Michal Kunc | Alexander Okhotin

Reversible two-way finite automata over a unary alphabet

Turku Centre *for* Computer Science

TUCS Technical Report
No 1024, December 2011

Reversible two-way finite automata over a unary alphabet

Michal Kunc
Department of Mathematics, Masaryk University,
Brno, Czech Republic
kunc@math.muni.cz
Alexander Okhotin
Department of Mathematics, University of Turku, *and*
Turku Centre for Computer Science
Turku FI–20014, Finland
alexander.okhotin@utu.fi

TUCS Technical Report
No 1024, December 2011

Abstract

It is established that transforming an n-state two-way deterministic finite automaton over a one-letter alphabet to an equivalent two-way reversible automaton requires between $2n-2$ and $2n+3$ states.

Keywords: Finite automata, two-way automata, reversible automata, unary languages, descriptional complexity, reversible computation.

TUCS Laboratory
Discrete Mathematics for Information Technology

1 Introduction

Consider a special case of *two-way deterministic finite automata* (2DFA), called *reversible 2DFAs*, in which every step of computation is logically reversible, that is, every configuration has a uniquely determined predecessor. Reversibility is an important property of computational devices in general, which is particularly relevant to the physics of computation [2, 7]. It has a long history of complexity-theoretic studies [3, 5, 11], leading to a notable result of Lange et al. [11] on the equivalence of reversible space and deterministic space. In the domain of finite automata, reversible 1DFAs define a proper subfamily of regular languages [13]; on the other hand, every regular language is accepted by a reversible 2DFA: as shown by Kondacs and Watrous [8], every n-state 1DFA can be simulated by a $2n$-state reversible 2DFA.

This paper presents a transformation of an n-state 2DFA over a one-letter alphabet to an equivalent reversible 2DFA with $2n + 3$ states, which is done by generalizing the method of Kondacs and Watrous [8] to any sweeping 2DFA rather than just a 1DFA. In conjunction with the transformation of an arbitrary one-letter 2DFA to a sweeping 2DFA [9], this yields the desired construction. This transformation is accompanied by a proof that transforming an n-state unary 2DFA to an equivalent reversible 2DFA requires at least $2n - 2$ states in the worst case.

2 Two-way automata

Given an input string w, a 2DFA operates on a tape containing the string $\vdash w \dashv$, where $\vdash$ and $\dashv$ are special symbols known as the left-end marker and the right-end marker, respectively. According to the standard definition, a 2DFA begins its computation at the left-end marker and accepts at the right-end marker. In this paper, as well as in the authors' previous work [9, 10], the definition is extended to allow acceptance on both sides: this leads to symmetric constructions and allows avoiding some awkward exceptions in the results. Changing the mode of acceptance affects the size of an automaton at most by one state.

A 2DFA is defined as a sextuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, F_\vdash, F_\dashv)$, in which Σ is a finite alphabet with $\vdash, \dashv \notin \Sigma$, Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta\colon Q \times (\Sigma \cup \{\vdash, \dashv\}) \to Q \times \{-1, +1\}$ is a partially defined transition function, and $F_\vdash, F_\dashv \subseteq Q$ are sets of states accepting on the left-end marker $\vdash$ and on the right-end marker $\dashv$, respectively. When $\mathcal{A}$ is in a state q and observes a square of the tape with a symbol $a \in \Sigma \cup \{\vdash, \dashv\}$, the value $\delta(q, a)$ indicates the next state and the direction of motion. It is assumed that every transition defined at any of the end-markers must move the head in the appropriate direction (that is, inside the string). Additionally, for a state

$q \in F_\vdash$ or $q \in F_\dashv$ the value of $\delta(q, \vdash)$, $\delta(q, \dashv)$ respectively, is not defined.

The computation of $\mathcal{A}$ on an input string $w = a_1 \dots a_\ell$, with $\ell \geqslant 0$ and $a_1, \dots, a_\ell \in \Sigma$, takes place on the tape containing the symbols $\vdash w \dashv = \vdash a_1 \dots a_\ell \dashv$, and begins in the state q_0, with the head observing the left-end marker $\vdash$. If it eventually reaches an accepting state in $F_\vdash$ or in $F_\dashv$ while on the corresponding end-marker, the string is accepted; otherwise, it either encounters an undefined transition or gets into an infinite loop.

Formally, let $a_0 = \vdash$ and $a_{\ell+1} = \dashv$. Then a *computation* of $\mathcal{A}$ on w, beginning with a configuration (p_0, i_0), is the longest sequence $(p_0, i_0), (p_1, i_1), \dots$, finite or infinite, in which

- $p_t \in Q$ and $0 \leqslant i_t \leqslant \ell + 1$ for each t-th step;
- every next element (p_t, i_t), if it is defined, satisfies $\delta(p_{t-1}, a_{i_{t-1}}) = (p_t, d_t)$ and $i_t = i_{t-1} + d_t$.

The computation beginning with a given configuration (p_0, i_0) is always uniquely defined. It is *accepting* if it is finite and its last configuration (p_f, i_f) satisfies either $i_f = 0$ and $p_f \in F_\vdash$, or $i_f = \ell + 1$ and $p_f \in F_\dashv$. The language recognized by the 2DFA $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of strings accepted from the configuration $(q_0, 0)$.

A 2DFA is called *sweeping* [14], if in every computation its head changes the direction of motion only on the end markers. It is an open problem, whether every n-state 2DFA has an equivalent sweeping 2DFA with polynomially many states.

Theorem 1 (Kunc, Okhotin [9])**.** *Let $n \geqslant 1$. Then for every 2DFA $\mathcal{A}$ over a unary alphabet, with n states, there exists an equivalent sweeping 2DFA with $n + 1$ states. For $n \geqslant 2$, this bound is the best possible.*

3 Reversible automata

Let a 2DFA be called *direction-determinate*, if for every state $q \in Q$, all transitions leading to q move the head in the same direction $d(q) \in \{-1, +1\}$, and $d(q_0) = +1$ for the initial state. A direction-determinate 2DFA is *reversible*, if the transitions from two different states by the same symbol cannot lead to a single state, that is, whenever $\delta(p, a) = \delta(p', a) = (q, d(q))$, the states p and p' must be the same. A reversible 2DFA is *strongly reversible*, if, furthermore, its transitions in all states and on all symbols are defined, with the following two exceptions: first, the transitions from those states on the end-markers that cannot be reached from the correct direction are not defined (that is, $\delta(q, \vdash)$ is undefined for $q \neq q_0$ with $d(q) = +1$, and $\delta(q, \dashv)$ is undefined for q with $d(q) = -1$); secondly, one of the end-markers is designated as the ending point of all computations, and there exist two special states, q_{acc} and q_{rej}, which have undefined transitions on this end-marker, of which q_{acc} is accepting.

Lemma 1. *A reversible 2DFA halts on every input.*

In a strongly reversible 2DFA, all computations end on the same designated end-marker, in one of the states $q_{\mathrm{acc}}, q_{\mathrm{rej}}$.

Proof. Let $\mathcal{A}$ be a reversible 2DFA, and suppose that it does not halt on an input string $w \in \Sigma^*$. Let (q, i) be the first repeating configuration; it cannot be the initial configuration, because $d(q_0) = +1$ implies that $(q_0, 0)$ cannot be entered for the second time.

Let $(p, i - d)$ and $(p', i - d')$ be the configurations, from which (q, i) is entered at the first time and at the second time, respectively. These configurations must be different, because otherwise (q, i) would not be the first repeating configuration. If $d \neq d'$, then the state q is enterable from both sides, and hence the 2DFA is not direction-determinate. Assume $d = d'$ and $p \neq p'$, and let a be the symbol in the position $i - d$. Then, $\delta(p, a) = \delta(p', a) = (q, d)$, and therefore the automaton is not reversible. The contradiction obtained proves the first statement of the lemma.

Now consider a strongly reversible 2DFA. Since, by the above arguments, it halts on every input, every computation leads either to an undefined transition, or to an accepting state (which also has the transition undefined). All undefined transitions are on the end-markers, and all of them, except those in the states q_{acc} and q_{rej} on the designated end-marker, take place on unreachable configurations; hence, no computation can end by reaching them. Thus, rejection is only possible in the state q_{rej}, on the designated marker, and acceptance can take place only on the same marker, in the state q_{acc}. Therefore, every computation must end in one of these two configurations. □

4 Transformation to reversible automata

It was proved by Kondacs and Watrous [8] that every n-state 1DFA can be simulated by a $2n$-state reversible 2DFA. Their construction shall now be generalized to apply to any sweeping 2DFA, rather than a 1DFA.

Lemma 2. *Let* $\mathcal{A}$ *be a direction-determinate 2DFA with* n *states over any alphabet. Then there exists and can be effectively constructed a strongly reversible 2DFA with* $2n+1$ *states recognizing the language* $L(\mathcal{A})^R$. *If* $\mathcal{A}$ *accepts only on the left-end marker* $\vdash$, *then* $L(\mathcal{A})$ *is recognized by a strongly reversible 2DFA with* $2n$ *states.*

Proof. Consider the case of acceptance only on the left-end marker $\vdash$. As the direction is always known, the notation for the transition function can be simplified as follows: for each $a \in \Sigma \cup \{\vdash, \dashv\}$, let $\delta_a \colon Q \to Q$ be a partial function defined by $\delta_a(p) = q$ if $\delta(p, a) = (q, d(q))$. Consider also its inverse, $\delta_a^{-1} \colon Q \to 2^Q$ with $\delta_a^{-1}(q) = \{\, p \mid \delta_a(p) = q \,\}$. Note that a reversible 2DFA has all functions δ_a injective, and hence $|\delta_a^{-1}(q)| \leqslant 1$ for all a and q; but in an arbitrary direction-determinate 2DFA considered in this proof, each set

$\delta_a^{-1}(q)$ may contain any number of elements between 0 and $|Q|$. Assume, without loss of generality, that whenever a state has any transitions defined at one of the end-markers, or is accepting there, this state must be reachable from the corresponding direction: that is, whenever, for any $q \in Q$, the transition $\delta_\dashv(q)$ is defined or $q \in F_\dashv$, the direction $d(q)$ must be $+1$, and if $\delta_\vdash(q)$ is defined or $q \in F_\vdash$ for any state $q \neq q_0$, then $d(q) = -1$ (if any transitions or accepting states violate this condition, they will never be used and can be safely removed). Assume any linear ordering on Q and let $\min S$ and $\max S$ denote the least and the greatest element of a nonempty set $S \subseteq Q$ with respect to this ordering. Let $\mathrm{next}_S(q)$ with $q \in S \subseteq Q$ denote the least element of S strictly greater than q, provided that it exists.

The new automaton begins its computation in the final configurations, and goes through the tree of computations leading to these configurations, searching for the initial configuration. This involves both backward simulation of the original automaton, when exploring each branch of this tree, as well as forward simulation, which is used when the backward search results in a configuration unreachable by the original automaton. For that purpose, the new automaton has the set of states $Q \cup \{\, [q] \mid q \in Q \,\}$, with a new copy $[q]$ for each state from Q. The states of the form $[q]$ simulate the computation backwards. Whenever the new automaton is in a state $[q]$ with the head scanning $a \in \Sigma \cup \{\vdash, \dashv\}$, this means that there exists a computation of the original automaton, beginning with the head in the next position in the string (determined by $d(q)$), while in the state q, and eventually leading to acceptance. In this way the backward computation traces the state and the position of the head of a forward computation, but the state and the position are always out of synchronization by one step. The general arrangement of the simulation is illustrated in Figure 1.

The initial state of the new automaton is $[\min F_\vdash]$, with the following transition by the left-end marker $\vdash$:

$$\delta'([\min F_\vdash], \vdash) = ([\min F_\vdash], +1). \tag{1}$$

The automaton accepts in the state $[\delta_\vdash(q_0)]$ on the left-end marker $\vdash$.

The backward transitions from $[q]$ by $a \in \Sigma \cup \{\vdash, \dashv\}$ are defined as follows. If the state q is reachable from some other states by the symbol a, then the automaton continues with the backward simulation by choosing the least of those predecessor states, $p = \min \delta_a^{-1}(q)$, and moving to the state $[p]$ in the direction $-d(p)$. That is,

$$\delta'([q], a) = \big([\min \delta_a^{-1}(q)], -d(\min \delta_a^{-1}(q))\big), \quad \text{if } \delta_a^{-1}(q) \neq \varnothing. \tag{2}$$

An example of this case is given in Figure 1, in the state $[q]$ and at the symbol a. When the constructed automaton reaches this configuration, it selects the minimal element of the set $\delta_a^{-1}(q) = \{p, p'\}$, which is p, and continues the backward simulation by moving the head to the left (as $-d(p) = -1$) and entering the state $[p]$.

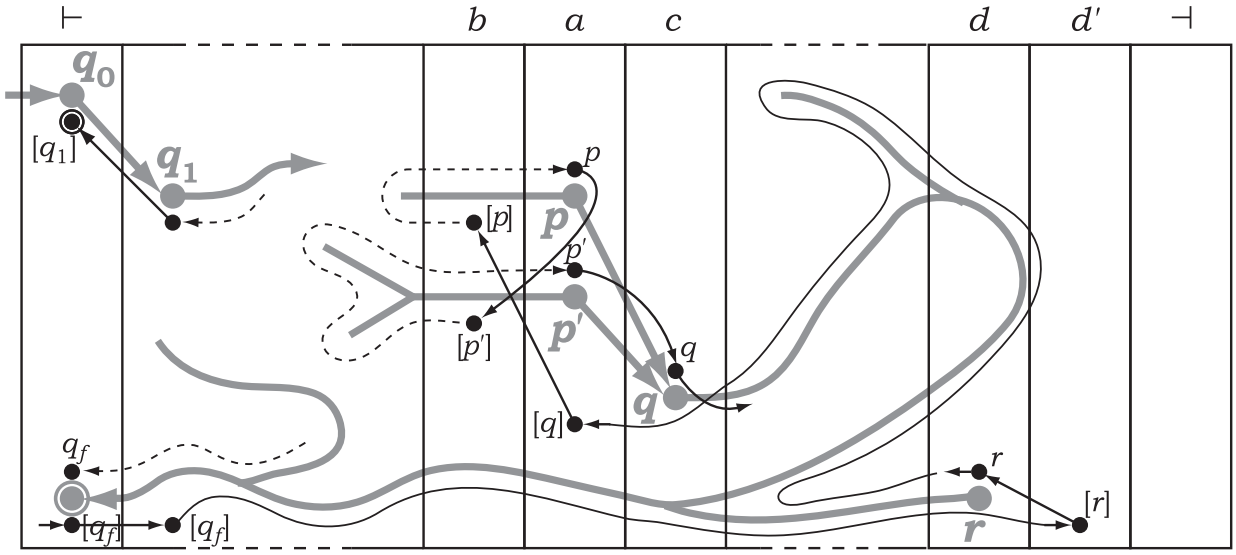

Figure 1: How a given 2DFA (thick grey lines) is simulated by a reversible 2DFA (thin black lines).

The other principal case in the backward simulation is when a branch of the tree of computations is traced back to a configuration without predecessors. Therefore, these computations cannot take place when starting from the initial configuration, and the constructed automaton switches to forward simulation, until it finds the next suitable branch of the tree. This is done by the following transition:

$$\delta'([q], a) = (q, d(q)), \quad \text{if } \delta_a^{-1}(q) = \varnothing, \tag{3}$$

defined unless $a = \vdash$ and $d(q) = -1$, and unless $a = \dashv$ and $d(q) = +1$. This case is illustrated in the figure by the configuration in state r over the symbol d, which has no predecessors.

Forward transitions are defined by the rule that if multiple braches of the tree of computations converge at the present point, and the current branch is not the last of them, then the simulation switches to backtracking the next branch by the following transition:

$$\delta'(p, a) = ([p'], -d(p')), \quad \text{if } \delta_a(p) \text{ is defined and } p' = \mathrm{next}_{\delta_a^{-1}(\delta_a(p))}(p). \tag{4}$$

An example of this can be found in Figure 1, in the state p over the symbol a, where $\delta_a(p) = q$ is the state where the branching occurs, the predecessors of this state are $\delta_a^{-1}(q) = \{p, p'\}$, and the current state p is not the last of them. Then the automaton takes the next branch by going to the left in the state $[p']$.

If there was no branching at the present point, or if there was a branching, but the current branch is already the last of them (this is a single case), then

the original automaton is simulated forward:

$$\delta'(p,a) = \big(\delta_a(p), d(\delta_a(p))\big), \quad \text{if } \delta_a(p) \text{ is defined and } p = \max \delta_a^{-1}(\delta_a(p)), \tag{5}$$

or, in other words, $\delta'(p,a) = \delta(p,a)$. In the figure, this case occurs in the state p' over the symbol a, where $\delta_a^{-1}(\delta_a(p')) = \{p, p'\}$ and p' is the last of these states.

It remains to define the transitions in the case of undefined $\delta_a(p)$:

$$\delta'(p,a) = ([p], -d(p)), \quad \text{if } \delta_a(p) \text{ is undefined, and } (p,a) \notin F_{\vdash} \times \{\vdash\}. \tag{6}$$

These transitions shall never be used, but they are necessary to comply with the definition of strong reversibility.

Finally, if the left-end marker $\vdash$ is reached in one of the accepting states, this means that the backward computation tree leading to this state has been completely searched, and one should switch to the next accepting state, if it exists:

$$\delta'(p,\vdash) = \big([\mathrm{next}_{F_{\vdash}}(p)], +1\big), \quad \text{if } p \in F_{\vdash} \text{ and it is not the greatest element of } F_{\vdash}. \tag{7}$$

And if this was the last accepting state, the constructed automaton rejects, that is, $q_{\mathrm{rej}} = \max F_{\vdash}$ and $\delta'(\max F_{\vdash}, \vdash)$ is not defined.

Fix an input string $w \in \Sigma^*$. For every configuration (p,i), from which the original automaton accepts w, let $\pi(p,i)$ denote the (uniquely determined) finite path from (p,i) to the corresponding accepting configuration. The above ordering of states induces the following strict partial ordering on accepting computations of the original automaton: a computation $\pi(p_\ell, i_\ell) = (p_\ell, i_\ell)\dots(p_1,i_1)(p_0,i_0)$ is less than $\pi(p'_{\ell'}, i'_{\ell'}) = (p'_{\ell'}, i'_{\ell'})\dots(p'_1,i'_1)(p'_0,i'_0)$ if there exists such an $\hat{\ell} \leqslant \ell, \ell'$ that $(p_j,i_j) = (p'_j,i'_j)$ for all $j \in \{0,\dots,\hat{\ell}-1\}$ and $p_{\hat{\ell}} < p'_{\hat{\ell}}$. Then two computations are incomparable if and only if one of them is a suffix of the other.

The correctness statement of the construction reads as follows:

Claim 1. *If the new automaton reaches a configuration $([q],i)$ in one or more steps, then the original automaton accepts from $(q, i+d(q))$ and, if the original automaton accepts from $(q_0,0)$, the computation $\pi(q_0,0)$ is not less than $\pi(q, i+d(q))$.*

If the new automaton reaches (q,i), then the original automaton accepts from (q,i) and if the computation $\pi(q_0,0)$ is accepting, then it is greater than $\pi(q,i)$.

The claim is proved by induction on the length of the computation of the new automaton.

Basis. At the first step of the computation, the new automaton goes from its initial configuration $([\min F_{\vdash}],0)$ to the configuration $([\min F_{\vdash}],1)$. Since the state $\min F_{\vdash}$ is enterable on the left-end marker $\vdash$ by the assumption,

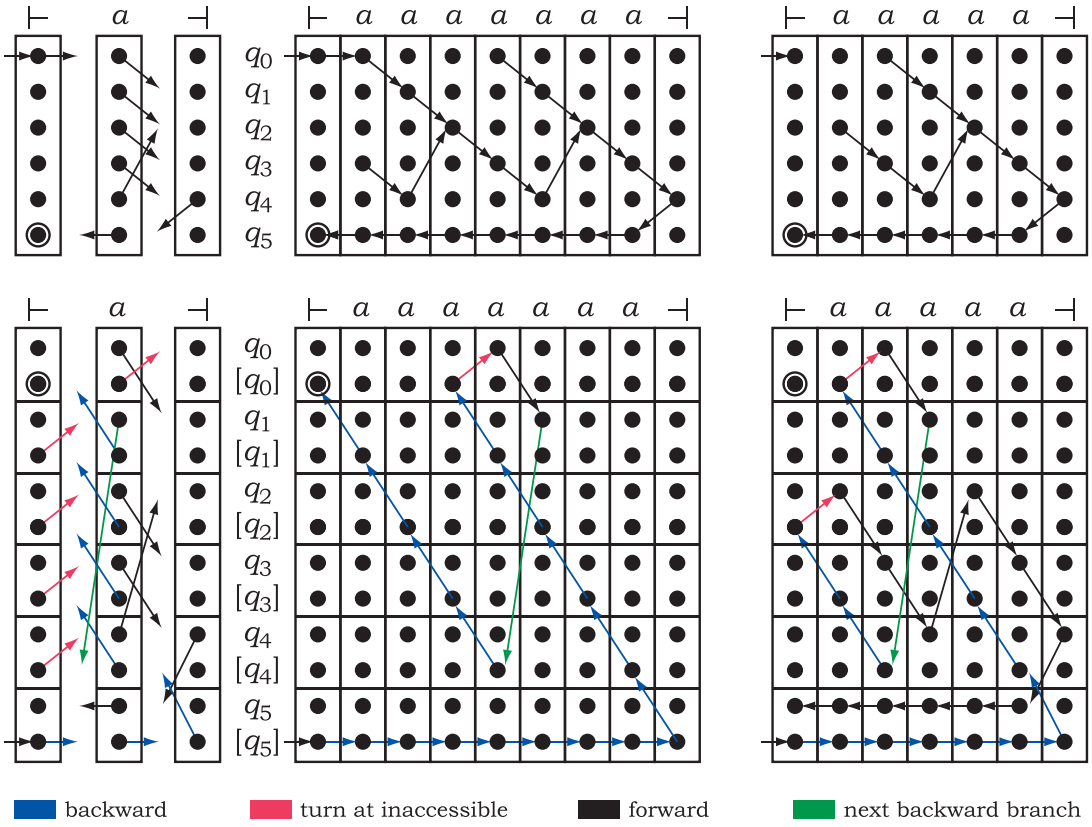

Figure 2: Example of a transformation to a reversible 2DFA.

$d(\min F_\vdash) = -1$, and $(\min F_\vdash, 1 + d(\min F_\vdash)) = (\min F_\vdash, 0)$ is an accepting configuration of the original automaton. Because $\min F_\vdash$ is the least accepting state, there is no accepting computation less than $\pi(\min F_\vdash, 0) = (\min F_\vdash, 0)$.

Induction step I. Assume that the constructed automaton reaches a configuration $([q], i)$ and the statement of the lemma holds true, that is, the original automaton accepts from $(q, i + d(q))$ and $\pi(q_0, 0)$ is not less than $\pi(q, i + d(q))$ (provided that the original automaton accepts from $(q_0, 0)$). If the constructed automaton accepts in this configuration, then $([q], i) = ([\delta_\vdash(q_0)], 0)$, and the acceptance is justified by the fact that the original automaton has an accepting computation starting from $(\delta_\vdash(q_0), 0 + d(\delta_\vdash(q_0))) = (\delta_\vdash(q_0), 1)$.

If the constructed automaton does not accept here, consider the next step of its computation. Let $a \in \Sigma \cup \{\vdash, \dashv\}$ be the symbol in the position i and first assume that $\delta_a^{-1}(q) \neq \varnothing$. Let $p = \min \delta_a^{-1}(q)$. Then the new automaton makes a transition from $([q], i)$ to $([p], i - d(p))$, while the original automaton goes from $\big(p, (i - d(p)) + d(p)\big) = (p, i)$ to $(q, i + d(q))$, from whence it accepts by the induction assumption. Suppose, for the sake of contradiction, that the original automaton accepts from $(q_0, 0)$ and $\pi(q_0, 0) < \pi(p, i)$. Since

$\pi(p,i) = (p,i) \cdot \pi(q, i + d(q))$ and $\pi(q_0, 0)$ is not less than $\pi(q, i + d(q))$ by the assumption, it follows that $\pi(q_0, 0)$ and $\pi(p, i)$ must be different in the first step of $\pi(p, i)$, that is, $\pi(q_0, 0) = (q_0, 0) \ldots (p', i) \cdot \pi(q, i + d(q))$ and $p' < p$. Then $\delta_a(p') = q$ and $p' \in \delta_a^{-1}(q)$, which contradicts the assumption that p is the least element of $\delta_a^{-1}(q)$.

The other possible next step of the new automaton from the configuration $([q], i)$ occurs for $\delta_a^{-1}(q) = \varnothing$: then it proceeds to the configuration $(q, i + d(q))$, and it has to be established that the original automaton accepts starting from $(q, i + d(q))$. That is directly given by the induction hypothesis for $([q], i)$, which furthermore states that if the original automaton accepts starting from $(q_0, 0)$, then $\pi(q_0, 0)$ cannot be less than $\pi(q, i + d(q))$. It remains to argue that $\pi(q, i + d(q))$ is not a suffix of $\pi(q_0, 0)$. Indeed, if it were a proper suffix, then there would have been a previous configuration (p, i), from which the original automaton would go to $(q, i + d(q))$; but this would imply that $p \in \delta_a^{-1}(q)$ and thus contradict the assumption. If $\pi(q, i + d(q))$ and $\pi(q_0, 0)$ are actually the same computation, then $i + d(q) = i + d(q_0) = 0$, which is impossible, because $d(q_0) = +1$.

Induction step II. Let the statement of the lemma hold for a configuration (p, i) of the constructed automaton, that is, $\pi(p, i)$ is assumed to be an accepting computation, and in case the original automaton accepts from $(q_0, 0)$, the computation $\pi(q_0, 0)$ is neither less than $\pi(p, i)$, nor has $\pi(p, i)$ as a suffix. The statement of the lemma is to be established for the configuration of the constructed automaton at the next step after (p, i). Since the computation $\pi(p, i)$ is accepting, there are two possibilities: either it consists of a unique accepting configuration, or begins with a transition to $\big(\delta_a(p), i + d(\delta_a(p))\big)$, where a is the symbol at the position i.

In the former case, $i = 0$ and $\pi(p, i) = (p, 0)$, and it is known that if the computation $\pi(q_0, 0)$ is accepting, then its last configuration may not be $(p, 0)$ (as $\pi(p, 0) = (p, 0)$ would then be a suffix of $\pi(q_0, 0)$), and its last configuration also cannot be $(q, 0)$ with $q < p$ (because then $\pi(q_0, 0) < \pi(p, 0)$). Thus this branch of the tree of accepting computations was found not to come from the initial state. If p is **not the greatest** of the accepting states, the simulation continues backtracking from the next accepting state $p' = \mathrm{next}_{F_\vdash}(p)$. Since p' is enterable from the right by assumption, $d(p')$ must be -1. In this case the constructed automaton proceeds to the configuration $([p'], 1) = ([p'], -d(p'))$, while the original automaton accepts from the configuration $(p', 1 + d(p')) = (p', 0)$, because it *is* an accepting configuration. Furthermore, if the original automaton accepts from $(q_0, 0)$, and $\pi(q_0, 0)$ were supposed to be less than $\pi(p', 0)$, then $\pi(q_0, 0)$ and $\pi(p', 0)$ would differ at the last step, that is, $\pi(q_0, 0)$ would accept in a state p or less, which is known to be untrue. Turning to the other possibility, if p is **the greatest** state in $F_\vdash$, this implies that the original automaton does not accept this string, because if it did, then its accepting computation must have ended with one of the accepting configurations. This is the case when the constructed automaton

rightfully rejects.

Finally, assume that the computation $\pi(p,i)$ of the original automaton begins with a transition from (p,i) to $\big(\delta_a(p), i + d(\delta_a(p))\big)$. Let $q = \delta_a(p)$. If p is **the greatest** state in $\delta_a^{-1}(q)$, then the constructed automaton similarly goes from (p,i) to $\big(q, i+d(q)\big)$. The computation $\pi(q, i+d(q))$ of the original automaton is accepting as a suffix of the accepting computation $\pi(p,i)$. If the original automaton accepts from $(q_0,0)$ and $\pi(q_0,0)$ is less than $\pi(q, i + d(q))$, then $\pi(q_0,0)$ is also less than the longer computation $\pi(p,i)$, which contradicts the induction hypothesis. Suppose $\pi(q, i + d(q))$ is a suffix of $\pi(q_0,0)$. These two computations cannot coincide, because $i+d(q_0) = i+1 \neq 0$, and so $\pi(q, i + d(q))$ must be a proper suffix of $\pi(q_0,0)$. Then, since $\pi(p,i)$ is not a suffix of $\pi(q_0,0)$ by the induction hypothesis, the computations $\pi(p,i) = (p,i)\cdot\pi(q, i+d(q))$ and $\pi(q_0,0)$ differ on the first step of $\pi(p,i)$, and $\pi(q_0,0) = \ldots (p',i)\cdot\pi(q, i+d(q))$ for some state $p' \neq p$ with $\delta_a(p') = q$. Then $p' < p$, because p is known to be the greatest of such states, and therefore $\pi(q_0,0) < \pi(p,i)$, a contradiction.

Otherwise, if **there is a greater** state $p' > p$ with $\delta_a(p') = q$, the constructed automaton proceeds from (p,i) to $([p'], i - d(p'))$, where p' is the least of such states, that is, $p' = \mathrm{next}_{\delta_a^{-1}(q)}(p)$. Then the original automaton in the configuration $(p', i - d(p') + d(p')) = (p', i)$ will apply the transition $\delta(p', a) = (q, d(q))$ and arrive to the configuration $(q, i+d(q))$; this is the next configuration after (p,i), and hence the original automaton accepts from it. Now assume that the original automaton accepts from $(q_0,0)$, and suppose its accepting computation $\pi(q_0,0)$ is less than $\pi(p',i) = (p',i)\cdot\pi(q, i+d(q))$. Since $\pi(q_0,0)$ is not less than $\pi(q, i + d(q))$ by the assumption, the computation $\pi(q_0,0)$ must be different from $\pi(p',i)$ in the first configuration of $\pi(p',i)$, that is, $\pi(q_0,0)$ has a suffix $(p'',i)\cdot\pi(q, i+d(q))$ with $\delta_a(p'') = q$ and $p'' < p'$. Now, if $p'' = p$, then $\pi(p,i)$ is a suffix of $\pi(q_0,0)$, and if $p'' < p$, then $\pi(p,i)$ is less than $\pi(q_0,0)$; either possibility contradicts the assumption. This completes the proof of Claim 1.

Claim 2. *The constructed automaton is reversible.*

The new automaton is direction-determinate, with $d'(q) = d(q)$ and $d'([q]) = -d(q)$, and so it remains to check that every state is reached from every symbol in a unique way. Fix a symbol $a \in \Sigma \cup \{\vdash, \dashv\}$ and consider how each state in $Q \cup \{\, [q] \mid q \in Q \,\}$ can be reached by a transition reading a.

Consider any state $q \in Q$ that is reachable from some state by a, that is, with $\delta_a^{-1}(q) \neq \varnothing$. Then, q can only be reached by a transition (5) from the state $\max \delta_a^{-1}(q)$. An unreachable state $q \in Q$ with $\delta_a^{-1}(q) = \varnothing$ can be reached only from by a transition (3) from the state $[q]$.

States from $\{\, [p] \mid p \in Q \,\}$ can be reached by five different types of transitions: (1) (2), (4), (6), (7). First consider a state $[p]$ with $p \in Q$, for which $\delta_a(p)$ is defined. Then, if p is the least state in $\delta_a^{-1}(\delta_a(p))$, it is reached by a transition (2) from $[\delta_a(p)]$, and if p is not the least among the pre-images of

$\delta_a(p)$, then it is reached by a transition (4) from the state p', defined as the greatest element of $\delta_a^{-1}(\delta_a(p))$ strictly less than p.

Consider transitions leading to a state $[p]$, with undefined $\delta_a(p)$, and first assume that the state p is accepting on a, that is, $(p, a) \in F_{\vdash} \times \{\vdash\}$: then $\delta_a(p)$ is undefined by the assumption. If p is the least state in $F_{\vdash}$, then it is reached by the starting transition (1) of the constructed automaton. If p not the least among the states in $F_{\vdash}$, let p' be the greatest state in $F_{\vdash}$ that is less than p; then p is reached by a transition (7) from p'.

Finally, a state $[p]$, with $\delta_a(p)$ undefined and with $(p, a) \notin F_{\vdash} \times \{\vdash\}$, is reached only by a transition (6) from the state p.

Thus, it has been determined that all seven types of transitions by a lead to seven disjoint groups of states, proving Claim 1.

Thus, the constructed automaton is reversible. It is strongly reversible, because all transitions on symbols from Σ, as well as all transitions in reachable states on the markers, are defined, and the only undefined transitions are those in the accepting state $[\delta_{\vdash}(q_0)]$ and the rejecting state $\max F_{\vdash}$ on the left-end marker $\vdash$. Now Lemma 1 asserts that every computation of the constructed automaton ends in one of these two configurations, and Claim 1 implies that the new automaton recognizes the same language as the original one.

Assume that the new automaton accepts a string $w \in \Sigma^*$. Then it does so in the configuration $([\delta_{\vdash}(q_0)], 0)$. Then, by Claim 1, the original automaton accepts from $(\delta_{\vdash}(q_0), 1)$ on the same input w. Since that is the second configuration in its computation of the original automaton, it therefore accepts w.

If the new automaton rejects, it does so in the configuration $(\max F_{\vdash}, 0)$. According to Claim 1, if the original automaton accepts, then its accepting computation $\pi(q_0, 0)$ must be greater than $\pi(\max F_{\vdash}, 0) = (\max F_{\vdash}, 0)$. However, no computation may be greater $(\max F_{\vdash}, 0)$, and therefore the original automaton either rejects or goes to an infinite loop. □

Consider the modifications of the construction needed to support automata with acceptance on both sides of the string. The constructed automaton begins its computation on the right-end marker $\dashv$, and first proceeds as the mirror image of the above construction, searching through the accepting computations of the original automaton ending on the right-end marker $\dashv$, in each state from $F_{\dashv}$. This is done exactly as in the above proof, until the state $\max F_{\dashv}$ is reached on the right-end marker $\dashv$. At this point, the new automaton uses a new extra state $q_{\leftarrow}$ to move from the state $\max F_{\dashv}$ on the right-end marker $\dashv$ to the configuration $([\min F_{\vdash}], 1)$, from whence it proceeds with searching the tree of accepting computations ending on the left-end marker $\vdash$. If the state $[\max F_{\vdash}]$ is reached on the left-end marker $\vdash$, the automaton rejects. As before, the acceptance takes place in the state $[\delta_{\vdash}(q_0)]$ on the left-end marker.

5 Lower bound on the size of reversible automata

The next lemma shows that the number of states in the construction in Lemma 2 is optimal up to an additive constant.

Lemma 3. *For every $n \geqslant 2$, every reversible 2DFA for any co-finite language $L \subseteq a^*$, such that the longest string not in L is a^{n-2}, must have at least $2n-2$ states.*

Proof. Suppose there exists a reversible 2DFA recognizing L with at most $2n-3$ states. Consider the string $u = a^{n-2}$, which is the longest string not belonging to L, and let $v = a^{n-2+(2n-3)!} \in L$. It will be verified by induction that the computations of the automaton on both strings reach the markers in the same states. This, in particular, implies that u is accepted by the automaton if and only if v is accepted, which is in contradiction with $u \notin L$ and $v \in L$.

Consider a computation of the automaton on the string v, which begins in some state q at one of the markers (say $\vdash$). First assume that the computation returns back to the left-end marker $\vdash$ before reaching the right-end marker $\dashv$. This means that the computation is of the form $(q, 0)$, (q_1, i_1), (q_2, i_2), $\ldots$, (q_{k-1}, i_{k-1}), $(q_k, 0)$, with $1 \leqslant i_j \leqslant |v|$ for $1 < j < k-1$ and $i_1 = i_{k-1} = 1$. The first claim is that all the states $q_1, q_2, \ldots, q_{k-1}, q_k$ are different. If $q_s = q_t$ for $s < t \leqslant k$, then $q_{s-1} = q_{t-1}$ due to the reversibility of the automaton, and applying this property $s-1$ times leads to $q_1 = q_{t-s+1}$. Accordingly, it is left to show that $q_1 \neq q_j$ for $1 < j \leqslant k$. First, note that q_1 cannot be equal to q_k, since they are reached from different directions. And if $q_1 = q_j$ for $1 < j < k$, then (q_j, i_j) is reached from the left, which implies $i_j > 1$, because $i_{j-1} > 0$. Hence, the computation keeps revisiting the state q_1 while moving to the right, and therefore it cannot return to the left-end marker $\vdash$ before visiting the right-end marker $\dashv$.

Thus, the states $q_1, \ldots, q_k$ must be pairwise distinct, that is, $k \leqslant 2n-3$. This shows that this computation cannot visit the $(n-1)$th letter of v, and therefore the computation on u starting in the configuration $(q, 0)$ proceeds to $(q_k, 0)$ in the same way as the computation on v.

Now assume that the computation on v starting in q on the left-end marker $\vdash$ reaches the right-end marker $\dashv$ before visiting the left-end marker $\vdash$ again, so that the computation is of the form $(q, 0)$, (q_1, i_1), (q_2, i_2), $\ldots$, (q_{k-1}, i_{k-1}), $(q_k, |v|+1)$, where $1 \leqslant i_j \leqslant |v|$ for $1 < j < k-1$, $i_1 = 1$ and $i_{k-1} = |v|$. Since the string v is of length at least $2n-3$, the length k of this computation must be at least $2n-2$. Because the automaton has at most $2n-3$ states, two of the states $q_1, q_2, \ldots, q_{2n-2}$ must be equal. Due to the reversibility of the automaton, this means that $q_1 = q_j$ for some $1 < j \leqslant 2n-2$. Therefore, the whole computation between (q_1, i_1) and $(q_k, |v|+1)$ proceeds by repeating the sequence of states $q_1, \ldots, q_j$, in each iteration moving to the right by $i_j - 1$

symbols, with $i_j - 1 \leqslant j - 1 \leqslant 2n - 3$. Let m be the rightmost position reached during the computation from $(q_1, 1)$ to (q_j, i_j), that is, the maximum of the numbers $i_1, \ldots, i_j$. Since the automaton is direction-determinate, the configuration $(q_j, i_j) = (q_1, i_j)$ must be reached from the left, so the previous configuration is $(q_{j-1}, i_j - 1)$. The number of steps needed to get from $(q_1, 1)$ to the position m and then return back to the configuration $(q_{j-1}, i_j - 1)$ is at least $2(m-1) - (i_j - 2) = 2m - i_j$. Because the actual number of these steps is $j - 2 \leqslant 2n - 4$, the number m is at most $n - 2 + \frac{i_j}{2}$.

Note that for the computation on u, the right-end marker is in the position $n-1$ and the difference between the lengths of u and v is a multiple of $i_j - 1$. Therefore, in order to verify that the computations on u and v reach the right-end marker in the same state, it is sufficient to prove that the computation on v reaches all positions of the form $n-1+t\cdot(i_j-1)$, with $t \geqslant 0$, for the first time in the same state. Since every iteration of the cycle $q_1, \ldots, q_j$ is shifted by exactly $i_j - 1$ positions to the right from the previous iteration, this task can be achieved by showing that if the position $n-1+t\cdot(i_j-1)$ is reached for the first time during the kth iteration of the cycle, then the following position $n-1+(t+1)\cdot(i_j-1)$ is reached for the first time during the $(k+1)$th iteration of the cycle. Clearly, it is reached at latest during the $(k+1)$th iteration, so it remains to prove that this cannot happen already during the kth iteration. Assume that the position $n - 1 + (t+1) \cdot (i_j - 1)$ is indeed reached during the kth iteration. As the rightmost position reached during this iteration is $m+(k-1)\cdot(i_j-1)$, this means that $n-1+(t+1)\cdot(i_j-1) \leqslant m+(k-1)\cdot(i_j-1)$, which implies $n - 1 + t \cdot (i_j - 1) \leqslant m + (k-2) \cdot (i_j - 1)$. If $k \geqslant 2$, then this shows that the position $n - 1 + t \cdot (i_j - 1)$ is reached already during the $(k-1)$th iteration, which is a contradiction. If $k = 1$, then

$$n - 2 + i_j \leqslant n - 1 + (t+1) \cdot (i_j - 1) \leqslant m \leqslant n - 2 + \frac{i_j}{2},$$

which is a contradiction as well. □

Theorem 2. *For every n-state unary 2DFA there exists and can be effectively constructed an equivalent reversible 2DFA with $2n + 3$ states that halts on every input. At least $2n - 2$ states are necessary in the worst case.*

Sketch of a proof. A given n-state unary 2DFA is first converted to a sweeping 2DFA with $n + 1$ states, according to Theorem 1. Then, by Lemma 2, there is a reversible 2DFA with $2(n + 1) + 1$ states recognizing the reversal of the original language, which is the same as the original language due to the unary alphabet.

For the lower bound, consider the co-finite language $L_n = \{\, a^\ell \mid \ell \geqslant n - 2 \,\}$, which is recognized by a 1DFA with n states. By Lemma 3, every reversible 2DFA for this language needs to have at least $2n - 2$ states. □

6 Application to multiple-letter alphabets

Since Lemma 2 is applicable to any alphabet, it can be used to make an arbitrary 2DFA reversible. In order to meet its conditions, the following pre-processing step is required:

Lemma 4. *For every 2DFA over any alphabet and with n states, there exists and can be effectively constructed an equivalent direction-determinate 2DFA with at most $2n$ states.*

Sketch of a proof. Each state enterable from both directions is split into two states: the one entered from the left and the one entered from the right. These two states have the same outgoing transitions (exception: on each marker, only one of these states has a transition) □

Stacking these two constructions together leads to a new, more manageable construction and an actual proof of the result of Geffert et al. [6] on making a 2DFA halt on every input:

Theorem 3 (Geffert et al. [6])**.** *For every n-state 2DFA over any alphabet there exists and can be effectively constructed an equivalent strongly reversible 2DFA with $4n + 2$ states.*

Proof. The given n-state 2DFA is first transformed to an equivalent $2n$-state direction-determinate 2DFA according to Lemma 4. Then, Lemma 2 is applied to produce a $(4n + 1)$-state strongly reversible 2DFA recognizing the reversal of the original language. Finally, the latter 2DFA can be reversed again by adding an extra state, leading to the desired $(4n + 2)$-state strongly reversible 2DFA. □

References

[1] C. H. Bennett, "Logical reversibility of computation", *IBM Journal of Research and Development*, 17:6 (1973).

[2] C. H. Bennett, "The thermodynamics of computation—a review", *International Journal of Theoretical Physics*, 21:12 (1982), 905–940.

[3] C. H. Bennett, "Time/space trade-offs for reversible computation", *SIAM Journal on Computing*, 81 (1989), 766–776.

[4] M. Chrobak, "Finite automata and unary languages", *Theoretical Computer Science*, 47 (1986), 149–158. Errata: 302 (2003), 497–498.

[5] P. Crescenzi, C. H. Papadimitriou, "Reversible simulation of space-bounded computations", *Theoretical Computer Science*, 143:1 (1995), 159–165.

[6] V. Geffert, C. Mereghetti, G. Pighizzini, "Complementing two-way finite automata", *Information and Computation*, 205:8 (2007), 1173–1187.

[7] J. Kari, "Reversible cellular automata", *Developments in Language Theory* (DLT 2005, Palermo, Italy, 4–8 July 2005), LNCS 3572, 57–68.

[8] A. Kondacs, J. Watrous, "On the power of quantum finite state automata", *38th Annual Symposium on Foundations of Computer Science* (FOCS 1997, Miami Beach, Florida, USA, 19–22 October 1997), IEEE, 66–75.

[9] M. Kunc, A. Okhotin, "Describing periodicity in two-way deterministic finite automata using transformation semigroups", *Developments in Language Theory* (DLT 2011, Milan, Italy, 19–22 July 2011), LNCS 6795, 324–336.

[10] M. Kunc, A. Okhotin, "State complexity of operations on two-way deterministic finite automata over a unary alphabet", *Descriptional Complexity of Formal Systems* (DCFS 2011, Limburg, Germany, 25–27 July 2011), LNCS 6808, 222–234.

[11] K.-J. Lange, P. McKenzie, A. Tapp, "Reversible space equals deterministic space", *Journal of Computer and System Sciences*, 60:2 (2000), 354–367.

[12] S. Lombardy, "On the construction of reversible automata for reversible languages", *Automata, Languages and Programming* (ICALP 2002, Málaga, Spain, 8–13 July 2002), LNCS 2380, 170–182.

[13] J.-E. Pin, "On the languages accepted by finite reversible automata", *Automata, Languages and Programming* (ICALP 1987, Karlsruhe, Germany, 13–17 July 1987), LNCS 267, 237–249.

[14] M. Sipser, "Lower bounds on the size of sweeping automata", *STOC 1979*, 360–364.

Turku
Centre *for*
Computer
Science

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematical Sciences

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration
- Institute of Information Systems Sciences

ISBN 978-952-12-2661-8
ISSN 1239-1891