Reversible and Irreversible Computations of Deterministic Finite-State Devices

Martin Kutrib(✉)

Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany
kutrib@informatik.uni-giessen.de

Abstract. Finite-state devices with a read-only input tape that may be equipped with further resources as queues or pushdown stores are considered towards their ability to perform reversible computations. Some aspects of the notion of logical reversibility are addressed. We present some selected results on the decidability, uniqueness, and size of minimal reversible deterministic finite automata. The relations and properties of reversible automata that are equipped with storages are discussed, where we exemplarily stick with the storage types queue and pushdown store. In particular, the computational capacities, decidability problems, and closure properties are the main topics covered, and we draw attention to the overall picture and some of the main ideas involved.

Keywords: Reversibility · Finite state devices · Minimality · Queue and pushdown storage · Decidability · Closure properties

1 Introduction

Reversibility is a practically motivated property that has been investigated for several automata models. Computers can be seen as information processing devices which are physical realizations of such abstract models. From this viewpoint it is natural to study the fundamental physical principle of reversibility, which means in essence that every configuration has at most one unique successor configuration and at most one unique predecessor configuration. Moreover, in [29] it has been argued that only the logically irreversible operations in a computer necessarily dissipate energy by generating a corresponding amount of entropy for every irreversibly erased bit of information. This observation strongly suggests to study reversible computations without loss of information. A main question in this setting is whether or not the computation of a given automaton model can be made reversible in general.

The first investigations of reversible computations date back to the sixties of the last century when the massively parallel model of cellular automata was studied in this respect. It has been shown that the injectivity of the global transition function is equivalent to the reversibility of the automaton. It turned out that global reversibility is decidable for one-dimensional cellular automata [1], whereas the problem is undecidable for higher dimensions [16]. Nowadays it

G.F. Italiano et al. (Eds.): MFCS 2015, Part I, LNCS 9234, pp. 38–52, 2015.
DOI: 10.1007/978-3-662-48057-1_3

is known from [33] that every, possibly irreversible, one-dimensional cellular automaton can always be simulated by a reversible one-dimensional cellular automaton in a constructive way.

Later, in [6] reversible sequential machines, more precisely, Turing machines have been studied. Again, a fundamental result is that every Turing machine can be made reversible or, in other words, that any recursively enumerable language can be accepted in a reversible way. Given this result, the question for the efficiency of such a simulation almost suggests itself. Let the irreversible computation take t time and s space. In [7] a first efficient reversible simulation is proposed that uses $s \cdot t^{\log(3)}$ time and $s \cdot \log(t)$ space. So, for maximal t it uses s^2 space. In [30] a different method has been shown that uses only $O(s)$ space but at the cost of exponential time. In [9] a general upper bound on the tradeoff between time and space that suffices for the reversible simulation of irreversible computations is proved. It has the exponential time simulation and the quadratic space simulation as extremes. The result shows that it is possible to achieve subexponential time and subquadratic space simultaneously.

Valuable surveys with further references to literature are, for example, [17] for cellular automata and [34], where one may find a summary of results on reversible Turing machines, reversible cellular automata, and other reversible models such as logic gates, logic circuits, or logic elements with memory, and [20] for further aspects of reversibility (see also [4,21,22,25] for further investigations).

Logical reversibility has been studied also for further models such as time-bounded Turing machines [5], two-way multi-head finite automata [3,35], one-way multi-head finite automata [24], queue automata [26], and limited automata [27].

Here we consider some aspects of reversibility in sequential devices that have a read-only input tape and may be equipped with further storage resources. The discussion is mainly restricted exemplarily to finite automata as well as to queue and pushdown automata. The notion of reversibility and its possible definitions are discussed. Then we turn to the size of reversible finite automata. It is well known that the minimal DFA accepting a given regular language is unique up to isomorphism. So the relations between minimality and reversibility are of natural interest, in particular, questions concerning the decidability, uniqueness, and the size of a minimal reversible DFA in terms of the size of the equivalent minimal DFA. Finally, the relations and properties of reversible automata that are equipped with a pushdown store or a queue are discussed, where the computational capacities, decidability problems, and closure properties are the main topics.

2 Preliminaries and the Notion of Logical Reversibility

The reader is assumed to be familiar with the basic notions of automata theory as contained, for example, in [12,15]. In the present paper we will use the following notational conventions. An *alphabet* Σ is a non-empty finite set, its elements are called *letters* or *symbols*. We write Σ^* for the *set of all words* over the finite

alphabet Σ. The *empty word* is denoted by λ, and $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. The *reversal* of a word w is denoted by w^R and for the *length* of w we write $|w|$. We use $\subseteq$ for *inclusions* and $\subset$ for *strict inclusions*. In the following, two devices are said to be *equivalent* if they accept the same language.

The devices we are interested in are computing machines with a finite number of discrete internal states. The machines have a read-only input tape, may be equipped with further resources, and evolve in discrete time, where each computation step is driven by a *deterministic transition function*. Given a *configuration* representing the complete "global state" of a device, the transition function is used to compute the successor configuration. The transition function depends on the current internal state and on the status of further resources the machine is equipped with. It gives the successor state and maybe changes the status of the resources. In general, these devices are considered in terms of formal language recognition. However, reversibility is a property of machines and not a property of languages. So, notions as "the family of reversible regular languages" are meaningless unless the reversibility of a regular language is defined by the reversibility of a certain type of device that accepts it. For example, the deterministic finite automaton (DFA) depicted in Fig. 1 accepts the regular language a^*bb^*. Since any equivalent DFA must have a state with two incoming edges which are labeled by the same input symbol, it cannot be reversible. So, from the viewpoint of (deterministic) finite automata the language a^*bb^* is irreversible. However, in [19] it has been shown that reversible *two-way* deterministic finite automata characterize the regular languages. This implies that every regular language is accepted by some reversible two-way deterministic finite automaton and, thus, from the viewpoint of (deterministic) two-way finite automata the language a^*bb^* is reversible.

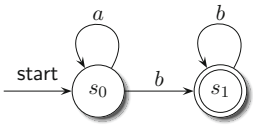

Fig. 1. An irreversible DFA accepting the language a^*bb^*, that cannot be accepted by any reversible DFA.

Basically, the definition of logical reversibility of some type of device requires that the device is deterministic and that any configuration must have at most one predecessor. But these requirements do not define reversibility sufficiently. For example, in which way is the predecessor configuration computed? May we use a universal device? Do we have to use a device of the same type? Or else a device with the same computational power? The idea to step the computation back and forth anticipates not to use a universal machine in general. For the latter question consider again the DFA of Fig. 1 that accepts the language a^*b^+. If the predecessor configuration has to be computed by a DFA as well, the given DFA

is irreversible. Once in state s_1, it is impossible to get back uniquely on input symbol b. However, if a DFA is equipped with an input window of size two, it can compute the predecessor state. Moreover, deterministic finite automata with window size two have the same power as DFA. So, if the predecessor configuration may be computed with lookahead two, the given DFA is reversible. Further results on gradual reversibility dependent on the size of lookaheads can be found in [2,28].

Another question that comes up in connection with the computability of predecessor configurations concerns the set of configurations that count. Do we have to consider all possible configurations as potential predecessors? Or only configurations that are reachable from some initial configurations, that is, configurations that actually occur in computations? Consider for example the DFA in Fig. 2. It is reversible for all reachable configurations, but it is irreversible if all possible configurations count. Reversibility on reachable configurations is a wider notion than reversibility on all configurations. It turns out that this makes a big difference if *machines* are considered, but it does not make any difference if *languages* are considered (see Sect. 4).

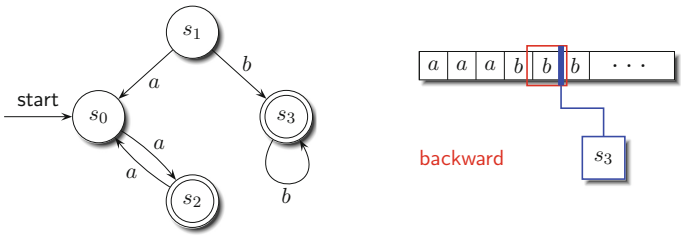

Fig. 2. [20] A DFA (left) and an unreachable configuration (right).

Unless stated otherwise, in the sequel we tacitly make the appointment that the backward steps of a computation are performed by another device of the same type and that all configurations count.

3 Size of Reversible Finite Automata

It is well known that the minimal DFA accepting a given regular language is unique up to isomorphism. So the relations between minimality and reversibility are of natural interest, in particular, questions concerning the decidability, uniqueness, and the size of a minimal reversible DFA in terms of the size of the equivalent minimal DFA.

Before we turn to the discussion of these relations, we recall some definitions. A *deterministic finite automaton* (DFA) is a system $M = \langle S, \Sigma, \delta, s_0, F \rangle$, where S is the finite set of *internal states*, Σ is the alphabet of *input symbols*, $s_0 \in S$ is the

initial state, $F \subseteq S$ is the set of *accepting states*, and $\delta\colon S \times \Sigma \to S$ is the partial *transition function*. Note, that here the transition function is not required to be *total*. By $\delta^{\leftarrow}\colon S \times \Sigma \to 2^S$, with $\delta^{\leftarrow}(q,a) = \{\, p \in S \mid \delta(p,a) = q \,\}$, we denote the *reverse* transition function of δ.

A DFA is *reversible* if every letter $a \in \Sigma$ induces an *injective partial mapping* from S to itself *via* the mapping $\delta_a\colon S \to S$ with $p \mapsto \delta(p,a)$. In this case, the reverse transition function $\delta^{\leftarrow}$ can be seen as a (partial) injective function $\delta^{\leftarrow}\colon S \times \Sigma \to S$.

A restricted variant of reversible deterministic finite automata has been introduced and studied in the context of algorithmic learning theory in [2]; see also [18]. The definition there requires that any reversible DFA has only one sole accepting state. Sometimes these devices are called *bideterministic* DFA. For this notion of reversibility the question for uniqueness and the size of a minimal reversible DFA is settled, as a language L is accepted by a bideterministic DFA if and only if the minimal DFA for L is reversible and has a unique final state (see [36]).

Later this concept of reversibility has been extended in [36], so that multiple accepting as well as multiple initial states are allowed. In particular, this means that reversible DFA in this sense are nondeterministic devices. However, even these devices cannot accept the regular language a^*b^* reversibly [36]. A further generalization of reversibility to quasi-reversibility, which even allows nondeterministic transitions was introduced in [32] (see also [11]). However, these quasi-reversible DFA may be exponentially more succinct than the minimal reversible DFA.

Next we turn to discuss the question for decidability, uniqueness, and the size of a minimal reversible DFA in the standard definition from above.

The example depicted in Fig. 3 answers the question whether a minimal reversible DFA is unique.

Fig. 3. Non-isomorphic minimal reversible DFA for the finite language $L = \{aa, ab, ba\}$.

Theorem 1 ([14]). *Let L be a regular language accepted by some reversible DFA. Then a minimal reversible DFA accepting L is not necessarily unique, even not up to isomorphism.*

3.1 Trade-Offs

The first exponential lower bound for the state trade-off between a minimal DFA and an equivalent minimal reversible DFA originates in [13]. By the $2n$-fold concatenation L^{2n} of the finite language $L = \{aa, ab, ba\}$ a lower bound of $\Omega(1.001^n)$ has been derived. So, the minimal reversible finite automaton for some language can be exponentially larger than the minimal automaton. In [14] the exact number of states of a minimal reversible DFA for the language L^{2n} has been shown; it is $2^{2n+2} - 3$. Since the minimal DFA for L^{2n} has $6n + 1$ states, the blow-up in the number of states is in the order of $2^{n/3} = (\sqrt[3]{2})^n$, which is approximately 1.259^n.

However, in the same paper [14] the lower bound has been improved. A witness DFA is depicted in Fig. 4. Notice that no transitions are defined from the sole accepting state. Clearly, the DFA is minimal, but not reversible. However, since the language accepted is finite, one readily sees that it can be accepted by a reversible DFA. It is shown that the number of states of the minimal equivalent reversible DFA is $\sum_{i=1}^{n} F_n$, where F_n denotes the nth Fibonacci number. This is equal to $F_{n+2} - 1$. From the closed form

$$F_n = \frac{1}{\sqrt{5}} \cdot \left(\frac{1+\sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \cdot \left(\frac{1-\sqrt{5}}{2} \right)^n$$

and the fact that $\left(\frac{1-\sqrt{5}}{2}\right)^n$ tends to zero, for large n, we see that the state blow-up is in the order of $\left(\frac{1+\sqrt{5}}{2}\right)^n$, that is, approximately 1.618^n, the golden ratio Φ to the power of n.

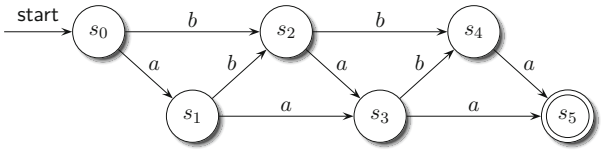

Fig. 4. [14] A minimal DFA, for $n = 6$ states, where the minimal equivalent reversible DFA needs $\sum_{i=1}^{n} F_i = F_{n+2} - 1$ states.

Theorem 2 ([14]). *For every n with $n \geq 3$ there is an n-state DFA over a binary input alphabet accepting a reversible language, such that any equivalent reversible DFA needs at least $\Omega(\Phi^n)$ states with $\Phi = (1+\sqrt{5})/2$, the golden ratio.*

It is worth mentioning that the lower bound for the witness languages of Theorem 2 is for a binary alphabet. It can be increased at the cost of more symbols. For a k-ary alphabet one can derive the lower bound from the k-ary Fibonacci function $F_n = F_{n-1} + F_{n-2} + \cdots + F_{n-k}$. For $k = 3$ the lower bound is of order 1.839^n and for $k = 4$ it is of order 1.927^n. For growing alphabet sizes the bound asymptotically tends to 2^{n-1}, that is, $\Omega(2^{n-1})$. This is precisely the upper bound (for arbitrary alphabet sizes).

Theorem 3 ([14]). *Let M be a minimal deterministic finite automaton with n states, that accepts a reversible language. Then a minimal reversible deterministic finite automaton for $L(M)$ has at most 2^{n-1} states.*

3.2 Decidability

Now we turn to decidability questions in connection with (minimal) reversible DFA. The first problem that comes into mind is the problem to decide whether a given DFA is reversible or, more involved, whether it accepts a *language* that is also accepted by some reversible DFA. The decision of the reversibility of DFA is almost trivial. An inspection of the transition function and the set of accepting states suffices. Moreover, this observation transfers also to languages accepted by bideterministic automata in the notion of [2], because it is sufficient to verify the reversibility of the minimal DFA for the language, which must have a unique final state (see remark above). For languages accepted by nondeterministic DFA, where the nondeterminism is limited to multiple initial states, it has been shown in [36], that there is a polynomial time algorithm for testing whether the language can be accepted by a reversible finite automaton.

For DFA the problem has been solved in [14] by proving the following structural characterization of regular languages that can be accepted by reversible DFA in terms of their minimal DFA.

Theorem 4 ([14]). *Let $M = \langle S, \Sigma, \delta, s_0, F\rangle$ be a minimal deterministic finite automaton. The language $L(M)$ can be accepted by a reversible deterministic finite automaton if and only if there do not exist useful states $p, q \in S$, a letter $a \in \Sigma$, and a word $w \in \Sigma^*$ such that $p \neq q$, $\delta(p, a) = \delta(q, a)$, and $\delta(q, aw) = q$.*

So, the characterization is based on the absence of a forbidden pattern in the (minimal) deterministic state graph. Now, checking the absence of the forbidden patterns yields an NL-complete decidability algorithm. The idea of proving NL containment is to decide in NL whether a given DFA accepts a *non-reversible* language by witnessing the forbidden pattern. Since NL is closed under complementation the containment of the reversibility problem within NL follows. For the NL hardness, the NL-complete graph reachability problem is reduced to the problem in question (with respect to deterministic logspace reductions).

Theorem 5 ([14]). *Given a DFA M, the problem to decide whether $L(M)$ is accepted by any reversible DFA is* NL*-complete.*

Another interesting decidability problem is to determine whether a given reversible DFA is already minimal. Again with a forbidden pattern approach, it is shown in [14] that the minimality of reversible DFA can be decided by an NL-complete algorithm.

Theorem 6 ([14]). *Given a DFA M, the problem to decide whether M is already a minimal reversible deterministic finite automaton is* NL*-complete.*

A further result in [14] is the *effective construction* of a minimal reversible DFA out of a given DFA that accepts a reversible language. The basic idea how to make a given DFA reversible is very intuitive: as long as there is an irreversible state, copy this state and all states reachable from it, and distribute the incoming transitions to the new copies. The absence of the forbidden pattern ensures that this procedure eventually comes to an end.

4 Queues and Pushdown Stores

This section is devoted to discuss relations and properties of reversible automata that are equipped with a pushdown store (DPDA) or a queue (DQA). Their reversible variants have been introduced and studied in [23] and [26], where only reachable configurations are relevant for reversibility. However, for pushdown automata and queue automata this makes a difference only for *machines*, that is, there are machines that are reversible on reachable but not on all configurations. It does not make a difference for *languages*, that is, for any machine that is reversible on reachable configurations there is an equivalent machine that is reversible on all configurations. So, from the perspective of languages and language classes it is safe to stick with either notion of reversibility.

Recall that a queue automaton, at each time step, may remove or keep the symbol at the front and enters a (possibly empty) symbol at the end of the queue. The transition depends on the current state, the current input symbol or λ, and the symbols currently at the front and end of the queue. Often queue automata are defined that can only see the symbol at the front of the queue. However, with an eye towards reversible computations we extend the definition as described. It is worth mentioning that the additional knowledge of the last queue symbol does not increase the computational power of queue automata. For reverse computation steps the head of the input tape is again moved to the *left*. Moreover, the roles played by the front and end of the queue are interchanged. That is, in reverse computation steps the symbols are removed from the end and added to the front of the queue. We denote the relation from one configuration to the next by $\vdash$.

A DQA M with transition function δ is said to be reversible (REV-DQA), if there exists a reverse transition function $\delta^{\leftarrow}$ inducing a relation $\vdash^{\leftarrow}$ from one configuration to the next, so that $c' \vdash^{\leftarrow} c$ if and only if $c \vdash c'$, for any two configurations c, c' of M (Fig. 5). See [26] for detailed definitions.

The following example is interesting insofar as the language is known not to be accepted by any reversible pushdown automaton.

Example 7 ([26]). The deterministic linear context-free language $\{ a^n b^n \mid n \geq 1 \}$ is accepted by the quasi realtime REV-DQA with state set $\{q_0, q_1, q_2\}$, queue alphabet $\{A_0, B_0, B_1\}$, initial state q_0, set of accepting states $\{q_2\}$, and empty queue symbol $\perp$, where the transition functions δ and $\delta^{\leftarrow}$ are as follows. Let $X \in \{A_0, B_0, B_1\}$.

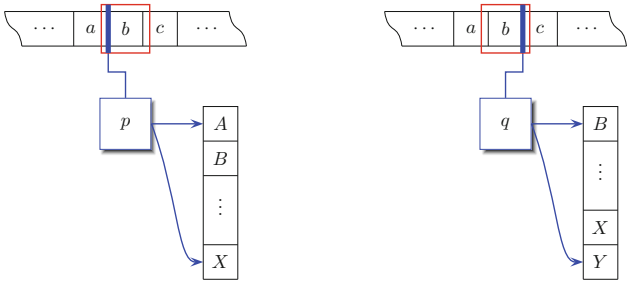

Fig. 5. Successive configurations of a REV-DQA, where $\delta(p, b, A, X) = (q, Y, \text{remove})$ (left to right) and $\delta^{\leftarrow}(q, b, B, Y) = (p, A, \textbf{remove})$ (right to left).

Transition function δ	Reverse transition function $\delta^{\leftarrow}$
(1) $\delta(q_0, a, \bot, \bot) = (q_0, A_0, \texttt{keep})$	(1) $\delta^{\leftarrow}(q_0, a, A_0, A_0) = (q_0, \lambda, \texttt{remove})$
(2) $\delta(q_0, a, A_0, X) = (q_0, A_0, \texttt{keep})$	
(3) $\delta(q_0, b, A_0, X) = (q_1, B_0, \texttt{remove})$	(2) $\delta^{\leftarrow}(q_1, b, X, B_0) = (q_0, A_0, \texttt{remove})$
(4) $\delta(q_1, b, A_0, X) = (q_1, B_1, \texttt{remove})$	(3) $\delta^{\leftarrow}(q_1, b, X, B_1) = (q_1, A_0, \texttt{remove})$
(5) $\delta(q_1, \lambda, B_0, B_0) = (q_2, \lambda, \texttt{keep})$	(4) $\delta^{\leftarrow}(q_2, \lambda, B_0, B_0) = (q_1, \lambda, \texttt{keep})$
(6) $\delta(q_1, \lambda, B_0, B_1) = (q_2, \lambda, \texttt{keep})$	(5) $\delta^{\leftarrow}(q_2, \lambda, B_0, B_1) = (q_1, \lambda, \texttt{keep})$

The main idea of the construction is to provide two flags for the enqueued symbols, namely the letter and its index. The letters characterize whether the machine is in mode A or mode B. The machine is in mode A while reading a's from the input and is in mode B when reading b's and checking whether the number of a's and b's coincide. The index describes in which state the machine was in its last step. This information is necessary for the reverse transition function.

In order to obtain a reversible automaton some transition rules have to be provided that cannot be used in any reachable configuration. For example, let the REV-DQA perform the transition $\delta(q_0, a, A_0, A_0) = (q_0, A_0, \text{keep})$. Then the reverse transition rule $\delta^{\leftarrow}(q_0, a, A_0, A_0) = (q_0, \lambda, \text{remove})$ has to be defined. However, applying this rule to the unreachable configuration, where the queue content is $A_0B_1A_0$ gives the predecessor configuration with queue content A_0B_1. This implies that the (forward) transition rule $\delta(q_0, a, A_0, B_1) = (q_0, A_0, \text{keep})$ has to be defined as well. ■

For the sake of completeness, recall that the transition function of a deterministic pushdown automaton maps the current state, the current input symbol or λ, and the symbol at the top of the stack to the successor state and a new (possibly empty) string at the top of the stack.

A DPDA M with transition function δ is said to be reversible (REV-DPDA), if there exists a reverse transition function $\delta^{\leftarrow}$ inducing a relation $\vdash^{\leftarrow}$ from one configuration to the next, so that $c' \vdash^{\leftarrow} c$ if and only if $c \vdash c'$, for any two configurations c, c' of M. See [23] for detailed definitions.

4.1 Computational Capacity

In order to explore the general computational capacity of reversible queue automata, the next result has been shown in [26]. It contrasts the situation for pushdown automata and complements the situation for Turing machines, where every Turing machine can be made reversible [6].

Theorem 8 ([26]). *Let M be a deterministic queue automaton. Then there exists a reversible queue automaton accepting the language $L(M)$.*

Combining this construction with the result from [37] that says that queue automata without any time restriction describe the recursively enumerable languages, we obtain that reversible queue automata and Turing machines have the same computational power.

Theorem 9 ([26]). *Every recursively enumerable language is accepted by some reversible queue automaton.*

Giving a queue automaton arbitrary time for the computation may be a little unfair compared with pushdown automata, because the former can always cycle through the queue, thus, reading the whole storage content without destroying any information. A natural possibility to overcome this advantage is studied in [10] where queue automata are considered that work in quasi realtime. Quasi realtime means that the number of consecutive λ-transitions is bounded by a constant. It is shown in [10] that quasi realtime queue automata are less powerful than queue automata without time restriction. However, for reversible queue as well as pushdown automata there is the nice correspondence that both types of automata if working in quasi realtime can be sped up to realtime.

Theorem 10 ([26]). *For every quasi realtime reversible DQA an equivalent realtime reversible DQA can effectively be constructed.*

Theorem 11 ([23]). *For every reversible DPDA an equivalent realtime reversible DPDA can effectively be constructed.*

Though both types of reversible devices have strictly less power when restricted to (quasi) realtime computations, they still can accept all regular languages. The idea is simply to simulate a given DFA, whereby the state history is remembered in the storage.

On the other hand, any language known not to be accepted in realtime by a pushdown automaton is a witness for the fact that reversible pushdown automata are strictly weaker than the general pushdown automata. This raises the natural question whether *all* realtime DPDA languages are accepted by reversible DPDA. This question has been answered negatively.

Theorem 12 ([23]). *The realtime deterministic linear context-free language $\{ a^n b^n \mid n \geq 0 \}$ is not accepted by any reversible DPDA.*

A similar result for queue automata has been established in [26]. In particular, a language L_{mcp} is exhibited which is accepted by some realtime queue automaton, but not by any realtime reversible queue automaton. In fact, the stronger result is obtained that any reversible queue automaton accepting L_{mcp} takes at least $\Omega\left(\frac{n^2}{\log(n)}\right)$ time steps.

Example 13 ([26]). We consider the regular language $L_{bin} = ((aa+a)(bb+b))^+$. Then the language

$$L_{mcp} = \{\, p\$w_1\$w_1\$w_2\$w_2\$\cdots\$w_n\$w_n \mid p \in L_{bin}, n \geq 0, w_i \in \{a,b\}^* \,\}$$

is accepted by a realtime DQA. Informally, a DQA works as follows. The prefix up to the first $ can be tested without using the queue, because it belongs to a regular language. Then the first copy of each w_i is stored in the queue and subsequently compared and removed from the queue while reading the second copy. Whenever the second copy matches the first copy, the automaton enters an accepting state. Whenever a mismatch is detected, a non-accepting state is entered so that the input is rejected. ■

By using Kolmogorov complexity and incompressibility arguments, the lower bound mentioned above has been shown.

Theorem 14 ([26]). *Any reversible DQA accepting L_{mcp} has a time complexity of $\Omega\left(\frac{n^2}{\log(n)}\right)$.*

This lower bound result raises the question for the costs of simulating any realtime DQA, not necessarily reversible, by an equivalent reversible DQA. It turned out that quadratic time is sufficient for such simulations.

Theorem 15 ([26]). *Every realtime DQA can be simulated by a reversible DQA that needs at most quadratic time.*

This upper bound shows that quadratic time is the trade-off for making realtime DQA reversible. On the other hand, the language L_{mcp} provides a lower bound, which shows that there are cases where a quadratic time trade-off is almost reached. Moreover, we have derived that for both types of automata the reversible variant is strictly weaker than the realtime general variant.

We conclude the subsection by an incomparability result showing that the language accepting capabilities of reversible (quasi) realtime DQA are different from those of reversible pushdown automata. The context-free language $\{\, w\#w^R \mid w \in \{a,b\}^+ \,\}$ is accepted by a reversible pushdown automaton [23], but not by any even irreversible quasi realtime queue automaton [8,31]. On the other hand, it is not hard to see that the non-context-free language $\{\, a^n b^n c^n \mid n \geq 1 \,\}$ is accepted by some realtime reversible DQA.

Theorem 16 ([26]). *The families of languages accepted by realtime reversible DQA and by reversible pushdown automata are incomparable.*

4.2 Decidability and Closure Properties

Let us now turn to decidability aspects of reversible pushdown and queue automata. Problems which are decidable for DPDA are decidable for reversible DPDA as well.

Corollary 17. *Finiteness, infiniteness, universality, equivalence, and regularity are decidable for reversible DPDA.*

On the other hand, inclusion is known to be undecidable for DPDA. By reduction of the Post's correspondence problem it has been shown that inclusion is undecidable for reversible DPDA, too.

Theorem 18 ([23]). *Inclusion is undecidable for reversible DPDA.*

The situation for reversible queue automata is in considerable contrast to the situation for reversible pushdown automata. By reduction of the emptiness problem for deterministic linearly space bounded one-tape, one-head Turing machines, so-called linear bounded automata, it has been shown that the emptiness problem for realtime reversible DQA is not even semidecidable. From this result it is derived that all the commonly studied decidability questions are non-semidecidable, too.

Theorem 19 ([26]). *Emptiness, finiteness, infiniteness, universality, inclusion, equivalence, regularity, and context-freeness are not semidecidable for realtime reversible DQA.*

These results bring us to the problem whether reversibility itself is decidable. Here again, we have to distinguish between the reversibility of a machine and the reversibility of the language accepted by a machine. Let us first shortly consider the languages. The following theorem contrasts the situation for finite automata, where the problem is decidable (see above).

Theorem 20 ([23]). *It is undecidable whether the language accepted by a nondeterministic pushdown automaton can be accepted by a reversible DPDA.*

The same problem for deterministic pushdown automata is open.

Next, we consider the question of whether the reversibility of a machine in question can be decided. Now we have to distinguish between the reversibility on all or only on reachable configurations. This makes a big difference. While the question of reversibility on *all configurations* is decidable just by by inspection of the transition function, the question of reversibility on reachable configurations becomes more involved. In particular, we obtain a difference between queue and pushdown automata.

Theorem 21 ([26]). *Reversibility on reachable configurations is not semidecidable for realtime DQA.*

However, we have the decidability for pushdown automata. The size of a pushdown automaton is the length of its representation.

Theorem 22 ([23]). *Let M be a deterministic pushdown automaton of size n. Then it is decidable in time $O(n^4)$, whether M is reversible on reachable configurations. Moreover, the decision problem is P-complete.*

Given a nondeterministic pushdown automaton, by inspecting the transition function one can decide whether or not it is a DPDA. If the answer is yes, then it can be decided whether it is reversible on reachable configurations by the previous theorem. If it is not a DPDA, then it cannot be a reversible DPDA. Therefore, the previous result transfers to nondeterministic devices.

Corollary 23 ([23]). *Let M be a nondeterministic pushdown automaton of size n. Then it is decidable in time $O(n^4)$, whether M is reversible on reachable configurations. Moreover, the decision problem is P-complete.*

Finally, we consider closure properties of the families in question. It turns out that the families of languages accepted by realtime reversible queue automata and reversible pushdown automata have similar closure properties. For example, they are closed under complementation and inverse homomorphism, but are not closed under union, intersection, intersection with regular languages, concatenation, reversal, and homomorphism. The closure properties of the language families discussed are summarized in Table 1. The closure properties for general (realtime) DQA may be found in [10].

Table 1. Closure properties of language families discussed. REG denotes the family of regular languages.

Language class	$\cup$	$\bullet$	R	h_λ	h^{-1}	$\cap_{\mathrm{REG}}$	$\cup_{\mathrm{REG}}$	$\cap$	$\sim$
REG	+	+	+	+	+	+	+	+	+
realtime REV-DQA	−	−	−	−	+	−	−	−	+
realtime DQA	−	−	−	−	+	+	+	−	+
REV-DPDA	−	−	−	−	+	−	−	−	+
DPDA	−	−	−	−	+	+	+	−	+

References

1. Amoroso, S., Patt, Y.N.: Decision procedures for surjectivity and injectivity of parallel maps for tesselation structures. J. Comput. Syst. Sci. **6**, 448–464 (1972)
2. Angluin, D.: Inference of reversible languages. J. ACM **29**, 741–765 (1982)
3. Axelsen, H.B.: Reversible multi-head finite automata characterize reversible logarithmic space. In: Dediu, A.-H., Martín-Vide, C. (eds.) LATA 2012. LNCS, vol. 7183, pp. 95–105. Springer, Heidelberg (2012)
4. Axelsen, H.B., Glück, R.: A simple and efficient universal reversible turing machine. In: Dediu, A.-H., Inenaga, S., Martín-Vide, C. (eds.) LATA 2011. LNCS, vol. 6638, pp. 117–128. Springer, Heidelberg (2011)

5. Axelsen, H.B., Jakobi, S., Kutrib, M., Malcher, A.: A hierarchy of fast reversible turing machines. In: Krivine, J., Stefani, J.B. (eds.) Reversible Computation (RC 2015). LNCS, vol. 9138, pp. 29–44. Springer, Heidelberg (2015)
6. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**, 525–532 (1973)
7. Bennett, C.H.: Time/space trade-offs for reversible computation. SIAM J. Comput. **18**, 766–776 (1989)
8. Brandenburg, F.J.: Intersections of some families of languages. In: Kott, L. (ed.) International Colloquium on Automata, Languages and Programming (ICALP 1986). LNCS, vol. 226, pp. 60–68. Springer, Heidelberg (1986)
9. Buhrman, H., Tromp, J., Vitányi, P.M.B.: Time and space bounds for reversible simulation. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 1017–1027. Springer, Heidelberg (2001)
10. Cherubini, A., Citrini, C., Crespi-Reghizzi, S., Mandrioli, D.: QRT FIFO automata, breadth-first grammars and their relations. Theoret. Comput. Sci. **85**, 171–203 (1991)
11. García, P., de Parga, M.V., López, D.: On the efficient construction of quasi-reversible automata for reversible languages. Inform. Process. Lett. **107**, 13–17 (2008)
12. Harrison, M.A.: Introduction to Formal Language Theory. Addison-Wesley, Reading (1978)
13. Héam, P.C.: A lower bound for reversible automata. RAIRO Inform. Théor. **34**, 331–341 (2000)
14. Holzer, M., Jakobi, S., Kutrib, M.: Minimal reversible deterministic finite automata. In: Potapov, I. (ed.) Developments in Language Theory (DLT 2015). LNCS, Springer (to appear 2015)
15. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Boston (1979)
16. Kari, J.: Reversibility and surjectivity problems of cellular automata. J. Comput. Syst. Sci. **48**, 149–182 (1994)
17. Kari, J.: Reversible cellular automata. In: De Felice, C., Restivo, A. (eds.) DLT 2005. LNCS, vol. 3572, pp. 57–68. Springer, Heidelberg (2005)
18. Kobayashi, S., Yokomori, T.: Learning approximately regular languages with reversible languages. Theoret. Comput. Sci. **174**, 251–257 (1997)
19. Kondacs, A., Watrous, J.: On the power of quantum finite state automata. In: Foundations of Computer Science (FOCS 1997), pp. 66–75. IEEE Computer Society (1997)
20. Kutrib, M.: Aspects of reversibility for classical automata. In: Calude, C.S., Freivalds, R., Kazuo, I. (eds.) Gruska Festschrift. LNCS, vol. 8808, pp. 83–98. Springer, Heidelberg (2014)
21. Kutrib, M., Malcher, A.: Fast reversible language recognition using cellular automata. Inform. Comput. **206**, 1142–1151 (2008)
22. Kutrib, M., Malcher, A.: Real-time reversible iterative arrays. Theoret. Comput. Sci. **411**, 812–822 (2010)
23. Kutrib, M., Malcher, A.: Reversible pushdown automata. J. Comput. Syst. Sci. **78**, 1814–1827 (2012)
24. Kutrib, M., Malcher, A.: One-way reversible multi-head finite automata. In: Glück, R., Yokoyama, T. (eds.) RC 2012. LNCS, vol. 7581, pp. 14–28. Springer, Heidelberg (2013)

25. Kutrib, M., Malcher, A., Wendlandt, M.: Real-time reversible one-way cellular automata. In: Isokawa, T., Imai, K., Matsui, N., Peper, F., Umeo, H. (eds.) AUTOMATA 2014. LNCS, vol. 8996, pp. 56–69. Springer, Heidelberg (2015)
26. Kutrib, M., Malcher, A., Wendlandt, M.: Reversible queue automata. In: Bensch, S., Freund, R., Otto, F. (eds.) Non-Classical Models of Automata and Applications (NCMA 2014), vol. 304, pp. 163–178. Austrian Computer Society, Vienna (2014). www.books@ocg.at
27. Kutrib, M., Wendlandt, M.: Reversible limited automata. In: Machines, Computations, and Universality (MCU 2015). LNCS, Springer (to appear, 2015)
28. Kutrib, M., Worsch, T.: Degrees of reversibility for DFA and DPDA. In: Yamashita, S., Minato, S. (eds.) RC 2014. LNCS, vol. 8507, pp. 40–53. Springer, Heidelberg (2014)
29. Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5**, 183–191 (1961)
30. Lange, K.J., McKenzie, P., Tapp, A.: Reversible space equals deterministic space. J. Comput. Syst. Sci. **60**, 354–367 (2000)
31. Li, M., Longpré, L., Vitányi, P.M.B.: The power of the queue. SIAM J. Comput. **21**, 697–712 (1992)
32. Lombardy, S.: On the construction of reversible automata for reversible languages. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, p. 170. Springer, Heidelberg (2002)
33. Morita, K.: Reversible simulation of one-dimensional irreversible cellular automata. Theoret. Comput. Sci. **148**, 157–163 (1995)
34. Morita, K.: Reversible computing and cellular automata - a survey. Theoret. Comput. Sci. **395**, 101–131 (2008)
35. Morita, K.: Two-way reversible multi-head finite automata. Fund. Inform. **110**, 241–254 (2011)
36. Pin, J.E.: On reversible automata. In: Simon, I. (ed.) LATIN 1992. LNCS, vol. 583, pp. 401–416. Springer, Heidelberg (1992)
37. Vollmar, R.: Über einen Automaten mit Pufferspeicherung. Computing **5**, 57–70 (1970)