Cycle Height of Finite Automata

Chris Keeler(✉) and Kai Salomaa

School of Computing, Queen's University, Kingston, ON K7L 2N8, Canada
{keeler,ksalomaa}@cs.queensu.ca

Abstract. A nondeterministic finite automaton (NFA) A has cycle height $\mathcal{K}$ if any computation of A can visit at most $\mathcal{K}$ cycles, and A has finite cycle height if it has cycle height $\mathcal{K}$ for some $\mathcal{K}$. We give a polynomial time algorithm to decide whether an NFA has finite cycle height and, in the positive case, to compute its optimal cycle height. Nondeterministic finite automata of finite cycle height recognize the polynomial density regular languages.

1 Introduction

Deterministic and nondeterministic finite automata define the class of regular languages and have been systematically studied for over 60 years. At the same time, many important questions on finite automata and regular languages remain open [8,10]. The last decades have seen much work on the descriptional complexity, or state complexity, of regular languages [4,6,7].

In this paper we consider a structural property of finite automata called *cycle height.* A nondeterministic finite automaton (NFA) is said to have finite cycle height if no two cycles overlap. A finite cycle height NFA A has cycle height $\mathcal{K}$ if all computations of A visit no more than $\mathcal{K}$ non-equivalent cycles.[1] The acyclic NFAs have cycle height zero and the nearly acyclic NFAs [9] have cycle height one.

Note that cycle height differs from the notion of *cycle rank* [5] which counts the degree of nesting of cycles in an NFA. Also Msiska and van Zijl [12] estimate the size blow-up of the subset construction by counting how many times a computation passes through a simple cycle. The notion is in some sense related to cycle height, but their point of view is different because the algorithm modifies the NFA by removing nested cycles.

A language L has *polynomial density* if the number of strings of length n in L is bounded by a polynomial in n. Szilard et al. [15] have shown that a language recognized by a deterministic finite automaton (DFA) A has polynomial density if all strings have a certain *tiered* property with respect to A. The tiered property is related to our notion of cycle height of NFAs, although in [15] the tiered words are defined only with respect to a DFA. As noted by

[1] By non-equivalent cycles we mean cycles that are not permutations of each other. The notion will be defined formally in Sect. 2.

S. Konstantinidis and G. Pighizzini (Eds.): DCFS 2018, LNCS 10952, pp. 200–211, 2018.
https://doi.org/10.1007/978-3-319-94631-3_17

Kutrib et al. [11], from [15] it follows that, for a polynomial density regular language L, the degree of the polynomial giving the density of L is computable. Using more advanced techniques, Gawrychowski et al. [3] have shown that for an m-state DFA A recognizing a polynomial density language the degree of the polynomial giving the density of the language can be computed in $O(m)$ time, assuming the alphabet size is constant.

The contributions of this paper are as follows. In Sect. 3 we give a polynomial time algorithm to decide whether an NFA A has finite cycle height and, in the positive case, to compute the cycle height of A. We show that NFAs with finite cycle height recognize the polynomial density regular languages, but an NFA recognizing a polynomial density language need not have finite cycle height. Based on results from [15] it then follows that a DFA A has finite cycle height if and only if the language $L(A)$ has polynomial density. Furthermore, if A has finite cycle height, the degree of the polynomial bounding the density of A is the cycle height of A minus one. This would give a polynomial time algorithm to compute the density of a language recognized by a DFA, however, the time complexity is worse than in the known algorithm from Gawrychowski et al. [3]. Finally in Sect. 4 we study upper and lower bounds for the *depth path width* [9] of NFAs with finite cycle height. The depth path width of an NFA A, roughly speaking, quantifies the overall path expansion in computations of A by counting the number of complete computations of A on all possible inputs of a given length.

2 Preliminaries

We assume the reader to be familiar with the basics of formal languages and finite automata [14]. The set of strings over a finite alphabet Σ is Σ^*, the set of strings of length $m \geq 0$ is Σ^m and ε is the empty string. The cardinality of a finite set F is denoted $|F|$ and $\mathbb{N}$ is the set of non-negative integers.

A *nondeterministic finite automaton* (NFA) is a tuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is the finite set of states, Σ is the input alphabet, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of final states. The transition function δ is in the usual way extended as a function $Q \times \Sigma^* \to 2^Q$ and the language recognized by A is $L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset\}$. If $|\delta(q, b)| \leq 1$ for all $q \in Q$ and $b \in \Sigma$, the automaton A is a *deterministic finite automaton* (DFA). It is well known that the NFAs and DFAs recognize the class of *regular languages*.

Unless otherwise mentioned, we always assume that an NFA or a DFA does not have useless states, that is, states that cannot be used in an accepting computation. Note that we can avoid a DFA having a "useless" dead state because we allow DFAs to have undefined transitions.

The *density* of a language L is a function $\mathbb{N} \to \mathbb{N}$ defined as $\varrho_L(n) = |L \cap \Sigma^n|$. A language L is said to have polynomial density if there exists an integer $d \geq 0$ such that $\varrho_L(n) \in O(n^d)$. Density is sometimes instead defined as the ratio $|L \cap \Sigma^n|/|\Sigma^n|$, with our notion of density instead being referred to as *population* [15].

2.1 Cycle Height and Depth Path Width

First we recall some definitions related to computations and cycles of an NFA. In the following $A = (Q, \Sigma, \delta, q_0, F)$ is always an NFA.

A *(state) path* of the NFA A with underlying string $w = b_1 b_2 \cdots b_k$, $b_i \in \Sigma$, $i = 1, \ldots, k$, $k \geq 0$, is a sequence of states $(p_0, p_1, \ldots, p_k)$, where $p_j \in \delta(p_{j-1}, b_j)$, $j = 1, \ldots k$. A path beginning in the start state q_0, is a *computation* of A on the underlying string w. A computation that ends in an accepting state of F is an accepting computation. A computation $(q_0, p_1, \ldots, p_\ell)$ is a *complete computation* on a string $b_1 b_2 \cdots b_k$ if $\ell = k$. The set of all computations of A on the string w is denoted $\mathrm{comp}_A(w)$.

A path $(p_0, p_1, \ldots, p_k)$, $k \geq 1$, with underlying string $b_1 b_2 \cdots b_k$ is a *cycle* if $p_0 = p_k$. A cycle with one transition from a state to itself is called a *self-loop*.

Cycles that are obtained from each other by a cyclical shift are said to be *equivalent:* For $0 < i < k$, the above cycle (with $p_0 = p_k$) is equivalent to the cycle $(p_i, \ldots, p_k, p_1, \ldots p_{i-1}, p_i)$ having underlying string $b_{i+1} \cdots b_k b_1 \cdots b_i$. In the following, unless otherwise mentioned, by a cycle we always mean an equivalence class of cycles and, thus, by two distinct cycles we mean two non-equivalent cycles.

We say that an NFA A has *finite cycle height* if for any distinct cycles C_1 and C_2 of A, either C_1 is unreachable from C_2 or C_2 is unreachable from C_1. It is easy to see that an NFA has finite cycle height if and only if no two different cycles overlap. Additionally, this condition implies a strict ordering on the cycles, since the reachability between two distinct cycles holds in at most in one direction.

A finite cycle height NFA A has cycle height $\mathcal{K} \in \mathbb{N}$ if no computation of A can contain states belonging to $\mathcal{K} + 1$ different cycles. Intuitively, this means that no computation can "pass through" $\mathcal{K} + 1$ different cycles. Note that since A has finite cycle height and a strict ordering on its cycles, a computation can "pass through" a cycle at most once. We say that A has *strict cycle height* $\mathcal{K}$ if it has cycle height $\mathcal{K}$ but not cycle height $\mathcal{K} - 1$. Note that if A has cycle height $\mathcal{K}$, then there exists a unique $0 \leq \mathcal{K}' \leq \mathcal{K}$ such that A has strict cycle height $\mathcal{K}'$.

An acyclic NFA has cycle height zero and a nearly acyclic NFA [9] has cycle height one (by its definition). Since a finite cycle height NFA cannot have overlapping cycles, the Lemma below is immediate. An n-state NFA with cycle height n is given in Fig. 2.

Lemma 1. *The finite cycle height of an n-state NFA is at most n. For each $n \in \mathbb{N}$ and $\mathcal{K} \leq n$ there exists an n-state NFA with strict cycle height $\mathcal{K}$.*

To conclude this section we recall the notion of depth path width, which counts the number of complete computations of given length. The *depth path width* [9] of A on strings of length $\ell \in \mathbb{N}$ is

$$\mathrm{DPW}(A, \ell) = \sum_{w \in \Sigma^\ell} |\mathrm{comp}_A(w)|.$$

The depth path width of the NFA A is defined as $\mathrm{DPW}^{\sup}(A) = \sup_{\ell \in \mathbb{N}}(\mathrm{DPW}(A, \ell))$.

In Sect. 4 we will use the following Lemma.

Lemma 2. *Let A be an NFA and $\ell \in \mathbb{N}$.*

(i) If A' is an NFA obtained from A by changing the alphabet symbol labeling one transition, then $\mathrm{DPW}(A', \ell) = \mathrm{DPW}(A, \ell)$.
(ii) If A'' is an NFA obtained from A by adding one new transition, then

$$\mathrm{DPW}(A'', \ell) \geq \mathrm{DPW}(A, \ell).$$

3 Polynomial Time Algorithm for Cycle Height

We present an algorithm which determines whether or not an NFA A has finite cycle height, and if so, returns the strict cycle height of A.

The idea of Algorithm 1 is as follows: For an NFA A, we first split A into its strongly connected components (SCCs). We then ensure that A does not have any nested cycles, which would prevent finite cycle height. This is done by checking that each SCC is either an acyclic singleton (consisting of only one state and no transitions) or a simple cycle (where consecutive states are connected by a unique transition).

After this, the algorithm creates an acyclic graph $G = (V, E)$, $V = \{v_0, \ldots, v_{k-1}\}$, where each vertex represents one of A's strongly connected components. Each edge (v_i, v_j) represents a connection in A between the two SCCs s_i and s_j, for $0 \leq i < j \leq k-1$. The weights of these edges represent the type of SCC to which the edge leads. That is, if (v_i, v_j) is a 0-weight edge in E, then s_j is (and v_j represents) an acyclic singleton. If (v_i, v_j) is a 1-weight edge in E, then s_j is (and v_j represents) a simple cycle SCC.

The algorithm then determines the minimum distance from v_0 (the vertex representing the SCC containing q_0) to all other v_j. Since each 1-weight edge leads to a vertex representing a cyclical SCC, the maximum-cost path starting from v_0 will lead through the most vertices representing cyclical SCCs. In fact, the length of the maximum-cost path in G starting from v_0 is the integer $\mathcal{K}$, such that A has strict cycle height $\mathcal{K}$.

In the algorithm, for states q_a and q_b of A, the *distance* from q_a to q_b is the length of the shortest string that takes q_a to q_b. If q_b is not reachable from q_a the distance is ∞.

Complexity analysis of Algorithm 1: The input is an NFA $(Q, \Sigma, \delta, q_0, F)$ with m states. Creating the distance matrix takes $\Theta(m^3)$ time and $\Theta(m^2)$ space using the Floyd-Warshall reachability algorithm [2]. Creating the set of strongly connected components can be done in $\Theta(m + |\delta|)$ time using Tarjan's SCC algorithm [16]. Checking for the existence of an SCC which is not a simple cycle is naturally bounded by the number of states and transitions in A. Clearly then, the "if" part on line 4 is not as computationally hard as the "else" part on line 6. For the else part, the two for all statements multiply the inner statements' complexity by $O(\binom{m}{2})$, as they enumerate all ordered pairs and there are maximally m SCCs.

Algorithm 1. Computing the Cycle Height of an NFA

Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA where $|Q| = m$.
Create a distance matrix M, where $M[q_a, q_b]$ is the distance from state $q_a \in Q$ to state $q_b \in Q$.
Let $s_0, s_1, \ldots, s_{k-1}$, $k \geq 1$, be the strongly connected components of A.
if there exists s_i, $0 \leq i \leq k-1$, such that s_i is not a simple cycle or acyclic singleton **then**
  return "A does not have finite cycle height"
else
  if the start state q_0 is in an acyclic singleton **then**
    startBias = 0
  else
    startBias = 1
  end if
  Create an acyclic graph $G{=}(V, E), V{=}\{v_0, \ldots, v_{k-1}\}, E{=}\emptyset$, where each v_i represents the strongly connected component s_i, for $0 \leq i \leq k-1$.
  for all s_i, $0 \leq i < k-1$ select one state q_i in s_i and **do**
    for all s_j, $i < j \leq k-1$ select one state q_j in s_j and **do**
      if $M[q_i, q_j] \neq \infty$ **then**
        if s_j is an acyclic singleton consisting of state s_j **then**
          Add a 0-weight edge to E from vertex v_i to vertex v_j.
        else
          Add a 1-weight edge to E from vertex v_i to vertex v_j.
        end if
      end if
    end for
  end for
  Let D be the distances from v_0 to all other vertices in V.
  return $\max_{v \in V}(D[v]) + startBias$
end if

Since we know that there is a strict ordering on the cycles, we do not need to compare all pairs of SCCs, as q_i is never reachable from q_j when $i < j$.

Determining the reachability between SCCs is done in constant time with the help of the distance matrix M. We create the shortest-path tree D for G using the modified Dijkstra's algorithm given in [13], which takes $O(|E| + |V| \cdot logC)$ time, where C is the largest edge value. An upper bound for the runtime of the algorithm is

$$\Theta(m^3 + m + |\delta| + \binom{m}{2} + |E| + |V| \cdot \log C)$$

Since 0 and 1 are the only edge values used, the constant C is ignored and the runtime simplifies to $\Theta(\max(m^3, |\delta|))$.

Using Algorithm 1, we obtain the following two results. Note that Theorem 1 assumes that the input NFA has finite cycle height. This property can be decided using Theorem 2.

Theorem 1. *If A is an NFA with m states, transition function δ and finite cycle height, then we can compute in time $O(\max(m^3, |\delta|))$ and space $O(\max(m^2, |\delta|))$ the strict cycle height of A.*

If we consider the alphabet to be fixed, as is often done, the time bound of Theorem 1 simplifies to $O(m^3)$.

Second, if we modify the algorithm so that the distance matrix M is never calculated, and the else part on line 6 just returns 1 (instead of lines 10 through 25), then we can decide whether an NFA has finite cycle height just using Tarjan's SCC algorithm, checking that each SCC is a simple cycle or an acyclic singleton.

Theorem 2. *If A is an NFA with m states and a transition function δ, we can decide in time $O(m + |\delta|)$ whether or not A has finite cycle height.*

3.1 Relationship with Polynomial Density Languages

The cycle height of an NFA A can be related to language classes recognized by A. It is known that nearly acyclic NFAs, or NFAs of cycle height one, recognize exactly the constant density languages [9]. We recall the following characterization of polynomial density regular languages from [15].

Proposition 1 ([15]). *A regular language is of polynomial density (of degree at most k) if and only if it can be represented as a finite union of regular expressions of the form $x \cdot y_1^* z_1 \dots y_t^* z_t$, with each $t \leq k+1$ and $x, y_1, z_1, \dots, y_t, z_t \in \Sigma^*$.*

Using the above characterization we can verify that the languages recognized by finite cycle height NFAs have polynomial density. Note that Szilard et al. [15] define a notion of a *t-tiered string*, and this notion is closely related to our notion of cycle height.[2] If all strings are t-tiered with respect to an automaton A then the language of A has a representation as in Proposition 1. Using this observation we can extend Proposition 1 for NFAs of given cycle height.

Proposition 2. *If A is an NFA with cycle height $\mathcal{K}$, then*

$$L(A) = \bigcup_{i=1}^{r} x_i \cdot [y_{i,1}^* z_{i,1} \cdot \ldots \cdot y_{i,t_i}^* z_{i,t_i}],$$

for $x_i, y_{i,j}, z_{i,j} \in \Sigma^*$, and some integers j, t_i and r, such that $t_i \leq \mathcal{K}$ for all i.

As a corollary, we get an explicit upper bound for the density of a language recognized by an NFA of cycle height $\mathcal{K}$.

Corollary 1. *If A is an NFA with cycle height $\mathcal{K}$, then $\varrho_{L(A)}(n) \in O(n^{\mathcal{K}-1})$.*

The reverse is not true: an NFA recognizing a polynomial density language need not have finite cycle height. As a counterexample, we give the NFA in Fig. 1, which does not have finite cycle height, but whose language, $L = a^*$, has constant density.

For a DFA A that has strict cycle height $\mathcal{K}$, we can use results of [15] to strengthen Corollary 1 such that it gives the precise density of $L(A)$.

[2] Strictly speaking, the t-tiered words are defined in [15] only with respect to a DFA.

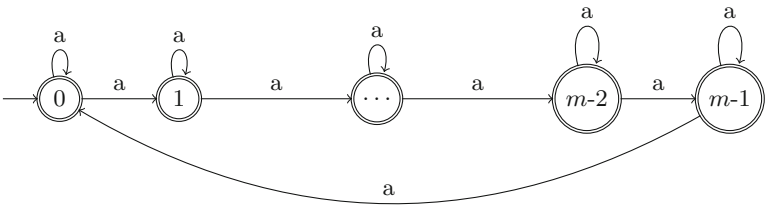

Fig. 1. NFA with language a^*

Lemma 3. *Let A be a DFA with strict cycle height $\mathcal{K}$. Then the density of $L(A)$ is in $\Omega(n^{\mathcal{K}-1})$.*

Proof. Since A has strict cycle height $\mathcal{K}$, there exists a string w such that the accepting computation of A on w visits $\mathcal{K}$ different cycles. (Here we rely on the assumption that A has no useless states, which implies that any computation can be extended to an accepting computation.) This means that w is $\mathcal{K}$-tiered with respect to A (as defined in [15]) and the claim follows from Lemma 1 of [15]. □

From Lemma 3 and Corollary 1 we see that the cycle height of a DFA exactly characterizes the density of the recognized language. Note that cycle height zero DFAs recognize finite languages.

Corollary 2. *If A is a DFA with strict cycle height $\mathcal{K} \geq 1$ then the density of $L(A)$ is $\Theta(n^{\mathcal{K}-1})$.*

From Theorem 6 of [15] we know that if the density of a regular language L is non-polynomial, then $\varrho_L(n) = 2^{\Omega(n)}$. The gap between polynomial and exponential densities occurs because there do not exist any regular languages whose density functions contain non-integer exponents, e.g. $\sqrt{n}$, or $2^{\sqrt{n}}$. Together with Corollary 2 this gives:

Corollary 3. *If the cycle height of a DFA A is not finite then the density of $L(A)$ is $2^{\Omega(n)}$.*

Corollary 2 and Theorem 1 would yield a polynomial time algorithm to compute the exact density of a regular language, however, the time complexity cannot compete with the algorithm of Gawrychowski et al. [3]. For an m-state DFA over a fixed alphabet, an algorithm based on our Theorem 1 to compute the degree of the polynomial giving the density of the language would require $O(m^3)$ time. In comparison, for a DFA A over a fixed alphabet the algorithm given by Theorem 9 of Gawrychowski et al. [3] works in linear time and, even for an NFA A, the algorithm of Gawrychowski et al. [3] does the computation in time $O(m^2)$, where m is the number of states of A.

4 Depth Path Width of Finite Cycle Height NFAs

First we consider bounds for the number of transitions of an NFA with finite cycle height. The definition of NFAs with strict cycle height m implies that an NFA of the form given in Fig. 2 has the minimal number of transitions among all NFAs of strict cycle height m.

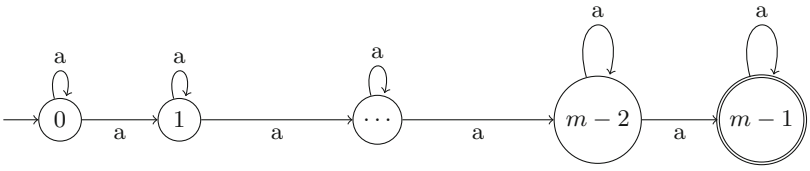

Fig. 2. Unary NFA with strict cycle height m having a minimal number of transitions

For NFAs with strict cycle height $\mathcal{K}$, the following Lemma gives bounds for the number of cycles and transitions as a function of the number of states.

Lemma 4. *If $A = (Q, \Sigma, \delta, q_0, F)$ is an NFA with cycle height $\mathcal{K}$, then $\mathcal{K} \leq |Q|$, and $|\delta| \leq \mathcal{K} + |\Sigma| \cdot \binom{|Q|}{2}$. If A has strict cycle height $\mathcal{K}$, then we have also that $2 \cdot \mathcal{K} - 1 \leq |\delta|$.*

We examine the number of complete computations of a given length of a unary NFA A with strict cycle height $\mathcal{K}$, that is, the depth path width of A.

Lemma 5. *Let $A_{\mathcal{K}} = (Q, \{a\}, \delta, q_0, F)$ be a unary NFA with strict cycle height $\mathcal{K}$. Then* $\mathrm{DPW}(A_{\mathcal{K}}, \ell) \geq \sum_{i=0}^{\mathcal{K}-1} \binom{\ell}{i}$, $\ell \in \mathbb{N}$.

Next we compute an upper bound for the depth path width of $\mathcal{K}$-state unary NFAs having strict cycle height $\mathcal{K}$.

Lemma 6. *Let $A_{\mathcal{K}}$ be a $\mathcal{K}$-state unary NFA with strict cycle height $\mathcal{K}$. Then for $\ell \in \mathbb{N}$,* $\mathrm{DPW}(A_{\mathcal{K}}, \ell) \leq \binom{\ell+\mathcal{K}-1}{\mathcal{K}-1}$.

Proof. Since $A_{\mathcal{K}}$ has strict cycle height $\mathcal{K}$, each of the $\mathcal{K}$ states must be part of a distinct cycle, that is, each state has a self-loop and no other transition can be part of a cycle. By Lemma 2 (ii) to get an upper bound for the depth path width, we can add to $A_{\mathcal{K}}$ a maximal number of acyclic transition. That is, without loss of generality $A_{\mathcal{K}}$ is as in Fig. 3 (with $\mathcal{K} = m$).

It remains to compute the depth path width of $A_{\mathcal{K}}$. As the base case, we observe that $\mathrm{DPW}(A_1, \ell) = \binom{\ell+1-1}{0} = \binom{\ell}{0} = 1$. Inductively, we assume that the claim holds for $A_{\mathcal{K}}$ and now the inductive claim is that

$$\mathrm{DPW}(A_{\mathcal{K}+1}, \ell) = \binom{\ell + \mathcal{K}}{\mathcal{K}}$$

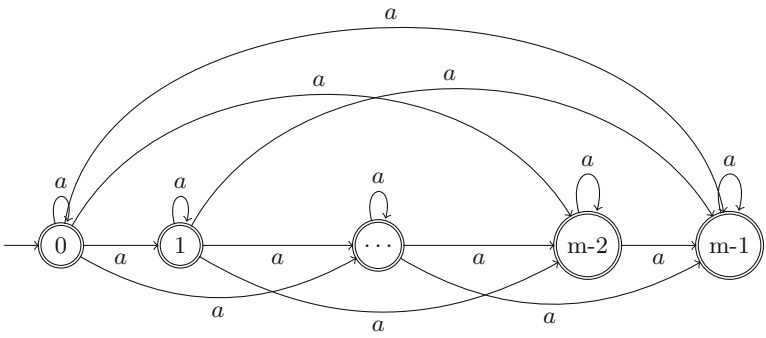

Fig. 3. An m-state unary NFA with strict cycle height m and maximal number of transitions

The value $\mathrm{DPW}(A_{\mathcal{K}+1}, \ell)$ counts the number of computations of length ℓ that are in $A_{\mathcal{K}}$, as well as all of the computations with $\ell - 1$ transitions from $A_{\mathcal{K}}$ and one final transition, $\delta(q, a) = \mathcal{K}$, for some $0 \leq q \leq \mathcal{K} - 1$. More formally:

$$DPW(A_{\mathcal{K}+1}, \ell) = DPW(A_{\mathcal{K}}, \ell) + DPW(A_{\mathcal{K}+1}, \ell - 1) \tag{1}$$

Using our inductive assumption to replace the values in the right hand side of (1), we get:

$$DPW(A_{\mathcal{K}+1}, \ell) = \binom{\ell + \mathcal{K} - 1}{\mathcal{K} - 1} + \binom{\ell + \mathcal{K} - 1}{\mathcal{K}} = \binom{\ell + \mathcal{K}}{\mathcal{K}}$$

where the last equality uses Pascal's triangle recursion rule [17]. □

Lemma 6 gives an upper bound for the depth path width of a $\mathcal{K}$-state unary NFA with strict cycle height $\mathcal{K}$. Next we consider the depth path width of unary NFAs with strict cycle height $\mathcal{K}$ that have more than $\mathcal{K}$ states.

Lemma 7. *Let $A_{\mathcal{K}} = (Q, \{a\}, \delta, q_0, F)$ be an $\mathcal{K}$-state NFA with strict cycle height $\mathcal{K}$ (as in Lemma 6). Let $B_{\mathcal{K}+nc}$ be $A_{\mathcal{K}}$ with one additional state that is not involved in any cycle. Then $DPW(B_{\mathcal{K}+nc}, \ell) \leq DPW(A_{\mathcal{K}+1}, \ell)$, $\ell \in \mathbb{N}$.*

Since NFAs with finite strict cycle height will have maximal depth path width when they have the same number of states and cycles, the result from Lemma 6 is an upper bound for all $\mathcal{K}$-state NFAs with strict cycle height $\mathcal{K}$. Combining the results from these Lemmas, we observe the following corollary.

Corollary 4. *If $A_{\mathcal{K}} = (Q, \Sigma, \delta, q_0, F)$ is a $\mathcal{K}$-state unary NFA with strict cycle height $\mathcal{K}$, then*

$$\sum_{i=0}^{\mathcal{K}-1} \binom{\ell}{i} \leq \mathrm{DPW}(A_{\mathcal{K}}, \ell) \leq \binom{\ell + \mathcal{K} - 1}{\mathcal{K} - 1}$$

The upper bound also holds for $\mathcal{K}$-state unary NFAs with cycle height $\mathcal{K}$. The lower bound also holds for $\mathcal{K}$-state non-unary NFAs with strict cycle height $\mathcal{K}$.

The results of Sect. 4 give upper and lower bounds for the number of complete computations of NFAs whose number of states matches exactly their strict cycle height.

If the number of states exceeds the number of cycles, there will naturally be fewer complete computations. In this case, however, the depth path width of these NFAs cannot be just a function of the number of cycles, and must necessarily involve the number of states.

Problem 1. What is the maximum number of complete computations for an m-state NFA with strict cycle height $\mathcal{K}$, when $m > \mathcal{K}$?

4.1 Experimental Results

To acquire results for the maximum number of complete computations of NFAs with strict finite cycle height and a non-unary alphabet, we first need the following definition.

Definition 1 ([18]). *Pascal's generalized triangle, denoted $\mathcal{P}^N$, is an extension of Pascal's triangle, defined as:*

$$\mathcal{P}^N(0,0) = \mathcal{P}^N(x,0) = \mathcal{P}^N(0,y) = 1$$

$$\mathcal{P}^N(x,y) = \mathcal{P}^N(x-1,y) + \mathcal{P}^N(x,y-1) + (N \cdot \mathcal{P}^N(x-1,y-1)),$$

where $\mathcal{P}^N(x,y)$ is the y^{th} element in the x^{th} row of the N^{th} generalized triangle, for $x, y \geq 0$. It is obvious that $\mathcal{P}^0$ reduces to the normal Pascal's triangle.

Recalling the result from Lemma 6 for unary machines, we can see easily that:

$$DPW(A_{\mathcal{K}}, \ell) = \mathcal{P}^0(\mathcal{K}-1, \ell) = \binom{\ell + \mathcal{K} - 1}{\mathcal{K} - 1}$$

For NFAs with a binary alphabet we look to $\mathcal{P}^1$, which corresponds to the Delannoy numbers [1], and has the following closed form:

$$\mathcal{P}^1(x,y) = \sum_{i=0}^{\min(x,y)} 2^i \cdot \binom{x}{i} \cdot \binom{y}{i}$$

By extrapolating upon the closed form for the Delannoy numbers, we established a candidate equation upper bounding the number of complete computations of NFAs of the form given in Fig. 4. Based on experimental testing of NFAs of this form (for $1 \leq \mathcal{K} \leq 7$, $1 \leq |\Sigma| \leq 8$, and $1 \leq \ell \leq 10$), we believe that the following conjecture captures the number of complete computations as the length of the computation increases.

The selection of characters used on the self-loops (in the case of Fig. 4, c_1) is arbitrary for the purposes of counting computations, and does not have to be the same for every state.

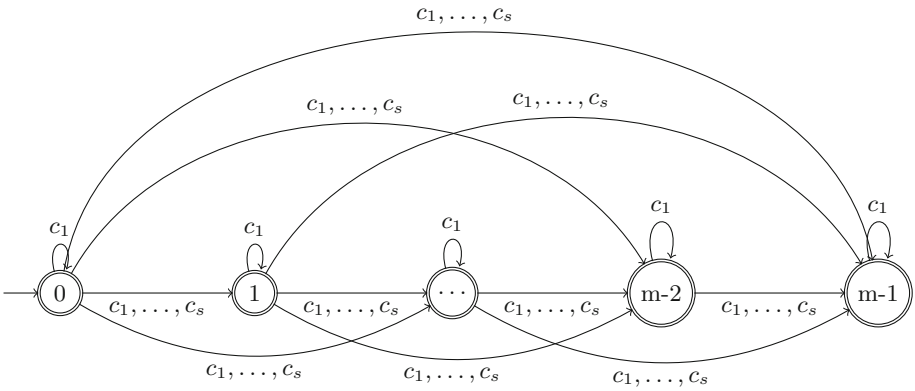

Fig. 4. $\mathcal{K}$-state NFA with strict cycle height $\mathcal{K}$ and $|\Sigma| = s$

Conjecture 1. Let $A_{\mathcal{K}}^{|\Sigma|} = (Q, \Sigma, \delta, q_0, F)$ be a $\mathcal{K}$-state NFA with strict cycle height $\mathcal{K}$, as in Fig. 4. Then

$$DPW(A_{\mathcal{K}}^{|\Sigma|}, \ell) = \mathcal{P}^{|\Sigma|-1}(\mathcal{K}-1, \ell) = \sum_{i=0}^{\min(\mathcal{K}-1,\ell)} |\Sigma|^i \cdot \binom{\mathcal{K}-1}{i} \cdot \binom{\ell}{i} \tag{2}$$

Furthermore, since (2) and the structure of NFAs of the form given in Fig. 4 scale down to the unary case, we believe that this is an upper bound on the number of complete computations of any $\mathcal{K}$-state NFA with strict cycle height $\mathcal{K}$.

Conjecture 2. Let $A_{\mathcal{K}}^{|\Sigma|} = (Q, \Sigma, \delta, q_0, F)$ be a $\mathcal{K}$-state NFA with strict cycle height $\mathcal{K}$. Then

$$DPW(A_{\mathcal{K}}^{|\Sigma|}, \ell) \leq \sum_{i=0}^{\min(\mathcal{K}-1,\ell)} |\Sigma|^i \cdot \binom{\mathcal{K}-1}{i} \cdot \binom{\ell}{i} \tag{3}$$

Acknowledgments. Research supported by NSERC grant OGP0147224.

References

1. Banderier, C., Schwer, S.R.: Why Delannoy numbers? J. Statist. Plann. Inference **135**, 40–54 (2004)
2. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)
3. Gawrychowski, P., Krieger, D., Rampersad, N., Shallit, J.: Finding the growth rate of a regular or context-free language in polynomial time. Int. J. Found. Comput. Sci. **21**(4), 597–618 (2010)
4. Goldstine, J., Kappes, M., Kintala, C.M.R., Leung, H., Malcher, A., Wotschke, D.: Descriptional complexity of machines with limited resources. J. Univ. Comput. Sci. **8**(2), 193–234 (2002)

5. Gruber, H.: Digraph complexity measures and applications in formal language theory. Discrete Math. Theor. Comput. Sci. **14**(2), 189–204 (2012)
6. Gruber, H., Holzer, M.: From finite automata to regular expressions and back - a summary on descriptional complexity. Int. J. Found. Comput. Sci. **26**(8), 1009–1040 (2015)
7. Han, Y.-S., Salomaa, A., Salomaa, K.: Ambiguity, nondeterminism and state complexity of finite automata. Acta Cybernetica **23**, 141–157 (2017)
8. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata - a survey. Inform. Comput. **209**(3), 456–470 (2011)
9. Keeler, C., Salomaa, K.: Branching measures and nearly acyclic NFAs. In: Pighizzini, G., Câmpeanu, C. (eds.) DCFS 2017. LNCS, vol. 10316, pp. 202–213. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60252-3_16
10. Kutrib, M., Pighizzini, G.: Recent trends in descriptional complexity of formal languages. Bull. EATCS (111) (2013)
11. Kutrib, M., Meckel, K., Wendlandt, M.: Parameterized prefix distance between regular languages. In: Geffert, V., Preneel, B., Rovan, B., Štuller, J., Tjoa, A.M. (eds.) SOFSEM 2014. LNCS, vol. 8327, pp. 419–430. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04298-5_37
12. Msiska, M., van Zijl, L.: Interpreting the subset construction using finite sublanguages. In: Proceedings of Prague Stringology Conference 2016, pp. 48–62 (2016)
13. Ahuja, R.K., Mehlhorn, K., Orlin, J.B., Tarjan, R.E.: Faster algorithms for the shortest path problem. J. ACM **37**(2), 213–223 (1990)
14. Shallit, J.: Second Course in Formal Languages and Automata Theory. Cambridge University Press, New York (2009)
15. Szilard, A., Yu, S., Zhang, K., Shallit, J.: Characterizing regular languages with polynomial densities. In: Havel, I.M., Koubek, V. (eds.) MFCS 1992. LNCS, vol. 629, pp. 494–503. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55808-X_48
16. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
17. Tattersall, J.J.: Elementary Number Theory in Nine Chapters, Cambridge University Press (2005)
18. Wong, C.K., Maddocks, T.W.: A generalized Pascal's triangle. Fibonacci Quart. **13**(2), 134–136 (1975)