Aspects of Reversibility for Classical Automata

Martin Kutrib(✉)

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

Abstract. Some aspects of logical reversibility for computing devices with a finite number of discrete internal states are addressed. These devices have a read-only input tape, may be equipped with further resources, and evolve in discrete time. The reversibility of a computation means in essence that every configuration has a unique successor configuration and a unique predecessor configuration. The notion of reversibility is discussed. 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? Do we have to consider all possible configurations as potential predecessors? Or only configurations that are reachable from some initial configurations? We present some selected aspects as gradual reversibility and time-symmetry as well as results on the computational capacity and decidability mainly of finite automata and pushdown automata, and draw attention to the overall picture and some of the main ideas involved.

1 Introduction

Computers are information processing devices which are physical realizations of abstract computational models. So, it is interesting to know whether an abstract model is able to obey physical laws. Since reversibility is a fundamental principle in physics, it is interesting to study the models from this point of view. Moreover, the observation that loss of information results in heat dissipation [26] strongly suggests to study reversible computations without loss of information.

First studies of this kind have been done for the massively parallel model of cellular automata since the sixties of the last century. Nowadays it is known from [29] 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 [4] reversible sequential machines, more precisely, Turing machines have been introduced. Again, a fundamental result is that every Turing machine can be made reversible. These two types of devices received a lot of attention in connection with reversibility. They are beyond the scope of this discussion. Valuable surveys with further references to literature are, for example, [15] for cellular automata and [30], 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 (see also [3,17,18,21] for further investigations). Logical reversibility has

C.S. Calude et al. (Eds.): Gruska Festschrift, LNCS 8808, pp. 83–98, 2014.
DOI: 10.1007/978-3-319-13350-8_7

been studied also for other computational devices such as space-bounded Turing machines [27], two-way multi-head finite automata [2,31], one-way multi-head finite automata [20], and queue automata [22].

Here we focus on some aspects of reversibility in sequential devices, where we mainly restrict the discussion exemplarily to finite automata and pushdown automata. In Section 2 the notion of reversibility and its possible definitions are discussed. In Section 3 the simplest device in question is considered. Gradual reversibility, computational capacity, and decidability of finite automata are the topics presented. The next level in the basic hierarchy of automata are deterministic pushdown automata. Their reversible variants are dealt with in Section 4. Finally, a further aspect of reversibility, the so-called time-symmetry, is discussed in Section 5. Basically, this means that one can go back in time by applying the same transition function as for forward steps after a specific transformation of the phase-space. So, time-symmetric machines themselves cannot distinguish whether they run forward or backward in time.

The reader is assumed to be familiar with the basic notions of automata theory as contained, for example, in [11,14]. 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*. The *family of languages* accepted by devices of type X is denoted by $\mathcal{L}(X)$. In the following, two devices are said to be *equivalent* if they accept the same language.

2 Reversibility of Automata – What Is It?

In general, here we consider 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.

Since we are particularly interested in reversible computations of such devices, we discuss the notion of reversibility first. Basically, reversibility is meant with respect to the possibility of stepping the computation back and forth. To this end, the devices have to be also backward deterministic. That is, any configuration must have at most one predecessor. This simple observation raises several questions.

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? While the idea to step the computation back and forth anticipates not to use a universal machine in general,

Fig. 1. A DFA accepting the language a^*b^+ (left), and its reverse DFA with lookahead two (right). The labels on the edges indicate the *complete* content of the input window (but only 1 symbol is "consumed").

the answer to the latter questions is not that clear. Consider the deterministic finite automaton of Figure 1 that accepts the language a^*b^+. If the predecessor configuration has to be computed by a device of the same type, the DFA is irreversible since there are two different transitions entering the same state s_1 with the same input symbol b (see Figure 2). On the other hand, if the predecessor configuration may be computed by a device with the same computational power, the DFA is reversible. In this case we may provide a lookahead of size two, that is, the input window of the backward DFA has size two while nevertheless only one symbol is processed per time step. The lookahead helps to overcome the crucial situation of the computation at the borderline between the a's and b's (Figure 3). However, a lookahead does not increase the computational capacity of DFA. Such devices still characterize the regular languages only.

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

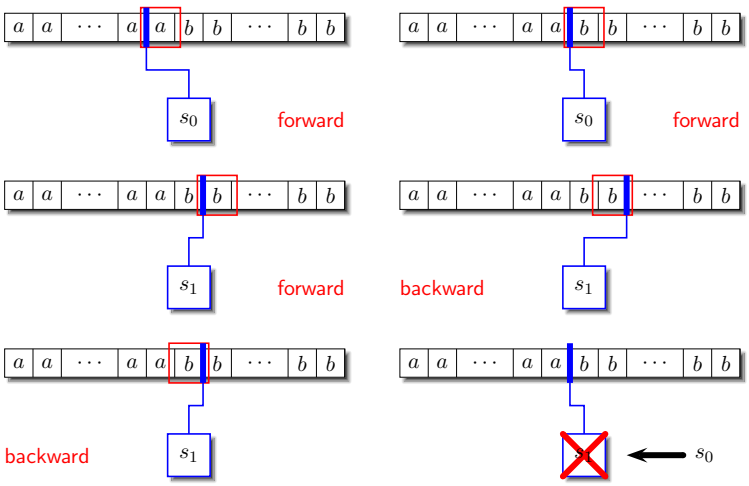

Fig. 2. An irreversible computation of the DFA accepting the language a^*b^+

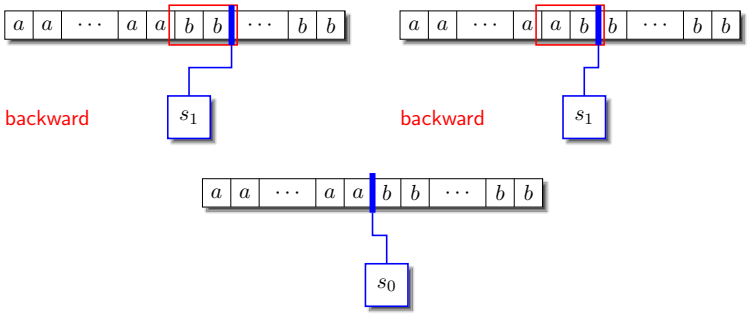

Fig. 3. Backward computation of the DFA with lookahead 2

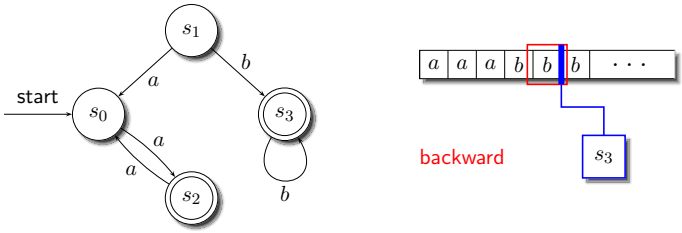

Fig. 4. A DFA (left) and an unreachable configuration (right)

in Figure 4. The configuration on the right-hand side is unreachable from any initial configuration. Is the DFA reversible? It is 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. A gentle argument towards the former notion is given by the big bang theory. If one believes in it everything we are living in evolved from an initial situation.

So, in the sequel we carefully have to distinguish which notion of reversibility is meant. Unless stated otherwise, we require that the backward steps of a computation are performed by another device of the same type.

3 Finite-State Machines

Here we turn to the simplest type of device in question. Reversible deterministic finite automata (REV-DFA) have been introduced and studied in the context of algorithmic learning theory in [1] (see also [16]). Given a DFA M, the inverse $M^{\leftarrow}$ of M is defined by interchanging initial and final states and reversing each transition arrow. In [1], the finite automaton M is defined to be reversible if and only if both M and $M^{\leftarrow}$ are deterministic. Incomplete transition functions are allowed. In particular, this definition implies that for reversible DFA only one final state is allowed. Since there are regular languages that are not accepted

by any DFA with a sole accepting state, by definition, there are non-reversible regular languages in this setting.

The reversible DFA of [1] are called *bideterministic* in [33]. The definition of reversibility has been extended in the latter reference. Now multiple accepting as well as multiple initial states are allowed. So, reversible DFA in the sense of [33] may have limited nondeterminism plugged in from the outside world at the outset of the computation. On the other hand, there are no non-reversible languages per definition any more.

Here we stick with standard definitions. That is, a REV-DFA has a unique initial state and may have multiple accepting states. Essentially, in [33] it has been shown that the regular language a^*b^+ of Figure 1 cannot be accepted by any REV-DFA.

Theorem 1 ([1,33]). *There are regular languages which are not reversible, so there are deterministic finite automata that cannot be simulated by any reversible finite automaton.*

An observation in [1] is that the inherent irreversibility of some regular language may depend on the size of the input window of the devices. If this size is increased *for backward computations*, more languages become reversible. For example, the inherent irreversible language a^*b^+ of Figure 1 is reversibly accepted with lookahead size 2. This result led to the definition of so-called k-reversible languages. In [7] this notion has been generalized to DFA in the definition of [33], and in [24] to pushdown automata (and DFA in standard definition).

We denote DFA with lookahead size $k \geq 1$ by (k)-DFA. So, a classical deterministic finite automaton is a (1)-DFA. A forward DFA is said to be *reversible of degree* k (REV(k)-DFA) if the predecessor configuration is unique for all computations that lead to the current configuration along the last k symbols read in a forward computation. Note, the lookahead for the forward DFA is still 1 (see [24] for formal definitions). So, the lookahead of the backward DFA is used to determine the unique predecessor configuration from all computations that lead to the current configuration along the symbols seen in the input window.

This definition includes also non-reachable configurations. Consider once more the language a^*b^+ from above that is accepted by some REV(2)-DFA. The configuration $(baab, s_1, bb)$ is unreachable in any computation starting from an initial configuration, where $baab$ is the input read so far, s_1 is the current state, and bb is the unread input. The configuration $(aaab, s_1, bb)$ is reachable. Both have two predecessor configurations, namely (baa, s_0, bbb), (baa, s_1, bbb) and (aaa, s_0, bbb), (aaa, s_1, bbb). However, in both cases the predecessor configuration is unique, when the computation comes along the last two input symbols. For the first case we have $(ba, s_0, abbb) \vdash (baa, s_0, bbb) \vdash (baab, s_1, bb)$ and $(ba, s_1, abbb) \nvdash^* (baab, s_1, bb)$.

The next example yields an infinite and strict hierarchy of regular languages dependent on the *degree of reversibility*.

Example 1. Let $k \geq 1$ be an integer. Then the language $\{ a^m b^n \mid m \geq 0, n \geq k \}$ is accepted by some REV$(k+1)$-DFA as indicated in Figure 5. However, the language *cannot* be accepted by any REV(k)-DFA. □

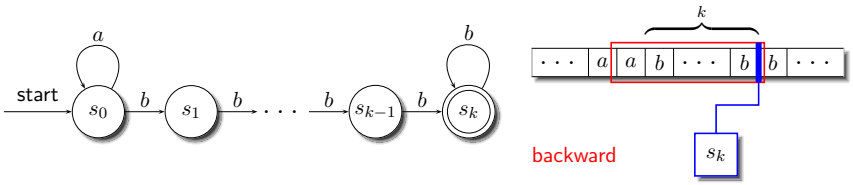

Fig. 5. A REV$(k+1)$-DFA accepting the language $a^* b^k b^*$. This DFA is not a REV(k)-DFA, and there is no other REV(k)-DFA accepting this language.

Theorem 2 ([1,24]). *For any integer $k \geq 1$, there are regular languages accepted by REV$(k+1)$-DFA that cannot be accepted by any REV(k)-DFA.*

So far, it turned out that lookaheads on the input gradually increase the capability to perform reverse computations. Now we are interested in the question whether all regular languages are captured by REV-DFA. Or else, whether there are regular languages that cannot be accepted by any REV-DFA of any degree. For the important subclass of *finite* languages, the answer to the latter question is *no*.

Proposition 1 ([24]). *Any finite language is accepted by some REV(1)-DFA.*

For a second important subclass, the *unary* languages, reversibility is always obtained as well, but the degree for REV-DFA cannot be bounded by any number.

Proposition 2 ([24]). *For any unary regular language L, there is an integer $k \geq 1$ so that L is accepted by some REV(k)-DFA.*

Finally, we consider the general cases where there are languages for which even an arbitrarily large degree cannot help.

Theorem 3 ([24]). *There are regular languages which cannot be accepted by any REV(k)-DFA for any degree $k \geq 1$.*

The idea of the proof is to use a language L over an alphabet Σ that is accepted with lookahead size 2 but cannot be accepted with lookahead size 1, and a regular substitution $s(a) = a\#^*$, for $a \in \Sigma$ and a new symbol #. Language $s(L)$ consists of all words from L with an arbitrary number of # between each two symbols from Σ. Clearly, $s(L)$ is still accepted by some DFA. On the other hand, for any $k \geq 1$, language $s(L)$ contains all words from $s(L) \cap (\Sigma \#^k)^*$. So, when accepting such words there is always at most one symbol of Σ in the

lookahead. Therefore, if $s(L)$ would be reversible for input lookahead size k, a direct construction would show that it is reversible for input lookahead size 1 as well, a contradiction.

Summarizing the results so far, there is an infinite proper hierarchy of k-reversible languages. The union (REV($*$)-DFA) $= \bigcup_{k\geq 1}$ (REV(k)-DFA) of all levels of the hierarchy is properly included in the family of regular languages. In order to justify the power of REV(k)-DFA we compare the union with other well-known subregular language families (see, for example, [5,12] for further results and references on subregular language families). It turned out that (REV($*$)-DFA) includes finite as well as unary regular languages properly. Moreover, in [1] it is shown that (REV($*$)-DFA) includes the definite languages [32] and the reverse definite languages [10] properly. On the other hand, (REV($*$)-DFA) is incomparable with the subregular language families of generalized definite [10] and locally testable languages [28].

Coming to another aspect, we recall that it is well know that the minimal DFA accepting a given regular language is unique. So there is the natural question asking for the relations between minimality and reversibility. It turned out that in this connection the different notions of reversibility do matter. In [33], the following proposition is cited.

Proposition 3. *A language L is accepted by a bideterministic finite automaton if and only if the minimal finite automaton of L is reversible and has a unique final state.*

This answers the question about the notion of reversibility in [1]. However, for the other notions of reversibility considered, the *minimal reversible* finite automaton for some language can be exponentially larger than the minimal automaton.

Example 2. Let L be the finite language $\{aa, ab, ba\}$. The minimal DFA accepting the $2n$-fold concatenation of L is depicted in Figure 6. It has $6n + 1$ states. Since L^{2n} is finite, it is reversible. □

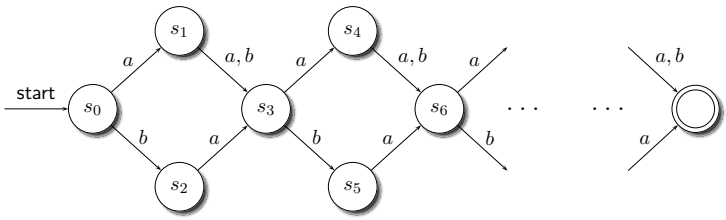

Fig. 6. A minimal DFA accepting the language L^{2n}, for $n \geq 1$

Theorem 4 ([13]). *Let $n \geq 1$. The minimal REV-DFA accepting L^{2n} has $\Omega(r^n)$ states, for some $r > 1$.*

Another aspect concerns the problem to decide whether a given language or automaton is reversible. In case of finite automata, deciding this problem for devices is almost trivial. An inspection of the transition function and the set of accepting states suffices. This observation transfers to languages in the notion of [1] by Proposition 3. In general, the problem is more involved.

Theorem 5 ([33]). *There is a polynomial time algorithm for testing whether the language accepted by a minimal finite automaton can be accepted by a reversible finite automaton.*

4 Pushdown Automata

The next level in the basic hierarchy of automata are deterministic pushdown automata (DPDA). Their reversible variants have been introduced and studied in [19], where only reachable configurations are relevant for reversibility and the predecessor configurations have to be computed by a device of the same type. Recall that the transition function δ of DPDA 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. We denote the relation from one configuration to the next by $\vdash$.

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 $c_{i+1} \vdash^{\leftarrow} c_i$, $0 \leq i \leq n-1$, for any sequence $c_0 \vdash c_1 \vdash \cdots \vdash c_n$ of configurations passed through by M beginning with an initial configuration c_0 (cf. Figure 7). See [19,23] for detailed definitions.

Example 3 ([19]). The linear context-free languages $\{ wcw^R \mid w \in \{a,b\}^* \}$ as well as $\{ a^n c b^n \mid n \geq 0 \}$ are accepted by REV-DPDA. A sketch of the basic idea is shown in Figure 8. □

A simple observation reveals that the transitions of a REV-DPDA either pop a symbol, change the symbol at the top of the pushdown store, or push a single

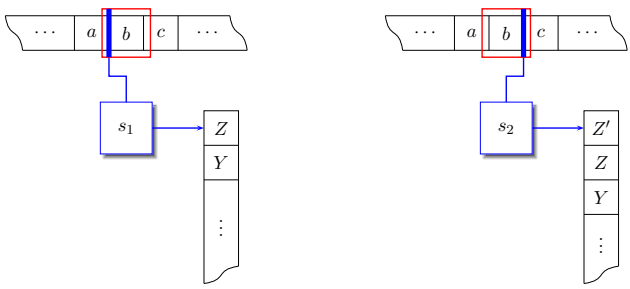

Fig. 7. Successive configurations of a reversible deterministic pushdown automaton, where $\delta(s_1, b, Z) = (s_2, Z'Z)$ (left to right) and $\delta^{\leftarrow}(s_2, b, Z') = (s_1, \lambda)$ (right to left)

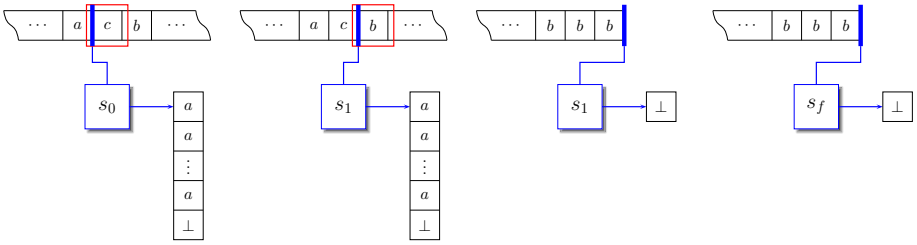

Fig. 8. Scheme of a forward computation of a REV-DPDA accepting $\{ a^n cb^n \mid n \geq 0 \}$. The crucial point for backward computations is to determine the time step at which state s_0 has to be reentered. This is given by the c in the input.

symbol. The reason is that the reverse transition has only access to the topmost symbol. The next result clarifies the role played by λ-steps.

Theorem 6 ([19]). *For every REV-DPDA an equivalent realtime REV-DPDA can effectively be constructed, that is, a REV-DPDA without λ-steps.*

It is well known that general deterministic pushdown automata which are not allowed to perform λ-steps are weaker than DPDA that may move on λ input [11]. So, Theorem 6 provides a class of irreversible deterministic context-free languages. Every deterministic context-free language that is not realtime is not accepted by any REV-DPDA. For example, the language

$$\{ a^m eb^n ca^m \mid m, n \geq 0 \} \cup \{ a^m eb^n da^n \mid m, n \geq 0 \}$$

does not belong to the family (REV-DPDA) (see, for example, [8,11]). This result immediately raises the question of whether *all realtime* deterministic context-free languages are reversible. The next theorem answers this question negatively. In particular, it shows that the c in the center of the input of Example 3 is essential.

Theorem 7 ([19]). *The realtime deterministic linear context-free language $\{ a^n b^n \mid n \geq 0 \}$ is not accepted by any REV-DPDA.*

So, we conclude that the family (REV-DPDA) is strictly included in the family of languages accepted by realtime deterministic pushdown automata. Not only in connection with reversibility it is interesting to consider realtime deterministic context-free languages whose reversals are also realtime deterministic context-free languages. It turned out that this family is incomparable with the family (REV-DPDA). Furthermore, it is known that the families of linear context-free languages and (REV-DPDA) are incomparable [19].

In [33], it has been shown that there are regular languages which are not accepted by any reversible finite automaton. However, the regular languages are strictly included in (REV-DPDA) [19]. Summarizing the results so far, we

have obtained the following strict hierarchy, where REG denotes the regular and $_{rt}$(DPDA) the realtime deterministic context-free languages:

$$\text{REG} \subset \quad \text{(REV-DPDA)} \subset \quad {}_{rt}\text{(DPDA)} \subset \quad \text{(DPDA)}.$$

Let us now turn to decidability aspects of REV-DPDA. Problems which are decidable for DPDA are decidable for REV-DPDA as well. Therefore, emptiness, universality, equivalence, and regularity are decidable for REV-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 REV-DPDA, too [19].

The following theorem contrasts the situation for finite automata, where the problem is decidable in polynomial time (Theorem 5).

Theorem 8 ([19]). *It is undecidable whether the language accepted by a nondeterministic pushdown automaton can be accepted by a REV-DPDA.*

The same problem for deterministic pushdown automata is open. However, if we consider devices instead of accepted languages, we have the decidability of reversibility. The size of a pushdown automaton is the length of its representation.

Theorem 9 ([19]). *Let M be a deterministic pushdown automaton of size n. Then it is decidable in time $O(n^4)$ whether M is a REV-DPDA. 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 a REV-DPDA by the previous theorem. If it is not a DPDA, then it cannot be a REV-DPDA. Therefore, the previous result transfers to nondeterministic devices.

Corollary 1. *Let M be a nondeterministic pushdown automaton of size n. Then it is decidable in time $O(n^4)$ whether M is a REV-DPDA. Moreover, the decision problem is P-complete.*

Next, the degree of reversibility is considered for pushdown automata. Compared with DFA the additional resource pushdown storage allows a more involved definition of lookaheads and, thus, degrees of reversibility. On the one hand, there is the possible lookahead on the input as for (k)-DFA. On the other hand, we consider a lookahead on the stack, that is, the machine can see the topmost l stack symbols.

Without going into the details of the definition, we say that a *deterministic pushdown automaton with lookaheads k and l* ((k,l)-DPDA) is a pushdown automaton having an input lookahead of size k and a lookahead of size l on the stack. A classical deterministic pushdown automaton is a DPDA with lookaheads $k = 1$ and $l = 1$.

In this connection we consider all configurations, not only reachable ones. As for DFA, a DPDA is said to be *reversible of degree* (k, l) (REV(k, l)-DPDA) if and only if there exists a reverse (k, l)-DPDA with transition function $\delta^{\leftarrow}$ inducing a relation $\vdash^{\leftarrow}$ from one configuration to the next, so that any configuration has a unique predecessor for all computations that lead to the configuration along the symbols seen in the input window and are consistent with the symbols at the top of the stack. Details of the definition can be found in [24].

Example 4. For any integer $k \geq 1$, the deterministic linear context-free language $\{ a^n b a^m b a^n \mid n \geq 1, m \geq k \}$ is accepted by some REV$(k+1, 1)$-DPDA. □

The languages of Example 4 are used as witnesses for an infinite and tight hierarchy of languages acceptable by reversible pushdown automata of a degree that depends on the size of the input window only. Large stack windows do not help.

Theorem 10 ([24]). *For any integer $k \geq 1$, there are deterministic linear context-free languages accepted by REV$(k+1, 1)$-DPDA that cannot be accepted by any REV(k, l)-DPDA, for an arbitrary $l \geq 1$.*

On the other hand, it turned out that the lookahead on the stack is interesting from a descriptional complexity point of view only. The question whether there are hierarchies with respect to the size of the stack lookahead has been answered negatively. In fact, any reversible pushdown automaton of degree $(k, l+1)$ can be simulated by a reversible pushdown automaton of degree $(k, 1)$. So, in general, a lookahead on the stack does not help to obtain reversibility. We present the results obtained from two different simulation principles based on where the information of the topmost stack symbols is maintained. This could be in additional registers of the states or in the stack symbols. Both methods are constructive. From a practical point of view, states are somehow more active resources while stack symbols are more passive. So, it depends on the application which principle is more suitable.

Theorem 11 ([24]). *Let $k, l \geq 1$ be integers and M be a REV(k, l)-DPDA with m states and n stack symbols. Then an equivalent REV$(k, 1)$-DPDA with n stack symbols and at most $m \cdot \frac{n^{l+1}}{n-1}$ states can effectively be constructed.*

The second construction groups up to l stack symbols into one. However, the construction has to overcome the problem, that, when the original automaton pops a symbol, the simulating one has to access the symbol below the topmost.

Theorem 12 ([24]). *Let $k, l \geq 1$ be integers and M be a REV(k, l)-DPDA with m states and n stack symbols. Then an equivalent REV$(k, 1)$-DPDA with m states and at most $\frac{n^{l+1}}{n-1} \cdot (n^l + 1)$ stack symbols can effectively be constructed.*

So, as for DFA it turned out that lookaheads on the input gradually increase the capability to perform reversible computations. On the other hand, lookaheads on the stack do not. Now we take a look beyond the degrees. Are all realtime

deterministic context-free languages captured by REV-DPDA? From above it is known that any finite language is accepted by some REV(1)-DFA and, thus, by a REV(1, 1)-DPDA. However, the necessity to provide arbitrary degrees to DFA to accept unary regular languages is not applicable for pushdown automata.

Proposition 4 ([24]). *Any unary (deterministic) context-free language is accepted by some REV*$(1, 1)$*-DPDA.*

Finally, by a similar idea of translation as for regular languages the next result is obtained.

Theorem 13 ([24]). *There are realtime deterministic context-free languages which cannot be accepted by any REV*(k, l)*-DPDA for any degree* (k, l), $k, l \geq 1$.

5 Time Symmetry

A further aspect of reversibility in real systems is discussed, for example, in [25]. In particular, physical reality reveals that often one can go back in time by applying the same transition function after a specific transformation of the phase-space. In [6] it is motivated that, for example, in Newtonian mechanics, relativity, or quantum mechanics one can go back in time by applying the same dynamics, provided that the sense of time direction is changed by a specific transformation of the phase-space. For Newtonian mechanics, the transformation leaves masses and positions unchanged but reverses the sign of the momenta. This aspect is called *time symmetry*. So, time-symmetric machines themselves cannot distinguish whether they run forward or backward in time. In this connection, computational models with discrete internal states, more precisely cellular automata, have been studied for the first time in [6].

Aspects of time-symmetry for reversible DFA and reversible DPDA have been considered in [23]. The "direction of time" is adjusted by a weak transformation of the phase-space, that is, an involution.

Let A, B, C be arbitrary sets, and $f : B \to C$, $g : A \to B$ two mappings. For their *composition* we write $f \circ g : A \to C$. A mapping $\tau : A \to A$ is said to be an *involution* if $\tau \circ \tau = \mathrm{id}$, where id denotes the identity mapping. In general, we say that an automaton M is *time symmetric* if there exists an involution τ on the phase-space so that $\tau \circ \delta \circ \tau = \delta^{\leftarrow}$. So, given a configuration c, an application of the involution τ transforms it, then δ is used to compute a new configuration, which is again transformed by a second application of τ. The result is the predecessor configuration of c. Precise definitions naturally depend on the specific type of automaton considered.

First we turn to REV-DFA. A reversible DFA with state set S is *time symmetric* if and only if there is an involution $\tau : S \to S$ so that $\delta_x^{-1} = \tau \circ \delta_x \circ \tau$ holds for all input symbols x, where δ_x is the next-state function for input symbol x. Looking at two successive steps, we obtain $\delta_x^{-1} \circ \delta_y^{-1} = \tau \circ \delta_x \circ \tau \circ \tau \circ \delta_y \circ \tau = \tau \circ \delta_x \circ \delta_y \circ \tau$. Obviously, this generalizes to arbitrary numbers of steps. In some sense τ reverses the direction in time permanently (that is, until τ is applied again).

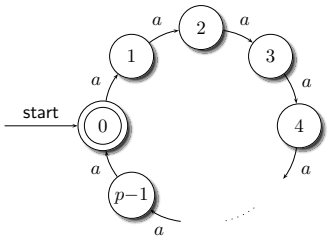

Fig. 9. Example of a unary reversible DFA accepting the language $\{ a^{j \cdot p} \mid j \geq 0 \}$

Example 5. Consider the p-state DFA M depicted in Figure 9. It turns out that M *is* time symmetric. As witness involution one can take $\tau(i) = p - i - 1$ (see Figure 10 for $p = 8$). We have $\tau(\delta_a(\tau(i))) = \tau(\delta_a(p - i - 1)) = \tau(p - i) = i - 1 = \delta_a^{-1}(i) = \delta^{\leftarrow}(i, a)$, for all i (all arithmetic being done mod p). □

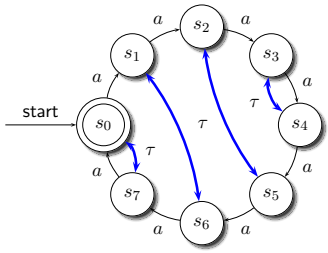

Fig. 10. Time symmetry of a reversible DFA accepting the language $\{ a^{j \cdot 8} \mid j \geq 0 \}$

The natural question whether all reversible DFA are already time symmetric has been answered in the negative.

Theorem 14 ([23]). *There are reversible DFA which are not time symmetric.*

Figure 11 shows a witness for Theorem 14. Nevertheless, the relation between reversible and time-symmetric DFA is different from the relation between arbitrary and reversible DFA.

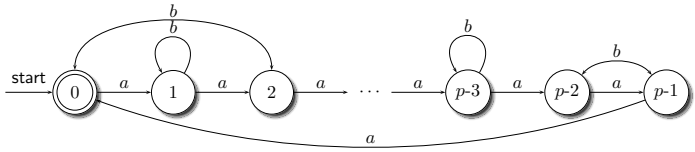

Fig. 11. Example of a reversible DFA that is not time symmetric

Theorem 15 ([23]). *Let $p \geq 1$ and M be a p-state reversible DFA. Then there exists an equivalent $2p$-state time-symmetric DFA. This upper bound is tight, that is, for $p \geq 6$ there is a p-state reversible DFA so that every equivalent time-symmetric DFA has at least $2p$ states.*

While there are reversible DFA that are not time symmetric in general, every *unary* REV-DFA *is* time-symmetric.

Theorem 16 ([23]). *Each reversible* unary *DFA is time symmetric.*

The second type of devices we are going to discuss from the viewpoint of time symmetry are reversible pushdown automata. The handling of the additional resource makes the definition of time symmetry more involved. While the usual presentation of DPDA uses the transition function δ, in the present context it is advantageous to consider its induced *extended transition function* $\hat{\delta} : S \times (\Sigma \cup \{\lambda\} \times \Gamma^* \to S \times \Gamma^*$, where Γ denotes the set of stack symbols, as follows. For any $p, q \in S$, $x \in \Sigma \cup \{\lambda\}$, $Z \in \Gamma$, and $\beta, \gamma \in \Gamma^*$ set $\hat{\delta}(q, x, Z\gamma) = (p, \beta\gamma)$ if and only if $\delta(q, x, Z) = (p, \beta)$.

While in Section 4 reversibility is considered for reachable configurations, here – to remain in context – we require reversibility for all configurations.

A reversible DPDA is *time symmetric* if and only if there is an involution $\tau : S \times \Gamma^* \to S \times \Gamma^*$ so that, $\hat{\delta}_x^{-1} = \tau \circ \hat{\delta}_x \circ \tau$ holds for all input symbols x.

For example, the linear context-free language $\{ wcv \mid w \in \{a, b\}^*, w^R = vu \}$ is accepted by a time-symmetric DPDA showing that time-symmetric DPDA can accept non-regular languages. However, the next result contrasts the situation for DFA.

Theorem 17 ([23]). *There are reversible unary DPDA which are not time symmetric.*

The unary language $\{ a^i \mid i \geq n \}$ is a witness for Theorem 17 (see Figure 12 for an example). Since it is regular, it is accepted by a REV-DPDA M. To this

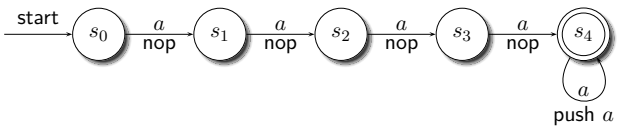

Fig. 12. A non-time-symmetric but reversible REV-DPDA that accepts a^4a^*

end, on any sufficiently long input M runs into a cycle and, thus, has to push the history to get out of it in backward computations. In order to give evidence that M is not time symmetric we note that in the cycle symbols have to be pushed. In particular, this means that backward steps have to pop symbols. However, there are no states in the cycle that pop a symbol. So, there is no state

to which a push state could be mapped by an involution witnessing the time symmetry.

As for DFA any reversible DPDA can be simulated by a time-symmetric DPDA with at most twice as many states.

Theorem 18 ([23]). *Let $p \geq 1$ and M be a p-state reversible DPDA. Then there exists an equivalent $2p$-state time-symmetric DPDA.*

Though it is well known that every unary context-free language is regular [9], the families of reversible and time-symmetric unary languages are different.

Theorem 19 ([23]). *(1) The family of languages accepted by time-symmetric (unary) DFA is properly included in the family of languages accepted by time-symmetric (unary) DPDA.*

(2) The family of languages accepted by reversible (unary) DFA is properly included in the family of languages accepted by reversible (unary) DPDA.

Finally, without going into further details, we just mention that the technique to put a reversible device and its inverse "side by side" also works for other types of reversible automata. However, whether the increase in size caused by doubling the number of states is always necessary is an interesting and almost untouched question.

References

1. Angluin, D.: Inference of reversible languages. J. ACM **29**, 741–765 (1982)
2. 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)
3. Axelsen, H.B., Glück, R.: A simple and efficient universal reversible turing machine. In: Dediu, A.-H., Inenaga, S., Martin-Vide, C. (eds.) LATA 2011. LNCS, vol. 6638, pp. 117–128. Springer, Heidelberg (2011)
4. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**, 525–532 (1973)
5. Bordihn, H., Holzer, M., Kutrib, M.: Determinization of finite automata accepting subregular languages. Theoret. Comput. Sci. **410**, 3209–3222 (2009)
6. Gajardo, A., Kari, J., Moreira, A.: On time-symmetry in cellular automata. J. Comput. System Sci. **78**, 1115–1126 (2012)
7. García, P., Vázquez de Parga, M., Cano, A., López, D.: On locally reversible languages. Theoret. Comput. Sci. **410**, 4961–4974 (2009)
8. Ginsburg, S., Greibach, S.A.: Deterministic context-free languages. Inform. Control **9**, 620–648 (1966)
9. Ginsburg, S., Rice, H.G.: Two families of languages related to ALGOL. J. ACM **9**, 350–371 (1962)
10. Ginzburg, A.: About some properties of definite, reverse-definite and related automata. IEEE Trans. Elect. Comput. **EC–15**, 806–810 (1966)
11. Harrison, M.A.: Introduction to Formal Language Theory. Addison-Wesley, Reading (1978)

12. Havel, I.M.: The theory of regular events II. Kybernetica **6**, 520–544 (1969)
13. Héam, P.C.: A lower bound for reversible automata. RAIRO Inform. Théor. **34**, 331–341 (2000)
14. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. AddisonWesley, Reading (1979)
15. Kari, J.: Reversible cellular automata. In: De Felice, C., Restivo, A. (eds.) DLT 2005. LNCS, vol. 3572, pp. 57–68. Springer, Heidelberg (2005)
16. Kobayashi, S., Yokomori, T.: Learning approximately regular languages with reversible languages. Theoret. Comput. Sci. **174**, 251–257 (1997)
17. Kutrib, M., Malcher, A.: Fast reversible language recognition using cellular automata. Inform. Comput. **206**, 1142–1151 (2008)
18. Kutrib, M., Malcher, A.: Real-time reversible iterative arrays. Theoret. Comput. Sci. **411**, 812–822 (2010)
19. Kutrib, M., Malcher, A.: Reversible pushdown automata. J. Comput. System Sci. **78**, 1814–1827 (2012)
20. 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)
21. Kutrib, M., Malcher, A.: Real-time reversible one-way cellular automata. In: Cellular Automata and Discrete Complex Systems (AUTOMATA 2014) (to appear, 2014)
22. Kutrib, M., Malcher, A., Wendlandt, M.: Reversible Queue Automata. In: Non-Classical Models of Automata and Applications (NCMA 2014), vol. 304, pp. 163–178. Autralian Computer Society (2014)
23. Kutrib, M., Worsch, T.: Time-symmetric machines. In: Dueck, G.W., Miller, D.M. (eds.) RC 2013. LNCS, vol. 7948, pp. 168–181. Springer, Heidelberg (2013)
24. 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)
25. Lamb, J.S., Roberts, J.A.: Time-reversal symmetry in dynamical systems: A survey. Phys. D **112**, 1–39 (1998)
26. Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5**, 183–191 (1961)
27. Lange, K.J., McKenzie, P., Tapp, A.: Reversible space equals deterministic space. J. Comput. System Sci. **60**, 354–367 (2000)
28. McNaughton, R., Papert, S.: Counter-Free Automata. No. 65 in Research Monographs. MIT Press (1971)
29. Morita, K.: Reversible simulation of one-dimensional irreversible cellular automata. Theoret. Comput. Sci. **148**(1), 157–163 (1995)
30. Morita, K.: Reversible computing and cellular automata - a survey. Theoret. Comput. Sci. **395**, 101–131 (2008)
31. Morita, K.: Two-way reversible multi-head finite automata. Fund. Inform. **110**, 241–254 (2011)
32. Perles, M., Rabin, M.O., Shamir, E.: The theory of definite automata. IEEE Trans. Elect. Comput. **EC–12**, 233–243 (1963)
33. Pin, J.E.: On reversible automata. In: Simon, I. (ed.) Latin 1992. LNCS, vol. 583, pp. 401–416. Springer, Heidelberg (1992)