Describing Periodicity in Two-Way Deterministic Finite Automata Using Transformation Semigroups

Michal Kunc[1],[*] and Alexander Okhotin[2,3],[**]

[1] Masaryk University, Czech Republic
kunc@math.muni.cz
[2] Department of Mathematics, University of Turku, Finland
[3] Academy of Finland
alexander.okhotin@utu.fi

Abstract. A framework for the study of periodic behaviour of two-way deterministic finite automata (2DFA) is developed. Computations of 2DFAs are represented by a two-way analogue of transformation semigroups, every element of which describes the behaviour of a 2DFA on a certain string x. A subsemigroup generated by this element represents the behaviour on strings in x^+. The main contribution of this paper is a description of all such monogenic subsemigroups up to isomorphism. This characterization is then used to show that transforming an n-state 2DFA over a one-letter alphabet to an equivalent sweeping 2DFA requires exactly $n + 1$ states, and transforming it to a one-way automaton requires exactly $\max_{0 \leqslant \ell \leqslant n} G(n - \ell) + \ell + 1$ states, where $G(k)$ is the maximum order of a permutation of k elements.

1   Introduction

Two-way deterministic finite automata (2DFA) were introduced in the famous paper by Rabin and Scott [15] alongside the one-way nondeterministic automata (1NFA). Both kinds of automata recognize the same language family as the one-way deterministic finite automata (1DFA). However, they are substantially different in terms of succinctness of description, and the number of states needed to represent a language by one type of finite automata is sometimes much greater than for another type.

While the methods for determining the number of states in one-way automata, both deterministic and nondeterministic, are well-known, and the main descriptional complexity questions [6] have been researched to extinction, the succinctness issues of two-way automata have proved to be truly challenging. The question of whether 2DFAs can simulate their nondeterministic counterparts (2NFA) with only a polynomial blowup has attracted a lot of attention due to its relation

to the *L vs. NL* problem in the complexity theory [2,16], yet no definite answers could be found. Even such a basic question as the precise number of states in a 1DFA needed to simulate an n-state 2DFA had not been settled for almost half a century: the $(n+1)^{n+1}$ upper bound by Shepherdson [17] was approached by a relatively close $(\frac{n-5}{2})^{\frac{n-5}{2}}$ lower bound by Moore [13], but only a few years ago the exact value $n(n^n - (n-1)^n)$ was finally determined by Kapoutsis [8]. Simulations of 2NFAs by simpler automata, first studied by Vardi [19], were also determined precisely by Kapoutsis [8]. The complexity of operations on 2DFAs has recently been investigated by Jirásková and Okhotin [7].

In spite of some individual results on the succinctness of 2DFAs, what exactly can a 2DFA with a given number of states do, remains a mystery. This paper undertakes to study the behaviour of 2DFAs in one particular case: on a con-catenation of multiple instances of the same string, on which an automaton may first keep track of the number of instances, but eventually loses count and follows a periodic trajectory, repeatedly passing through the same sequence of states. The paper develops a general framework for reasoning about such computations, which describes both the periodic and the non-periodic behaviour in a unified way.

Consider the well-known algebraic representation of a 1DFA by a monoid of *partial transformations* of its set of states [14]. A generalization of this concept to semigroups of *two-way transformations* representing bi-directional motion was given in the work of Birget [3], who used it to represent 2DFAs of a special form. Recalling this notion, this paper applies it to arbitrary 2DFAs in Section 2. Each element of a two-way transformation semigroup defines the behaviour of some 2DFA on some nonempty string x, that is, for the 2DFA entering x from a given side in a given state, the two-way transformation specifies the result of its computation: whether the 2DFA ever leaves the string, and if it does, then from which side and in which state it emerges. The behaviour of the 2DFA on all strings in x^+ is then represented by the subsemigroup generated by the two-way transformation corresponding to x.

The properties of such *monogenic subsemigroups* of the semigroup of two-way transformations are gradually worked out in Section 3. For each two-way transformation on n states, an ordinary transformation on $n+1$ states is con-structed, so that the subsemigroups generated by these transformations are isomorphic. The final result is the precise characterization of monogenic semi-groups of two-way transformations on n states: their index ℓ (that is, the start-ing point of the periodicity) and period $\mathrm{lcm}(p_1, \ldots, p_k)$ satisfy the inequality $p_1 + \cdots + p_k + \ell \leqslant n + 1$.

The most direct application of this framework is to the state complexity of 2DFAs over a one-letter alphabet. The first study of unary 2DFAs was under-taken by Chrobak [4], who has sketched an argument that an n-state 2DFA over a unary alphabet can be simulated by a $\Theta(G(n))$-state 1DFA, where $G(n) = e^{(1+o(1))\sqrt{n \ln n}}$ is the maximal order of a permutation on n elements, known as *Landau's function* [9]. Further work in this direction was done by Mereghetti and Pighizzini [10] and by Geffert, Mereghetti and Pighizzini [5], who similarly esti-

mated the 2NFA–1DFA tradeoff. Their approach lies with considering only the periodic part of the language and using a general upper bound on the starting point of its periodicity, and thus leads only to asymptotic succinctness tradeoffs. No exact values of succinctness tradeoffs between unary two-way and one-way automata are known up to date. The first such results are obtained in this paper.

Based on the analysis of monogenic two-way transformation semigroups, the use of the states by a unary 2DFA is explained as follows. It is in fact found that 2DFAs can do just two things using fewer states than 1DFAs:

1. count divisibility separately for several numbers, where an equivalent 1DFA would need to count modulo the least common multiple of those numbers;
2. when counting up to a finite bound ℓ, they can count one step less than one would expect, and then use one of the cycles to distinguish the string of length ℓ from longer strings.

In Section 4, this understanding is used to show that every 2DFA over a unary alphabet can be transformed to an equivalent sweeping 2DFA with at most one extra state, thus more or less confirming the unproved claim by Chrobak [4], who stated that this can be done without increasing the number of states. The next Section 5 considers the standard question of converting an n-state unary 2DFA to an equivalent 1DFA. Chrobak's [4] asymptotic estimation $\Theta(G(n))$ is hereby improved to the precise expression, which is $\max_{0 \leqslant \ell \leqslant n} G(n - \ell) + \ell + 1$. The same function applies to the 2DFA to 1NFA transformation.

2   Two-Way Transformation Semigroups

A (partial) 1DFA is a quintuple $\mathcal{A} = (\Sigma, Q, q_1, \delta, F)$, where Σ is a finite alphabet, Q is a finite set of states, of which $q_1 \in Q$ is the initial state and $F \subseteq Q$ are the accepting states, and $\delta \colon Q \times \Sigma \to Q$ is a (partially defined) transition function. For every string $w \in \Sigma^*$, if $\mathcal{A}$ reads the string w beginning in a state $q \in Q$, it can either finish reading it in a state q', or reach an undefined transition. This is represented by a (partial) function $\delta_w^{\mathcal{A}} \colon Q \to Q$, called the *behaviour of $\mathcal{A}$ on* $w \in \Sigma^*$. The language recognized by a 1DFA is $L(\mathcal{A}) = \{\, w \mid \delta_w^{\mathcal{A}}(q_1) \in F \,\}$.

Behaviours of 1DFAs on different strings can be analyzed within the monoid $\mathcal{PT}_Q$ of partial functions from Q to Q (also known as *partial transformations of Q*) with the function composition $\circ$ as the product. The computations of the 1DFA $\mathcal{A}$ are characterized by its *transition monoid*, which is the submonoid of $\mathcal{PT}_Q$ consisting of the automaton's behaviours $\delta_w^{\mathcal{A}}$ on all strings $w \in \Sigma^*$. This submonoid is generated by the behaviours on letters, and the function composition takes the form $\delta_v^{\mathcal{A}} \circ \delta_u^{\mathcal{A}} = \delta_{uv}^{\mathcal{A}}$. A detailed introduction was given by Perrin [14]. In this section, this construction of a monoid associated with an automaton is generalized to the case of bi-directional motion.

Consider 2DFAs operating on strings $\vdash w \dashv$ bounded by end-markers, which begin their computation at $\vdash$, and accept by going beyond either of the markers. Thus, a 2DFA is defined as a quadruple $\mathcal{A} = (\Sigma, Q, q_1, \delta)$, in which Σ is a finite alphabet with $\vdash, \dashv \notin \Sigma$, Q is a finite set of states, $q_1 \in Q$ is the initial state and

$\delta \colon Q \times (\Sigma \cup \{\vdash, \dashv\}) \to Q \times \{-1, +1\}$ is a partially defined transition function: for $\mathcal{A}$ in a state q observing a symbol a, the value $\delta(q, a)$ indicates the next state and the direction of motion. Consider the behaviour of a 2DFA on any nonempty string w. It enters the string in a certain state either from its first or from its last symbol. Then the automaton may either loop inside the string, or reach an undefined transition, or eventually leave the string by going to the left of its first symbol or to the right of its last symbol in a certain new state.

This behaviour is represented similarly to the behaviour of a 1DFA, using additional notation for the right and the left sides of the string. Entering the leftmost symbol of w in a state q is described using the notation $\overrightarrow{q}$. Then, leaving w by going to the right of its last symbol in a state r is denoted by $\overrightarrow{r}$, because this means entering the string to the right of w on its first symbol. Symmetrically, the notation $\overleftarrow{q}$ represents entering a string from the right in a state q, and if such a computation results in leaving w by going to the left beyond its first symbol in a state r, this is denoted by $\overleftarrow{r}$.

Thus the behaviour of a 2DFA with the set of states Q on any string can be represented as a partial transformation of a set $N = \overrightarrow{Q} \cup \overleftarrow{Q}$, where $\overrightarrow{Q} = \{\, \overrightarrow{q} \mid q \in Q \,\}$ and $\overleftarrow{Q} = \{\, \overleftarrow{q} \mid q \in Q \,\}$. Transformations of this kind were introduced by Birget [3], who used them to describe the behaviour of a restricted case of 2DFAs. These transformations will be called *two-way transformations* on Q, denoted by symbols f and g, and depicted by oriented graphs, such as in Figure 1. Let $f_w^{\mathcal{A}} \colon N \to N$ denote the behaviour of a 2DFA $\mathcal{A}$ on a string $w \in (\Sigma \cup \{\vdash, \dashv\})^*$. Then, the language recognized by $\mathcal{A}$ is $L(\mathcal{A}) = \{\, w \in \Sigma^* \mid f_{\vdash w \dashv}^{\mathcal{A}}(\overrightarrow{q_1}) \text{ is defined} \,\}$.

A natural question is whether all two-way transformations may occur as behaviours of some automata on some strings, and the answer is negative. Let f be a behaviour of some automaton on a string $w \in \Sigma^+$, let $\overrightarrow{q} \in \overrightarrow{Q}$ and assume $f(\overrightarrow{q}) = \overrightarrow{r} \in \overrightarrow{Q}$. Then the computation of the automaton going through w to the state r beyond w should pass through the last symbol of w in some state p, from where the automaton went to the right. Accordingly, there should exist a state p with $f(\overleftarrow{p}) = \overrightarrow{r}$.

A symmetric condition applies to elements of $\overleftarrow{Q}$, and the entire necessary condition can be succinctly stated as follows. For all $\alpha, \beta \in N$, if both α and β belong to $\overrightarrow{Q}$ or both belong to $\overleftarrow{Q}$, this is denoted by $\alpha \sim \beta$ and represents entering strings in the same direction. Then

$$\forall \alpha \in N \colon f(\alpha) \sim \alpha \implies \exists \beta \in N \colon \beta \nsim \alpha \ \wedge \ f(\beta) = f(\alpha). \qquad (1)$$

Denote by $\mathcal{TT}_Q$ the set of two-way transformations on Q satisfying this condition.

Two-way transformations that occur as behaviours of some automata on one-symbol strings $w = a \in \Sigma$ must satisfy a stronger condition:

$$f_a^{\mathcal{A}}(\overrightarrow{q}) = f_a^{\mathcal{A}}(\overleftarrow{q}) \quad \text{(for all } q \in Q) \qquad (2)$$

And conversely, if a two-way transformation satisfies condition (2), then it is a behaviour of some letter in some 2DFA. Such two-way transformations shall be called *distinguished*.
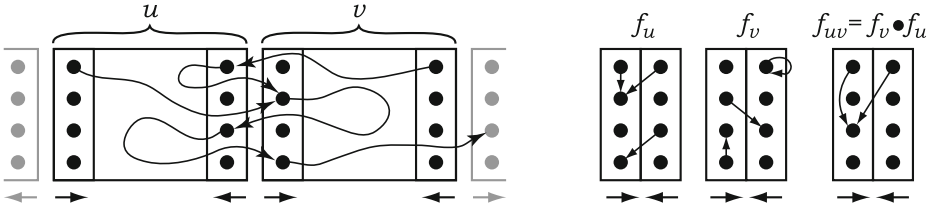
Fig. 1. Composition of behaviours on u and on v as a product $f_v \bullet f_u$

Once the behaviour of a 2DFA on some strings $u, v \in \Sigma^+$ is known to be f and g, respectively, its behaviour on their concatenation uv can be inferred from f and g, as depicted in Figure 1. It is calculated as a certain *product* $g \bullet f$ of two-way transformations $f, g \colon N \to N$, defined by Birget [3]. This product faithfully represents the behaviour of 2DFAs on the concatenation of two strings, that is, $f_{uv}^{\mathcal{A}} = f_v^{\mathcal{A}} \bullet f_u^{\mathcal{A}}$ for any strings $u, v \in (\Sigma \cup \{\vdash, \dashv\})^+$ and for any 2DFA $\mathcal{A}$.

It can be shown that $\mathcal{TT}_Q$ equipped with the product $\bullet$ is a semigroup. This semigroup is generated by distinguished two-way transformations, and therefore, elements of $\mathcal{TT}_Q$ are precisely behaviours of some 2DFA on some string. More precisely, every element of $\mathcal{TT}_Q$ can be obtained as a product of two distinguished transformations. Note that $\mathcal{TT}_Q$ is not a monoid for the lack of an identity element. Though there exists an identity two-way transformation defined by $e(\alpha) = \alpha$, it does not satisfy condition (1). For this reason, the semigroup representation of two-way automata considers only their computations on nonempty inputs. The empty string can be reintroduced later, when turning from semigroups back to automata.

There is the following formal connection between computations of 2DFAs and semigroups $\mathcal{TT}_Q$:

Proposition 1. *Let* $\mathcal{A} = (\Sigma, Q, q_1, \delta)$ *be a 2DFA, and consider the subsemigroup of* $\mathcal{TT}_Q$ *generated by distinguished two-way transformations* $f_a^{\mathcal{A}}$, *for* $a \in \Sigma \cup \{\vdash, \dashv\}$. *Then*

$$L(\mathcal{A}) = \left\{ a_1 \dots a_\ell \mid (f_{\dashv}^{\mathcal{A}} \bullet f_{a_\ell}^{\mathcal{A}} \bullet \dots \bullet f_{a_1}^{\mathcal{A}} \bullet f_{\vdash}^{\mathcal{A}})(\overrightarrow{q_1}) \text{ is defined} \right\}.$$

In particular, for a given fixed automaton $\mathcal{A}$, *the membership of a string* $a_1 \dots a_\ell \in \Sigma^+$ *in* $L(\mathcal{A})$ *depends only on the element* $f_{a_1 \dots a_\ell}^{\mathcal{A}} = f_{a_\ell}^{\mathcal{A}} \bullet \dots \bullet f_{a_1}^{\mathcal{A}}$ *of the subsemigroup generated by* $\{ f_a^{\mathcal{A}} \mid a \in \Sigma \}$.

3   Monogenic Subsemigroups of $\mathcal{TT}_Q$

Let $f \in \mathcal{TT}_Q$ be any fixed element of the two-way transformation semigroup. This element represents the behaviour of some 2DFA $\mathcal{A}$ on some string w. By Proposition 1, for arbitrary $u, v \in \Sigma^*$, the string $uw^\ell v$ belongs to $L(\mathcal{A})$ if and only if the two-way transformation $f_{v\dashv}^{\mathcal{A}} \bullet (f)^{\bullet \ell} \bullet f_{\vdash u}^{\mathcal{A}}$ is defined on $\overrightarrow{q_1}$, where $f^{\bullet \ell}$
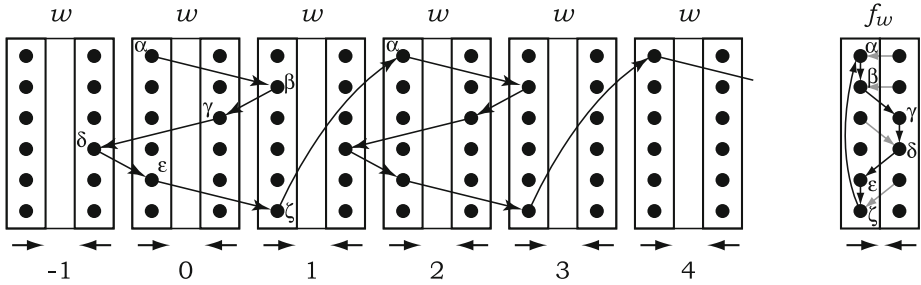
Fig. 2. Computation on a bi-infinite string of ws

stands for $f \bullet \cdots \bullet f$, ℓ times. Beginning with a certain $j \geqslant 1$, two-way transformations in the sequence $\{f^{\bullet \ell}\}_{\ell \geqslant 1}$ repeat periodically. Then, by the above observation, the membership of strings $uw^\ell v$ in $L(\mathcal{A})$ becomes periodic no later than starting from j, and the period divides the period of the powers of f. Therefore, any upper bound on the periodicity of powers $f^{\bullet \ell}$, which is described by the structure of the monogenic subsemigroup generated by f in $\mathcal{TT}_Q$, constitutes a bound on the periodicity of the membership of strings $uw^\ell v$.

In order to describe the structure of a monogenic subsemigroup S generated by some element s in a finite semigroup, one has to determine two numbers. The least positive integer i, for which there exists $j > i$ with $s^i = s^j$, is called the *index* of S, and the least number $p \geqslant 1$ with $s^i = s^{i+p}$ is called the *period* of S. The index and period determine a monogenic semigroup up to isomorphism.

3.1 The Distance Travelled After i Steps

For a string $w \in \Sigma^*$, assume a bi-infinite string of copies of w, numbered by integers. Let $f \in \mathcal{TT}_Q$ be the behaviour of some 2DFA on w. Consider a computation of this 2DFA, that begins by entering copy number 0 from the left in a certain state, which is represented by $\alpha \in \overrightarrow{Q}$. At every j-th step, the automaton proceeds to the neighbouring instance of w: to the right if $f^j(\alpha) \in \overrightarrow{Q}$, and to the left if $f^j(\alpha) \in \overleftarrow{Q}$. Such a computation is illustrated in Figure 2, where $f(\alpha) = \beta$, $f^2(\alpha) = \gamma, \ldots, f^6(\alpha) = \alpha$, etc. Similarly, one can consider computations starting from $\alpha' \in \overleftarrow{Q}$, numbering the instances of w in the reverse direction.

Consider the *distance* travelled in such a computation. In Figure 2, three steps of computation move the head back by one square, while six steps of computation result in moving forward by two squares. This shall be denoted by $\mathrm{d}(\alpha, 3) = -1$ and $\mathrm{d}(\alpha, 6) = 2$, respectively.

Definition 1. *For every $\alpha \in N$ and $i \geqslant 0$, such that $f^i(\alpha)$ is defined, let*

$$\mathrm{d}(\alpha, i) = \big|\{\, j \mid 1 \leqslant j \leqslant i, f^j(\alpha) \sim \alpha \,\}\big| - \big|\{\, j \mid 1 \leqslant j \leqslant i, f^j(\alpha) \nsim \alpha \,\}\big|.$$

In other words, $\mathrm{d}(\alpha, i)$ expresses how far one moves from the original position in the bi-infinite string of fs by means of i steps of the computation represented

by the two-way transformation f, where positive numbers mean continuing in the direction of α, while negative numbers mean that the direction was reversed. Observe that $d(\alpha, i)$ and $d(\alpha, i+1)$ always differ exactly by one.

In the following, it will be investigated how the graph structure of any $f \in \mathcal{TT}_Q$ determines its behaviour as a two-way transformation.

For any $m \geqslant 1$ and $\alpha \in N$, the value $f^{\bullet m}(\alpha)$ represents the computation on a block of m instances of f. This computation begins on the first or on the last instance of f in this block, depending on whether $\alpha \in \overrightarrow{Q}$ or $\alpha \in \overleftarrow{Q}$. Consider the case of $\alpha \in \overrightarrow{Q}$. Then the computation begins on the first instance of f, and at every j-th step the computation proceeds to the neighbouring instance as described above. Unless $f^{\bullet m}(\alpha)$ is undefined, the computation eventually leaves the block to the right or to the left.

The condition of leaving the block can be defined in terms of d as $d(\alpha, i) \in \{-1, m\}$ for some i. This is formally established in the following lemma, which handles the cases of $\alpha \in \overrightarrow{Q}$ and $\alpha \in \overleftarrow{Q}$ uniformly.

Lemma 1. *For every $m \in \mathbb{N}$ and $\alpha \in N$, $f^{\bullet m}(\alpha) = f^i(\alpha)$, where $i \in \mathbb{N}$ is the smallest number with $d(\alpha, i) \notin \{0, \ldots, m-1\}$, or, equivalently, with $d(\alpha, i) \in \{-1, m\}$. If such an i does not exist, then $f^{\bullet m}(\alpha)$ is undefined.*

3.2 $\tilde{f}$: Moving by f Until Advancing by One Position

Consider the example in Figure 2. What the f-cycle of length 6 does is, starting from α, first going one step forward, then two steps back and finally three steps forward. Provided that there is an extra instance of w in position -1, this results in going 2 steps forward. However, if the instance of w in position 0 is the leftmost one, then this computation would not take place.

In order to deal with computations on long blocks of ws, it is useful to assume that there is an unbounded supply of ws on both sides, and consider the computation starting from α that results in advancing by one position (that is, to the right if $\alpha \in \overrightarrow{Q}$ and to the left if $\alpha \in \overleftarrow{Q}$).

Definition 2. *For every $f \in \mathcal{TT}_Q$, define a partial transformation $\tilde{f} \in \mathcal{PT}_N$ by the rule $\tilde{f}(\alpha) = f^i(\alpha)$, where $i \in \mathbb{N}$ is the smallest number with $d(\alpha, i) = 1$. If such an i does not exist, $\tilde{f}(\alpha)$ is undefined.*

Returning to Figure 2, $\tilde{f}(\alpha) = f(\alpha) = \beta$, since $d(\alpha, 1) = 1$. The value of $\tilde{f}(\beta)$ is given by $f^5(\beta) = \alpha$, because $\{d(\beta, n)\}_{n \geqslant 1} = \{-1, -2, -1, 0, \mathbf{1}, \ldots\}$. For the elements $\gamma, \delta \in \overleftarrow{Q}$, $\tilde{f}$ represents going by one step to the left, and accordingly, $\tilde{f}(\gamma) = \delta$ and $\tilde{f}(\delta)$ is undefined.

Since for every $\alpha \in N$, the last step of the computation defining $\tilde{f}(\alpha)$ is a move in the same direction as α, one has $\tilde{f}(\alpha) \sim \alpha$. Therefore, $\tilde{f}$ is formed of two separate partial transformations on each of the sets $\overrightarrow{Q}$ and $\overleftarrow{Q}$.

While the partial transformation $\tilde{f}$ corresponds to advancing by one position, the m-th power of $\tilde{f}$ represents advancing by m positions, that is, $\tilde{f}^m(\alpha) = f^i(\alpha)$, where $i \in \mathbb{N}$ is the smallest number with $d(\alpha, i) = m$.

The first step in describing the subsemigroup generated by f in $\mathcal{TT}_Q$ is to prove that it is isomorphic to the subsemigroup generated by $\tilde{f}$ in $\mathcal{PT}_N$. This amounts to showing the following statement:

Lemma 2. *For all positive integers m and k, $f^{\bullet m} = f^{\bullet k}$ if and only if $\tilde{f}^m = \tilde{f}^k$.*

The forward implication follows immediately from the equality $\tilde{f}^m = \widetilde{f^{\bullet m}}$, which means that making m steps forward over f is the same as making one step forward over a block of m instances of f. The converse is more difficult to verify, since the inequality $f^{\bullet m}(\alpha) \neq f^{\bullet k}(\alpha)$ does not necessarily imply that $\tilde{f}^m(\alpha) \neq \tilde{f}^k(\alpha)$. However, a detailed analysis shows that one can always find some node $\beta \in N$, reachable from α by several applications of f, which satisfies $\tilde{f}^m(\beta) \neq \tilde{f}^k(\beta)$.

3.3   Index and Period of the Subsemigroup Generated by $\tilde{f}$

It is well known how to calculate the index and the period of a subsemigroup of $\mathcal{PT}_N$ generated by a partial transformation h, using the structure of h as a directed graph of out-degree 1, with the set of nodes N and with $h(\alpha) = \beta$ represented by an arc $\alpha \to \beta$. Every node from N belongs either to an h-cycle, or to an h-tail, which is any maximal subset of N consisting of elements not belonging to any h-cycle, and in which for any two elements α and β, either β is reachable from α or vice versa. Note that every h-tail leads either into a cycle, or into a *dead element* where h is not defined. The number of elements of N that belong to a given tail is called the *length* of the tail. Then the index of the subsemigroup generated by h is equal to the length of the longest h-tail (it equals 1 if there is no tail) and its period is equal to the least common multiple of the lengths of all h-cycles.

Therefore, both the index and the period of the subsemigroup generated by f are determined by the lengths of $\tilde{f}$-cycles and the longest $\tilde{f}$-tail. Let C be the set of all nodes from N belonging to $\tilde{f}$-cycles, and fix one of the longest $\tilde{f}$-tails. Let T denote this $\tilde{f}$-tail and denote $D = C \cup T \subseteq N$. Denote the restriction of $\tilde{f}$ to D by $\hat{f} \in \mathcal{PT}_D$.

Then the subsemigroups generated by $\tilde{f}$ and $\hat{f}$ have the same index and period, and hence are isomorphic. Therefore, some information about the index and period of the subsemigroup generated by f can be obtained by finding an upper bound on the size of D with respect to $|Q|$. However, it can be shown that condition (1), which is imposed on f by the fact that it arises as a behaviour of a 2DFA, significantly restricts the possible lengths of cycles and tails of $\tilde{f}$. More precisely, the sum of the lengths of its cycles and the length of the longest tail never exceeds $|Q| + 1$, and it can reach $|Q| + 1$ only in a very special case, which corresponds to the ability of a 2DFA to use one of its cycles to save one state when counting to a certain bound (an example of such an automaton is given in the following section).

The verification of the inequality $|D| \leqslant |Q| + 1$ is rather simple if f is a distinguished transformation, because it can be easily shown that there exists at

most one such state q, that both $\overrightarrow{q}$ and $\overleftarrow{q}$ belong to D. Indeed, assume there is such a $q \in Q$. Since $f(\overrightarrow{q}) = f(\overleftarrow{q})$, at least one of the nodes $\overrightarrow{q}$ and $\overleftarrow{q}$ (say the former one) belongs to an f-tail. As they cannot belong to the same f-tail, the other node $\overleftarrow{q}$ belongs to an f-cycle. Therefore, $\overrightarrow{q}$ is the last node of the unique f-tail, which contains some element of D. This implies that such q is uniquely determined. Moreover, one can also see that if such a q exists, then $\hat{f}$ contains a cycle (the node $\overrightarrow{q}$ cannot belong to the only $\hat{f}$-tail due to $\overrightarrow{q} \nsim \overleftarrow{q}$) and there exists a node in D (namely, the last node of the $\hat{f}$-tail of $\overrightarrow{q}$), on which $\hat{f}$ is not defined.

This argument can be generalised to arbitrary two-way transformations by proving that if a node γ is in the image of $\hat{f}$, then $\gamma = f(\alpha)$ for some node α satisfying $\alpha \nsim \gamma$ and not belonging to D. There is only one exception to this rule, namely when γ is the node where the unique f-tail containing elements of D joins an f-cycle. This leads to the following upper bound on the size of D.

Lemma 3. *It holds that $|N| \geqslant 2\,|D| - 2$. Additionally, if $|N| = 2\,|D| - 2$, then $\hat{f}$ is undefined on some element of D and contains a cycle.*

3.4   The Main Theorem and Its Implications

Because the subsemigroup generated by f in $\mathcal{TT}_Q$ is isomorphic to the subsemigroup generated by $\hat{f}$ in $\mathcal{PT}_D$, Lemma 3 can be used to describe the structure of all monogenic subsemigroups of $\mathcal{TT}_Q$:

Theorem 1. *For every monogenic subsemigroup S of $\mathcal{TT}_Q$ there exist $k \geqslant 1$ and numbers $p_1, \ldots, p_k \geqslant 1$ and $\ell \geqslant 1$, with $p_1 + \cdots + p_k + \ell \leqslant |Q| + 1$, such that S has index ℓ and period $\mathrm{lcm}(p_1, \ldots, p_k)$. More precisely, if S is generated by $f \in \mathcal{TT}_Q$, then ℓ can be obtained as the length of the longest $\tilde{f}$-tail and numbers $p_1, \ldots, p_k$ as the lengths of all $\tilde{f}$-cycles (if there is no cycle in $\tilde{f}$, let $k = p_1 = 1$).*

Conversely, for any integers $k \geqslant 1$, $p_1, \ldots, p_k \geqslant 1$ and $\ell \geqslant 1$ satisfying $p_1 + \cdots + p_k + \ell \leqslant |Q| + 1$, the semigroup $\mathcal{TT}_Q$ contains a distinguished transformation, which generates a subsemigroup with index ℓ and period $\mathrm{lcm}(p_1, \ldots, p_k)$.

This result provides a lower bound on the size of any such Q, that $\mathcal{TT}_Q$ contains a monogenic subsemigroup with a certain index and period:

Corollary 1. *Let S be a monogenic subsemigroup of $\mathcal{TT}_Q$, which has index ℓ and period p. Let $p = p_1 \cdots p_k$, where $p_1, \ldots, p_k$ are powers of distinct primes, be the prime factorization of p. Then, $|Q|$ must be at least $p_1 + \cdots + p_k + \ell - 1$.*

For a 2DFA $\mathcal{A}$, Proposition 1 guarantees that any strings $u, v \in \Sigma^+$ satisfying $f_u^{\mathcal{A}} = f_v^{\mathcal{A}}$ represent the same element of the syntactic semigroup of $L(\mathcal{A})$. This means that the bounds on the size of monogenic subsemigroups of $\mathcal{TT}_Q$ given in Theorem 1 and Corollary 1 are valid also for monogenic subsemigroups of the syntactic semigroup of any language accepted by an n-state 2DFA.

Consider computations of a 2DFA $\mathcal{A}$ on inputs $ux^i v$, in which the behaviour on the infix x^i is described by the two-way transformation $(f_x^{\mathcal{A}})^{\bullet i}$. Due to Proposition 1, the membership of a string $ux^i v$ in the language $L(\mathcal{A})$ depends only

on the element $f^{\mathcal{A}}_{ux^i v} = f^{\mathcal{A}}_v \bullet (f^{\mathcal{A}}_x)^{\bullet i} \bullet f^{\mathcal{A}}_u$ of $\mathcal{TT}_Q$. Therefore, the periodic behaviour of the set $\{\, i \geqslant 1 \mid ux^i v \in L(\mathcal{A}) \,\}$ depends only on the structure of the subsemigroup generated in $\mathcal{TT}_Q$ by $f^{\mathcal{A}}_x$. More precisely, the period of this set divides the period of the subsemigroup, and its index is bounded by the index of the subsemigroup. This leads to the following consequences of Theorem 1 and Corollary 1.

Corollary 2. *Let $\mathcal{A}$ be an n-state 2DFA over an arbitrary finite alphabet Σ, and let $u, v \in \Sigma^*$ and $x \in \Sigma^+$ be any strings. Then, there exist $k \geqslant 1$ and numbers $p_1, \ldots, p_k \geqslant 1$ and $\ell \geqslant 1$, with $p_1 + \cdots + p_k + \ell \leqslant n+1$, such that the set of numbers $\{\, i \geqslant 1 \mid ux^i v \in L(\mathcal{A}) \,\}$ is periodic from ℓ with period $p = \mathrm{lcm}(p_1, \ldots, p_k)$.*

Corollary 3. *Let L be a regular language over an arbitrary finite alphabet Σ, and let $u, v \in \Sigma^*$ and $x \in \Sigma^+$ be any strings. Let the set of numbers $\{\, i \geqslant 1 \mid ux^i v \in L(\mathcal{A}) \,\}$ have period p beginning from ℓ. Let $p = p_1 \cdots p_k$, where $p_1, \ldots, p_k$ are powers of distinct primes, be the prime factorization of p. Then, every 2DFA recognizing L must have at least $p_1 + \cdots + p_k + \ell - 1$ states.*

4   Transformation to Sweeping Automata

A 2DFA is called *sweeping* [18] if in every computation its head changes the direction of motion only on the markers. For an arbitrary alphabet, as independently proved by Berman [1] and by Micali [12], the succinctness blowup from general 2DFAs to sweeping 2DFAs is exponential. For a unary alphabet, Mereghetti and Pighizzini [11] established a transformation of an n-state 2NFA to a sweeping 2NFA with $O(n^2)$ states. Regarding the deterministic case, Chrobak [4] mentioned in passing that "it is easy to show that any unary 2DFA can be substituted by an equivalent sweeping 2DFA without increasing the number of its states". This claim was not substantiated, and the best result known in the literature is the $O(n^2)$ bound for unary 2NFAs due to Mereghetti and Pighizzini [11]. The framework developed in this paper allows finally settling this question:

Theorem 2. *Let $n \geqslant 1$. Then for every unary deterministic two-way automaton $\mathcal{A}$ with n states, there exists an equivalent sweeping deterministic two-way automaton with $n+1$ states. For $n \geqslant 2$, this bound is the best possible.*

In short, the intuition of Chrobak was generally right, though one extra state is needed.

The upper bound is proved by constructing a new sweeping automaton, that simulates the transformation $\tilde{f}$, where f is the behaviour of the original 2DFA on the letter. The construction is straightforward in itself: it produces $p_1 + \cdots + p_k + \ell$ states, where $p_1, \ldots, p_k$ are the lengths of the cycles in $\tilde{f}$, and ℓ is the length of its longest tail. The nontrivial part of the argument is the upper bound $n+1$ on this sum, given in Theorem 1.

The lower bound is witnessed by the following language.
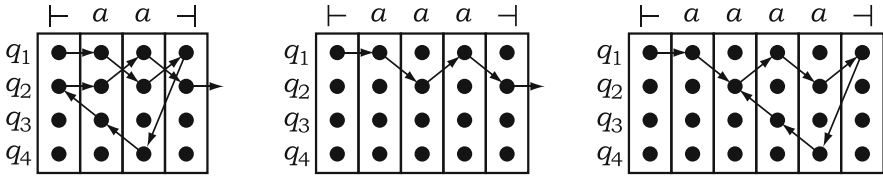
Fig. 3. Computation of the 2DFA for L_n with $n = 4$ in Theorem 2

Example 1. Let $n \geqslant 2$ and consider the language $L_n = a(a^2)^* \cup \{a^{n-2}\}$ if n is even, or $L_n = (a^2)^* \cup \{a^{n-2}\}$ for n odd. Then L_n is recognized by an n-state 2DFA with acceptance only on the right-end marker (that is, with the standard definition). However, every sweeping 2DFA for L_n needs $n + 1$ states.

The obvious automaton for this language is an $(n + 1)$-state 1DFA, and a sweeping 2DFA cannot have fewer states. However, one state can be saved at the expense of losing the sweeping property, as follows. Consider the case of $n = 4$, presented in Figure 3. The automaton begins by traversing the string from left to right, counting modulo 2. Once the right-end marker is reached, the string is either accepted because of its parity, or the automaton proceeds back to the left, counting up to $n - 2$. A sweeping 2DFA would need an extra rejecting state, reached if the string is of length n or greater; the proposed 2DFA can reuse the first two states instead, rejecting by entering an infinite loop.

5   Transformation to One-Way Automata

Now consider the question of the number of states in 1DFAs and 1NFAs needed to represent languages recognized by n-state unary 2DFAs. Chrobak [4] was the first to find out that both tradeoffs are asymptotically equivalent to the function

$$G(n) = \max\{\, \mathrm{lcm}\{p_1, \ldots, p_k\} \mid p_1 + \cdots + p_k \leqslant n \,\},$$

known as *Landau's function* and estimated as $G(n) = e^{(1+o(1))\sqrt{n \ln n}}$ [9].

For a 2DFA over an alphabet $\{a\}$, with $f \in \mathcal{TT}_Q$ representing the behaviour on a, the numbers $p_1, \ldots, p_k$ in the definition of G correspond, according to Theorem 1, to the cycles in $\tilde{f}$. The length of the tail ℓ in Theorem 1 is actually reflected by the number $n + 1 - (p_1 + \cdots + p_k)$. The contribution of ℓ to the size of a 1DFA is taken into account in the precise expression for the 2DFA-to-1DFA tradeoff given in the following theorem.

Theorem 3. *Let $n \geqslant 1$. Then for every unary two-way automaton $\mathcal{A}$ with n states, where the transitions on the letter are deterministic, there exists an equivalent complete 1DFA with $\max_{1 \leqslant \ell \leqslant n+1} G(n + 1 - \ell) + \ell$ states. For $n \geqslant 3$, this bound is tight already for the transformation of complete 2DFAs with acceptance on both sides to 1NFAs.*

The upper bound follows from Corollary 2 with $x = a$ and $u = v = \varepsilon$. Proving that this bound is tight requires some efforts. Given $n \geqslant 3$, let $k \geqslant 1$, $\ell \geqslant 1$, and let $p_1, \ldots, p_k \geqslant 2$ be powers of distinct primes, such that $p_1 + \cdots + p_k + (\ell - 1) = n$ and $G(n + 1 - \ell) = p_1 \cdots p_k = p$. A lower bound $\max_{1 \leqslant \ell \leqslant n+1} G(n + 1 - \ell) + \ell - 1$ on the 2DFA to 1DFA tradeoff is obtained straightforwardly by constructing an $(n + 1)$-state 2DFA for such language as, for instance, $a^{p+\ell-1}(a^p)^*$. To see that one extra state is necessary, one has to embed within such an example the same idea as in the lower bound in Theorem 2. This is achieved in the language

$$L_n = a^{p+\ell-1}(a^p)^* \cup \{\, a^i \mid i \equiv \ell \pmod{p_k} \text{ and } i \equiv \ell - 1 \pmod{p_1 \cdots p_{k-1}} \,\},$$

which is nontrivially recognized by a 2DFA with n states, while every 1DFA for this language obviously requires $p + \ell$ states.

References

1. Berman, P.: A note on sweeping automata. In: de Bakker, J.W., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 91–97. Springer, Heidelberg (1980)
2. Berman, P., Lingas, A.: On complexity of regular languages in terms of finite automata. Report 304, Institute of Computer Science, Polish Academy of Sciences, Warsaw (1977)
3. Birget, J.-C.: Concatenation of inputs in a two-way automaton. Theoretical Computer Science 63(2), 141–156 (1989)
4. Chrobak, M.: Finite automata and unary languages. Theoretical Computer Science 47, 149–158 (1986); Errata 302, 497–498 (2003)
5. Geffert, V., Mereghetti, C., Pighizzini, G.: Converting two-way nondeterministic unary automata into simpler automata. Theoretical Computer Science 295(1-3), 189–203 (2003)
6. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata. In: Dediu, A.H., Ionescu, A.M., Martín-Vide, C. (eds.) LATA 2009. LNCS, vol. 5457, pp. 23–42. Springer, Heidelberg (2009)
7. Jirásková, G., Okhotin, A.: On the state complexity of operations on two-way finite automata. In: Ito, M., Toyama, M. (eds.) DLT 2008. LNCS, vol. 5257, pp. 443–454. Springer, Heidelberg (2008)
8. Kapoutsis, C.A.: Removing bidirectionality from nondeterministic finite automata. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 544–555. Springer, Heidelberg (2005)
9. Landau, E.: Uber die Maximalordnung der Permutationen gegebenen Grades (On the maximal order of permutations of a given degree). Archiv der Mathematik und Physik, Ser. 3 5, 92–103 (1903)
10. Mereghetti, C., Pighizzini, G.: Optimal simulations between unary automata. SIAM Journal on Computing 30(6), 1976–1992 (2001)
11. Mereghetti, C., Pighizzini, G.: Two-way automata simulations and unary languages. Journal of Automata, Languages and Combinatorics 5(3), 287–300 (2000)
12. Micali, S.: Two-way deterministic finite automata are exponentially more succinct than sweeping automata. Information Processing Letters 12(2), 103–105 (1981)
13. Moore, F.R.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. IEEE Transactions on Computers 20, 1211–1214 (1971)

14. Perrin, D.: Finite Automata. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science vol. B, pp. 1–57. MIT Press, Cambridge (1990)
15. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM Journal of Research and Development 3, 114–125 (1959)
16. Sakoda, W.J., Sipser, M.: Nondeterminism and the size of two way finite automata. In: STOC 1978, pp. 275–286 (1978)
17. Shepherdson, J.C.: The reduction of two-way automata to one-way automata. IBM Journal of Research and Development 3, 198–200 (1959)
18. Sipser, M.: Lower bounds on the size of sweeping automata. In: STOC 1979, pp. 360–364 (1979)
19. Vardi, M.: A note on the reduction of two-way automata to one-way automata. Information Processing Letters 30(5), 261–264 (1989)