{m} \beta_i$. Then $w \in L(\beta)$ if and only if w satisfies some $\neg\varphi_i$. That is, $L(\beta) = \{0, 1\}^n$ if and only if $\neg\varphi$ is a tautology, which is if and only if φ is not satisfiable. Note that by the assumption, the length of every β_i is exactly n.

We now construct an rpoNFA $\mathcal{M}$ as follows, see Figure 2. The initial state of $\mathcal{M}$ is state 0. For every β_i, we construct a deterministic path consisting of $n + 1$ states $\{q_{i,0}, q_{i,1}, \dots, q_{i,n}\}$ with transitions $q_{i,\ell+1} \in q_{i,\ell} \cdot \beta_{i,\ell}$ and $q_{i,0} = 0$ accepting the words β_i. In addition, we add $n+1$ states $\{\alpha_1, \alpha_2, \dots, \alpha_{n+1}\}$ and transitions $\alpha_{\ell+1} \in \alpha_\ell \cdot a$, for $\ell < n+1$ and $\alpha_0 = 0$, and $\alpha_{n+1} \in \alpha_{n+1} \cdot a$, where $a \in \{0, 1\}$, accepting all words of length different from n. The accepting states of $\mathcal{M}$ are the states $\{0, q_{1,n}, \dots, q_{m,n}\} \cup \{\alpha_1, \dots \alpha_{n+1}\} \setminus \{\alpha_n\}$. Notice that $\mathcal{M}$ is restricted partially ordered. The automaton accepts the language $L(\mathcal{M}) = L(\beta) \cup \{w \in \{0, 1\}^* \mid |w| \neq n\}$, which is universal if and only if $L(\beta) = \{0, 1\}^n$. □

For a matching upper bound, we use Lemmas 14 and 15, which provide the main ingredients for showing that, if the size $|\Sigma|$ of the alphabet is bounded, then non-universality is witnessed by a word of polynomial length. Together with Lemma 22, this allows us to establish the following result.

Theorem 23. *Let Σ be a fixed non-unary alphabet, and let $\mathcal{B}$ be an rpoNFA over Σ. If $\mathcal{A}$ is an NFA (poNFA, rpoNFA, DFA, poDFA) over Σ, then the problem whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$ is* coNP*-complete.*

Proof. Hardness follows from Lemma 22 by letting $L(\mathcal{A}) = \Sigma^*$, which can be represented by a poDFA.

For membership, let $|\Sigma| = m$, and let $\mathcal{A}$ be an NFA. We show that $L(\mathcal{A})$ is not a subset of $L(\mathcal{B})$ if and only if there exists an NFA C of polynomial size with respect to $\mathcal{B}$ such that $L(\mathcal{A}) \cap L(C) \neq \emptyset$ and $L(\mathcal{B}) \cap L(C) = \emptyset$. Since such an NFA can be guessed by a nondeterministic algorithm, and the (non)emptiness of the intersection of the languages of two NFAs can be verified in polynomial time, we obtain that the problem whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$ is in coNP.

It remains to show that there exists such an NFA C. Without loss of generality, we assume that $\mathcal{B}$ is complete; otherwise, we make it complete in polynomial time by adding a single sink state and the missing transitions. Let k be the depth of $\mathcal{B}$. Then k is bounded by the number of states of $\mathcal{B}$. By Theorem 8, language $L(\mathcal{B})$ is k-$\mathcal{R}$-trivial, which means that it is a finite union of $\sim_k^{\mathcal{R}}$ classes. According to Lemmas 14 and 15, the length of the unique minimal representatives of the $\sim_k^{\mathcal{R}}$ classes is at most $\binom{k+m}{k} - 1 < \frac{(k+m)^m}{m!}$. Since m is a constant, the bound is polynomial in k. Now, $L(\mathcal{A})$ is not a subset of $L(\mathcal{B})$ if and only if there exists a word in $L(\mathcal{A})$ that is not in $L(\mathcal{B})$. This means that there exists a $\sim_k^{\mathcal{R}}$ class that intersects with $L(\mathcal{A})$ and is disjoint from $L(\mathcal{B})$. Let w be the unique minimal representative of this class. In Theorem 16, we constructed a DFA $\mathcal{D}$ with at most $|w| + 2$ states recognizing the class $w/_{\sim_k^{\mathcal{R}}}$. Notice that $\mathcal{D}$ is such that $L(\mathcal{A}) \cap L(\mathcal{D}) \neq \emptyset$, $L(\mathcal{B}) \cap L(\mathcal{D}) = \emptyset$, and that the size of $\mathcal{D}$ is polynomial with respect to the size of $\mathcal{B}$. This completes the proof. □

Corollary 24. *Let Σ be a fixed non-unary alphabet. Then the universality problem for rpoNFAs over Σ is* coNP*-complete.*

Proof. Hardness follows from Lemma 22, the containment from Theorem 23 by letting $L(\mathcal{A}) = \Sigma^*$. □

Notice that the proof of Theorem 4 also applies to rpoNFAs, and hence we immediately have the following result.

Corollary 25. *The universality problem for rpoNFAs over a unary alphabet is* NL*-complete.* □

$k\backslash n$	1	2	3
1	a_1	a_1a_2	$a_1a_2a_3$
2	a_1^2	$a_1^2a_2a_1a_2$	$a_1^2a_2a_1a_2a_3a_1a_2a_3$
3	a_1^3	$a_1^3a_2a_1^2a_2a_1a_2$	$a_1^3a_2a_1^2a_2a_1a_2a_3a_1^2a_2a_1a_2a_3a_1a_2a_3$
4	a_1^4	$a_1^4a_2a_1^3a_2a_1^2a_2a_1a_2$	$a_1^4a_2a_1^3a_2a_1^2a_2a_1a_2a_3a_1^3a_2a_1^2a_2a_1a_2a_3a_1^2a_2a_1a_2a_3a_1a_2a_3$

Table 2: Recursive construction of words $W_{k,n}$ as used in the proof of Lemma 26

Without fixing the alphabet, universality remains PSpace-hard even for rpoNFAs, but a proof along the lines of Theorem 3 is not straightforward. In essence, rpoNFAs lose the ability to navigate to an arbitrary position within a word for checking some pattern there. Expressions of the form $(\Sigma^* \cdots)$, which we frequently used, e.g., in (1), are therefore excluded. This is problematic since the run of a polynomially space-bounded Turing machine may be of exponential length, and we need to match patterns across the full length of our (equally exponential) encoding of this run. How can we navigate such a long word without using Σ^*? Our answer is to first define an rpoNFA that accepts all words except for a single, exponentially long word. This word will then be used as an rpoNFA-supported "substrate" for our Turing machine encoding, which again follows Theorem 3.

Lemma 26. *For all positive integers k and n, there exists an rpoNFA $\mathcal{A}_{k,n}$ over an n-letter alphabet with $n(k+2)$ states such that the unique word not accepted by $\mathcal{A}_{k,n}$ is of length $\binom{k+n}{k} - 1$.*

Proof. For integers $k, n \geq 1$, we recursively define words $W_{k,n}$ over the alphabet $\Sigma_n = \{a_1, a_2, \ldots, a_n\}$. For the base cases, we set $W_{k,1} = a_1^k$ and $W_{1,n} = a_1a_2\cdots a_n$. The cases for $k, n > 1$ are defined recursively by setting

$$\begin{aligned} W_{k,n} &= W_{k,n-1}\, a_n\, W_{k-1,n} \\ &= W_{k,n-1}\, a_n\, W_{k-1,n-1}\, a_n\, W_{k-2,n} \\ &= W_{k,n-1}\, a_n\, W_{k-1,n-1}\, a_n\, \cdots\, a_n\, W_{1,n-1}\, a_n\,. \end{aligned} \tag{5}$$

The recursive construction is illustrated in Table 2. The length of $W_{k,n}$ is $\binom{k+n}{n} - 1$ [28]. Notice that a_n appears exactly k times in $W_{k,n}$. We further set $W_{k,n} = \varepsilon$ whenever $kn = 0$, since this is useful for defining $\mathcal{A}_{k,n}$ below.

We construct an rpoNFA $\mathcal{A}_{k,n}$ over Σ_n that accepts the language $\Sigma_n^* \setminus \{W_{k,n}\}$. For $n = 1$ and $k \geq 0$, let $\mathcal{A}_{k,1}$ be the minimal DFA accepting the language $\{a_1\}^* \setminus \{a_1^k\}$. It consists of $k+2$ states of the form $(i; 1)$ as depicted in Figure 3, together with the given transitions. All states but $(k; 1)$ are final, and $(0; 1)$ is initial.

Given $\mathcal{A}_{k,n-1}$, we recursively construct $\mathcal{A}_{k,n}$ as defined next. The construction for $n = 2$ is illustrated in Figure 4. We obtain $\mathcal{A}_{k,n}$ from $\mathcal{A}_{k,n-1}$ by adding $k+2$ states $(0; n), (1; n), \ldots, (k+1; n)$, where $(0; n)$ is added to the initial states, and all states other than $(k; n)$ are added to the final states. $\mathcal{A}_{k,n}$ therefore has $n(k+2)$ states.

The additional transitions of $\mathcal{A}_{k,n}$ consist of four groups:

1. Self-loops $(i; n) \xrightarrow{a_j} (i; n)$ for every $i = 0, \ldots, k+1$ and $a_j = a_1, \ldots, a_{n-1}$;
2. Transitions $(i; n) \xrightarrow{a_n} (i+1; n)$ for every $i = 0, \ldots, k$, and the self-loop $(k+1; n) \xrightarrow{a_n} (k+1; n)$;
3. Transitions $(i; n) \xrightarrow{a_n} (i+1; m)$ for every $i = 0, \ldots, k$ and $m = 1, \ldots, n-1$;
4. Transitions $(i; m) \xrightarrow{a_n} (k+1; n)$ for every accepting state $(i; m)$ of $\mathcal{A}_{k,n-1}$.

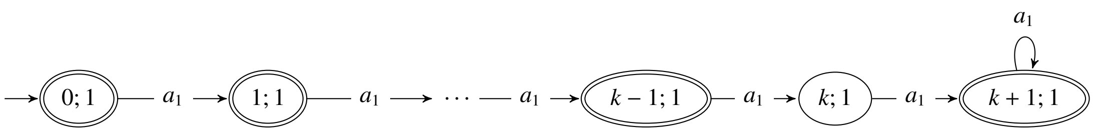

Figure 3: The rpoNFA $\mathcal{A}_{k,1}$ with $k+2$ states

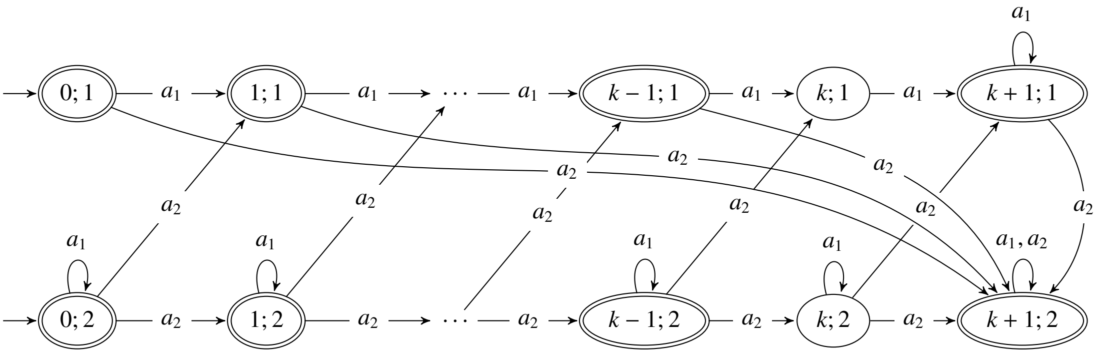

Figure 4: The rpoNFA $\mathcal{A}_{k,2}$ with $2(k+2)$ states

The additional states of $\mathcal{A}_{k,n}$ and transitions (1) and (2) ensure acceptance of every word that does not contain exactly k occurrences of a_n. The transitions (3) together with the transitions (4) ensure acceptance of all words in $(\Sigma_{n-1}^* a_n)^{i+1} L(\mathcal{A}_{k-(i+1),n-1}) a_n \Sigma_n^*$ for which the word between the $(i+1)$-st and the $(i+2)$-nd occurrence of a_n is not of the form $W_{k-(i+1),n-1}$, and hence not a correct subword of $W_{k,n} = W_{k,n-1} a_n \cdots a_n W_{k-(i+1),n-1} a_n \cdots a_n W_{1,n-1} a_n$. The transitions (4) ensure that all words with a prefix $w \cdot a_n$ are accepted, where w is any word $\Sigma_{n-1}^* \setminus \{W_{k,n-1}\}$ accepted by $\mathcal{A}_{k,n-1}$. Together, these conditions ensure that $\mathcal{A}_{k,n}$ accepts every input other than $W_{k,n}$.

It remains to show that $\mathcal{A}_{k,n}$ does not accept $W_{k,n}$, which we do by induction on (k, n). We start with the base cases. For $(0, n)$ and any $n \geq 1$, the word $W_{0,n} = \varepsilon$ is not accepted by $\mathcal{A}_{0,n}$, since the initial states $(0, m) = (k, m)$ of $\mathcal{A}_{0,n}$ are not accepting. Likewise, for $(k, 1)$ and any $k \geq 0$, we find that $W_{k,1} = a_1^k$ is not accepted by $\mathcal{A}_{k,1}$ (Figure 3).

For the inductive case $(k, n) \geq (1, 2)$, assume $\mathcal{A}_{k',n'}$ does not accept $W_{k',n'}$ for any $(k', n') < (k, n)$; here $\leq$ is the standard product order. We have $W_{k,n} = W_{k,n-1} a_n W_{k-1,n}$, and $W_{k,n-1}$ is not accepted by $\mathcal{A}_{k,n-1}$ by induction. In addition, there is no transition under a_n from any non-accepting state of $\mathcal{A}_{k,n-1}$ in $\mathcal{A}_{k,n}$. Therefore, if $W_{k,n}$ is accepted by $\mathcal{A}_{k,n}$, it must be accepted in a run starting from the initial state $(0; n)$. Since $W_{k,n-1}$ does not contain a_n, we find that $\mathcal{A}_{k,n}$ can only reach the states $(0; n) \cdot W_{k,n-1} a_n = \{(1; m) \mid 1 \leq m \leq n\}$ after reading $W_{k,n-1} a_n$. These are the initial states of automaton $\mathcal{A}_{k-1,n}$, which does not accept $W_{k-1,n}$ by induction. Hence $W_{k,n}$ is not accepted by $\mathcal{A}_{k,n}$. □

As a corollary, we find that there are rpoNFAs $\mathcal{A} = \mathcal{A}_{n,n}$ for which the shortest non-accepted word is exponential in the size of $\mathcal{A}$. Note that $\binom{2n}{n} \geq 2^n$.

Corollary 27. *For every integer $n \geq 1$, there is an rpoNFA $\mathcal{A}_n$ over an n-letter alphabet with $n(n+2)$ states such that the shortest word not accepted by $\mathcal{A}_n$ is of length $\binom{2n}{n} - 1$. Therefore, any minimal DFA accepting the same language has at least $\binom{2n}{n}$ states.*

Proof. This is immediate from Lemma 26 by setting $n = k$. □

To simulate exponentially long runs of a Turing machine, we start from an encoding of runs using words $\#w_1\# \cdots \#w_m\#$ as in Theorem 3, but we combine every letter of this encoding with one letter of the alphabet of $\mathcal{A}_n$. We then accept all words for which the projection to the alphabet of $\mathcal{A}_n$ is accepted by $\mathcal{A}_n$, i.e., all but those words of exponential length that are based on the unique word not accepted by $\mathcal{A}_n$. We ensure that, if there is an accepting run, it will have an encoding of this length. It remains to eliminate (accept) all words that correspond to a non-accepting or wrongly encoded run. We can check this as in Theorem 3, restricting to the first components of our combined alphabet. The self-loop that was used to encode Σ^* in poNFAs is replaced by a full copy of $\mathcal{A}_n$, with an additional transition from each state that allows us to leave this "loop". This does not simulate the full loop, but it allows us to navigate the entirety of our exponential word, which is all we need.

Theorem 28. *The universality problem for rpoNFAs is* PSpace*-complete.*

Proof. The membership follows since universality is in PSpace for NFAs. For hardness, we proceed as explained above. Consider a p-space-bounded DTM $M = \langle Q, T, I, \gamma, \sqcup, q_o, q_f \rangle$ as in the proof of Theorem 3. We encode runs of M as words over $T \times (Q \cup \{\varepsilon\}) \cup \{\#\}$ as before. We can use an unrestricted alphabet now, so no binary encoding is needed, and the regular expressions can be simplified accordingly.

If M has an accepting run, then this run does not have a repeated configuration. For an input word x, there are $C(x) = (|T \times (Q \cup \{\varepsilon\})|)^{p(|x|)}$ distinct configuration words in our encoding. Considering separator symbols #, the maximal length of the encoding of a run without repeated configurations therefore is $1 + C(x)(p(|x|) + 1)$, since every configuration word now ends with # and is thus of length $p(|x|) + 1$. Let n be the least number such that $|W_{n,n}| \geq 1 + C(x)(p(|x|) + 1)$, where $W_{n,n}$ is the word from the proof of Lemma 26. Since $|W_{n,n}| + 1 = \binom{2n}{n} \geq 2^n$, it follows that n is smaller than $\lceil \log_2(1 + C(x)(p(|x|) + 1)) \rceil$ and hence polynomial in the size of M and x.

Consider the automaton $\mathcal{A}_{n,n}$ with alphabet $\Sigma_n = \{a_1, \ldots, a_n\}$ of Lemma 26, and define $\Delta_{\#\$} = T \times (Q \cup \{\varepsilon\}) \cup \{\#, \$\}$. We consider the alphabet $\Pi = \Sigma_n \times \Delta_{\#\$}$, where the second letter is used for encoding a run as in Theorem 3. Since $|W_{n,n}|$ may not be a multiple of $p(|x|) + 1$, we add \$ to fill up any remaining space after the last configuration. For a word $w = \langle a_{i_1}, \delta_1 \rangle \cdots \langle a_{i_\ell}, \delta_\ell \rangle \in \Pi^\ell$, we define $w[1] = a_{i_1} \cdots a_{i_\ell} \in \Sigma_n^\ell$ and $w[2] = \delta_1 \cdots \delta_\ell \in \Delta_{\#\$}^\ell$. Conversely, for a word $v \in \Delta_{\#\$}^*$, we write $\mathrm{enc}(v)$ to denote the set of all words $w \in \Pi^{|v|}$ with $w[2] = v$. Similarly, for $v \in \Sigma_n^*$, $\mathrm{enc}(v)$ denotes the words $w \in \Pi^{|v|}$ with $w[1] = v$. We extend this notation to sets of words.

We say that a word w encodes an accepting run of M on x if $w[1] = W_{n,n}$ and $w[2]$ is of the form $\#w_1\# \cdots \#w_m\#\$^j$ such that there is an $i \in \{1, \ldots, m\}$ for which we have that

- $\#w_1\# \cdots \#w_i\#$ encodes an accepting run of M on x as in the proof of Theorem 3,
- $w_k = w_i$ for all $k \in \{i + 1, \ldots, m\}$, and
- $j \leq p(|x|)$.

In other words, we extend the encoding by repeating the accepting configuration until we have less than $p(|x|) + 1$ symbols before the end of $|W_{n,n}|$ and fill up the remaining places with \$.

The modified encoding requires slightly modified expressions for capturing conditions (A)–(D) from the proof of Theorem 3. Condition (A) is not necessary, since we do not encode symbols in binary. Condition (B) can use the same expression as in (2), adjusted to our alphabet:

$$\left(\varepsilon + \Pi + \Pi^2 + \cdots + \Pi^{p(|x|)+1}\right) + \sum_{0 \leq j \leq p(|x|)+1} (\Pi^j \cdot \bar{E}_j \cdot \Pi^*) \tag{6}$$

where $\bar{E}_j$ is the set $\Sigma_n \times (\Delta_{\#\$} \setminus \{v_j\})$ where v_j encodes the j-th symbol on the initial tape as in Theorem 3. All uses of Π^i in this expression encode words of polynomial length, which can be represented in rpoNFAs. Trailing expressions Π^* do not lead to nondeterministic self-loops of Figure 1.

Condition (C) uses the same ideas as in Theorem 3, especially the transition encoding function f, which we extend to $f : \Delta_{\#\$}^3 \to \Delta_{\#\$}$. For allowing the last configuration to be repeated, we define f as if the final state q_f of M had a self loop (a transition that does not modify the tape, state, or head position). Moreover, we generally permit \$ to occur instead of the expected next configuration symbol. We obtain:

$$\Pi^* \sum_{\delta_\ell, \delta, \delta_r \in \Delta_{\#\$}} \mathrm{enc}(\delta_\ell \delta \delta_r) \cdot \Pi^{p(|x|)-1} \cdot \hat{f}(\delta_\ell, \delta, \delta_r) \cdot \Pi^* \tag{7}$$

where $\hat{f}(\delta_\ell, \delta, \delta_r)$ is $\Pi \setminus \mathrm{enc}(\{f(\delta_\ell, \delta, \delta_r), \$\})$. Expression (7) is not readily encoded in an rpoNFA, due to the leading Π^*. To address this, we replace Π^* by the expression $\Pi^{\leq |W_{n,n}|-1}$, which matches every word $w \in \Pi^*$ with $|w| \leq |W_{n,n}| - 1$. Clearly, this suffices for our case. As $|W_{n,n}| - 1$ is exponential, we cannot encode this directly as for other expressions Π^i before and we use $\mathcal{A}_{n,n}$ instead.

In detail, let E be the expression obtained from (7) when omitting the initial Π^*, and let $\mathcal{A}$ be an rpoNFA that accepts the language of E. We can construct $\mathcal{A}$ so that it has a single initial state. Moreover, let $\mathrm{enc}(\mathcal{A}_{n,n})$ be the automaton $\mathcal{A}_{n,n}$ of Lemma 26 with each transition $q \xrightarrow{a_i} q'$ replaced by all transitions $q \xrightarrow{\pi} q'$ with $\pi \in \mathrm{enc}(a_i)$. We construct an rpoNFA $\mathcal{A}'$ that accepts the language of $(\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\}) + (\Pi^{\leq |W_{n,n}|-1} \cdot E)$ by merging $\mathrm{enc}(\mathcal{A}_{n,n})$ with at most $n(n+2)$ copies of $\mathcal{A}$, where we identify the initial state of each such copy with a different final state of $\mathrm{enc}(\mathcal{A}_{n,n})$, if it does not introduce nondeterministic self-loops. The fact that $\mathrm{enc}(\mathcal{A}_{n,n})$ alone already accepts $(\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\})$

was shown in the proof of Lemma 26. This also implies that it accepts all words of length $\leq |W_{n,n}| - 1$ as needed to show that $(\Pi^{\leq |W_{n,n}|-1} \cdot E)$ is accepted. Entering states of (a copy of) $\mathcal{A}$ after accepting a word of length $\geq |W_{n,n}|$ is possible, but all words accepted in such a way are longer than $W_{n,n}$ and hence in $(\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\})$.

It remains to show that for every strict prefix $w_{n,n}$ of $W_{n,n}$, there is a state in $\mathcal{A}_{n,n}$ reached by $w_{n,n}$ that is the initial state of a copy of $\mathcal{A}$, and hence the check represented by E in $\Pi^{\leq |W_{n,n}|-1} \cdot E$ can be performed. In other words, if $a_{n,n}$ denotes the letter following $w_{n,n}$ in $W_{n,n}$, then $w_{n,n}$ reaches a state in $\mathcal{A}_{n,n}$ that does not have a loop under $a_{n,n}$. However, this follows from the fact that $\mathcal{A}_{n,n}$ accepts everything but $W_{n,n}$, since then the DFA obtained from $\mathcal{A}_{n,n}$ by the standard subset construction has a path of length $\binom{2n}{n} - 1$ labeled with $W_{n,n}$ without any loop. Moreover, any state of this path in the DFA is a subset of states of $\mathcal{A}_{n,n}$. Therefore, at least one of the states reachable under $w_{n,n}$ in $\mathcal{A}_{n,n}$ does not have a self-loop under $a_{n,n}$.

Note that the acceptance of $(\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\})$, which is a side effect of this encoding, does not relate to expressing (7) but is still useful for our intended overall encoding.

The final condition (D) is minimally modified to allow for up to $p(|x|)$ trailing \$. For a word v, we use $v^{\leq i}$ to abbreviate $(\varepsilon + v + \cdots + v^i)$, and we define $\bar{E}_f = (T \times (Q \setminus \{q_f\}))$ as before. Since not all words with too many trailing \$ are accepted by (C), we add this here instead. Moreover, we need to check that all the symbols \$ appear only at the end, that is, the last expression accepts all inputs where \$ is followed by a different symbol:

$$\begin{aligned}
&\Pi^* \,\mathrm{enc}(\#)(\Pi + \cdots + \Pi^{p(|x|)})\,\mathrm{enc}(\$)^{\leq p(|x|)} + \\
&\Pi^* \,\mathrm{enc}(\bar{E}_f)(\varepsilon + \Pi + \cdots + \Pi^{p(|x|)-1})\,\mathrm{enc}(\#)\,\mathrm{enc}(\$)^{\leq p(|x|)} + \\
&\Pi^* \,\mathrm{enc}(\$)^{p(|x|)+1} + \\
&(\Pi \setminus \mathrm{enc}(\$))^* \,\mathrm{enc}(\$)\,\mathrm{enc}(\$)^*(\Pi \setminus \mathrm{enc}(\$))\Pi^*
\end{aligned} \tag{8}$$

As before, we cannot encode the leading Π^* directly as an rpoNFA, but we can perform a similar construction as in (7) to overcome this problem.

The union of the rpoNFAs for (6)–(8) constitutes an rpoNFA that is polynomial in the size of M and x, and that is universal if and only if M does not accept x. □

6. Inclusion and Equivalence of Partially Ordered NFAs

Universality is closely related to the inclusion and equivalence problems, which are of interest mainly from the point of view of optimization, e.g., in query answering. Given two languages K and L over Σ, the *inclusion problem* asks whether $K \subseteq L$ and the *equivalence problem* asks whether $K = L$. The relation of universality to inclusion and equivalence lies in the fact that the complexity of universality provides a lower bound on the complexity of both inclusion and equivalence. We now show that the complexities coincide, see Table 3.

The complexity of inclusion and equivalence for regular expressions of special forms has been investigated by Martens et al. [26]. For a few of them, PSpace-completeness of the inclusion problem has been achieved. The results are established for alphabets of unbounded size. Since some of the expressions define languages expressible by poNFAs, we readily have that the inclusion problem for poNFAs is PSpace-complete. However, using Theorem 3 and the well-known PSpace upper bound on inclusion and equivalence for NFAs, we obtain the following result.

Corollary 29. *The inclusion and equivalence problems for poNFAs are* PSpace*-complete even if the alphabet is binary.*

The expressions in Martens et al. [26] cannot be expressed as rpoNFAs. Hence the question for rpoNFAs was open. Using Theorem 28 and the upper bound for NFAs, we can easily establish the following result.

Corollary 30. *The inclusion and equivalence problems for rpoNFAs are* PSpace*-complete.*

If the alphabet is fixed, the complexity of inclusion (and of equivalence) is covered by Theorem 23.

Corollary 31. *The inclusion and equivalence problems for rpoNFAs over a fixed alphabet are* coNP*-complete.*

Finally, for the unary case, it is known that the inclusion and equivalence problems for NFAs over a unary alphabet are coNP-complete [18, 39]. For poNFAs we obtain the following result.

	Unary alphabet	Fixed alphabet	Arbitrary alphabet
DFA	L-comp.	NL-comp.	NL-comp.
rpoNFA	NL-comp.	coNP-comp.	PSpace-comp.
poNFA	NL-comp.	PSpace-comp.	PSpace-comp.
NFA	coNP-comp.	PSpace-comp.	PSpace-comp.

Table 3: Complexity of deciding inclusion and equivalence

Theorem 32. *The inclusion and equivalence problems for poNFAs over a unary alphabet are* NL*-complete.*

Proof. The proof is a modification of the proof of Theorem 4. Checking $L(\mathcal{A}) \subseteq L(\mathcal{B})$ is easy if $L(\mathcal{A})$ is finite, since there is at most depth$(\mathcal{A}) + 1$ strings to be checked. If $L(\mathcal{A})$ is infinite, then there must be a simple path from an initial state to an accepting state via a state with a self-loop. Let k denote the length of this path, which is bounded by the number of states. Then this path accepts all words of length at least k, that is, all words of the form $a^k a^*$. Then $L(\mathcal{B})$ must also be infinite and, similarly, we get ℓ such that $a^\ell a^*$ all belong to $L(\mathcal{B})$. For every $m \leq \max\{k, \ell\}$, we check that if $a^m \in L(\mathcal{A})$, then $a^m \in L(\mathcal{B})$. This requires to perform $m + 1$ nondeterministic logarithmic checks. As m is smaller than the inputs, the proof is complete. □

7. Deterministic Regular Expressions and Partially Ordered NFAs

In this section, we point out the relationship of partially ordered NFAs to deterministic regular expressions (DREs) [7]. DREs are of interest in schema languages for XML data – Document Type Definition (DTD) and XML Schema Definition (XSD) – since the World Wide Web Consortium standards require that the regular expressions in their specification must be deterministic.

The *regular expressions* (REs) over an alphabet Σ are defined as follows: $\emptyset$, ε and a, $a \in \Sigma$, are regular expressions. If r and s are regular expressions, then $(r \cdot s)$, $(r + s)$ and $(r)^*$ are regular expressions. The language defined by a regular expression r, denoted by $L(r)$, is inductively defined by $L(\emptyset) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$, $L(a) = \{a\}$, $L(r \cdot s) = L(r) \cdot L(s)$, $L(r + s) = L(r) \cup L(s)$, and $L(r^*) = \{\varepsilon\} \cup \bigcup_{i=1}^{\infty} L(r)^i$, where $L(r) \cdot L(s)$ denotes the concatenation of the languages $L(r)$ and $L(s)$. Let r be a regular expression, and let $\overline{r}$ be a regular expression obtained from r by replacing the i-th occurrence of symbol a in r by a_i. For instance, if $r = (a + b)^* b(a + b)$, then $\overline{r} = (a_1 + b_1)^* b_2(a_2 + b_3)$. A regular expression r is *deterministic* (one-unambiguous [7] or DRE) if there are no words wa_iv and wa_jv' in $L(\overline{r})$ such that $i \neq j$. For instance, the expression $(a + b)^* b(a + b)$ is not deterministic since the strings b_2a_2 and $b_1b_2a_2$ are both in $L((a_1 + b_1)^* b_2(a_2 + b_3))$. A regular language is *DRE definable* if there exists a DRE that defines it. Brüggemann-Klein and Wood [7] showed that not every regular language is DRE definable.

The important question is then whether a regular language is DRE definable. This problem has been shown to be PSpace-complete [12]. Since the language of the expression $(a + b)^* b(a + b)$ is not DRE definable [7], but it can be easily expressed by a poNFA, DRE definability is nontrivial for poNFAs. Its complexity however follows from existing results, namely from the proof in Bex et al. [5] showing PSpace-hardness of DRE-definability for regular expressions, since the regular expression constructed there can be expressed as a poNFA. Thus, we readily have the following:

Corollary 33. *To decide whether the language of a poNFA is DRE definable is* PSpace*-complete.*

On the other hand, the problem is trivial for the languages of rpoNFAs, which makes rpoNFAs interesting for the XML schema languages.

Theorem 34. *Every rpoNFA language is DRE definable.*

To prove the theorem, we need to introduce a few notions. For a state q of an NFA $\mathcal{A}$, the *orbit of* q is the maximal strongly connected component of $\mathcal{A}$ containing q. State q is called a *gate* of the orbit of q if q is accepting or has an outgoing transition that leaves the orbit. The orbit automaton of state q is the sub-automaton of $\mathcal{A}$ consisting of the orbit of q in which the initial state is q and the accepting states are the gates of the orbit of q. We denote the orbit automaton of q by $\mathcal{A}_q$. The orbit language of q is $L(\mathcal{A}_q)$. The orbit languages of $\mathcal{A}$ are the orbit languages of states of $\mathcal{A}$.

An NFA $\mathcal{A}$ has the *orbit property* if, for every pair of gates q_1, q_2 in the same orbit in $\mathcal{A}$, the following properties hold: (i) q_1 is accepting if and only if q_2 is accepting, and (ii) for all states q outside the orbit of q_1 and q_2, there is a transition $q \in q_1 \cdot a$ if a and only if there is a transition $q \in q_2 \cdot a$.

Brüggemann-Klein and Wood [7] have shown that the language of a minimal DFA $\mathcal{A}$ is DRE-definable if and only if $\mathcal{A}$ has the orbit property and all orbit languages of $\mathcal{A}$ are DRE-definable.

Lemma 35. *Every language of a minimal partially ordered DFA is DRE-definable.*

Proof. Every orbit of a partially ordered DFA is a singleton, and hence it satisfies the orbit property. The orbit language is either empty or A^* for some alphabet A, and therefore DRE-definable. □

Proof (of Theorem 34). Every language defined by an rpoNFA is $\mathcal{R}$-trivial (Theorem 7), and hence its minimal DFA is partially ordered [8]. By the previous lemma, the language is DRE-definable. □

Finally, note that the converse of Theorem 34 does not hold. The expression $b^*a(b^*a)^*$ is deterministic [12] and it can be easily verified that its minimal DFA is not partially ordered. Therefore, the expression defines a language that is not $\mathcal{R}$-trivial.

8. Conclusion

Our results regarding the complexity of deciding universality for partially ordered NFAs are summarized in Table 1. We found that poNFAs over a fixed, two-letter alphabet are still powerful enough to recognize the language of all non-accepting computations of a PSpace Turing machine. Restricting poNFAs further by forbidding nondeterministic self-loops, we could establish lower coNP complexity bounds for universality for alphabets of bounded size. We can view this as the complexity of universality of rpoNFAs in terms of the size of the automaton when keeping the alphabet fixed. Unfortunately, the complexity is PSpace-complete even for rpoNFAs over arbitrary (unbounded) alphabets. The proof uses an interesting construction where the encoding of a Turing machine computation is "piggybacked" on an exponentially long word, for which a dedicated rpoNFA is constructed.

We have characterized the expressive power of rpoNFAs by relating them to the class of $\mathcal{R}$-trivial languages. It is worth noting that the complexity bounds we established for recognizing $\mathcal{R}$-triviality for a given NFA agrees with the complexity of the rpoNFA universality problem for both fixed and arbitrary alphabets. Our results on universality therefore extend beyond rpoNFAs to arbitrary NFAs that recognize $\mathcal{R}$-trivial languages.

Moreover, the results on universality further extend to the complexity of inclusion and equivalence, and to the complexity of DRE-definability. Restricted poNFAs ($\mathcal{R}$-trivial languages) have been shown to be of interest in schema languages for XML data.

Our work can be considered as a contribution to the wider field of studying subclasses of star-free regular languages. The Straubing-Thérien hierarchy provides a large field for interesting future work in this area.

Acknowledgements

The authors gratefully acknowledge very useful suggestions and comments of the anonymous referees.

References

References

[1] Aho, A. V., Hopcroft, J. E., Ullman, J. D., 1974. The Design and Analysis of Computer Algorithms. Addison-Wesley.

[2] Almeida, J., Bartoňová, J., Klíma, O., Kunc, M., 2015. On decidability of intermediate levels of concatenation hierarchies. In: Developments in Language Theory (DLT). Vol. 9168 of LNCS. Springer, pp. 58–70.

[3] Bar-Hillel, Y., Perles, M. A., Shamir, E., 1961. On formal properties of simple phrase structure grammars. Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung 14, 143–172.

[4] Barceló, P., Libkin, L., Reutter, J. L., 2014. Querying regular graph patterns. Journal of the ACM 61 (1), 8:1–8:54.

[5] Bex, G. J., Gelade, W., Martens, W., Neven, F., 2009. Simplifying XML schema: Effortless handling of nondeterministic regular expressions. In: ACM International Conference on Management of Data (SIGMOD). ACM, pp. 731–744.

[6] Bouajjani, A., Muscholl, A., Touilim, T., 2007. Permutation rewriting and algorithmic verification. Information and Computation 205 (2), 199–224.

[7] Brüggemann-Klein, A., Wood, D., 1998. One-unambiguous regular languages. Information and Computation 142 (2), 182–206.

[8] Brzozowski, J. A., Fich, F. E., 1980. Languages of *R*-trivial monoids. Journal of Computer and System Sciences 20 (1), 32–49.

[9] Brzozowski, J. A., Knast, R., 1978. The dot-depth hierarchy of star-free languages is infinite. Journal of Computer and System Sciences 16 (1), 37–55.

[10] Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M. Y., 2003. Reasoning on regular path queries. ACM SIGMOD Record 32 (4), 83–92.

[11] Cohen, R. S., Brzozowski, J. A., 1971. Dot-depth of star-free events. Journal of Computer and System Sciences 5 (1), 1–16.

[12] Czerwinski, W., David, C., Losemann, K., Martens, W., 2013. Deciding definability by deterministic regular expressions. In: International Conference on Foundations of Software Science and Computation Structures (FoSSaCS). Vol. 7794 of LNCS. Springer, pp. 289–304.

[13] Ellul, K., Krawetz, B., Shallit, J., Wang, M.-W., 2005. Regular expressions: New results and open problems. Journal of Automata, Languages and Combinatorics 10 (4), 407–437.

[14] Friedman, E. P., 1976. The inclusion problem for simple languages. Theoretical Computer Science 1 (4), 297–316.

[15] Garey, M. R., Johnson, D. S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman.

[16] Glaßer, C., Schmitz, H., 2008. Languages of dot-depth 3/2. Theory of Computing Systems 42 (2), 256–286.

[17] Hofman, P., Martens, W., 2015. Separability by short subsequences and subwords. In: International Conference on Database Theory (ICDT). Vol. 31 of LIPIcs. pp. 230–246.

[18] Holzer, M., Kutrib, M., 2011. Descriptional and computational complexity of finite automata—A survey. Information and Computation 209 (3), 456–470.

[19] Hunt III, H. B., 1973. On the time and tape complexity of languages. Ph.D. thesis, Department of Computer Science, Cornell University, Ithaca, NY.

[20] Hunt III, H. B., Rosenkrantz, D. J., 1978. Computational parallels between the regular and context-free languages. SIAM Journal on Computing 7 (1), 99–114.

[21] Jones, N. D., 1975. Space-bounded reducibility among combinatorial problems. Journal of Computer and System Sciences 11 (1), 68–85.

[22] Klíma, O., Polák, L., 2013. Alternative automata characterization of piecewise testable languages. In: Developments in Language Theory (DLT). Vol. 7907 of LNCS. Springer, pp. 289–300.

[23] Krötzsch, M., Masopust, T., Thomazo, M., 2016. On the complexity of university for partially ordered NFAs. In: Mathematical Foundations of Computer Science (MFCS). Vol. 58 of LIPIcs. pp. 61:1–61:14.

[24] Kufleitner, M., Lauser, A., 2011. Partially ordered two-way Büchi automata. International Journal of Foundations of Computer Science 22 (8), 1861–1876.

[25] Lodaya, K., Pandya, P. K., Shah, S. S., 2010. Around dot depth two. In: Developments in Language Theory (DLT). Vol. 6224 of LNCS. Springer, pp. 303–315.

[26] Martens, W., Neven, F., Schwentick, T., 2009. Complexity of decision problems for XML schemas and chain regular expressions. SIAM Journal on Computing 39 (4), 1486–1530.

[27] Masopust, T., 2016. Piecewise testable languages and nondeterministic automata. In: Mathematical Foundations of Computer Science (MFCS). Vol. 58 of LIPIcs. pp. 67:1–67:14.

[28] Masopust, T., Thomazo, M., 2017. On boolean combinations forming piecewise testable languages. Theoretical Computer Science 682, 165–179.

[29] Meyer, A. R., Stockmeyer, L. J., 1972. The equivalence problem for regular expressions with squaring requires exponential space. In: Symposium on Switching and Automata Theory (SWAT/FOCS). IEEE Computer Society, pp. 125–129.

[30] Place, T., 2015. Separating regular languages with two quantifiers alternations. In: ACM/IEEE Symposium on Logic in Computer Science (LICS). IEEE Computer Society, pp. 202–213.

[31] Place, T., Zeitoun, M., 2015. Separation and the successor relation. In: Symposium on Theoretical Aspects of Computer Science (STACS). Vol. 30 of LIPIcs. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, pp. 662–675.

[32] Rampersad, N., Shallit, J., Xu, Z., 2012. The computational complexity of universality problems for prefixes, suffixes, factors, and subwords of regular languages. Fundamenta Informatica 116 (1-4), 223–236.

[33] Schmitz, H., 2000. The forbidden pattern approach to concatenation hierachies. Ph.D. thesis, University of Würzburg, Würzburg, Germany.

[34] Schützenberger, M. P., 1976. Sur le produit de concatenation non ambigu. Semigroup Forum 13 (1), 47–75.

[35] Schwentick, T., Thérien, D., Vollmer, H., 2001. Partially-ordered two-way automata: A new characterization of DA. In: Developments in Language Theory (DLT). Vol. 2295 of LNCS. Springer, pp. 239–250.

[36] Sénizergues, G., 1997. The equivalence problem for deterministic pushdown automata is decidable. In: International Colloquium on Automata, Languages and Programming (ICALP). Vol. 1256 of LNCS. Springer, pp. 671–681.

[37] Simon, I., 1972. Hierarchies of events with dot-depth one. Ph.D. thesis, Department of Applied Analysis and Computer Science, University of Waterloo, Canada.

[38] Stefanoni, G., Motik, B., Krötzsch, M., Rudolph, S., 2014. The complexity of answering conjunctive and navigational queries over OWL 2 EL knowledge bases. Journal of Artificial Intelligence Research 51, 645–705.

[39] Stockmeyer, L. J., Meyer, A. R., 1973. Word problems requiring exponential time: Preliminary report. In: ACM Symposium on the Theory of Computing (STOC). ACM, pp. 1–9.

[40] Straubing, H., 1981. A generalization of the Schützenberger product of finite monoids. Theoretical Computer Science 13, 137–150.

[41] Straubing, H., 1985. Finite semigroup varieties of the form $\mathbf{V}*\mathbf{D}$. Journal of Pure and Applied Algebra 36, 53–94.

[42] Thérien, D., 1981. Classification of finite monoids: The language approach. Theoretical Computer Science 14, 195–208.

[43] Wagner, K. W., 2004. Leaf language classes. In: Machines, Computations, and Universality (MCU). Vol. 3354 of LNCS. Springer, pp. 60–81.