Real-Time Reversible One-Way Cellular Automata

Martin Kutrib, Andreas Malcher(✉), and Matthias Wendlandt

Institut Für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
{kutrib,malcher,matthias.wendlandt}@informatik.uni-giessen.de

Abstract. Real-time one-way cellular automata (OCA) are investigated towards their ability to perform reversible computations with regard to formal language recognition. It turns out that the standard model with fixed boundary conditions is quite weak in terms of reversible information processing, since it is shown that in this case exactly the regular languages can be accepted reversibly. We then study a modest extension which allows that information may flow circularly from the leftmost cell into the rightmost cell. It is shown that this extension does not increase the computational power in the general case, but does increase it for reversible computations. On the other hand, the model is less powerful than real-time reversible two-way cellular automata. Additionally, we obtain that the corresponding language class is closed under Boolean operations, and we prove the undecidability of several decidability questions. Finally, it is shown that the reversibility of an arbitrary real-time circular one-way cellular automaton is undecidable as well.

Keywords: Reversibility · One-way cellular automata · Language recognition · Closure properties · Decidability

1 Introduction

Computational devices that are able to perform reversible computations have gained a lot of interest in the last years. The main property of reversible computations is that every configuration has a unique successor configuration as well as a unique predecessor configuration. Thus, in reversible computations no information is lost which is an appealing property from a physical point of view, because it has been observed that a loss of information results in heat dissipation [15]. Bennett [3] was the first who studied reversibility in computational devices, namely, in Turing machines. His fundamental result is that every Turing machine can be transformed into an equivalent reversible Turing machine. Thus, every recursively enumerable language can be processed in such a way that no information is lost. Less powerful classes with regard to the Chomsky hierarchy are the regular languages, which are accepted, for example, by deterministic finite automata (DFA), and the deterministic context-free languages, which are accepted, for example, by deterministic pushdown automata (DPDA). Reversible DFA are introduced in [2] and it is known [19] that there are regular languages for which no reversible DFA exists. This means that there are computations performed by DFA in which a loss of information cannot be avoided. Similar results are obtained for reversible DPDA in [13]. Recently, reversibility is also studied for multi-head finite automata. Morita shows in [18] that every multi-head finite automaton can be transformed into an equivalent reversible multi-head finite automaton under the condition of two-way motion of the multiple heads. In case of one-way motion, it is shown in [14] that the reversible variant is less powerful.

For cellular automata, injectivity of the global transition function is equivalent to the reversibility of the automaton. It is shown in [1] that global reversibility is decidable for one-dimensional cellular automata, whereas the problem is undecidable for higher dimensions [7]. For a detailed discussion we refer to the survey given in [17]. Additional information about some aspects of cellular automata may be found in [8]. All these results concern unbounded configurations. Moreover, in order to obtain a reversible device the neighborhood as well as the time complexity may be increased. In [5] it is shown that the neighborhood of a reverse CA is at most $n-1$ when the given reversible CA has n states. In connection with the ability to accept formal languages under real-time conditions, reversibility has been studied in [11,12] for real-time two-way cellular automata and real-time two-way iterative arrays with fixed boundary conditions. Cellular language acceptors are working on finite configurations with fixed boundary conditions (see, for example, [10]) and, thus, these devices cannot be reversible in the classical sense. Their number of different configurations is bounded. So, the system will run into loops that are reversible only if the initial configuration is reached again. In contrast to the traditional notion of reversibility, cellular language acceptors are considered that are reversible on the core of computation, that is, from initial configuration to the configuration given by the time complexity. This point of view is rather different from the traditional notion of reversibility since only configurations are considered that are reachable from initial configurations. At first glance, such a setting should simplify matters. But quite the contrary, we prove that real-time reversibility is undecidable.

In this paper, we continue the research on real-time reversibility and investigate one-way cellular automata (OCA). For the definition of reversible OCA, we first observe that information flow is from right to left in a forward computation and from left to right in a backward computation. Then, the following problem may occur: intuitively, every information which has passed the leftmost cell cannot be reconstructed since the leftmost cell always gets the border symbol from the left in the backward computation. Thus, such computations will be in general irreversible. We will show that this intuition is in fact right by proving that any real-time reversible OCA accepts a regular language. To obtain a more powerful model, we will allow that information may additionally flow circularly from the leftmost cell into the rightmost cell. It is shown that the language class accepted by real-time OCA with this extension, called circular OCA (COCA) is equivalent to the class accepted by classical real-time OCA.

Thus, the computational power of the general models is not increased, but it turns out that reversible COCA are more powerful than reversible OCA. First, we can prove that the computational power of reversible COCA lies properly in between reversible OCA and reversible two-way CA. Second, by a suitable simulation of reversible linearly bounded Turing machines by real-time COCA we obtain that emptiness and finiteness are undecidable for reversible real-time COCA. This implies the undecidability of inclusion and equivalence as well. Moreover, the problem of whether a given COCA is real-time reversible is also undecidable. These results are in line with the results for real-time two-way CA and real-time iterative arrays. Furthermore, we obtain also that the language class accepted by real-time COCA is closed under Boolean operations.

2 Preliminaries and Definitions

We denote the set of non-negative integers by $\mathbb{N}$. The reversal of a word w is denoted by w^R. For the length of w we write $|w|$. We write $\subseteq$ for set inclusion, and $\subset$ for strict set inclusion. In order to avoid technical overloading in writing, two languages L and L' are considered to be equal, if they differ at most by the empty word. Throughout the article two devices are said to be *equivalent* if and only if they accept the same language.

A one-way cellular automaton is a linear array of identical deterministic finite state machines, called cells, that are identified by natural numbers. In case of fixed boundary condition, each but the rightmost cell is connected to its nearest neighbor to the right. Cell 0 is in a distinguished permanent boundary state, and the rightmost cell is connected to cell 0. In case of circular boundary condition, the boundary state is not necessarily permanent [20] (see Fig. 1). The state transition depends on the current state of a cell itself and the current state of its neighbor. The state changes take place simultaneously at discrete time steps. The input mode for cellular automata is called parallel. One can suppose that all cells fetch their input symbol during a pre-initial step.

Definition 1. *A* (circular) one-way cellular automaton *((C)OCA) is a system* $\langle S, F, A, \#, \delta\rangle$*, where* S *is the finite, nonempty set of* cell states*,* $F \subseteq S$ *is the set of* accepting states*,* $A \subseteq S$ *is the nonempty set of* input symbols*,* $\# \in S \setminus A$ *is the distinguished* boundary state*, and* $\delta : S \times S \to S$ *is the* local transition function*.*

A configuration *of a (circular) one-way cellular automaton* $\langle S, F, A, \#, \delta\rangle$ *at time* $t \geq 0$ *is a mapping* $c_t : \{0, 1, 2, \dots, n\} \to S$*, for* $n \geq 1$*, that assigns a state to each cell. The operation starts at time 0 in a so-called* initial configuration*, which is defined by the given input* $w = a_1a_2 \cdots a_n \in A^+$*. We set* $c_0(i) = a_i$*, for* $1 \leq i \leq n$ *and* $c_0(0) = \#$*. Successor configurations are computed according to the global transition function* Δ*. Let* c_t*,* $t \geq 0$*, be a configuration with* $n \geq 1$*, then the successor* c_{t+1} *of a* one-way cellular automaton with fixed boundary condition *(OCA) is*

$$c_{t+1} = \Delta(c_t) \iff \begin{cases} c_{t+1}(0) = \# \\ c_{t+1}(i) = \delta(c_t(i), c_t(i+1)), i \in \{1, 2, \dots, n-1\} \\ c_{t+1}(n) = \delta(c_t(n), c_t(0)) \end{cases} .$$

The successor c_{t+1} *of a* one-way cellular automaton *with circular boundary condition (COCA) is*

$$c_{t+1} = \Delta(c_t) \iff \begin{cases} c_{t+1}(i) = \delta(c_t(i), c_t(i+1)), i \in \{0, 1, \dots, n-1\} \\ c_{t+1}(n) = \delta(c_t(n), c_t(0)) \end{cases} .$$

In order to distinguish between the boundary conditions, we write circular one-way cellular automaton for arrays with circular boundary conditions.

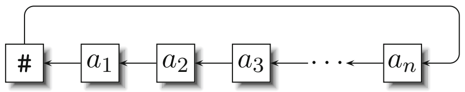

Fig. 1. A (circular) one-way cellular automaton.

An input w is accepted by a (circular) one-way cellular automaton if at some time step during its course of computation the leftmost cell receiving an input symbol, that is, cell 1, enters an accepting state. The *language accepted by* M is denoted by $L(M)$. Let $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, be a mapping. If all $w \in L(M)$ are accepted with at most $t(|w|)$ time steps, then M is said to be of time complexity t.

Observe that time complexities do not have to meet any further conditions. This general treatment is made possible by the way of acceptance. An input w is accepted if cell 1 enters an accepting state at some time $i \leq t(|w|)$. Subsequent states of the cell are not relevant. However, in the sequel we are particularly interested in fast devices operating in *real-time*, that is, obeying the time complexity $t(n) = n$. In general, the family of languages accepted by some device X with time complexity t is denoted by $\mathscr{L}_t(\text{X})$, where $\mathscr{L}_{rt}(\text{X})$ is written for real time.

Now we turn to cellular automata that are reversible on the core of computation, that is, from initial configuration to the configuration given by the time complexity. Consequently, we call them t-time reversible if the time complexity t is obeyed. One can imagine that the devices are switched off or reset after the computation. In this way only configurations are considered that are reachable from initial configurations. However, since the predecessor of such a configuration is unique, there cannot be an unreachable configuration as predecessor of a reachable one. Basically, reversibility is meant with respect to the possibility of stepping the computation back and forth. So, there must exist a reverse local transition function. Due to the domain S^2 and the range S, obviously, the local transition function cannot be injective in general. However, for reverse computation steps we may utilize the information which is available for the cells. In particular, the flow of information is reversed as well, and each cell receives the state of its *left* neighbor (the left neighbor of cell 0 is cell n).

For some mapping $t : \mathbb{N} \to \mathbb{N}$ let $M = \langle S, F, A, \#, \delta \rangle$ be a t-time ((C)OCA). Then M is said to be t *reversible* (REV-(C)OCA), if there exists a reverse local transition function $\delta_R : S \times S \to S$ so that $\Delta_R(\Delta(c_i)) = c_i$, for all configurations c_i

of M, $0 \leq i \leq t(n) - 1$. The global transition functions $\varDelta$ and $\varDelta_R$ are induced by δ and δ_R, respectively. For distinctness, we denote $\langle S, F, A, \texttt{\#}, \delta_R \rangle$ by M_R.

In order to clarify the notation we give an example.

Example 2. Language $L = \{\, a^n b^m \mid m \geq n \geq 1 \,\}$ is accepted by a real-time REV-COCA. For the construction we consider that each cell is divided into four tracks. Track 1 is used to store the original input permanently. Tracks 2 and 3 are used to shift input blocks of b's to the left. Here, every cell initially carrying an a (a-cell) uses both tracks, first track 3 and then track 2, to shift the input with speed 1/2. The #-cell and the remaining b-cells shift b's with speed 1 to the left using track 3 only. Using the circular structure of a COCA, any information shifted beyond the leftmost cell is stored step by step in the rightmost part of the COCA. Finally, track 4 is used to check the correct format and that there have been more b's than a's in the input. The latter check can be performed by every a-cell testing at the right time whether it is carrying a b on track 3. Only in this case an accepting state is entered.

Let us now argue why the automaton constructed accepts L. Assume that the input is $a^n b^n$. Since a-cells shift with speed 1/2 and b-cells shift with speed 1, the first b at cell $n+1$ enters track 3 of the first a at cell 1 at time $2n-1$. More generally, the ith b at cell $n+i$ enters track 3 of the ith a at cell i at time $2n-i$. Thus, it is possible for the signal started on track 4 of cell $2n$ at time 1 which reaches cell i at time $2n-i+1$ to check whether every cell i carries a b on its track 3 at time $2n-i$. If the input is $a^n b^m$ with $n \leq m$, the behavior is identical and an accepting state is entered if $n \geq 1$. If the input is $a^n b^m$ with $n > m$, then there is no b on track 3 of cell 1 at time $m+n$ and the input is not accepted. If the input is not of the form $a^+ b^+$, this can be detected by the check on track 4 and the entering of an accepting state is avoided. Nevertheless, the shifting and checking is continued.

To show that the automaton constructed is reversible we first note that the shifting of b's is reversible, since every b is shifted at every time step and the speed of shifting is uniquely determined by the original input stored on track 1. By the circular structure of the automaton, also no information is lost. Moreover, the actions on track 1 are reversible since its contents are never changing. The check of the correct format on track 4 can be done reversibly by simulating a deterministic finite automaton. Details are given in the proof of Theorem 4. For the remaining check we send a signal with maximum speed to the left started in the rightmost cell and enter an accepting state every time when the check on track 3 is successful. This can clearly be done reversibly.

Finally, we have to make sure that the original input is restored when going from time 1 to time 0. This can be ensured by a suitable interpretation of the input symbols. We identify symbol b with a state whose tracks 1 and 3 carry symbol b while tracks 2 and 4 are empty. In this way, the input symbol b can be used in later calculations and the need to restore them in the backward initial step is not occurring. On the other hand, we have to differentiate between an input symbol b and a state with b on its first track and empty track 3 which may occur when a blank is shifted into the cell. To this end, the latter states

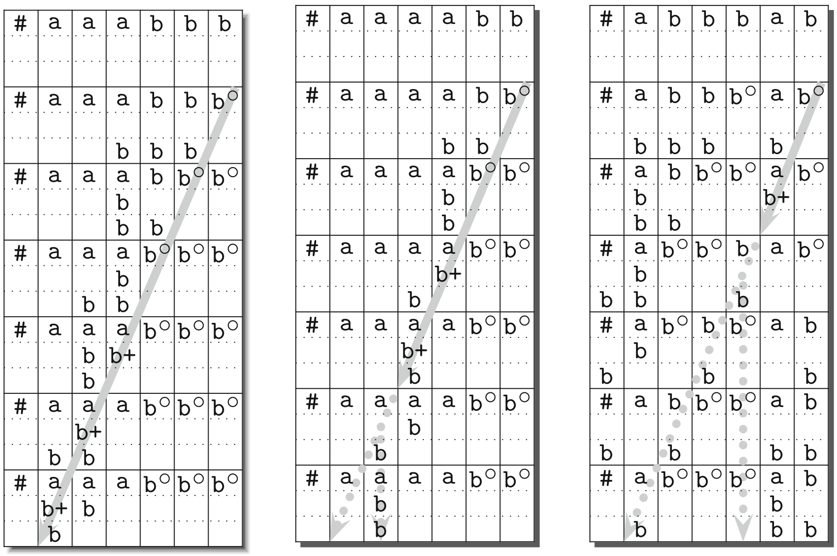

Fig. 2. Three example computations. A + denotes an accepting state, and a ∘ marks a cell into which a blank has been shifted on the third track. A solid arrow denotes the final check on the fourth track. If an error is encountered, the arrow is changed to a dotted arrow. Additionally, the time step is kept by storing a permanent information in the cell in which the error is encountered. This is depicted by the vertical dotted arrow. This construction is reversible since it occurs at most once.

are marked with ∘. Altogether, we obtain that L can be accepted by a real-time REV-COCA. Some example calculations to illustrate the construction are given in Fig. 2. □

3 Computational Capacity of Reversible (C)OCA

The classical definition of one-way cellular automata is the non-circular variant. However, for reversible real-time computations the slight generalization to circular cellular automata has a big impact. So, first we elaborate on this point.

Theorem 3. *Any language accepted by a real-time REV-OCA is regular.*

So, the condition to be reversible drastically reduces the computational capacity of OCA to that of deterministic finite automata, that is, a single cell. On the other hand, the next result says that every regular language can be accepted by a reversible real-time OCA. This is in contrast to the fact that there are regular languages that are not accepted by reversible DFA [2,19].

Theorem 4. *The family of regular languages and the family $\mathscr{L}_{rt}$(REV-OCA) are equal.*

Proof. By the previous theorem it remains to be shown that every regular language L can be accepted by some real-time REV-OCA. Since the regular languages are closed under reversal, L^R is also regular. Let L^R be accepted by a DFA M with state set S, input alphabet A, initial state s_0, set of accepting states F, and transition function $\delta : S \times A \rightarrow S$.

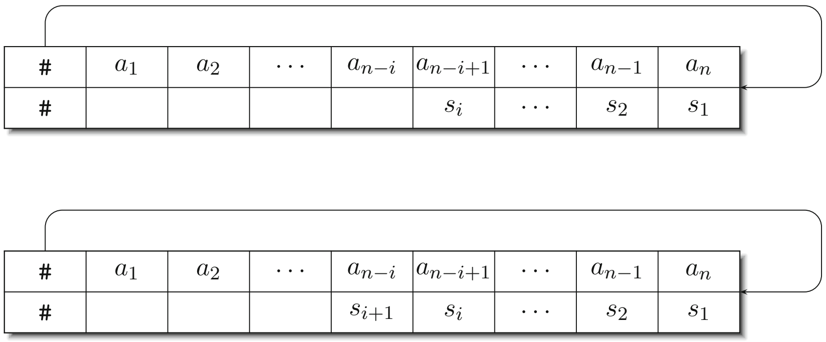

Fig. 3. Subsequent configurations of the REV-OCA M' in the proof of Theorem 4.

The idea for the simulation of L by a REV-OCA M' is first to divide its state set into two tracks. Track 1 is used to store the input permanently, while track 2 is used to send a signal from the rightmost cell with maximum speed to the left. The signal simulates the DFA M in such a way, that the state history is stored permanently on track 2. More precisely, let the input of M' be $a_1 a_2 \cdots a_n$. At time step 1, the rightmost cell n initiates the signal by calculating and storing $s_1 = \delta(s_0, a_n)$ on track 2. In general, for $0 \leq i \leq n-1$, the signal reaches cell $n-i$ at time $i+1$ and calculates and stores state $s_{i+1} = \delta(s_i, a_{n-i})$ on track 2 (see Fig. 3). The accepting states of M' are defined as those states having an accepting state of M on track 2. Clearly, M' accepts L. Trivially, the permanent storing of the input on track 1 is reversible. Moreover, since on track 2 the state history of M is stored, also the signal is reversible. $\square$

The previous results justify a slight generalization of reversible OCA. Moreover, the next result shows that for real-time computations the slight generalization to circular devices does not increase the computational capacity.

Theorem 5. *The language families $\mathscr{L}_{rt}(OCA)$ and $\mathscr{L}_{rt}(COCA)$ are equal.*

Proof. By definition, every OCA is a special case of a COCA. On the other hand, let M be a real-time COCA and $w \in L(M)$ be some accepted input. The states passed through by cell 1 up to time $|w|$ depend only on the states of cell 1 and 2 at time $|w|-1$, the states of cells 1 to 3 at time $|w|-2$, and so on until the states of cells 1 to $|w|$ at time 1, and the states of cells 1 to $|w|$ and 0 at time 0. Therefore, information sent by cell 1 to its left neighbor and further via cell $|w|$ towards cell 1 again can reach cell 1 not before time $|w|+1$. Thus, it cannot affect the overall computation result for real-time computations. $\square$

Since real-time OCA and real-time COCA characterize the same family of languages and the computational capacity of reversible OCA reduces to that of deterministic finite automata, we now turn to investigate the computational capacity of reversible real-time COCA. An immediate corollary is that the latter are strictly more powerful than the former, since Example 2 provides a non-regular language belonging to $\mathscr{L}_{rt}$(REV-COCA). So we have:

Lemma 6. *The family $\mathscr{L}_{rt}$(REV-COCA) properly includes $\mathscr{L}_{rt}$(REV-OCA).*

Next we turn to compare real-time reversible REV-COCA with real-time reversible *two-way* cellular automata (REV-CA). Basically, REV-CA are defined as REV-OCA with the exception that now the flow of information is two-way, that is, each cell is connected to its both nearest neighbors and the transition function δ maps $S \times S \times S$ to S.

Theorem 7. *The family $\mathscr{L}_{rt}$(REV-CA) properly includes $\mathscr{L}_{rt}$(REV-COCA).*

Proof. The inclusion follows for structural reasons. For the properness we use a unary witness language. It is well known that the language $L = \{ a^{2^n} \mid n \geq 0 \}$ is not accepted by any real-time OCA (see, for example, [9,10]).

On the other hand, in [4] it is shown that L is accepted by some CA in real time. The basic idea is depicted in Fig. 4. Initially a signal with speed 1/3 is sent to the right. Additionally, a signal with speed 1 that bounces between the slow signal and the leftmost cell is initiated. Now it is immediately verified that the fast signal is in the leftmost cell exactly at time steps 2^i, $i \geq 1$. Finally it suffices to send a signal from the rightmost cell to the left that accepts if and only if it arrives at the leftmost cell together with the fast signal. These three signals can be implemented in a reversible CA. $\square$

So, we have the following three level hierarchy:

$$\text{REG} = \mathscr{L}_{rt}(\text{REV-OCA}) \subset \mathscr{L}_{rt}(\text{REV-COCA}) \subset \mathscr{L}_{rt}(\text{REV-CA})$$

4 Closure Properties

This section is devoted to the closures of $\mathscr{L}_{rt}$(REV-COCA) under Boolean operations. A family of languages is said to be *effectively* closed under some operation if the result of the operation can be constructed from the given language(s).

Theorem 8. *The language family $\mathscr{L}_{rt}$(REV-COCA) is effectively closed under the Boolean operations complementation, union, and intersection.*

Proof. The effective closure under union and intersection can be proved the same way as for reversible real-time two-way CA [11]. The construction there is based on the well-known two-track technique and a suitable interpretation of accepting states. Both techniques do not require two-way communication and apply to REV-OCA as well.

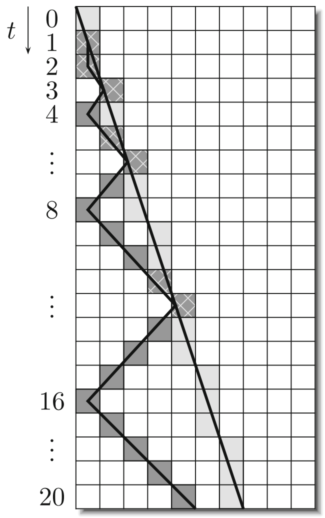

Fig. 4. Signals in a two-way cellular automaton accepting $\{ a^{2^n} \mid n \geq 0 \}$.

The principal idea for the construction showing the closure under complementation is to interchange accepting and non-accepting states. To enable this we have to make sure that the given REV-COCA accepts exactly at time step n on an input of length n and never before. This can be achieved by adding a copy S' of the state set S and by modifying the local transition function such that a state in S' is entered when an accepting state in S would have been entered. The transitions on S' are defined analogously to those of S. In this way, a cell remembers that it has entered an accepting state at some time step. Additionally, in the first time step a signal is started in the rightmost cell which moves with maximum speed to the left and makes any cell in some state from S' accepting. In this way acceptance in cell 1 at time n is ensured and accepting and non-accepting states can be interchanged. To guarantee the reversibility, we must be able to restore the time step in which cell 1 enters a state from S' for the first time. To this end, a signal Z is started in the next time step in cell 0 initially marked with #. This signal is shifted to the right part of the input by the circular structure of the automaton. Since only one such signal is started, we obtain the reversibility of the construction. A schematic example computation can be found in Fig. 5. $\square$

5 Decidability Questions

To show undecidability results for real-time REV-COCA we reduce the problems for deterministic one-tape one-head Turing machines whose space is limited by the length of the input, so-called linear bounded automata. It is well known that, for example, emptiness, finiteness, equivalence, regularity, and context-freeness is undecidable for such devices (see, for example, [6]).

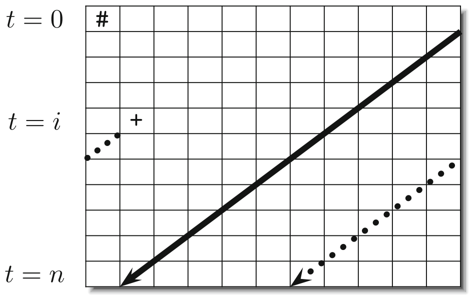

Fig. 5. Schematic construction of the signals used in the proof of closure under complementation. Cell 1 enters a state of S' at time step i for the first time. Signal Z is started at time step $i+1$ in cell 0 and is depicted by a dotted arrow. The solid arrow is the signal which arrives at real time in cell 1.

Theorem 9. *Emptiness is undecidable for real-time REV-COCA.*

Proof. Let T be an arbitrary deterministic linear bounded automaton. In [16] it has been shown that there is an equivalent reversible linear bounded automaton T'. Since T' works on limited space, we may assume that it is always halting. Moreover, by maintaining a counter as mentioned in [16], the backward computations of T' can be made always halting in the initial configuration. Next, T' can be modified to T'' without affecting the reversibility so that it starts with the head on the rightmost tape square and halts (in forward computations) and accepts only if the head is on the leftmost tape square.

Let Q denote the state set, q_0 the initial state, and I the input alphabet of T''. From T'' a new *reversible* linear bounded automaton $\hat{T}$ is constructed as follows. In a first phase, $\hat{T}$ simulates T'' from an initial configuration to a halting configuration. For a second phase a copy Q_b of the state set is used. After halting, $\hat{T}$ enters the copy of its current state. Now the states from Q_b are used to simulate the backward computation of T'' until the initial configuration (with the copy of the initial state) is reached and the backward computation halts.

Next we construct a REV-COCA $M = \langle S, F, A, \texttt{#}, \delta \rangle$ that, to some extent, simulates $\hat{T}$ as follows. The input alphabet A is $I \cup \{\texttt{\$}\}$, where $ is a new symbol. Basically, M uses three tracks. The input is provided on the third track, while the first and second one are initially empty.

The purpose of the third track is to simulate the tape of $\hat{T}$. The second track is used to store the current state of $\hat{T}$, and the first track is used to mark cells. Initially, every cell with an input symbol from I whose right neighbor carries either $ or # marks itself on the first track. In addition, it writes the initial state of $\hat{T}$ on its second track. See Fig. 6 for an example of the initial and first configuration.

Next, the simulation of $\hat{T}$ starts, where the simulation of one step of $\hat{T}$ takes two steps of M. To this end, the content of the third track is circularly shifted to the left at every other time step. Figure 7 shows how the transitions of $\hat{T}$ are simulated.

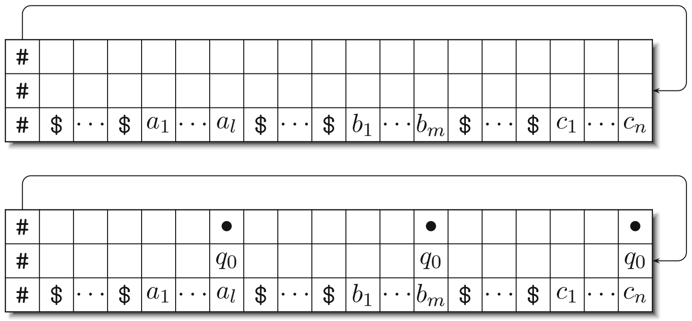

Fig. 6. Initial (top) and first (bottom) configuration of the REV-COCA M in the proof of Theorem 9.

When the simulation of $\hat{T}$ halts, that is, apart from the copy of its initial state, $\hat{T}$ is in its initial configuration again, the marked cells delete their mark and the state on their second track. The effect is that the original input is restored though cyclically shifted. Therefore, the whole simulation process repeats. Clearly, M is reversible since $\hat{T}$ is.

It remains to be explained how an input is accepted. To this end, M is extended by another, initially empty track that does not affect the behavior on the other tracks. In the first time step the rightmost cell initiates a signal on that track that moves to the left and simulates a deterministic finite automaton A. On its way the state history is stored so that the whole process is again reversible. Automaton A checks the structure of the input that has to be of the form $\$^+I^+\$^+$. Moreover, A enters an accepting state if and only if the structure is correct, and exactly at the moment it arrives in a cell the forward simulation of $\hat{T}$ halts accepting. The accepting states of M are now defined to be those states with an accepting state of A on the additional track.

Assume that $\hat{T}$ and, thus, the given linear bounded automaton T accepts an input. Then there exist appropriated numbers of $ to the left and to the right of the input so that M accepts as well. On the other hand, if $\hat{T}$ does not accept any input, also M does not accept any input. So, if emptiness would be decidable for real-time REV-COCA it would be decidable for linear bounded automata, a contradiction. □

The reduction in the proof of the previous theorem shows even more:

Theorem 10. *(In)finiteness is undecidable for real-time REV-COCA.*

Proof. If the real-time REV-COCA in the proof of Theorem 9 accepts, the numbers of $ to the left and to the right of the input over I is not unique. The simulation of $\hat{T}$ can go through further rounds. By adjusting the number of $ appropriately, we can find further inputs accepted by M without changing the input over I. Therefore, $L(M)$ is finite if and only if $L(T)$ is empty. This implies that neither finiteness nor infiniteness is decidable. □

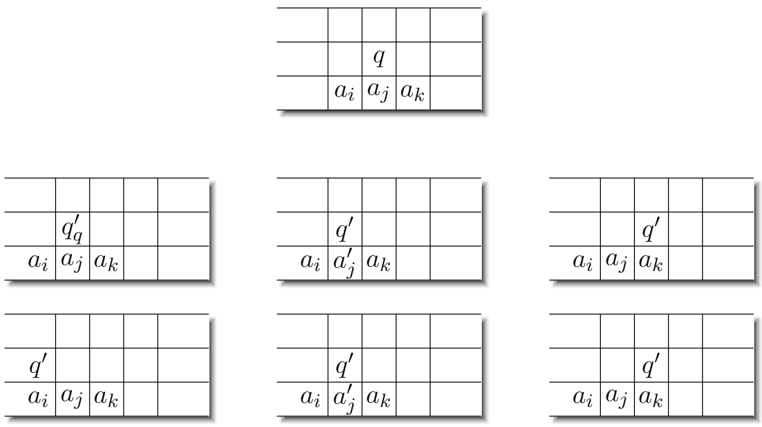

Fig. 7. Simulation of transitions of the linear bounded automaton. The initial situation is depicted at the top row. The two subconfigurations of the left column show the simulation of a left move, that is, (q, a_j) is mapped to (q', left). Here the intermediate state q'_q is used to indicate the following left move of the state. The two subconfigurations of the center column show the simulation of a stay/write move, that is, (q, a_j) is mapped to (q', a'_j), and the two subconfigurations of the right column show the simulation of a right move, that is, (q, a_j) is mapped to (q', right). The part of the state controlling the shift at *every other* step is omitted.

Theorem 11. *Equivalence and, thus, inclusion are undecidable for real-time REV-COCA.*

Proof. A COCA that simply does nothing is trivially reversible and accepts the empty language if the set of accepting states is empty. So, if equivalence would be decidable, emptiness would be decidable as well.

Two COCA M_1 and M_2 are equivalent if and only if $L(M_1) \subseteq L(M_2)$ and $L(M_2) \subseteq L(M_2)$. So, the decidability of inclusion would imply the decidability of equivalence. □

Theorem 12. *Let M be a real-time COCA. It is undecidable whether or not M is real-time reversible.*

Proof. Let M' be a real-time REV-COCA and F its set of accepting states. We modify M' to a real-time COCA M in such a way that we first add a new state g to the state set. Second, $\{g\}$ is defined to be the set of accepting states of M. Finally, the transition function is modified such that every cell in M enters state g whenever the cell would enter some state from the set F. Additionally, every cell in state g stays for the rest of the computation in this state and propagates state g with maximum speed to the left. We claim that the computation becomes irreversible whenever g is entered at least once. Let cell i enter state g at time step t on input w. Then, the behavior of state g destroys for the remaining time any information stored in cells $1, 2, \ldots, i$. Since the information flow in M is from right to left only, there are infinitely many inputs of arbitrary length with suffix w such that at real time all cells to the left of cell i are carrying only the information g. Then it is in particular not possible to restore the input: the only way would be to store the input using the circular structure. But since the inputs may be arbitrarily long, any information would reach a g-cell from the right and the information is lost.

Next, we claim that M is reversible if and only if $L(M')$ is empty. If M is reversible, then any cell can never enter state g which implies that $L(M)$ is empty. Then, by the construction, M' never enters an accepting state and $L(M')$ is empty as well. On the other hand, if $L(M')$ is empty, then M' never enters an accepting state and M never enters state g. Thus, M behaves the same way as M' and thus is reversible since M' is.

Now, we assume that the reversibility of a real-time COCA is decidable. This implies that we can decide the reversibility of M and so we can decide the emptiness of M'. This is a contradiction to Theorem 9. $\square$

6 Conclusion

Concerning the language recognition capacity of reversible cellular automata we obtained the strict hierarchy

$$\text{REG} = \mathscr{L}_{rt}(\text{REV-OCA}) \subset \mathscr{L}_{rt}(\text{REV-COCA}) \subset \mathscr{L}_{rt}(\text{REV-CA}).$$

Nevertheless, several questions remain unanswered. Exemplarily, we mention the relation between reversible real-time COCA and general real-time OCA. Can every language from $\mathscr{L}_{rt}(\text{OCA})$ reversibly be accepted by some real-time REV-COCA? In order to approach this problem, one can investigate counters. Is there a reversible COCA that passes through k^n different configurations on inputs of length n? The languages $L_k = \{\, a^n b^{k^n} \mid n \geq 1 \,\}$ belong to the family $\mathscr{L}_{rt}(\text{OCA})$. Moreover, it is known that all linear context-free languages are accepted by real-time OCA. For example, to accept the mirror language without center marker $\{\, w \mid w \in \{a,b\}^*, w = w^R \,\}$ a real-time OCA has to treat several different positions as centers. Can this computation be done reversibly by a REV-COCA?

The relations between $\mathscr{L}_{rt}(\text{REV-COCA})$ and the family of languages accepted by reversible as well as general iterative arrays, or between the families $\mathscr{L}_{lt}(\text{REV-COCA})$ and $\mathscr{L}_{rt}(\text{REV-CA})$ are also promising fields for further investigations.

References

1. Amoroso, S., Patt, Y.N.: Decision procedures for surjectivity and injectivity of parallel maps for tesselation structures. J. Comput. System Sci. **6**, 448–464 (1972)
2. Angluin, D.: Inference of reversible languages. J. ACM **29**, 741–765 (1982)
3. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**, 525–532 (1973)

4. Choffrut, C., Čulik II, K.: On real-time cellular automata and trellis automata. Acta Inform. **21**, 393–407 (1984)
5. Czeizler, E., Kari, J.: A tight linear bound on the neighborhood of inverse cellular automata. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 410–420. Springer, Heidelberg (2005)
6. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Cambridge (1979)
7. Kari, J.: Reversibility and surjectivity problems of cellular automata. J. Comput. System Sci. **48**, 149–182 (1994)
8. Kari, J.: Theory of cellular automata: a survey. Theoret. Comput. Sci. **334**, 3–33 (2005)
9. Kutrib, M.: Cellular automata - a computational point of view. In: Bel-Enguix, G., Jiménez-López, M.D., Martín-Vide, C. (eds.) New Developments in Formal Languages and Applications. Studies in Computational Intelligence, vol. 113, pp. 183–227. Springer, Heidelberg (2008)
10. Kutrib, M.: Cellular automata and language theory. In: Meyers, R.A. (ed.) Encyclopedia of Complexity and System Science, pp. 800–823. Springer, New York (2009)
11. Kutrib, M., Malcher, A.: Fast reversible language recognition using cellular automata. Inform. Comput. **206**, 1142–1151 (2008)
12. Kutrib, M., Malcher, A.: Real-time reversible iterative arrays. Theoret. Comput. Sci. **411**, 812–822 (2010)
13. Kutrib, M., Malcher, A.: Reversible pushdown automata. J. Comput. System Sci. **78**, 1814–1827 (2012)
14. 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)
15. Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5**, 183–191 (1961)
16. Lange, K.J., McKenzie, P., Tapp, A.: Reversible space equals deterministic space. J. Comput. System Sci. **60**, 354–367 (2000)
17. Morita, K.: Reversible computing and cellular automata - a survey. Theoret. Comput. Sci. **395**, 101–131 (2008)
18. Morita, K.: Two-way reversible multi-head finite automata. Fund. Inform. **110**, 241–254 (2011)
19. Pin, J.-C.: On reversible automata. In: Simon, I. (ed.) LATIN 1992. LNCS, vol. 583, pp. 401–416. Springer, Heidelberg (1992)
20. Umeo, H., Morita, K., Sugata, K.: Deterministic one-way simulation of two-way real-time cellular automata and its related problems. Inform. Process. Lett. **14**, 158–161 (1982)