Universality of Confluent, Self-Loop Deterministic Partially Ordered NFAs is Hard[*]

Tomáš Masopust[1] and Markus Krötzsch[2]

1   **Institute of Theoretical Computer Science and Center of Advancing Electronics Dresden (cfaed), TU Dresden, Germany,** `tomas.masopust@tu-dresden.de`
2   **Institute of Theoretical Computer Science and Center of Advancing Electronics Dresden (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. The expressivity of partially ordered NFAs (poNFAs) can be characterized by the Straubing-Thérien hierarchy. Level 3/2 is recognized by poNFAs, level 1 by confluent, self-loop deterministic poNFAs as well as by confluent poDFAs, and level 1/2 by saturated poNFAs. We study the universality problem for confluent, self-loop deterministic poNFAs. It asks whether an automaton accepts all words over its alphabet. Universality for both NFAs and poNFAs is a PSPACE-complete problem. For confluent, self-loop deterministic poNFAs, the complexity drops to CONP-complete if the alphabet is fixed but is open if the alphabet may grow. We solve this problem by showing that it is PSPACE-complete if the alphabet may grow polynomially. Consequently, our result provides a lower-bound complexity for some other problems, including inclusion, equivalence, and k-piecewise testability. Since universality for saturated poNFAs is easy, confluent, self-loop deterministic poNFAs are the simplest and natural kind of NFAs characterizing a well-known class of languages, for which deciding universality is as difficult as for general NFAs.

1   Introduction

McNaughton and Papert [29] showed that *first-order logic* describes *star-free languages*, a class of regular languages whose syntactic monoid is *aperiodic* [36]. Restricting the number of alternations of quantifiers in formulae in the prenex normal form results in a quantifier alternation hierarchy. The choice of predicates provides several hierarchies. The well-known and closely related are the *Straubing-Thérien (ST) hierarchy* [42, 44] and the *dot-depth hierarchy* [8, 11, 43].

We are interested in automata characterizations of the levels of the ST hierarchy, alternatively defined as follows. For an alphabet Σ, $\mathcal{L}(0) = \{\emptyset, \Sigma^*\}$ and, for integers $n \geq 0$, level $\mathcal{L}(n + 1/2)$ consists of all finite unions of languages $L_0 a_1 L_1 a_2 \ldots a_k L_k$ with $k \geq 0$, $L_0, \ldots, L_k \in \mathcal{L}(n)$, and $a_1, \ldots, a_k \in \Sigma$, and level $\mathcal{L}(n + 1)$ consists of all finite Boolean combinations of languages from level $\mathcal{L}(n + 1/2)$. The hierarchy does not collapse on any level [8]. For some levels, an algebraic or automata characterization is known. This characterization is particularly interesting in decision and complexity questions, such as the membership of a

─────────────────────
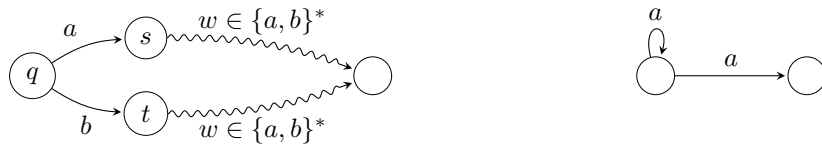
■ **Figure 1** Confluence (left) and the forbidden pattern of self-loop deterministic poNFAs (right)

language in a specific level of the hierarchy. Despite a recent progress [2, 31, 33], deciding whether a language belongs to level k of the ST hierarchy is still open for $k > 7/2$.

The most studied level of the ST hierarchy is level 1, known as *piecewise testable languages* introduced by Simon [39]. Simon showed that piecewise testable languages are those regular languages whose syntactic monoid is $\mathcal{J}$-trivial and that they are recognized by *confluent, partially ordered DFAs*. An automaton is *partially ordered* if its transition relation induces a partial order on states – 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. Figure 1 (left) for an illustration.

Omitting confluence results in *partially ordered DFAs* (poDFAs) studied by Brzozowski and Fich [7]. They showed that poDFAs characterize $\mathcal{R}$-*trivial languages*, a class of languages strictly between level 1 and level 3/2 of the ST hierarchy. Lifting the notion from DFAs to NFAs, Schwentick, Thérien and Vollmer [37] showed that partially ordered NFAs (poNFAs) characterize level 3/2 of the ST hierarchy. Hence poNFAs are more powerful than poDFAs. Languages of level 3/2 are also known as *Alphabetical Pattern Constraints* [5], which are regular languages effectively closed under permutation rewriting.

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 [23]. A poNFA is *self-loop deterministic* if it does not contain the pattern of Figure 1 (right). Our study further revealed that complete, confluent and self-loop deterministic poNFAs characterize piecewise testable languages [27, 28]. An NFA is *complete* if a transition under every letter is defined in every state. Complete, confluent and self-loop deterministic poNFAs are thus a natural extension of confluent poDFAs to nondeterministic automata.

In this paper, we study the *universality* problem of complete, confluent and self-loop deterministic poNFAs. The problem asks whether a given 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, 9, 40]. The problem is PSPACE-complete for NFAs [30]. Recent studies investigate it for specific types of regular languages, such as prefixes or factors [34].

In spite of 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 [23]. This might be because poNFAs possess quite a powerful nondeterminism. The pattern of Figure 1 (right), which may occur in poNFAs, admits an unbounded number of nondeterministic steps – the poNFA either stays in the same state or moves to another one. Forbidding the pattern results in self-loop deterministic poNFAs where the number of nondeterministic steps is bounded by the number of states. This restriction affects the complexity of universality. Deciding universality of self-loop deterministic poNFAs is CONP-complete if the alphabet is fixed but remains PSPACE-complete if the alphabet may grow polynomially [23]. The growth of the alphabet thus compensates for the restriction on the number of nondeterministic steps.

Table 1 Complexity of deciding universality for poNFAs and special classes thereof; ST stands for the corresponding level of the ST hierarchy; Σ denotes the input alphabet

	ST	$|\Sigma| = 1$	$|\Sigma| = k \geq 2$	Σ is growing
DFA		L-comp. [19]	NL-comp. [19]	NL-comp. [19]
spoNFA	$\frac{1}{2}$	AC^0 (Thm. 1)	AC^0 (Thm. 1)	AC^0 (Thm. 1)
ptNFA	1	NL-comp. (Thm. 3)	CoNP-comp. [27]	PSpace-comp. (Thm. 7)
rpoNFA		NL-comp. [23]	CoNP-comp. [23]	PSpace-comp. [23]
poNFA	$\frac{3}{2}$	NL-comp. [23]	PSpace-comp. [23]	PSpace-comp. [1]
NFA		CoNP-comp. [41]	PSpace-comp. [1]	PSpace-comp. [1]

The reduced complexity is also preserved by complete, confluent and self-loop deterministic poNFAs if the alphabet is fixed [27] but is open if the alphabet may grow.

We solve this problem by showing that deciding universality for complete, confluent and self-loop deterministic poNFAs is PSpace-complete if the alphabet may grow polynomially, which strengthens our previous result for self-loop deterministic poNFAs [23].

Consequently, the k-piecewise testability problem for complete, confluent and self-loop deterministic poNFAs, which was open [27], is PSpace-complete. The problem asks whether a given language is a finite boolean combination of languages of the form $\Sigma^* a_1 \Sigma^* \cdots \Sigma^* a_n \Sigma^*$, where $a_i \in \Sigma$ and $0 \leq n \leq k$. It is of interest in XML databases and separability [17, 26]. The result follows from the fact that 0-piecewise testability coincides with universality, and from the results in Masopust [27]. The problem is CoNP-complete for DFAs [20].

Another consequence is the worst-case complexity of the inclusion and equivalence problems for restricted NFAs, problems that are of interest, e.g., in optimization. The problems ask, given languages K and L, whether $K \subseteq L$, resp. $K = L$. Universality can be expressed as the inclusion $\Sigma^* \subseteq L$ or as the equivalence $\Sigma^* = L$. Although equivalence means two inclusions, complexities of these two problems may differ significantly – inclusion is undecidable for deterministic context-free languages [13] while equivalence is decidable [38]. The complexity of universality provides a lower bound on the complexity of both. Deciding inclusion or equivalence of two languages given as complete, confluent and self-loop deterministic poNFAs is thus PSpace-complete in general, CoNP-complete if the alphabet is fixed, and NL-complete if the alphabet is unary, see Masopust [27] for the lower bound of the second result and Krötzsch et al. [22] for the upper bound of the last two results.

To complete the picture, Héam [16] characterized level 1/2 of the ST hierarchy as languages recognized by saturated poNFAs (spoNFA), also called *shuffle ideals* or *upward closures*. A poNFA is *saturated* if it has a self-loop under every letter in every state. Deciding universality for spoNFAs is simple – it means to find a state that is both initial and accepting. Therefore, complete, confluent and self-loop deterministic poNFAs are the simplest and natural kind of NFAs recognizing a well-known class of languages for which the universality problem is as difficult as for general NFAs.

The following notation has been used for poNFAs and we adopt it in the sequel:

- self-loop deterministic poNFAs are denoted by *restricted poNFAs* or *rpoNFAs* [23], and
- complete, confluent and self-loop deterministic poNFAs are denoted by *ptNFAs* [27], where pt stands for piecewise testable.

An overview of the results is summarized in Table 1.

All proofs or their parts omitted in the paper may be found in the appendix.

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 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 all $i = 0, 1, \ldots, n-1$. 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 state $q \in Q$ and every letter $a \in \Sigma$.

The reachability relation $\leq$ on the set of states is defined by $p \leq q$ if there exists a word $w \in \Sigma^*$ 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 $p < q$. Partially ordered automata are sometimes called acyclic automata, where self-loops are allowed.

A *restricted partially ordered NFA (rpoNFA)* is a poNFA that is self-loop deterministic in the sense that the automaton contains no pattern of Figure 1 (right). Formally, for every state q and every letter a, if $q \in q \cdot a$ then $q \cdot a = \{q\}$.

A *saturated poNFA* (*spoNFA*) is a poNFA $\mathcal{A}$ such that, for every state q and every letter a, $q \xrightarrow{a} q$ is a transition in $\mathcal{A}$.

▶ **Theorem 1.** *Universality of spoNFAs is decidable in* AC^0 *(hence strictly simpler than* L*).*

Proof. The language of an spoNFA is universal if and only if ε belongs to it, which is if and only if one of the initial states is accepting. Given a binary encoding of states and their properties, a family of uniform constant-depth circuits can check if any state has initial and accepting bits set. Unbounded fan-in gates allow us to test any number of (initial) states. ◀

3 Confluent and Self-Loop Deterministic poNFAs – ptNFAs

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.

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 [21]. 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 $sw \cap tw \neq \emptyset$. The following relationship between ptNFAs and rpoNFAs holds [27].

▶ **Lemma 2.** *Complete and confluent rpoNFAs are exactly ptNFAs.*

As a result, complete and confluent rpoNFAs characterize piecewise testable languages [27], a class of languages that recently re-attracted attention because of some applications in logic on words [32] and in XML Schema languages [12, 17, 26]; see also Masopust [27] for a brief overview of their applications in mathematics and computer science.

We now study the universality problem for ptNFAs. Recall that if the alphabet is fixed, deciding universality for ptNFAs is CONP-complete and that hardness holds even if restricted to binary alphabets [27]. If the alphabet is unary, universality for ptNFAs is decidable in polynomial time [23]. We now show that it is NL-complete.

▶ **Theorem 3.** *The universality problem for ptNFAs over a unary alphabet is* NL-*complete.*

In the case the alphabet may grow polynomially, the universality problem for ptNFAs is open. In the rest of this paper we solve this problem by showing the following result.

▶ **Theorem 7.** *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, 4, 23]. 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) 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 7 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 Figure 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 ptNFA encoding such a word, which exists and is of polynomial size as shown in Lemma 4. 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 language K of (1) is finite, 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 Figure 1; see Figure 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
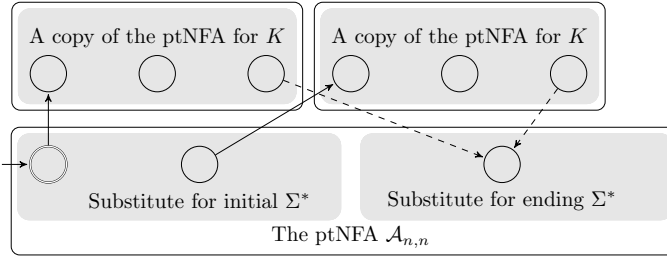
■ **Figure 2** Construction of an rpoNFA (solid edges) solving problem **(i)**, illustrated for two copies of the ptNFA for K, and its completion to a ptNFA (dashed edges) solving problem **(ii)**

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 [23], 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 2, 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, 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 4 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 5). The second condition ensures confluence, since all the transitions meet in the maximal state of $\mathcal{A}_{n,n}$. The idea is illustrated in Figure 2. The details follow.

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.

Thus, the first step 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 7 and its construction is described in the following lemma. The language considered there is the same as in our previous work [23, Lemma 17], where the constructed automaton is not a ptNFA.

▶ **Lemma 4.** *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

$$W_{k,n} = W_{k,n-1}\, a_n\, W_{k-1,n} = W_{k,n-1}\, a_n\, W_{k-1,n-1}\, a_n\, \cdots\, a_n\, W_{1,n-1}\, a_n\,. \tag{2}$$

The length of $W_{k,n}$ is $\binom{k+n}{n} - 1$ [28]. 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.
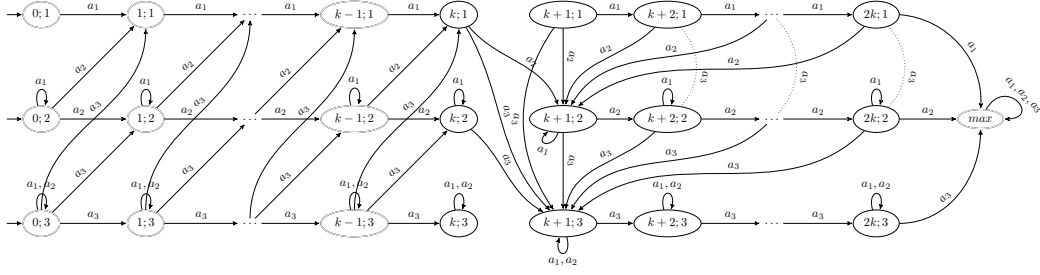
Figure 3 The ptNFA $\mathcal{A}_{k,3}$ with $3(2k+1)+1$ states; all undefined transitions go to state *max*; dotted lines depict arrows from $(k+i,1)$ to $(k+1,3)$ under a_3, for $i=2,3,\ldots,k$

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 5). $\mathcal{A}_{k,1}$ consists of $2k+1$ states of the form $(i;1)$ and a state *max*, as shown in the top-most row of states in Figure 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 Figure 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 Figure 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 every $i \in \{0, 1, \ldots, 2k\}$ and $a_j = a_1, a_2, \ldots, a_{n-1}$;
2. Transitions $(i;n) \xrightarrow{a_n} (i+1;n)$ for every $i \in \{0, 1, \ldots, 2k-1\} \setminus \{k\}$;
3. Transitions $(k,n) \xrightarrow{a_n} max$ and $(2k,n) \xrightarrow{a_n} max$, and the self-loop $max \xrightarrow{a_n} max$;
4. Transitions $(i;n) \xrightarrow{a_n} (i+1;m)$ for every $i = 0, 1, \ldots, k-1$ 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 $W_{k-i,n-1}$, 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. Figure 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 $(1; m)$, $1 \le m \le n$, or $(k + 1; n)$. However, states $(1; m)$, $1 \le m \le n$, 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 $W_{k,n} = W_{k,n-1}a_nW_{k-1,n}$ is not accepted from state $(k + 1; n)$. This can be generalized as follows.

▶ **Corollary 5.** *For any suffix a_iw of $W_{k,n}$, w is not accepted from state $(k + 1; i)$ of $\mathcal{A}_{k,n}$.*

The proof of Lemma 4 also shows that the transitions of **6** are redundant. We thus have the following observation.

▶ **Corollary 6.** *Removing from $\mathcal{A}_{k,n}$ the non-accepting states $(k + 1, i), \ldots, (2k, i)$, for $1 \le i \le n$, and the corresponding transitions results in an rpoNFA that accepts the same language.*

Since $\binom{2n}{n} \ge 2^n$, Lemma 4 implies that there are ptNFAs $\mathcal{A}_{n,n}$ for which the shortest non-accepted word $W_{n,n}$ is exponential in the size of $\mathcal{A}$.

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 [23].

▶ **Theorem 7.** *The universality problem for ptNFAs is PSPACE-complete.*

Proof. Membership follows from the fact that universality is in PSPACE for NFAs [14].

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 $q_o \ne q_f$. A configuration of $\mathcal{M}$ on x consists of a current state $q \in Q$, the position $1 \le \ell \le p(|x|)$ of the 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:
(A) we detect all words that do not start with the initial configuration;
(B) we detect all words that do not encode a valid run since they violate a transition rule;
(C) we detect all words that 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 $|W_{n,n}| \ge 1 + C(x)(p(|x|) + 1)$, where $W_{n,n}$ is the word constructed in Lemma 4. Since $|W_{n,n}| + 1 = \binom{2n}{n} \ge 2^n$, it follows that n is smaller than $\lceil \log(1 + C(x)(p(|x|) + 1)) \rceil$, 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 4, 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 $w[2] = \delta_1 \ldots \delta_\ell \in \Delta_{\#\$}^\ell$ as the projection to the second component. Conversely, for a word $v \in \Delta_{\#\$}^*$, we write $enc(v)$ to denote the set of all words $w \in \Pi^{|v|}$ with $w[2] = v$. Similarly, for $v \in \Sigma_n^*$, $enc(v)$ denotes the words $w \in \Pi^{|v|}$ with $w[1] = v$. We extend this notation to sets of words.

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

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 $0 \leq j \leq p(|x|) + 1$, 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

$$\left(\varepsilon + \Pi + \Pi^2 + \ldots + \Pi^{p(|x|)+1}\right) + \sum_{0 \leq j \leq p(|x|)+1} (\Pi^j \cdot \bar{E}_j \cdot \Pi^*) \tag{3}$$

Expression (3) 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 $0, 1, \ldots, j, j+1, max$ in Figure 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 $enc(\mathcal{A}_{n,n})$ and add the missing transitions from state j under $enc(x_j)$ to state $(n+1; i)$ if $enc(x_j)[1] = a_i$; see Figure 4. Notice that states $(n+1; i)$ are the states $(k+1; i)$ in Figure 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 $\Pi^* \setminus \{enc(W_{n,n})\}$. Assume that $w[1] = W_{n,n} = ua_iv$, where a_i is the position 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 5. Thus, the ptNFA accepts the language $\Pi^j \cdot \bar{E}_j \cdot \Pi^* + (\Pi^* \setminus \{enc(W_{n,n})\})$. 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 $\sum_{j=0}^{p(|x|)+1}(\Pi^j \cdot \bar{E}_j \cdot \Pi^*) + (\Pi^* \setminus \{enc(W_{n,n})\})$.
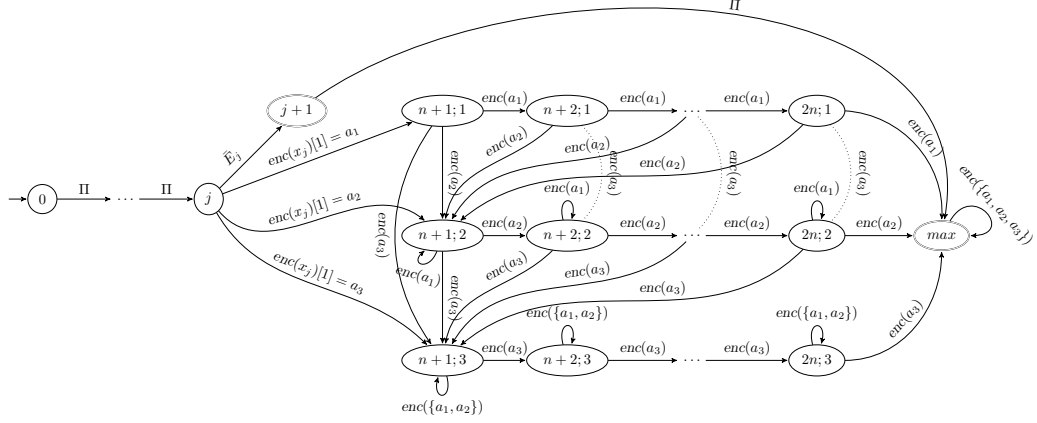
■ **Figure 4** A ptNFA accepting $\Pi^j \cdot \bar{E}_j \cdot \Pi^* + (\Pi^* \setminus \{enc(W_{n,n})\}$ illustrated for $\Sigma_n = \{a_1, a_2, a_3\}$; only the relevant part of $\mathcal{A}_{n,n}$ is depicted

Note 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 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

$$\Pi^* \sum_{\delta_\ell, \delta, \delta_r \in \Delta_{\#\$}} enc(\delta_\ell \delta \delta_r) \cdot \Pi^{p(|x|)-1} \cdot \hat{f}(\delta_\ell, \delta, \delta_r) \cdot \Pi^* \tag{4}$$

where $\hat{f}(\delta_\ell, \delta, \delta_r)$ is $\Pi \setminus enc(\{f(\delta_\ell, \delta, \delta_r), \$\})$. Note that (4) detects wrong transitions if a long enough next configuration exists. The case that the run stops prematurely is covered in **(C)**.

Expression (4) 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 $enc(\mathcal{A}_{n,n})$ instead.

In detail, let E be the expression obtained from (4) 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 $\sum_{\delta_\ell, \delta, \delta_r \in \Delta_{\#\$}} enc(\delta_\ell \delta \delta_r)$. To every leaf state, we add a path under Π of length $p(|x|) - 1$. The result corresponds to the language $\sum_{\delta_\ell, \delta, \delta_r \in \Delta_{\#\$}} enc(\delta_\ell \delta \delta_r) \cdot \Pi^{p(|x|)-1}$. Let $q_{\delta_\ell \delta \delta_r}$ denote the states uniquely determined by the words in $enc(\delta_\ell \delta \delta_r) \cdot \Pi^{p(|x|)-1}$. We
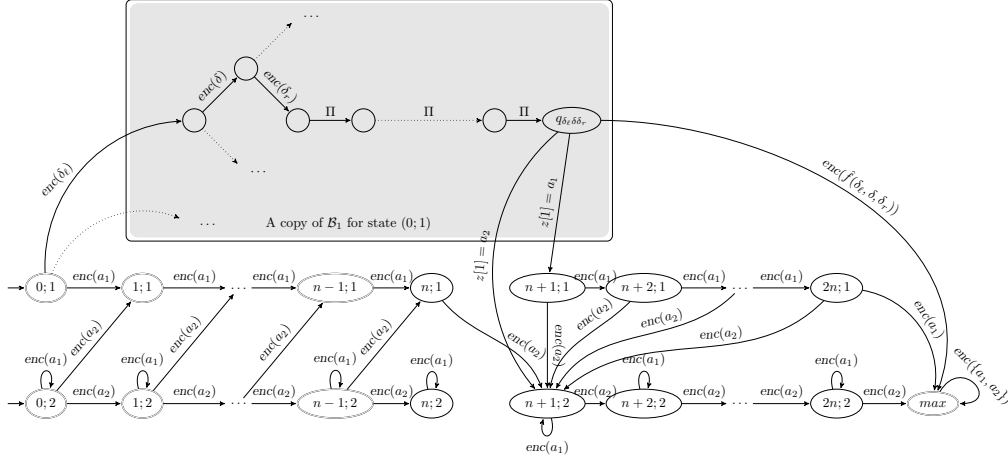
Figure 5 ptNFA $\mathcal{B}$ consisting of $enc(\mathcal{A}_{n,n})$, $n = 2$, with, for illustration, only one copy of ptNFA $\mathcal{B}_1$ for the case the initial state of $\mathcal{B}_1$ is identified with state $(0;1)$ and state max' with state max

add the transitions $q_{\delta_\ell\delta\delta_r} \xrightarrow{enc(\hat{f}(\delta_\ell,\delta,\delta_r))} max'$, where max' is a new accepting state. The automaton is illustrated in the upper part of Figure 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 $enc(f(\delta_\ell,\delta,\delta_r))$ and $enc(\$)$. We complete it by adding the missing transitions to the states of the ptNFA $\mathcal{A}_{n,n}$. Namely, for $z \in \{enc(f(\delta_\ell,\delta,\delta_r)), enc(\$)\}$, 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 $(\Pi^* \setminus \{enc(W_{n,n})\}) + (\Pi^{\leq |W_{n,n}|-1} \cdot E)$ by merging $enc(\mathcal{A}_{n,n})$ 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 $enc(\mathcal{A}_{n,n})$, if it does not violate the property of ptNFAs (Figure 1). This is justified by Corollary 6, 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 $enc(\mathcal{A}_{n,n})$ alone accepts $(\Pi^* \setminus \{enc(W_{n,n})\})$ was shown in Lemma 4. 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 $(\Pi^* \setminus \{enc(W_{n,n})\})$.

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 $enc(\mathcal{A}_{n,n})$ 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 $enc(\mathcal{A}_{n,n})$ 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 5, 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 $\mathcal{B}_1$, 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 $\mathcal{B}$ therefore accepts the language $\Pi^{\leq |W_{n,n}|-1} \cdot E + (\Pi^* \setminus \{enc(W_{n,n})\})$.

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 $\bar{E}_f = (T \times (Q \setminus \{q_f\}))$.

(C.1) $\Pi^* enc(\#)(\Pi + \ldots + \Pi^{p(|x|)}) enc(\$)^{\leq p(|x|)} +$

(C.2) $\Pi^* enc(\bar{E}_f)(\varepsilon + \Pi + \ldots + \Pi^{p(|x|)-1}) enc(\#) enc(\$)^{\leq p(|x|)} +$ (5)

(C.3) $\Pi^* enc(\$)^{p(|x|)+1} +$

(C.4) $(\Pi \setminus enc(\$))^* enc(\$) enc(\$)^* (\Pi \setminus enc(\$)) \Pi^*$

As before, we cannot encode the expression directly as a ptNFA, but we can perform a similar construction as the one used for encoding (4).

The expressions (3)–(5) together then detect all non-accepting or wrongly encoded runs of $\mathcal{M}$. In particular, if we start from the correct initial configuration ((3) does not match), then for (4) 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.   ◀

4   Discussion

Regular languages as well as recursively enumerable languages possess both deterministic and nondeterministic automata models. It is not typical – deterministic pushdown automata are strictly less powerful than nondeterministic pushdown automata, and the relationship between deterministic and nondeterministic linearly-bounded Turing machines is a longstanding open problem. Surprisingly, piecewise testable languages as well as $\mathcal{R}$-trivial languages possess such a property – $\mathcal{R}$-trivial languages are characterized by poDFAs [7] as well as by self-loop deterministic poNFAs [23], and piecewise testable languages by confluent poDFAs [21] as well as by complete, confluent and self-loop deterministic poNFAs [27].

We also point out that the languages of self-loop deterministic poNFAs (and of their restrictions) are definable by deterministic regular expressions [23]. Deterministic regular expressions [6] are of interest in schema languages for XML data, since the W3C standards require the regular expressions in their specification to be deterministic.

Whether a language is definable by a poNFA or a type thereof has also been investigated. Bouajjani, Muscholl and Touili [5] showed that deciding whether a regular language is an Alphabetical Pattern Constraints (hence recognizable by a poNFA) is PSPACE-complete for NFAs, and NL-complete for DFAs. The complexity is preserved for self-loop deterministic poNFAs [23], for complete, confluent and self-loop deterministic poNFAs [45, 10], and for saturated poNFAs [16]. In all cases, PSPACE-hardness is a consequence of a more general result by Hunt III and Rosenkrantz [18]. Although the problem whether there is an equivalent self-loop deterministic poNFA for a given DFA was not discussed in the literature, it can be seen that it reduces to checking whether the minimal DFA is partially order [23], which is an NL-complete problem.

A characterization of languages in terms of automata with forbidden patterns can be compared to the results of Glaßer and Schmitz [15, 35], who used DFAs with a forbidden pattern to obtain a characterization of level 3/2 of the dot-depth hierarchy.

Other relevant classes of partially ordered automata include partially ordered Büchi automata [24], two-way poDFAs [37], and two-way poDFAs with look-around [25].

References

1 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

2 Jorge Almeida, Jana Bartoňová, Ondřej Klíma, and Michal Kunc. On decidability of intermediate levels of concatenation hierarchies. In *Developments in Language Theory*, volume 9168 of *LNCS*, pages 58–70. Springer, 2015.

3 Pablo Barceló, Leonid Libkin, and Juan L. Reutter. Querying regular graph patterns. *Journal of the ACM*, 61(1):8:1–8:54, 2014.

4 Geert Jan Bex, Wouter Gelade, Wim Martens, and Frank Neven. Simplifying XML schema: effortless handling of nondeterministic regular expressions. In *ACM International Conference on Management of Data (SIGMOD)*, pages 731–744. ACM, 2009.

5 Ahmed Bouajjani, Anca Muscholl, and Tayssir Touili. Permutation rewriting and algorithmic verification. *Information and Computation*, 205(2):199–224, 2007.

6 Anne Brüggemann-Klein and Derick Wood. One-unambiguous regular languages. *Information and Computation*, 142(2):182–206, 1998.

7 Janusz A. Brzozowski and Faith E. Fich. Languages of R-trivial monoids. *Journal of Computer and System Sciences*, 20(1):32–49, 1980.

8 Janusz A. Brzozowski and Robert Knast. The dot-depth hierarchy of star-free languages is infinite. *Journal of Computer and System Sciences*, 16(1):37–55, 1978.

9 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Reasoning on regular path queries. *ACM SIGMOD Record*, 32(4):83–92, 2003.

10 Sang Cho and Dung T. Huynh. Finite-automaton aperiodicity is PSPACE-complete. *Theoretical Computer Science*, 88(1):99–116, 1991.

11 Rina S. Cohen and Janusz A. Brzozowski. Dot-depth of star-free events. *Journal of Computer and System Sciences*, 5(1):1–16, 1971.

12 Wojciech Czerwiński, Wim Martens, and Tomáš Masopust. Efficient separability of regular languages by subsequences and suffixes. In *Internation Colloquium on Automata, Languages and Programming*, volume 7966 of *LNCS*, pages 150–161, 2013.

13 Emily P. Friedman. The inclusion problem for simple languages. *Theoretical Computer Science*, 1(4):297–316, 1976.

14 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

15 Christian Glaßer and Heinz Schmitz. Languages of dot-depth 3/2. *Theory of Computing Systems*, 42(2):256–286, 2008.

16 Pierre-Cyrille Héam. On shuffle ideals. *RAIRO – Theoretical Informatics and Applications*, 36(4):359–384, 2002.

17 Piotr Hofman and Wim Martens. Separability by short subsequences and subwords. In *International Conference on Database Theory*, volume 31 of *LIPIcs*, pages 230–246. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

18 Harry B. Hunt III and Daniel J. Rosenkrantz. Computational parallels between the regular and context-free languages. *SIAM Journal on Computing*, 7(1):99–114, 1978.

19 Neil D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11(1):68–85, 1975.

20 Ondřej Klíma, Michal Kunc, and Libor Polák. Deciding k-piecewise testability. Submitted manuscript, 2014.

21 Ondřej Klíma and Libor Polák. Alternative automata characterization of piecewise testable languages. In *Developments in Language Theory*, volume 7907 of *LNCS*, pages 289–300. Springer, 2013.

22   Markus Krötzsch, Tomáš Masopust, and Michaël Thomazo. Complexity of universality and related problems for partially ordered NFAs. Extended version of [23]. Available at `http://arxiv.org/abs/1609.03460`, 2016.

23   Markus Krötzsch, Tomáš Masopust, and Michaël Thomazo. On the complexity of universality for partially ordered NFAs. In *Mathematical Foundations of Computer Science*, volume 58 of *LIPIcs*, pages 61:1–61:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

24   Manfred Kufleitner and Alexander Lauser. Partially ordered two-way Büchi automata. *International Journal of Foundations of Computer Science*, 22(8):1861–1876, 2011.

25   Kamal Lodaya, Paritosh K. Pandya, and Simoni S. Shah. Around dot depth two. In *Developments in Language Theory*, volume 6224 of *LNCS*, pages 303–315. Springer, 2010.

26   Wim Martens, Frank Neven, Matthias Niewerth, and Thomas Schwentick. Bonxai: Combining the simplicity of DTD with the expressiveness of XML schema. In *Principles of Database Systems*, pages 145–156. ACM, 2015.

27   Tomáš Masopust. Piecewise testable languages and nondeterministic automata. In *Mathematical Foundations of Computer Science*, volume 58 of *LIPIcs*, pages 67:1–67:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

28   Tomáš Masopust and Michaël Thomazo. On boolean combinations forming piecewise testable languages. *Theoretical Computer Science*, 2017. In press.

29   Robert McNaughton and Seymour A. Papert. *Counter-Free Automata*. The MIT Press, 1971.

30   Albert R. Meyer and Larry J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Symposium on Switching and Automata Theory*, pages 125–129. IEEE Computer Society, 1972.

31   Thomas Place. Separating regular languages with two quantifiers alternations. In *Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 202–213. IEEE Computer Society, 2015.

32   Thomas Place and Marc Zeitoun. Going higher in the first-order quantifier alternation hierarchy on words. In *International Colloquium on Automata, Languages and Programming*, volume 8573 of *LNCS*, pages 342–353. Springer, 2014.

33   Thomas Place and Marc Zeitoun. Separation and the successor relation. In *Symposium on Theoretical Aspects of Computer Science*, volume 30 of *LIPIcs*, pages 662–675. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

34   Narad Rampersad, Jeffrey Shallit, and Zhi Xu. The computational complexity of universality problems for prefixes, suffixes, factors, and subwords of regular languages. *Fundamenta Informatica*, 116(1–4):223–236, 2012.

35   Heinz Schmitz. *The forbidden pattern approach to concatenation hierachies*. PhD thesis, University of Würzburg, Germany, 2000.

36   Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

37   Thomas Schwentick, Denis Thérien, and Heribert Vollmer. Partially-ordered two-way automata: A new characterization of DA. In *Developments in Language Theory*, volume 2295 of *LNCS*, pages 239–250. Springer, 2001.

38   Géraud Sénizergues. $L(A) = L(B)$? *Electronic Notes in Theoretical Computer Science*, 9:43, 1997.

39   Imre Simon. *Hierarchies of Events with Dot-Depth One*. PhD thesis, University of Waterloo, Canada, 1972.

40   Giorgio Stefanoni, Boris Motik, Markus Krötzsch, and Sebastian Rudolph. The complexity of answering conjunctive and navigational queries over OWL 2 EL knowledge bases. *Journal of Artificial Intelligence Research*, 51:645–705, 2014.

41 Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In *ACM Symposium on the Theory of Computing*, pages 1–9. ACM, 1973.

42 Howard Straubing. A generalization of the Schützenberger product of finite monoids. *Theoretical Computer Science*, 13:137–150, 1981.

43 Howard Straubing. Finite semigroup varieties of the form V*D. *Journal of Pure and Applied Algebra*, 36:53–94, 1985.

44 Denis Thérien. Classification of finite monoids: The language approach. *Theoretical Computer Science*, 14:195–208, 1981.

45 Štěpán Holub, Tomáš Masopust, and Michaël Thomazo. Alternating towers and piecewise testable separators. Available at `http://arxiv.org/abs/1409.3943`, 2014.

A   Proofs

In this part, we present proofs omitted in the main body of the paper.

The following result was reported without proof [27]. For the convenience of reviewers, we provide the proof here.

▶ **Lemma 2.** *Complete and confluent rpoNFAs are exactly ptNFAs.*

Proof. First, we show that if $\mathcal{A}$ is a ptNFA, then $\mathcal{A}$ is a complete and confluent rpoNFA. Indeed, the definition of ptNFAs says that $\mathcal{A}$ is complete, partially ordered, and does not contain the pattern of Figure 1, since the pattern violates the UMS property. Thus, it is a complete rpoNFA. To show that $\mathcal{A}$ is confluent, let r be a state of $\mathcal{A}$, and let a and b be letters of its alphabet ($a = b$ is not excluded) such that $ra \ni s \neq t \in rb$. Let s' and t' be any maximal states reachable from s and t under the alphabet $\{a, b\}$, respectively. By the UMS property of $\mathcal{A}$, there is a path from t' to s' under $\Sigma(s')$ and a path from s' to t' under $\Sigma(t')$. Since $\mathcal{A}$ is partially ordered, $s' = t'$, which shows that $\mathcal{A}$ is confluent.

On the other hand, assume that $\mathcal{A}$ is a complete and confluent rpoNFA. To show that it is a ptNFA, we show that it satisfies the UMS property. For the sake of contradiction, assume that the UMS property is not satisfied, that is, there is a state q in $\mathcal{A}$ such that the component $G(\mathcal{A}, \Sigma(g))$ of $\mathcal{A}$ containing q and consisting only of transitions labeled with $\Sigma(q)$ has at least two maximal states with respect to $\Sigma(q)$. Let r be the biggest state in $G(\mathcal{A}, \Sigma(g))$ with respect to the partial order on states such that at least two different maximal states, say $s \neq t$, are reachable from r under $\Sigma(q)$. Such a state exists by assumption. We have that $r \notin \{s, t\}$; indeed, if $r = s$, then $t \in r \cdot au$, for some $a \in \Sigma(q)$ and $u \in \Sigma(q)^*$. Since $\mathcal{A}$ is an rpoNFA, it does not have any patter of Figure 1, which means that $a \notin \Sigma(s) \supseteq \Sigma(q)$, a contradiction. Let $s' \in ra$ and $t' \in rb$ be two different states on the path from r to s and t, respectively, for some letters $a, b \in \Sigma(q) \setminus \Sigma(r)$. Then $r < \min\{s', t'\}$. Since $\mathcal{A}$ is confluent, there exists r' such that $r' \in s'w \cap t'w$, for some $w \in \{a, b\}^*$. Let r'' denote a maximal state that is reachable from r' under $\Sigma(q)$. There are three cases: (i) if $r'' = s$, then $r < t'$ and both s and t are reachable from t' under $\Sigma(q)$, which yields a contradiction with the choice of r; (ii) $r'' = t$ yields a contradiction with the choice of r as in (i) by replacing t' with s'; and (iii) $r'' \notin \{s, t\}$ yields also a contradiction with the choice of r, since $r < \min\{s', t'\}$ and, e.g., s and r'' are two different maximal states with respect to $\Sigma(q)$ reachable from s' under $\Sigma(q)$. Thus, $\mathcal{A}$ satisfies the UMS property, which completes the proof. ◀

▶ **Theorem 3.** *The universality problem for ptNFAs over a unary alphabet is* NL*-complete.*

Proof. The problem is in NL even for unary poNFAs [22]. We prove hardness by reduction from the NL-complete DAG-reachability problem [19]. Let G be a directed acyclic graph

with n nodes, and let s and t be two nodes of G. We define a ptNFA $\mathcal{A}$ 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 a transition $i \xrightarrow{a} j$ to $\mathcal{A}$. In addition, we add $n-1$ new non-accepting states $f_1, \ldots, f_{n-1}$ together with the transitions $f_i \xrightarrow{a} f_{i+1}$, for $i = 1, \ldots, n-2$. For every state $q \notin \{t, f_1, \ldots, f_{n-1}\}$, we add a transition $q \xrightarrow{a} f_1$. Finally, we add a self-loop $t \xrightarrow{a} t$ and a transition $f_{n-1} \xrightarrow{a} t$. The initial state of $\mathcal{A}$ is s and all states corresponding to nodes of G are final. The automaton is partially ordered, complete and satisfies the UMS property, since state t is the only state with a self-loop and every path under a^* ends up in it.

It remains to show that $\mathcal{A}$ is universal if and only if there is a path from s to t in G. If t is reachable from s in G, then $L(\mathcal{A}) = \Sigma^*$, since t is reachable from s via states corresponding to nodes of G, which are all accepting in $\mathcal{A}$. If t is not reachable from s in G, then t is reachable from s in $\mathcal{A}$ via the path $s \xrightarrow{a^k} q \xrightarrow{a} f_1 \xrightarrow{a} f_2 \xrightarrow{a} \ldots \xrightarrow{a} f_{n-1} \xrightarrow{a} t$, for any q corresponding to a node of $V \setminus \{t\}$ reachable from s in G. We show that a^{n-1} does not belong to $L(\mathcal{A})$. The shortest path from state s to state t in $\mathcal{A}$ is of length n for $q = s$. Thus, any word accepted in t is of length at least n. On the other hand, every word accepted in a state corresponding to a node of $V \setminus \{t\}$ is of length at most $n-2$, since $|V \setminus \{t\}| = n-1$ and $\mathcal{A}$ is acyclic (without self-loops) on those states. This gives that a^{n-1} is not accepted by $\mathcal{A}$, hence $L(\mathcal{A})$ is not universal. ◀

▶ **Corollary 5.** *For any suffix $a_i w$ of $W_{k,n}$, w is not accepted from state $(k+1;i)$ of $\mathcal{A}_{k,n}$.*

Proof. Consider the word $W_{k,n}$ over $\Sigma_n = \{a_1, a_2, \ldots, a_n\}$ constructed in the proof of Lemma 4, and let $i \in \{1, \ldots, n\}$ be the maximal number for which there is a suffix $a_i w$ of $W_{k,n}$ such that w is accepted by $\mathcal{A}_{k,n}$ from state $(k+1;i)$. Then $W_{k,n} = w_1 a_i w_2 w_3$, where $w_2 \in \{a_1, \ldots, a_i\}^*$ is the shortest word labeling the path from state $(k+1;i)$ to state max. By the construction of $\mathcal{A}_{k,n}$, word $a_i w_2$ must contain $k+1$ letters a_i. We shown that $W_{k,n}$ does not contain more than k letters a_i interleaved only with letters a_j for $j < i$, which yields a contradiction that proves the claim.

By definition, every longest factor of $W_{k,n}$ over $\{a_1, \ldots, a_i\}$ is of the form $W_{k-\ell,i}$, for $\ell \in \{0, \ldots, k-1\}$. Since $W_{k-\ell,i} = W_{k-\ell,i-1} \, a_i \, W_{k-\ell-1,i-1} \, a_i \, \cdots \, a_i \, W_{1,i-1} \, a_i$, the number of occurrences of a_i interleaved only with letters a_j for $j < i$ is at most $k - \ell$, which results in k for $\ell = 0$ as claimed above. ◀

Here we present the omitted part of the proof of the main theorem.

▶ **Theorem 3.** *The universality problem for ptNFAs over a unary alphabet is* NL-*complete.*

Proof (omitted parts). 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 $\bar{E}_f = (T \times (Q \setminus \{q_f\}))$.

$$
\begin{aligned}
\textbf{(C.1)} \quad & \Pi^* enc(\#)(\Pi + \ldots + \Pi^{p(|x|)}) enc(\$)^{\leq p(|x|)} + \\
\textbf{(C.2)} \quad & \Pi^* enc(\bar{E}_f)(\varepsilon + \Pi + \ldots + \Pi^{p(|x|)-1}) enc(\#) enc(\$)^{\leq p(|x|)} + \\
\textbf{(C.3)} \quad & \Pi^* enc(\$)^{p(|x|)+1} + \\
\textbf{(C.4)} \quad & (\Pi \setminus enc(\$))^* enc(\$) enc(\$)^* (\Pi \setminus enc(\$)) \Pi^*
\end{aligned}
\tag{5}
$$

As before, we cannot encode the expression directly as a ptNFA, but we can perform a similar construction as in (4). Namely, a ptNFA for C.1 is illustrated in Figure 6, for C.2 in

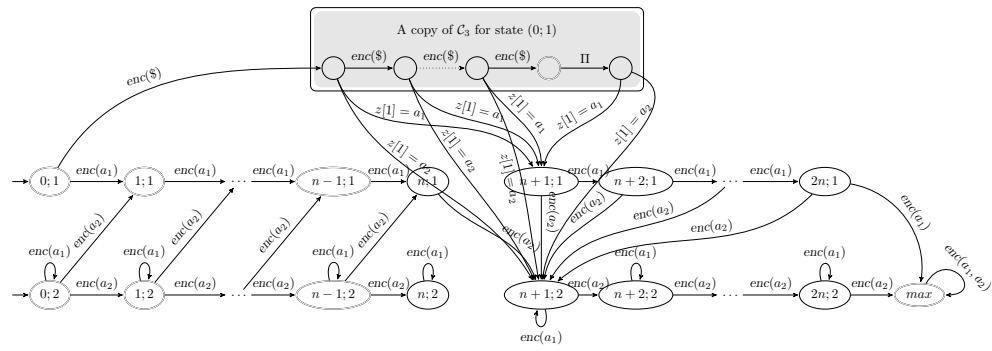Figure 7, and for C.3 in Figure 8. Finally, C.4 can be represented by a three-state partially ordered and confluent DFA.   ◀
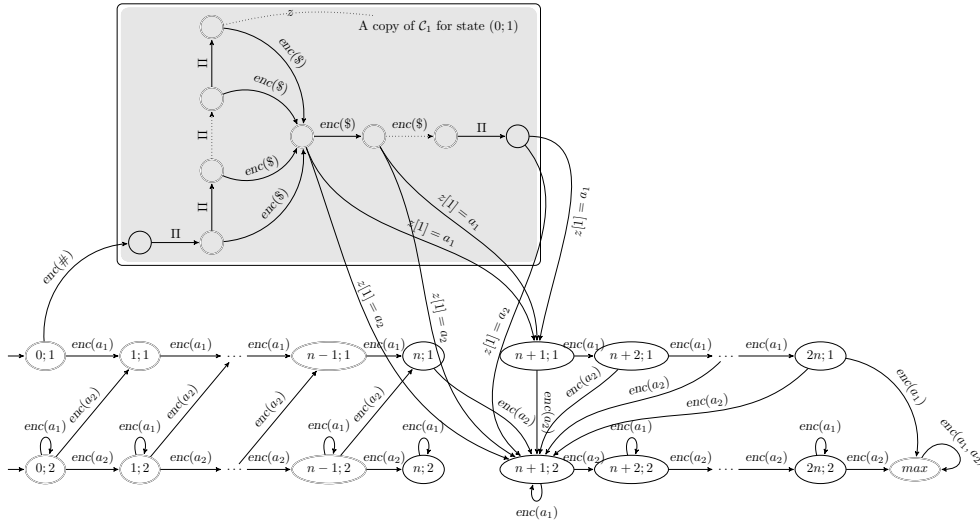
■ **Figure 6** The ptNFA for expression C.1 illustrated for $n = 2$
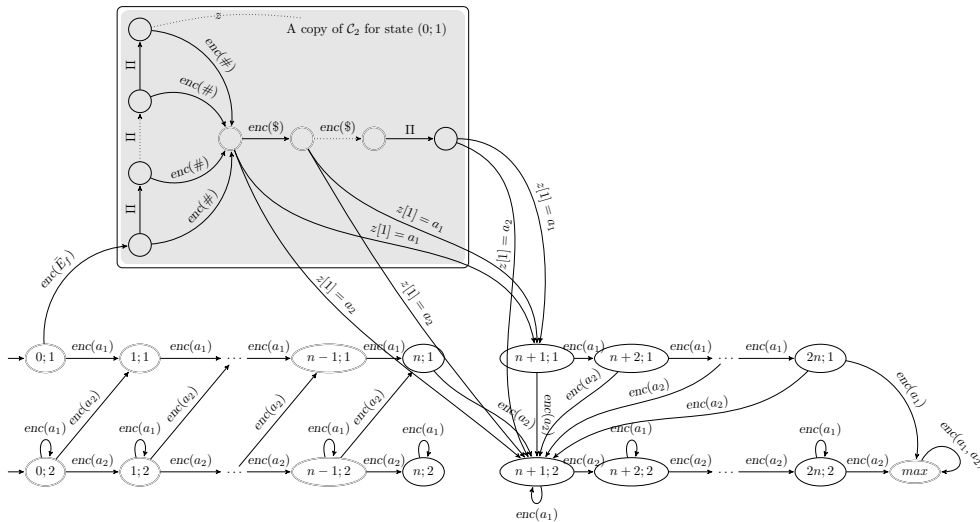
■ **Figure 7** The ptNFA for expression C.2 illustrated for $n = 2$

Figure 8 The ptNFA for expression C.3 illustrated for $n = 2$