Deciding Universality of ptNFAs is PSPACE-Complete*

Tomáš Masopust [1] and Markus Krötzsch [2]

[1] Institute of Mathematics, Czech Academy of Sciences, masopust@math.cas.cz
[2] cfaed, TU Dresden, Germany, markus.kroetzsch@tu-dresden.de

Abstract. An automaton is partially ordered if the only cycles in its transition diagram are self-loops. We study the universality problem for ptNFAs, a class of partially ordered NFAs recognizing piecewise testable languages. The universality problem asks if an automaton accepts all words over its alphabet. Deciding universality for both NFAs and partially ordered NFAs is PSPACE-complete. For ptNFAs, the complexity drops to coNP-complete if the alphabet is fixed but is open if the alphabet may grow. We show, using a novel and nontrivial construction, that the problem is PSPACE-complete if the alphabet may grow polynomially.

1 Introduction

Piecewise testable languages form a strict subclass of *star-free languages* or, in other words, of the languages definable by the linear temporal logic. They are investigated and find applications in semigroup theory [2, 25], in logic on words [9], in formal languages and automata theory [17], recently mainly in applications of separability [26], in natural language processing [10, 28], in cognitive and sub-regular complexity [29], in learning theory [11, 18], or in database theory in the context of schema languages for XML data [8, 14, 15, 20]. They have been extended from words to trees [4, 12].

Simon [31] showed that piecewise testable languages are exactly those regular languages whose syntactic monoid is $\mathcal{J}$-trivial and that they are characterized by *confluent, partially ordered DFAs*. An automaton is *partially ordered* if the only cycles are self-loops, and it is *confluent* if for any state q and any two of its successors s and t accessible from q by transitions labeled by a and b, respectively, there is a word $w \in \{a, b\}^*$ such that a common state is reachable from both s and t under w; cf. Fig. 1 (left) for an illustration.

Omitting confluence results in *partially ordered DFAs* (poDFAs) characterizing *$\mathcal{R}$-trivial languages* [6]. Lifting the notion of partial order from DFAs to NFAs, partially ordered NFAs (poNFAs) characterize the languages of level $\frac{3}{2}$ of the Straubing-Thérien hierarchy [30]; hence poNFAs are strictly more powerful than poDFAs. These languages are better known as *Alphabetical Pattern Constraints*, which are regular languages effectively closed under permutation rewriting used in algorithmic verification [5].

* Supported by DFG grants KR 4381/1-1 & CRC 912 (HAEC), and by RVO 67985840.

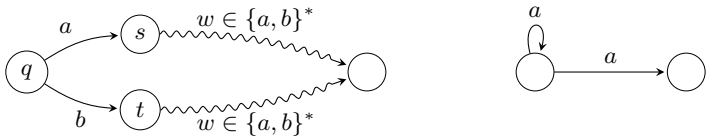

Fig. 1. Confluence (left) and the forbidden pattern of self-loop det. poNFAs (right)

In our recent work, we showed that the increased expressivity of poNFAs is caused by self-loop transitions involved in nondeterminism. Consequently, $\mathcal{R}$-trivial languages are characterized by self-loop deterministic poNFAs (denoted by rpoNFAs from *restricted* poNFAs) [19]. A poNFA is *self-loop deterministic* if it does not contain the pattern of Fig. 1 (right). Our study further revealed that complete, confluent and self-loop deterministic poNFAs (denoted by ptNFAs from *piecewise testable*) characterize piecewise testable languages [21, 23]. An NFA is *complete* if a transition under every letter is defined in every state.

In this paper, we study the *universality* problem of ptNFAs. The problem asks if an automaton accepts all words over its alphabet. The study of universality (and its dual, emptiness) has a long tradition in formal languages with many applications across computer science, e.g., in knowledge representation and database theory [3, 7, 32]. The problem is PSPACE-complete for NFAs [24]. Recent studies investigate the problem for specific types of regular languages, such as prefixes or factors [27].

Despite a rather low expressivity of poNFAs, the universality problem for poNFAs has the same worst-case complexity as for general NFAs, even if restricted to binary alphabets [19]. This is because poNFAs have a powerful nondeterminism. The pattern of Fig. 1 (right) admits an unbounded number of nondeterministic steps—the poNFA either stays in the same state or moves to another. Forbidding the pattern results in rpoNFAs where the number of nondeterministic steps is bounded by the number of states. This restriction affects the complexity of universality. Deciding universality for rpoNFAs is coNP-complete if the alphabet is fixed but remains PSPACE-complete if the alphabet may grow polynomially [19]. The growth of the alphabet thus compensates for the restriction on the number of nondeterministic steps. The reduced complexity is also preserved by ptNFAs if the alphabet is fixed [21] but is open if the alphabet may grow.

We solve this problem by showing that deciding universality for ptNFAs is PSPACE-complete if the alphabet may grow polynomially. To this aim, we use a novel and nontrivial extension of the construction for rpoNFAs [19]. Consequently, our result provides lower-bound complexities for the problems of inclusion, equivalence, and k-piecewise testability [21]. The results are summarized in Table 1.

2 Preliminaries

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. The empty word is

	$\lvert\Sigma\rvert = 1$	$\lvert\Sigma\rvert = k \geq 2$	Σ is growing
DFA	L-comp. [16]	NL-comp. [16]	NL-comp. [16]
ptNFA	NL-comp. (Thm. 1)	coNP-comp. [21]	PSPACE-comp. (Thm. 2)
rpoNFA	NL-comp. [19]	coNP-comp. [19]	PSPACE-comp. [19]
poNFA	NL-comp. [19]	PSPACE-comp. [19]	PSPACE-comp. [1]
NFA	coNP-comp. [33]	PSPACE-comp. [1]	PSPACE-comp. [1]

Table 1. Complexity of deciding universality

denoted by ε. For a word $w = xyz$, x is a *prefix*, y a *factor*, and z a *suffix* of w. A prefix (factor, suffix) of w is *proper* if it is different from w.

Let $\mathcal{A} = (Q, \Sigma, \cdot, I, F)$ be a *nondeterministic finite automaton* (NFA). 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 Iw instead of $I \cdot w$. 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 $q_0 a_1 q_1 a_2 \ldots q_{n-1} a_n q_n$ such that $q_{i+1} \in q_i \cdot a_{i+1}$, for $i = 0, \ldots, n-1$. Path π is *accepting* if $q_0 \in I$ and $q_n \in F$. We write $q_0 \xrightarrow{a_1 a_2 \cdots a_n} q_n$ to denote that there is a path from q_0 to q_n under the word $a_1 a_2 \cdots a_n$. Automaton $\mathcal{A}$ is *complete* if for every state q of $\mathcal{A}$ and every letter $a \in \Sigma$, the set $q \cdot a$ is nonempty. An NFA $\mathcal{A}$ is *deterministic* (DFA) if $|I| = 1$ and $|q \cdot a| = 1$ for every $q \in Q$ and every $a \in \Sigma$.

The reachability relation $\leq$ on states is defined by $p \leq q$ if there is a $w \in \Sigma^*$ such that $q \in p \cdot w$. An NFA $\mathcal{A}$ is *partially ordered (poNFA)* if the reachability relation $\leq$ is a partial order. For two states p and q of $\mathcal{A}$, we write $p < q$ if $p \leq q$ and $p \neq q$. A state p is *maximal* if there is no state q such that $p < q$.

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

A poNFA $\mathcal{A}$ over Σ with the state set Q can be turned into a directed graph $G(\mathcal{A})$ with the set of vertices Q where a pair $(p, q) \in Q \times Q$ is an edge in $G(\mathcal{A})$ if there is a transition from p to q in $\mathcal{A}$. For an alphabet $\Gamma \subseteq \Sigma$, we define the directed graph $G(\mathcal{A}, \Gamma)$ with the set of vertices Q by considering only those transitions corresponding to letters in Γ. For a state p, let $\Sigma(p) = \{a \in \Sigma \mid p \xrightarrow{a} p\}$ denote all letters labeling self-loops in p. We say that $\mathcal{A}$ satisfies the *unique maximal state* (UMS) property if, for every state q of $\mathcal{A}$, state q is the unique maximal state of the connected component of $G(\mathcal{A}, \Sigma(q))$ containing q.

Definition 1. *An NFA $\mathcal{A}$ is a* ptNFA *if it is partially ordered, complete and satisfies the UMS property.*

An equivalent notion to the UMS property for DFAs is confluence [17]. A DFA $\mathcal{D}$ over Σ is *(locally) confluent* if, for every state q of $\mathcal{D}$ and every pair of letters $a, b \in \Sigma$, there is a word $w \in \{a, b\}^*$ such that $(qa)w = (qb)w$. We generalize this notion to NFAs as follows. An NFA $\mathcal{A}$ over Σ is *confluent* if, for every state q of $\mathcal{A}$ and every pair of (not necessarily distinct) letters $a, b \in \Sigma$, if $s \in qa$ and $t \in qb$, then there is a word $w \in \{a, b\}^*$ such that $sw \cap tw \neq \emptyset$.

Lemma 1 ([21]). *Complete and confluent rpoNFAs are exactly ptNFAs.*

3 Complexity of Universality for ptNFAs

We now study the universality problem for ptNFAs. If the alphabet is fixed, deciding universality for ptNFAs is coNP-complete and the hardness holds even if restricted to binary alphabets [21]. For unary alphabets, universality for ptNFAs is decidable in polynomial time [19]. The following theorem improves this result.

Theorem 1. *Deciding universality of ptNFAs over a unary alphabet is an* NL*-complete problem.*

If the alphabet may grow polynomially, the universality problem for ptNFAs is open. In the rest of this paper we solve this problem by showing that the universality problem for ptNFAs is PSpace-complete.

A typical proof showing PSpace-hardness of universality for NFAs is to take a p-space bounded deterministic Turing machine $\mathcal{M}$, for a polynomial p, together with an input x, and to encode the computations of $\mathcal{M}$ on x as words over some alphabet Σ that depends on the alphabet and the state set of $\mathcal{M}$. One then constructs a regular expression (or an NFA) R_x representing all computations that do not encode an accepting run of $\mathcal{M}$ on x. That is, $L(R_x) = \Sigma^*$ if and only if $\mathcal{M}$ does not accept x [1].

The form of R_x is relatively simple, consisting of a union of expressions of the form

$$\Sigma^* \, K \, \Sigma^* \tag{1}$$

where K is a finite language with words of length bounded by $O(p(|x|))$.

Intuitively, K encodes possible violations of a correct computation of $\mathcal{M}$ on x, such as the initial configuration does not contain the input x, or the step from a configuration to the next one does not correspond to any rule of $\mathcal{M}$. These checks are local, involving at most two consecutive configurations of $\mathcal{M}$, each of polynomial size. They can therefore be encoded as a finite language with words of polynomial length.

The initial Σ^* of (1) then nondeterministically guesses a position in the word where a violation encoded by K occurs, and the last Σ^* reads the rest of the word if the violation check was successful.

This idea cannot be directly used to prove Theorem 2 for two reasons:

(i) Although expression (1) can easily be translated to a poNFA, it is not true for ptNFAs. The translation of the leading part $\Sigma^* K$ may result in the forbidden pattern of Fig. 1;

(ii) The constructed poNFA may be incomplete and its "standard" completion by adding the missing transitions to a new sink state may violate the UMS property.

A first observation to overcome these problems is that the length of the encoding of a computation of $\mathcal{M}$ on x is at most exponential with respect to the size of $\mathcal{M}$ and x. It would therefore be sufficient to replace the initial Σ^* in (1) by prefixes of an exponentially long word. However, such a word cannot be constructed by a polynomial-time reduction. Instead, we replace Σ^* with a

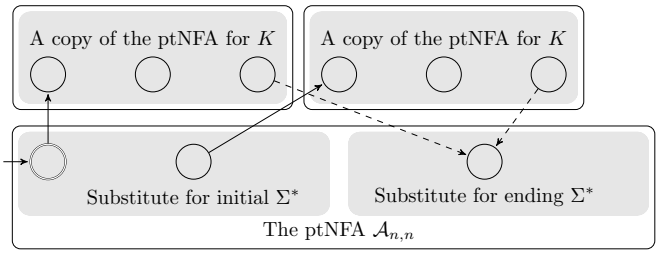

Fig. 2. Const. of an rpoNFA (solid edges) solving prob. **(i)**, illustrated for two copies of the ptNFA for K, and its completion to a ptNFA (dashed edges) solving prob. **(ii)**

ptNFA encoding such a word, which exists and is of polynomial size as shown in Lemma 2. There we construct, in polynomial time, a ptNFA $\mathcal{A}_{n,n}$ that accepts all words but a single one, $W_{n,n}$, of exponential length.

Since the language K of (1) is finite, and hence piecewise testable, there is a ptNFA for K. For every state of $\mathcal{A}_{n,n}$, we make a copy of the ptNFA for K and identify its initial state with the state of $\mathcal{A}_{n,n}$ if it does not violate the forbidden pattern of Fig. 1; see Fig. 2 for an illustration. We keep track of the words read by both $\mathcal{A}_{n,n}$ and the ptNFA for K by taking the Cartesian product of their alphabets. A letter is then a pair of symbols, where the first symbol is the input for $\mathcal{A}_{n,n}$ and the second is the input for the ptNFA for K. A word over this alphabet is accepted if the first components do not form $W_{n,n}$ or the second components form a word that is not a correct encoding of a run of $\mathcal{M}$ on x. This results in an rpoNFA that overcomes problem **(i)**.

However, this technique is not sufficient to resolve problem **(ii)**. Although the construction yields an rpoNFA that is universal if and only if the regular expression R_x is [19], the rpoNFA is incomplete and its "standard" completion by adding the missing transitions to an additional sink state violates the UMS property. According to Lemma 1, to construct a ptNFA from the rpoNFA, we need to complete the latter so that it is confluent. This is not possible for every rpoNFA, but it is possible for our case because the length of the input that is of interest is bounded by the length of $W_{n,n}$. The maximal state of $\mathcal{A}_{n,n}$ is accepting, and therefore all the missing transitions can be added so that the paths required by confluence meet in the maximal state of $\mathcal{A}_{n,n}$. Since all words longer than $|W_{n,n}|$ are accepted by $\mathcal{A}_{n,n}$, we could complete the rpoNFA by adding paths to the maximal state of $\mathcal{A}_{n,n}$ that are longer than $|W_{n,n}|$. However, this cannot be done by a polynomial-time reduction, since the length of $W_{n,n}$ is exponential. Instead, we add a ptNFA to encode such paths in the formal definition of $\mathcal{A}_{n,n}$ as given in Lemma 2 below. We then ensure confluence by adding the missing transitions to states of the ptNFA $\mathcal{A}_{n,n}$ from which the unread part of $W_{n,n}$ is not accepted and from which the maximal state of $\mathcal{A}_{n,n}$ is reachable under the symbol of the added transition (cf. Corollary 1). The second condition ensures confluence, since all the transitions meet in the maximal state of $\mathcal{A}_{n,n}$. The idea is illustrated in Fig. 2. The details follow.

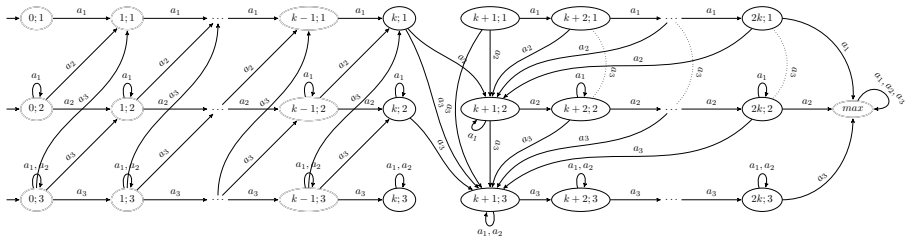

Fig. 3. The ptNFA $\mathcal{A}_{k,3}$ with $3(2k+1)+1$ states; all undefined transitions go to state *max*; dotted lines depict arrows from $(k+i,1)$ to $(k+1,3)$ under a_3, for $i = 2,3,\ldots,k$

By this construction, we do not get the same language as defined by the regular expression R_x, but the language of the constructed ptNFA is universal if and only if R_x is, which suffices for universality.

Thus, the first step of the construction is to construct the ptNFA $\mathcal{A}_{n,n}$ that accepts all words but $W_{n,n}$ of exponential length. This automaton is the core of the proof of Theorem 2. The language considered there is the same as in our previous work [19, Lemma 17], where the constructed automaton is not a ptNFA.

Lemma 2. *For all integers $k, n \geq 1$, there exists a ptNFA $\mathcal{A}_{k,n}$ over an n-letter alphabet with $n(2k+1)+1$ states, such that the unique non-accepted word of $\mathcal{A}_{k,n}$ is of length $\binom{k+n}{k} - 1$.*

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

$$W_{k,n} = W_{k,n-1}\, a_n\, W_{k-1,n} = W_{k,n-1}\, a_n\, W_{k-1,n-1}\, a_n \cdots a_n\, W_{1,n-1}\, a_n\,.$$

The length of $W_{k,n}$ is $\binom{k+n}{n} - 1$ [23]. Notice that letter 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 a ptNFA $\mathcal{A}_{k,n}$ 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 a DFA for $\{a_1\}^* \setminus \{a_1^k\}$ with k additional unreachable states used to address problem **(ii)** and included here for uniformity (see Corollary 1). $\mathcal{A}_{k,1}$ consists of $2k+1$ states of the form $(i;1)$ and a state *max*, as shown in the top-most row of states in Fig. 3, together with the given a_1-transitions. All states but $(i;1)$, for $i = k,\ldots,2k$, are accepting, and $(0;1)$ is initial. All undefined transitions in Fig. 3 go to state *max*.

Given a ptNFA $\mathcal{A}_{k,n-1}$, we recursively construct $\mathcal{A}_{k,n}$ as defined next. The construction for $n = 3$ is illustrated in Fig. 3. We obtain $\mathcal{A}_{k,n}$ from $\mathcal{A}_{k,n-1}$ by adding $2k+1$ states $(0;n), (1;n), \ldots, (2k;n)$, where $(0;n)$ is added to the initial states, and all states $(i;n)$ with $i < k$ are added to the accepting states. The automaton $\mathcal{A}_{k,n}$ therefore has $n(2k+1)+1$ states. The additional transitions of $\mathcal{A}_{k,n}$ consist of the following groups:

1. Self-loops $(i;n) \xrightarrow{a_j} (i;n)$ for $i \in \{0,1,\ldots,2k\}$ and $a_j = a_1, a_2, \ldots, a_{n-1}$;
2. Transitions $(i;n) \xrightarrow{a_n} (i+1;n)$ for $i \in \{0,1,\ldots,2k-1\} \setminus \{k\}$;
3. Transitions $(k,n) \xrightarrow{a_n} max$, $(2k,n) \xrightarrow{a_n} max$, and the self-loop $max \xrightarrow{a_n} max$;
4. Transitions $(i;n) \xrightarrow{a_n} (i+1;m)$ for $i = 0,1,\ldots,k-1$ and $m = 1,\ldots,n-1$;
5. Transitions $(i;m) \xrightarrow{a_n} max$ for every accepting state $(i;m)$ of $\mathcal{A}_{k,n-1}$;
6. Transitions $(i;m) \xrightarrow{a_n} (k+1,n)$ for every non-accepting state $(i;m)$ of $\mathcal{A}_{k,n-1}$.

By construction, $\mathcal{A}_{k,n}$ is complete and partially ordered. It satisfies the UMS property because if there is a self-loop in a state $q \neq max$ under a letter a, then there is no other incoming or outgoing transition of q under a. This means that the component of the graph $G(\mathcal{A}_{k,n}, \Sigma(q))$ containing q is only state q, which is indeed the unique maximal state. Hence, it is a ptNFA. Equivalently, to see that the automaton is confluent, the reader may notice that the automaton has a single sink state.

We show that $\mathcal{A}_{k,n}$ accepts $\Sigma_n^* \setminus \{W_{k,n}\}$. The additional states of $\mathcal{A}_{k,n}$ and transitions 1, 2, and 3 ensure acceptance of every word that does not contain exactly k occurrences of a_n. The transitions 4 and 5 ensure acceptance of all words in $(\Sigma_{n-1}^* a_n)^i L(\mathcal{A}_{k-i,n-1}) a_n \Sigma_n^*$, for which the longest factor before the $(i+1)$th occurrence of a_n is not of the form $W_{k-i,n-1}$, and hence not a correct factor of $W_{k,n} = W_{k,n-1}a_n \cdots a_n W_{k-i,n-1} a_n \cdots a_n W_{1,n-1} a_n$. 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 $\mathcal{A}_{k,1}$ (cf. Fig. 3).

For the inductive case $(k,n) \geq (1,2)$, assume that $\mathcal{A}_{k',n'}$ does not accept $W_{k',n'}$ for any $(k',n') < (k,n)$. 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. Therefore, after reading $W_{k,n-1}a_n$, automaton $\mathcal{A}_{k,n}$ must be in one of the states $(1;m)$, $1 \leq m \leq n$, or $(k+1;n)$. However, states $(1;m)$, $1 \leq m \leq n$, are the initial states of $\mathcal{A}_{k-1,n}$, which does not accept $W_{k-1,n}$ by induction. Assume that $\mathcal{A}_{k,n}$ is in state $(k+1;n)$ after reading $W_{k,n-1}a_n$. Since $W_{k-1,n}$ has exactly $k-1$ occurrences of letter a_n, $\mathcal{A}_{k,n}$ is in state $(2k;n)$ after reading $W_{k-1,n}$. Hence $W_{k,n}$ is not accepted by $\mathcal{A}_{k,n}$. □

The last part of the previous proof shows that the suffix $W_{k-1,n}$ of the word $W_{k,n} = W_{k,n-1}a_n W_{k-1,n}$ is not accepted from state $(k+1;n)$. This can be generalized as follows.

Corollary 1. *For any suffix $a_i w$ of $W_{k,n}$, w is not accepted from state $(k+1;i)$ of $\mathcal{A}_{k,n}$.*

The proof of Lemma 2 also shows that the transitions of 6 are redundant.

Corollary 2. *Removing from $\mathcal{A}_{k,n}$ the non-accepting states $(k+1,i),\ldots,(2k,i)$, for $1 \leq i \leq n$, and the corresponding transitions results in an rpoNFA that accepts the same language.*

A *deterministic Turing machine* (DTM) is a tuple $M = (Q, T, I, \delta, \sqcup, q_o, q_f)$, 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 δ is the transition function mapping $Q \times T$ to $Q \times T \times \{L, R, S\}$; see Aho et al. [1] for details.

We now prove the main result, whose proof is a nontrivial generalization of our previous construction showing PSPACE-hardness of universality for rpoNFAs [19].

Theorem 2. *The universality problem for ptNFAs is* PSPACE*-complete.*

Proof. Membership follows since universality is in PSPACE for NFAs [13].

To prove PSPACE-hardness, we consider a polynomial p and a p-space-bounded DTM $\mathcal{M} = (Q, T, I, \delta, \sqcup, q_o, q_f)$. Without loss of generality, we assume that $q_o \neq q_f$. A configuration of $\mathcal{M}$ on x consists of a current state $q \in Q$, the position $1 \leq \ell \leq p(|x|)$ of the read/write head, and the 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 $\Delta = T \times (Q \cup \{\varepsilon\})$. A run of $\mathcal{M}$ on x is represented as a 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 $\mathcal{M}$ (in particular are not of the form $\#w_1\#w_2\#\cdots\#w_m\#$) or that encode a run that is not accepting [1]. Such a regular expression can be constructed in the following three steps: we detect all words that

(A) do not start with the initial configuration;
(B) do not encode a valid run since they violate a transition rule;
(C) encode non-accepting runs or runs that end prematurely.

If $\mathcal{M}$ has an accepting run, it has one without repeated configurations. For an input x, there are $C(x) = (|T \times (Q \cup \{\varepsilon\})|)^{p(|x|)}$ distinct configuration words in our encoding. Considering a separator symbol #, the length of the encoding of a run without repeated configurations is at most $1 + C(x)(p(|x|) + 1)$, since every configuration word 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 constructed in Lemma 2. Since $|W_{n,n}| + 1 = \binom{2n}{n} \geq 2^n$, it follows that n is smaller than $\lceil \log(1 + C(x)(p(|x|) + 1)) \rceil$, and hence polynomial in the size of $\mathcal{M}$ and x.

Consider the ptNFA $\mathcal{A}_{n,n}$ over the alphabet $\Sigma_n = \{a_1, \ldots, a_n\}$ of Lemma 2, and define the alphabet $\Delta_{\#\$} = T \times (Q \cup \{\varepsilon\}) \cup \{\#, \$\}$. We consider the alphabet $\Pi = \Sigma_n \times \Delta_{\#\$}$ where the first letter is an input for $\mathcal{A}_{n,n}$ and the second letter is used for encoding a run as described above. Recall that $\mathcal{A}_{n,n}$ accepts all words different from $W_{n,n}$. Therefore, only those words over Π are of our interest, where the first components form the word $W_{n,n}$. Since the length of $W_{n,n}$ 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 $w[1] = a_{i_1} \cdots a_{i_\ell} \in \Sigma_n^\ell$ as the projection of w to the first component and $w[2] = \delta_1 \dots \delta_\ell \in \Delta_{\#\$}^\ell$ as the projection to the second component. 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.

Let $\mathrm{enc}(\mathcal{A}_{n,n})$ denote the automaton $\mathcal{A}_{n,n}$ with each transition $q \xrightarrow{a_i} q'$ replaced by all transitions $q \xrightarrow{\pi} q'$ with $\pi \in \mathrm{enc}(a_i)$. Then $\mathrm{enc}(\mathcal{A}_{n,n})$ accepts the language $\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\}$. We say that a word w encodes an accepting run of $\mathcal{M}$ 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 $i \in \{1, 2, \dots, m\}$ for which $\#w_1\# \cdots \#w_i\#$ encodes an accepting run of $\mathcal{M}$ on x, $w_k = w_i$ for all $k \in \{i+1, \dots, m\}$, and $j \le p(|x|)$. That is, 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 symbol \$.

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

$$w[2] = \#\langle x_1, q_0\rangle\langle x_2, \varepsilon\rangle \cdots \langle x_{|x|}, \varepsilon\rangle\langle \sqcup, \varepsilon\rangle \cdots \langle \sqcup, \varepsilon\rangle\#$$

of length $p(|x|) + 2$. This happens if (A.1) the word is shorter than $p(|x|) + 2$, or (A.2) at position j, for $0 \le j \le p(|x|) + 1$, there is a letter from the alphabet $\Delta_{\#\$} \setminus \{x_j\}$. Let $\bar{E}_j = \Sigma_n \times (\Delta_{\#\$} \setminus \{x_j\})$ where x_j is the jth symbol on the initial tape of $\mathcal{M}$. We can capture (A.1) and (A.2) in the regular expression

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

Expression (2) is polynomial in size. It can be captured by a ptNFA as follows. Each of the first $p(|x|) + 2$ expressions defines a finite language and can easily be captured by a ptNFA (by a confluent DFA) of size of the expression. The disjoint union of these ptNFAs then form a single ptNFA recognizing the language $\varepsilon + \Pi + \Pi^2 + \ldots + \Pi^{p(|x|)+1}$.

To express the language $\Pi^j \cdot \bar{E}_j \cdot \Pi^*$ as a ptNFA, we first construct the minimal incomplete DFA recognizing this language (states $0, 1, \dots, j, j+1, max$ in Fig. 4). However, we cannot complete it by simply adding the missing transitions to a new sink state because it results in a DFA with two maximal states, max and the sink state, violating the UMS property. Instead, we use a copy of the ptNFA $\mathrm{enc}(\mathcal{A}_{n,n})$ and add the missing transitions from state j under $\mathrm{enc}(x_j)$ to state $(n+1; i)$ if $\mathrm{enc}(x_j)[1] = a_i$; see Fig. 4. Notice that states $(n+1; i)$ are the states $(k+1; i)$ in Fig. 3. The resulting automaton is a ptNFA, since it is complete, partially ordered, and satisfies the UMS property—for every state q different from max, the component co-reachable and reachable under the letters of self-loops in q is only state q itself. The automaton accepts all words of $\Pi^j \cdot \bar{E}_j \cdot \Pi^*$.

We now show that any word w that is accepted by this automaton and that does not belong to $\Pi^j \cdot \bar{E}_j \cdot \Pi^*$ is such that $w[1] \neq W_{n,n}$, that is, it belongs to $\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\}$. Assume that $w[1] = W_{n,n} = ua_iv$, where a_i is the position

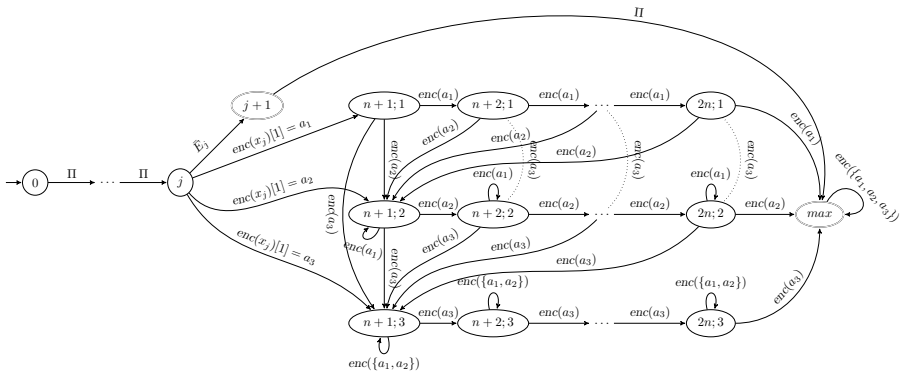

Fig. 4. A ptNFA accepting the language $\Pi^j \cdot \bar{E}_j \cdot \Pi^* + (\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\}$ illustrated for $\Sigma_n = \{a_1, a_2, a_3\}$; only the relevant part of $\mathcal{A}_{n,n}$ is depicted

and the letter under which the state $(n+1;i)$ of $\mathcal{A}_{n,n}$ is reached. Then v is not accepted from $(n+1;i)$ by Corollary 1. Thus, the ptNFA accepts the language $\Pi^j \cdot \bar{E}_j \cdot \Pi^* + (\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\})$. Constructing such a ptNFA for polynomially many expressions $\Pi^j \cdot \bar{E}_j \cdot \Pi^*$ and taking their union results in a polynomially large ptNFA accepting the language $\sum_{j=0}^{p(|x|)+1}(\Pi^j \cdot \bar{E}_j \cdot \Pi^*) + (\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\})$.

Notice that we ensure that the surrounding # in the initial configuration are present.

For **(B)**, we check for incorrect transitions. Consider again the encoding $\#w_1\# \ldots \#w_m\#$ of a sequence of configurations with a word over $\Delta \cup \{\#\}$. We can assume that w_1 encodes the initial configuration according to **(A)**. 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 extend f to $f\colon \Delta^3_{\#\$} \to \Delta_{\#\$}$. For allowing the last configuration to be repeated, we define f as if the accepting state q_f of $\mathcal{M}$ 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 can then check for invalid transitions using the regular expression

$$\Pi^* \sum_{\delta_\ell, \delta, \delta_r \in \Delta_{\#\$}} \mathrm{enc}(\delta_\ell \delta \delta_r) \cdot \Pi^{p(|x|)-1} \cdot \hat{f}(\delta_\ell, \delta, \delta_r) \cdot \Pi^* \tag{3}$$

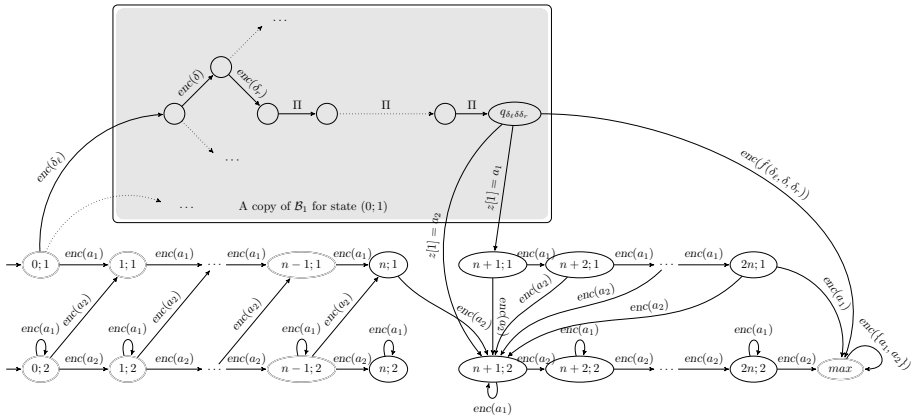

Fig. 5. ptNFA $\mathcal{B}$ consisting of $\mathrm{enc}(\mathcal{A}_{n,n})$, $n = 2$, with, for illustration, only one copy of ptNFA $\mathcal{B}_1$ for the case the initial state of $\mathcal{B}_1$ is identified with state $(0;1)$ and state max' with state max

where $\hat{f}(\delta_\ell, \delta, \delta_r)$ is $\Pi \setminus \mathrm{enc}(\{f(\delta_\ell, \delta, \delta_r), \$\})$. Note that (3) only detects wrong transitions if a long enough next configuration exists. The case that the run stops prematurely is covered in **(C)**.

Expression (3) is not readily encoded in a ptNFA because of 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 because the computations of interest are of length $|W_{n,n}|$ and a violation of a correct computation must occur. As $|W_{n,n}| - 1$ is exponential, we cannot encode it directly and we use $\mathrm{enc}(\mathcal{A}_{n,n})$ instead.

In detail, let E be the expression obtained from (3) by omitting the initial Π^*, and let $\mathcal{B}_1$ be an incomplete DFA that accepts the language of E constructed as follows. From the initial state, we construct a tree-shaped DFA corresponding to all words of length three of the finite language $\sum_{\delta_\ell,\delta,\delta_r \in \Delta_{\#\$}} \mathrm{enc}(\delta_\ell\delta\delta_r)$. To every leaf state, we add a path under Π of length $p(|x|) - 1$. The result corresponds to the language $\sum_{\delta_\ell,\delta,\delta_r \in \Delta_{\#\$}} \mathrm{enc}(\delta_\ell\delta\delta_r) \cdot \Pi^{p(|x|)-1}$. Let $q_{\delta_\ell\delta\delta_r}$ denote the states uniquely determined by the words in $\mathrm{enc}(\delta_\ell\delta\delta_r) \cdot \Pi^{p(|x|)-1}$. We add the transitions $q_{\delta_\ell\delta\delta_r} \xrightarrow{\mathrm{enc}(\hat{f}(\delta_\ell,\delta,\delta_r))} \mathit{max}'$, where max' is a new accepting state. The automaton is illustrated in the upper part of Fig. 5, denoted $\mathcal{B}_1$. It is an incomplete DFA for language E of polynomial size. It is incomplete only in states $q_{\delta_r\delta\delta_\ell}$ due to the missing transitions under $\mathrm{enc}(f(\delta_\ell, \delta, \delta_r))$ and $\mathrm{enc}(\$)$. We complete it by adding the missing transitions to the states of the ptNFA $\mathcal{A}_{n,n}$. Namely, for $z \in \{\mathrm{enc}(f(\delta_\ell, \delta, \delta_r)), \mathrm{enc}(\$)\}$, we add $q_{\delta_\ell\delta\delta_r} \xrightarrow{z} (n+1; i)$ if $z[1] = a_i$.

We construct a ptNFA $\mathcal{B}$ accepting the language $(\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\}) + (\Pi^{\leq |W_{n,n}|-1} \cdot E)$ by merging $\mathrm{enc}(\mathcal{A}_{n,n})$ with at most $n(n+1)$ copies of $\mathcal{B}_1$, where we identify the initial state of each such copy with a unique accepting state of $\mathrm{enc}(\mathcal{A}_{n,n})$, if it does not violate the property of ptNFAs (Fig. 1). This

is justified by Corollary 2, since we do not need to consider connecting $\mathcal{B}_1$ to non-accepting states of $\mathcal{A}_{n,n}$ and it is not possible to connect it to state max. We further identify state max' of every copy of $\mathcal{B}_1$ with state max of $\mathcal{A}_{n,n}$. The fact that $\mathrm{enc}(\mathcal{A}_{n,n})$ alone accepts $(\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\})$ was shown in Lemma 2. 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) $\mathcal{B}_1$ after accepting a word of length $\geq |W_{n,n}|$ is possible but all such words are longer than $W_{n,n}$ and hence in $(\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\})$.

Let w be a word that is not accepted by (a copy of) $\mathcal{B}_1$. Then, there are words u and v such that u leads $\mathrm{enc}(\mathcal{A}_{n,n})$ to a state from which w is read in a copy of $\mathcal{B}_1$. Since w is not accepted, there is a letter z and a word v such that uwz goes to state $(n+1;i)$ of $\mathcal{A}_{n,n}$ (for $z[1] = a_i$) and v leads $\mathrm{enc}(\mathcal{A}_{n,n})$ from state $(n+1;i)$ to state max. If $u[1]w[1]a_iv[1] = W_{n,n,}$, then v is not accepted from $(n+1;i)$ by Corollary 1, and hence $uwzv[1] \neq W_{n,n}$.

It remains to show that for every proper 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 $\mathcal{B}_1$, 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 there must be a state reachable by $w_{n,n}$ in $\mathcal{A}_{n,n}$ that does not have a self-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 $a_{n,n}$.

The ptNFA $\mathcal{B}$ thus accepts the language $\Pi^{\leq |W_{n,n}|-1} \cdot E + (\Pi^* \setminus \{\mathrm{enc}(W_{n,n})\})$.

Finally, for **(C)**, we detect all words that (C.1) end in a configuration that is incomplete (too short), (C.2) end in a configuration that is not in the accepting state q_f, (C.3) end with more than $p(|x|)$ trailing \$, or (C.4) contain \$ not only at the last positions, that is, we detect all words where \$ is followed by a different symbol. For a word v, we use $v^{\leq i}$ to abbreviate $\varepsilon + v + \ldots + v^i$, and we define $\bar{E}_f = (T \times (Q \setminus \{q_f\}))$.

$$\begin{aligned}
&\textbf{(C.1)}\ \ \Pi^* \,\mathrm{enc}(\#)(\Pi + \ldots + \Pi^{p(|x|)})\,\mathrm{enc}(\$)^{\leq p(|x|)} + \\
&\textbf{(C.2)}\ \ \Pi^* \,\mathrm{enc}(\bar{E}_f)(\varepsilon + \Pi + \ldots + \Pi^{p(|x|)-1})\,\mathrm{enc}(\#)\,\mathrm{enc}(\$)^{\leq p(|x|)} + \\
&\textbf{(C.3)}\ \ \Pi^* \,\mathrm{enc}(\$)^{p(|x|)+1} + \\
&\textbf{(C.4)}\ \ (\Pi \setminus \mathrm{enc}(\$))^* \,\mathrm{enc}(\$)\,\mathrm{enc}(\$)^* (\Pi \setminus \mathrm{enc}(\$))\Pi^*
\end{aligned} \tag{4}$$

As before, we cannot encode the expression directly as a ptNFA, but we can perform a similar construction as the one used for encoding (3).

The expressions (2)–(4) together then detect all non-accepting or wrongly encoded runs of $\mathcal{M}$. 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 ptNFA of polynomial size, we have thus reduced the word problem of polynomially space-bounded Turing machines to the universality problem for ptNFAs. □

All missing proofs can be found in the full version of this paper [22].

References

1. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The Design and Analysis of Computer Algorithms. Addison-Wesley (1974)
2. Almeida, J., Costa, J.C., Zeitoun, M.: Pointlike sets with respect to R and J. Journal of Pure and Applied Algebra 212(3), 486–499 (2008)
3. Barceló, P., Libkin, L., Reutter, J.L.: Querying regular graph patterns. Journal of the ACM 61(1), 8:1–8:54 (2014)
4. Bojanczyk, M., Segoufin, L., Straubing, H.: Piecewise testable tree languages. Logical Methods in Computer Science 8(3) (2012)
5. Bouajjani, A., Muscholl, A., Touili, T.: Permutation rewriting and algorithmic verification. Information and Computation 205(2), 199–224 (2007)
6. Brzozowski, J.A., Fich, F.E.: Languages of R-trivial monoids. Journal of Computer and System Sciences 20(1), 32–49 (1980)
7. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: Reasoning on regular path queries. ACM SIGMOD Record 32(4), 83–92 (2003)
8. Czerwiński, W., Martens, W., Masopust, T.: Efficient separability of regular languages by subsequences and suffixes. In: Internation Colloquium on Automata, Languages and Programming. LNCS, vol. 7966, pp. 150–161. Springer (2013)
9. Diekert, V., Gastin, P., Kufleitner, M.: A survey on small fragments of first-order logic over finite words. Int. Journal of Foundations of Computer Science 19(3), 513–548 (2008)
10. Fu, J., Heinz, J., Tanner, H.G.: An algebraic characterization of strictly piecewise languages. In: Theory and Applications of Models of Computation, LNCS, vol. 6648, pp. 252–263. Springer (2011)
11. García, P., Ruiz, J.: Learning k-testable and k-piecewise testable languages from positive data. Grammars 7, 125–140 (2004)
12. García, P., Vidal, E.: Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 12(9), 920–925 (1990)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
14. Hofman, P., Martens, W.: Separability by short subsequences and subwords. In: International Conference on Database Theory. LIPIcs, vol. 31, pp. 230–246. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2015)
15. Holub, Š., Jirásková, G., Masopust, T.: On upper and lower bounds on the length of alternating towers. In: Mathematical Foundations of Computer Science. LNCS, vol. 8634, pp. 315–326. Springer (2014)
16. Jones, N.D.: Space-bounded reducibility among combinatorial problems. Journal of Computer and System Sciences 11(1), 68–85 (1975)
17. Klíma, O., Polák, L.: Alternative automata characterization of piecewise testable languages. In: Developments in Language Theory. LNCS, vol. 7907, pp. 289–300. Springer (2013)
18. Kontorovich, L., Cortes, C., Mohri, M.: Kernel methods for learning languages. Theoretical Computer Science 405(3), 223–236 (2008)
19. Krötzsch, M., Masopust, T., Thomazo, M.: Complexity of universality and related problems for partially ordered NFAs. Information and Computation (2017), accepted. Preprint available at http://arxiv.org/abs/1609.03460
20. Martens, W., Neven, F., Niewerth, M., Schwentick, T.: Bonxai: Combining the simplicity of DTD with the expressiveness of XML schema. In: Principles of Database Systems. pp. 145–156. ACM (2015)

21. Masopust, T.: Piecewise testable languages and nondeterministic automata. In: Mathematical Foundations of Computer Science. LIPIcs, vol. 58, pp. 67:1–67:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
22. Masopust, T., Krötzsch, M.: Universality of confluent, self-loop deterministic partially ordered NFAs is hard (2017), http://arxiv.org/abs/1704.07860
23. Masopust, T., Thomazo, M.: On boolean combinations forming piecewise testable languages. Theoretical Computer Science 682, 165–179 (2017)
24. Meyer, A.R., Stockmeyer, L.J.: The equivalence problem for regular expressions with squaring requires exponential space. In: Symposium on Switching and Automata Theory. pp. 125–129. IEEE Computer Society (1972)
25. Perrin, D., Pin, J.E.: Infinite words: Automata, semigroups, logic and games, Pure and Applied Mathematics, vol. 141. Academic Press (2004)
26. Place, T., van Rooijen, L., Zeitoun, M.: Separating regular languages by piecewise testable and unambiguous languages. In: Mathematical Foundations of Computer Science. LNCS, vol. 8087, pp. 729–740. Springer (2013)
27. Rampersad, N., Shallit, J., Xu, Z.: The computational complexity of universality problems for prefixes, suffixes, factors, and subwords of regular languages. Fundamenta Informatica 116(1–4), 223–236 (2012)
28. Rogers, J., Heinz, J., Bailey, G., Edlefsen, M., Visscher, M., Wellcome, D., Wibel, S.: On languages piecewise testable in the strict sense. In: The Mathematics of Language. LNAI, vol. 6149, pp. 255–265. Springer (2010)
29. Rogers, J., Heinz, J., Fero, M., Hurst, J., Lambert, D., Wibel, S.: Cognitive and sub-regular complexity. In: Formal Grammar. LNCS, vol. 8036, pp. 90–108. Springer (2013)
30. Schwentick, T., Thérien, D., Vollmer, H.: Partially-ordered two-way automata: A new characterization of DA. In: Developments in Language Theory. LNCS, vol. 2295, pp. 239–250. Springer (2001)
31. Simon, I.: Hierarchies of Events with Dot-Depth One. Ph.D. thesis, University of Waterloo, Canada (1972)
32. Stefanoni, G., Motik, B., Krötzsch, M., Rudolph, S.: The complexity of answering conjunctive and navigational queries over OWL 2 EL knowledge bases. Journal of Artificial Intelligence Research 51, 645–705 (2014)
33. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time: Preliminary report. In: ACM Symposium on the Theory of Computing. pp. 1–9. ACM (1973)