Self-verifying Cellular Automata

Martin Kutrib[1] and Thomas Worsch[2(✉)]

[1] Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany
kutrib@informatik.uni-giessen.de
[2] Karlsruhe Institute of Technology, Karlsruhe, Germany
worsch@kit.edu

Abstract. We study the computational capacity of self-verifying cellular automata with an emphasis on one-way information flow (SVOCA). A self-verifying device is a nondeterministic device where each computation path can give one of the answers *yes*, *no*, or *do not know*. For every input word, at least one computation path must give either the answer *yes* or *no*, and the answers given must not be contradictory. Realtime SVOCA are strictly more powerful than realtime deterministic one-way cellular automata. They can be sped-up from lineartime to realtime and are capable to simulate any lineartime computation of deterministic two-way CA. Closure and decidability properties are considered as well.

1 Introduction

What is the power of nondeterminism in bounded-resource computations? Traditionally, nondeterministic devices have been viewed as having as many nondeterministic guesses as time steps. The studies of this concept of unlimited nondeterminism led, for example, to the famous open LBA-problem or the unsolved question whether or not P equals NP. In order to gain further understanding of the nature of nondeterminism, in for example [9] it has been viewed as an additional limited resource at the disposal of time or space bounded computations. We study the computational power of self-verifying cellular automata (SVCA). A self-verifying device is a nondeterministic device with symmetric conditions for acceptance/rejection. Each computation path can give one of the answers *yes*, *no*, or *do not know*. For every input word, at least one computation path must give either the answer *yes* or *no*, and the answers given must not be contradictory. So, if a computation path gives the answer *yes* or *no*, in both cases the answer is definitely correct. This justifies the notion *self-verifying* and is in contrast to general nondeterministic computations, where an answer that is not *yes* does not allow to conclude whether or not the input belongs to the language.

Self-verifying finite automata have been introduced and studied in [6,11,12] mainly in connection with randomized Las Vegas computations. Descriptional complexity issues for self-verifying finite automata have been studied in [14]. The computational and descriptional complexity of self-verifying pushdown automata has been studied in [8].

© Springer Nature Switzerland AG 2018
G. Mauri et al. (Eds.): ACRI 2018, LNCS 11115, pp. 340–351, 2018.
https://doi.org/10.1007/978-3-319-99813-8_31

The paper is organized as follows. In Sect. 2 we present the basic notation and the definitions of self-verifying (one-way) cellular automata as well as an introductory example. In Sect. 3 a strong speed-up result is derived that allows the conversion of lineartime SVOCA to realtime. Section 4 is devoted to explore the computational capacity of realtime SVOCA. It turns out that they are even capable to simulate any lineartime computation of a two-way CA. Moreover, the closure properties of the family of languages accepted by realtime SVOCA are studied. It is shown that the family is closed under the set-theoretic operations, reversal, concatenation, and inverse homomorphisms. Finally, decidability problems are considered. In particular, the property of being self-verifying turns out to be non-semidecidable.

Because of a page limit not all proofs are included in the version for the conference proceedings, but a research report with all details is available at [17].

2 Preliminaries

We denote the positive integers $\{1, 2, \dots\}$ by $\mathbb{N}$, the set $\mathbb{N} \cup \{0\}$ by $\mathbb{N}_0$, and the *powerset* of a set S by 2^S. We write $|S|$ for the *cardinality* of S. Let Σ denote a finite set of letters. Then we write Σ^* for the *set of all finite words* (strings) consisting of letters from Σ. The *empty word* is denoted by λ, and we set $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. For the *reversal of a word* w we write w^R and for its *length* we write $|w|$. A subset of Σ^* is called a *language* over Σ. The devices we will consider cannot accept the empty word. So, 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, that is, if $L \setminus \{\lambda\} = L' \setminus \{\lambda\}$. Set *inclusion* is denoted by $\subseteq$ and *strict set inclusion* by $\subset$.

A two-way cellular automaton is a linear array of identical finite automata, called cells, numbered $1, \dots, n$. Except for border cells the state transition depends on the current state of a cell itself and those of its both nearest neighbors. Border cells receive a boundary symbol on their free input lines. Synchronous state changes take place at discrete time steps.

We first define *nondeterministic* cellular automata. The nondeterminism is restricted to the first step. All further transitions are deterministic [2,16]. Although this is a very restricted case, we call such devices nondeterministic.

A *nondeterministic two-way cellular automaton* (NCA, for short) is a system $M = \langle S, \Sigma, F, \#, \delta_{nd}, \delta_d \rangle$, where

1. S is the finite, nonempty set of *cell states*,
2. $\Sigma \subseteq S$ is the nonempty set of *input symbols*,
3. $F \subseteq S$ is the set of *accepting states*,
4. $\# \notin S$ is the *boundary symbol*,
5. $\delta_{nd} \colon (S \cup \{\#\}) \times S \times (S \cup \{\#\}) \to (2^S \setminus \emptyset)$ is the *nondeterministic local transition function* applied in the first state transition,
6. $\delta_d \colon (S \cup \{\#\}) \times S \times (S \cup \{\#\}) \to S$ is the *deterministic local transition function* applied in all further state transitions.

In a *one-way* cellular automaton the next state of each cell only depends on the state of the cell itself and the state of its immediate neighbor to the right. So the domain of the transition functions is $S \times (S \cup \{\#\})$.

A *configuration* c_t of M at time $t \geq 0$ is a mapping $c_t : \{1, 2, \ldots, n\} \to S$, for $n \geq 1$, occasionally represented as a word over S. The *initial configuration* c_0 for an input $w = a_1 a_2 \cdots a_n \in \Sigma^+$ is defined by $c_0(i) = a_i$, for $1 \leq i \leq n$. For example, the initial configuration of an NOCA for w is represented by $a_1 a_2 \cdots a_n$. Successor configurations are computed according to the global transition function Δ mapping each configuration to a set of successor configurations.

For an NCA configuration c_t the set of its successors c_{t+1} is defined as:

$$c_{t+1} \in \Delta(c_t) \iff \begin{cases} c_{t+1}(1) \in \sigma(\#, c_t(1), c_t(2)) \\ c_{t+1}(i) \in \sigma(c_t(i-1), c_t(i), c_t(i+1)), i \in \{2, \ldots, n-1\} \\ c_{t+1}(n) \in \sigma(c_t(n-1), c_t(n), \#) \end{cases}$$

where $\sigma = \delta_{nd}$ if $t = 0$, and $\sigma = \delta_d$ if $t \geq 1$. For NOCA the global transition function is defined analogously. Thus, Δ is induced by δ_{nd} and δ_d. An NCA (NOCA) is *deterministic* if $\delta_{nd}(s_1, s_2, s_3)$ ($\delta_{nd}(s_1, s_2)$) is a singleton for all states $s_1, s_2, s_3 \in S \cup \{\#\}$. Deterministic cellular automata are denoted by CA and OCA.

An input w is *accepted* by a cellular automaton if at some time step during some computation the leftmost cell enters an accepting state. The *language accepted by* M is $L(M) = \{ w \in \Sigma^+ \mid w \text{ is accepted by } M \}$. Let $t : \mathbb{N} \to \mathbb{N}$ 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 (see [15] for a more on this general treatment of time complexity functions). If $t(n) = n$ acceptance is said to be in *realtime*. If $t(n) = k \cdot n$ for a rational number $k \geq 1$, then acceptance is in *lineartime*. The set of all languages accepted by devices X with time complexity t is denoted by $\mathscr{L}_t(X)$. We write $\mathscr{L}_{rt}(X)$ for real time and $\mathscr{L}_{lt}(X)$ for linear time.

Now we turn to *self-verifying* (one-way) cellular automata (SV(O)CA). As for NCA during the first step cells may choose between several new states. But the definition of acceptance is different from nondeterministic CA.

There are now three disjoint sets of states representing answers *yes*, *no*, and *do not know*. Moreover, for every input word, at least one computation path must give either the answer *yes* or *no*, and the answers given must not be contradictory. In order to implement the three possible answers the state set is partitioned into three disjoint subsets $S = F_+ \dot\cup F_- \dot\cup F_0$, where F_+ is the set of accepting states, F_- is the set of rejecting states, and $F_0 = S \setminus (F_+ \cup F_-)$ is referred to as the set of neutral states. We specify F_+ and F_- in place of the set F. of SVCA and SVOCA. So, let $M = \langle S, \Sigma, F_+, F_-, \#, \delta_{nd}, \delta_d \rangle$ be an SVOCA. For each input $w \in \Sigma^+$, the set of states reachable by cell 1 is defined as $S_w = \{ s \in S \mid s \in \left(\Delta^{[t]}(w\#)\right)(1) \text{ for some } t \geq 0 \}$, where $\Delta^{[t]}$ denotes the t-fold composition of Δ, that is, the set of configurations reachable in t time steps. For the "self-verifying property" it is required that for each $w \in \Sigma^+$, $S_w \cap F_+$ is empty if and only if $S_w \cap F_-$ is nonempty.

If all $w \in L(M)$ are accepted and all $w \notin L(M)$ are rejected after at most $t(|w|)$ time steps, then the self-verifying cellular automaton M is said to be of time complexity t. We illustrate the definitions with an example.

Example 1. The non-semilinear unary language $\{ a^{2^n} \mid n \geq 0 \}$ is accepted by the SVOCA $M = \langle \{a, -, 1, X, \sim, <_1, <_2, \ominus, \otimes, \oplus, 0\}, \{a\}, F_+, F_-, \#, \delta_{nd}, \delta_d \rangle$ in realtime, where $F_+ = \{\oplus\}$, $F_- = \{\ominus, \otimes\}$, and the transition functions δ_{nd} and δ_d are defined as follows.

$$
\begin{array}{lll}
(1) & \delta_{nd}(a,a) & = \{1,-\} \\
(2) & \delta_{nd}(a,\#) & = \{\oplus\} \\
(3) & \delta_d(1,-) & = X \\
(4) & \delta_d(-,-) & = - \\
(5) & \delta_d(-,1) & = <_1 \\
(6) & \delta_d(-,<_1) & = - \\
(7) & \delta_d(-,<_2) & = <_1 \\
(8) & \delta_d(X,-) & = X \\
(9) & \delta_d(X,<_1) & = X \\
(10) & \delta_d(<_1,X) & = <_2 \\
(11) & \delta_d(<_1,\sim) & = <_2 \\
(12) & \delta_d(<_2,X) & = \sim \\
(13) & \delta_d(<_2,\sim) & = \sim \\
(14) & \delta_d(\sim,\sim) & = \sim \\
(15) & \delta_d(1,\oplus) & = \oplus \\
(16) & \delta_d(X,\otimes) & = \oplus \\
(17) & \delta_d(<_1,\oplus) & = \otimes \\
(18) & \delta_d(<_1,\ominus) & = \otimes \\
(19) & \delta_d(<_2,\ominus) & = \ominus \\
(20) & \delta_d(\sim,\oplus) & = \ominus \\
(21) & \delta_d(\sim,\ominus) & = \ominus
\end{array}
$$

In addition to these transitions, δ_d maps any state from $\{\ominus, \otimes, \oplus, 0\}$ to itself, regardless of its neighbor. And all still undefined transitions map to the state 0.

The idea of the construction is as follows (see Fig. 1). Assume that the cells are numbered from 1 to n from right to left. In the first step, each cell guesses whether its position is 2^i, for some $i \geq 1$ (1). Accordingly they enter state 1 or $-$. The rightmost cell can identify itself and always enters state $\oplus$ (2). Next, each cell in state 1 sends a signal with speed 1/2 to the left. The signal is realized by states $<_1$ and $<_2$ (5–7 and 10–13). Moreover, cells in state 1 change to state X (3) and each cell passed through by such a signal changes to state $\sim$ (12–14).

In addition, initially a signal s is sent by the rightmost cell to the left with speed 1. This signal is realized by the states $\{\ominus, \otimes, \oplus\}$ and possibly by state 0 if an initial guess is wrong. The states $\{\ominus, \otimes, \oplus\}$ represent accepting and rejecting decisions of the cells. Once such state is entered it is never left again. Therefore the decisions are not contradictory. Now the idea is that the initial guess is verified if and only if signal s meets a 1/2-speed signal in a cell that initially guessed to be at some position 2^i and, thus, is now in state X (16–21).

In order to evidence the correctness of the construction, let us first assume the initial guesses are correct. Then cells 1 and 2 behave as required by Transitions 2 and 15. Now let some cell 2^i enter the accepting state $\oplus$ at time 2^i (which is true for cells 1 and 2). Then the 1/2-speed signal sent by that cell has reached cell $2^i + 2^{i-1}$. This implies that the fast and slow signal will meet in cell 2^{i+1}, as required. Altogether, for the case of initially correct guesses, the decisions are never contradictory, they are correct, and the guesses are verified to be true.

For the cases where one of the initial guesses is wrong, the neutral state 0 is used. Whenever a slow and the fast signal do not meet in a cell being in state X, state 0 is entered. Moreover, it is entered whenever two neighboring cells are in state 1. In particular, since the state 0 is never left, the fast signal checks the correctness of the initial guesses from right to left. It is stopped by any cell in the neutral state 0. Again, no contradictory decisions are made and, no decision

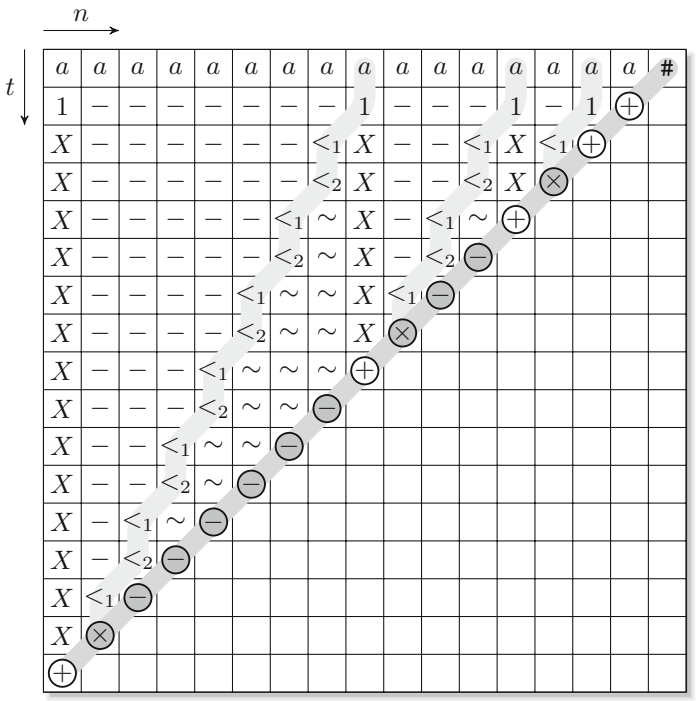

Fig. 1. Computation of a realtime SVOCA accepting the language $\{ a^{2^n} \mid n \geq 0 \}$. Slow signals moving with speed 1/2 are depicted in light gray, the fast signal with states $\ominus, \otimes, \oplus$ in a darker gray.

is made by the leftmost cell in case of wrong guesses. So, this realtime one-way cellular automaton accepts language $\{ a^{2^n} \mid n \geq 0 \}$ and it is self-verifying. ■

3 Characterization and Speed-Up

First we give evidence that self-verifying (one-way) cellular automata are in fact a generalization of deterministic (one-way) cellular automata. To this end, it is reasonable to consider only time complexities t that allow the leftmost cell to recognize the time step $t(n)$. Such functions are said to be *time-computable*. For example, the identity $t(n) = n$ is a time-computable time complexity for (O)CA. A signal which is initially emitted by the rightmost cell and moves with maximal speed, arrives at the leftmost cell exactly at time step n. By slowing down the signal to speed $\frac{x}{y}$ (that is, the signal moves x cells to the left and then stays in a cell for $y - x$ time steps), it is seen that the time complexities $\lfloor \frac{y}{x} \cdot n \rfloor$, for any positive integers $x < y$, are time-computable. More details can be found in [3].

Lemma 2. *Any (one-way) deterministic cellular automaton with a time-computable time complexity t can effectively be converted into an equivalent (one-way) self-verifying cellular automaton with the same time complexity t.*

For any time-computable time complexity t, the closures of the families $\mathscr{L}_t$(SVOCA) and $\mathscr{L}_t$(SVCA) under complementation are immediately seen. In order to construct an SVOCA that accepts the complement of the language accepted by a given SVOCA, it is sufficient to interchange the accepting and rejecting states while the neutral states remain as they are. On the other hand, Example 1 gives a witness for the strictness of the inclusion $\mathscr{L}_{rt}(\text{OCA}) \subset \mathscr{L}_{rt}(\text{SVOCA})$ since all unary languages accepted by realtime OCA are regular. This observation raises the natural question whether every language accepted by some realtime NOCA, whose complement is again accepted by some realtime NOCA, is accepted by a realtime SVOCA.

Proposition 3. *Let t be a time-computable time complexity. Every language $L \in \mathscr{L}_t(NCA)$ whose complement $\overline{L}$ belongs to $\mathscr{L}_t(NCA)$ as well is accepted by some t-time SVCA. The same is true for one-way devices.*

Proof. Let M_1 be a device accepting L and M_2 be a device accepting $\overline{L}$ with time complexity t. Now a t-time self-verifying devices M simulates M_1 and M_2 on different tracks, that is, it uses the same two channel technique of [7,19].

Then it remains to define the set of accepting states as $F_+ = \{ (s, s') \mid s \in F_1 \}$ and the set of rejecting states as $F_- = \{ (s, s') \mid s' \in F_2 \}$, where F_1 is the set of accepting states of M_1 and F_2 is the set of accepting states of M_2. □

Since it is straightforward to extract an NOCA accepting the complement of $L(M)$ from a given SVOCA M, the characterizations of the next theorem have been derived.

Theorem 4. *Let t be a time-computable time complexity. The family of languages $L \in \mathscr{L}_t(NCA)$ such that $\overline{L}$ belongs to $\mathscr{L}_t(NCA)$ as well coincides with the family $\mathscr{L}_t(SVCA)$. The same is true for one-way devices.*

Several types of cellular automata can be sped-up by a constant amount of time as long as the remaining time complexity does not fall below realtime. A proof in terms of trellis automata can be found in [4]. In [13] the speed-up results are shown for deterministic and nondeterministic cellular and iterative automata. The proofs are based on sequential machine characterizations of the parallel devices. In particular, deterministic CA and OCA can be sped-up from $(n + t(n))$-time to $(n + \frac{t(n)}{k})$-time [1,13]. Thus, lineartime is close to realtime. The question whether every lineartime CA can be sped-up to realtime is an open problem. The problem is solved for OCA. The realtime OCA languages are a proper subfamily of the lineartime OCA languages [4,20].

Next we are going to derive a stronger result for SVOCA from which follows that realtime is as powerful as lineartime. The result follows from the characterization of Theorem 4 and known results for NCA and NOCA [2], where the

so-called *packing-and-checking* technique is introduced and used. The basic principle is to guess the input in a packed form on the left of the array. Then the verification of the guess can be done by a deterministic OCA in realtime.

Theorem 5. *Let $k \geq 1$ and t be a time-computable time complexity. Then $\mathscr{L}_{k\cdot t}(SVCA) = \mathscr{L}_t(SVCA)$. The same is true for one-way devices.*

Proof. Given a $(k \cdot t)$-time SVCA M, there are $(k \cdot t)$-time NCA M_1 and M_2 with $L(M_1) = L(M)$ and $L(M_1) = \overline{L(M)}$ by Theorem 4. Both can be sped-up to t-time as shown in [2]. Applying Theorem 4 again yields a t-time SVCA that accepts $L(M)$. The reasoning for one-way devices is similar. □

In particular, we have:

Corollary 6. *The families $\mathscr{L}_{rt}(SVOCA)$ and $\mathscr{L}_{lt}(SVOCA)$ coincide and the families $\mathscr{L}_{rt}(SVCA)$ and $\mathscr{L}_{lt}(SVCA)$ coincide.*

4 Self-verifying One-Way Cellular Automata

4.1 Computational Capacity

First we recall that Example 1 gives a witness for the strictness of the following inclusion.

Theorem 7. *The family $\mathscr{L}_{rt}(OCA)$ is properly included in $\mathscr{L}_{rt}(SVOCA)$.*

The inclusion of the previous result can be pushed higher in the hierarchy of language families. However, the strictness of the inclusion gets lost. The question of the strictness is strongly related to the famous open problem whether or not the realtime CA languages are a proper subfamily of the CA languages.

Theorem 8. *The family $\mathscr{L}_{lt}(CA)$ is included in $\mathscr{L}_{rt}(SVOCA)$.*

Proof. Let $L \in \mathscr{L}_{lt}(\mathrm{CA})$. Since the family $\mathscr{L}_{lt}(\mathrm{CA})$ is closed under reversal [19], there exists a lineartime CA accepting L^R. This CA, in turn, can be sped-up by a multiplicative and additive constant [13]. Hence there is a CA $M = \langle S, \Sigma, F, \#, \delta\rangle$ that accepts L^R with time complexity $2n-1$.

First a deterministic OCA $M' = \langle S', \Sigma', F, \#, \delta'\rangle$ is constructed such that M' accepts the language $\{ \sqcup^{|w|} w^R \mid w \in L(M) \}$ with time complexity $2n-2$, (where $\sqcup \notin S$ and $n > 1$): Let $S' = (S \cup \{\sqcup\}) \cup (S \cup \{\sqcup\})^2$, $A' = A \cup \{\sqcup\}$, and $\forall s_1, s_2 \in S \cup \{\sqcup\}$: let $\delta'(s_1, \#) = (s_1, \sqcup)$ and $\delta'(s_1, s_2) = (s_1, s_2)$. Furthermore $\forall (s_1, s_2), (s_2, s_3) \in (S \cup \{\sqcup\})^2$:

$$\delta'((s_1, s_2), (s_2, s_3)) = \begin{cases} \delta(s_3, s_2, s_1) & \text{if } (s_1 \neq \sqcup \wedge s_2 \neq \sqcup \wedge s_3 \neq \sqcup) \\ \delta(\#, s_2, s_1) & \text{if } (s_1 \neq \sqcup \wedge s_2 \neq \sqcup \wedge s_3 = \sqcup) \\ \delta(s_3, s_2, \#) & \text{if } (s_1 = \sqcup \wedge s_2 \neq \sqcup \wedge s_3 \neq \sqcup) \\ \sqcup & \text{otherwise} \end{cases}.$$

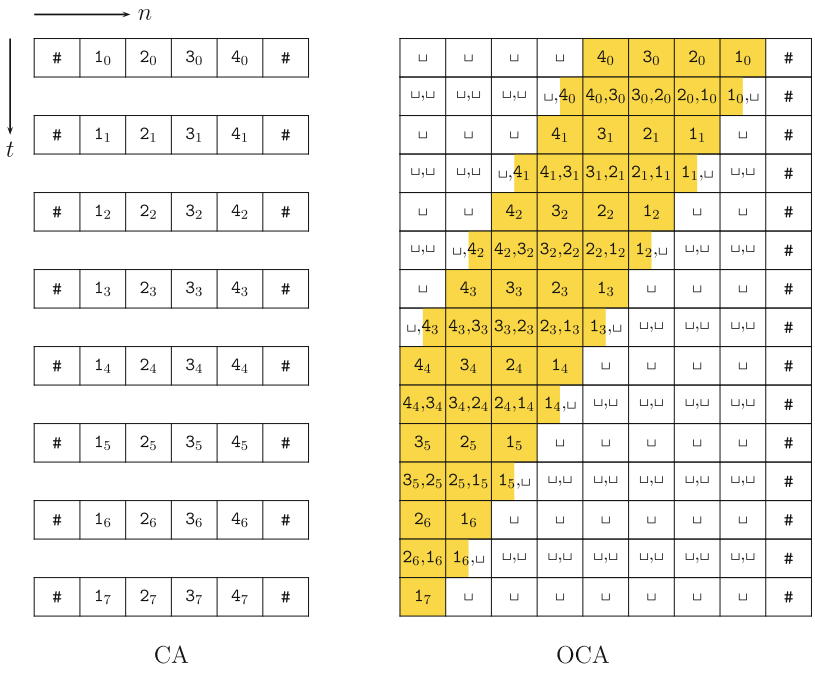

Fig. 2. Example for the proof of Theorem 8.

The basic idea is that during an intermediate step the cells of M' are collecting the information needed to simulate one step of the CA (see Fig. 2). Due to the one-way information flow a cell i thereby can collect information from the cells $i+1$ and $i+2$ and, thus, simulate one step of the CA cell $i+1$. Therefore, the relevant part of the configuration shifts in space to the left.

The cells of an SVOCA M'' that accepts the language $\{ w^R \mid w \in L(M) \}$ are constructed such that they can store two input symbols. Under input w^R the SVOCA M'' guesses in its first step the configuration $\sqcup^{|w|} w^R$ whereby two adjacent symbols are stored in one cell, respectively. The subsequent verification of the guess can be done by a deterministic realtime OCA as shown by the packing-and-checking technique in [2]. In parallel to the verification M'' simulates the OCA M' with double speed on the compressed input. Therefore, M'' has time complexity $1 + \frac{2n-2}{2} = n$. Since $L(M'') = \{ w^R \mid w \in L(M) \} = L^R(M) = (L^R)^R = L$ the theorem follows.

In order to make M'' self-verifying it enters accepting states if the guesses are correct and the simulation ends accepting, and enters rejecting states when the guesses are correct and the simulation does *not* end in an accepting state. All other states, in particular those entered in case of wrong guesses, are neutral. □

4.2 Closure Properties

This section is devoted to the closure properties of the family of realtime SVOCA languages, summarized in Table 1. From above we know already that the family of languages accepted by realtime SVOCA is closed under complementation, union, and intersection.

It is known that $\mathscr{L}_{rt}(\text{OCA})$ is closed under reversal [4], which is a long-standing open problem for $\mathscr{L}_{rt}(\text{CA})$.

Proposition 9. *The family of languages accepted by realtime SVOCA is closed under reversal.*

Proposition 10. *The family of languages accepted by realtime SVOCA is closed under concatenation.*

Proof. Let $L_1, L_2 \in \mathscr{L}_{rt}(\text{SVOCA})$. If the empty word belongs to L_1 then language L_2 belongs to the concatenation and vice versa. Since the family of languages accepted by realtime SVOCA is closed under union, it remains to consider languages $L_1, L_2 \in \mathscr{L}_{rt}(\text{SVOCA})$ that do not contain the empty word. Let M_1, M_2 be acceptors for L_1 and L_2. As an intermediate step, we construct a self-verifying cellular automaton M with two-way information flow, that is, each cell is connected to its both nearest neighbors and the leftmost cell receives a boundary symbol on its free input line.

Since the family $\mathscr{L}_{rt}(\text{SVOCA})$ is closed under reversal, there is a realtime SVOCA M_1^R that accepts the reversal L_1^R of L_1. Now M has two tracks with identical inputs. On one track it simulates M_2, whereby each cell that enters an accepting or rejecting state is marked accordingly. On the second track, M simulates M_1^R from left to right. That is, the simulation is such that each cell receives the state from its left neighbor. So, the information flow is from left to right. Again, each cell that enters an accepting or rejecting state is marked accordingly.

Let the input be $x_1x_2\cdots x_n$. If a cell at position i is marked accepting by the simulation of M_2, the word $x_ix_{i+1}\cdots x_n$ belongs to the language L_2. If a cell at position i is marked accepting by the simulation of M_1^R, the word $x_ix_{i-1}\cdots x_1$ belongs to the language L_1^R and, thus, $x_1x_2\cdots x_i$ belongs to the language L_1. So, the input $x_1x_2\cdots x_n$ belongs to the concatenation L_1L_2 if and only if M_1^R may mark a cell at position i and M_2 a cell at position $i+1$ accepting, for $1 \leq i < n$.

In order to check this condition, M uses a signal that is emitted from the rightmost cell when the simulation of M_1^R reaches that cell at time step n. The signal moves to the left and informs the leftmost cell at time step $2n$.

When the signal arrives, the leftmost cell enters an accepting state if and only if the signal has found two adjacent cells marked accepting. So, M accepts any input from L_1L_2 and only inputs from the concatenation L_1L_2. If the signal found neither two adjacent cells marked accepting, nor two adjacent cells that are marked accepting and unmarked, nor two adjacent cells unmarked the leftmost cell enters a rejecting state. In this case, no matter between which two adjacent symbols one assumes the cut between first and second factor, M has explicitly

rejected at least one of them. Clearly, in this case the input cannot belong to the concatenation. On the other hand, if some input does not belong to the concatenation, then there is always a computation of M that results in such a marking. So, M rejects any input that does not belong to L_1L_2 and only inputs that do not belong to L_1L_2. In any other case, the leftmost cell remains in a neutral state.

So far, we have constructed a two-way self-verifying cellular automaton with time complexity $2n$. The proof of Theorem 8 can almost literally be used to show that also a lineartime two-way self-verifying cellular automaton can be simulated by a realtime SVOCA. □

Next, we turn to the operations homomorphism and inverse homomorphism.

Proposition 11. *The family of languages accepted by realtime SVOCA is not closed under homomorphisms.*

Proof. It is well known that every recursively enumerable language is the homomorphic image of the intersection of two context-free languages [10]. Moreover, every context-free language is the homomorphic image of the intersection of a regular language and a Dyck language [5].

The Dyck languages as well as the regular languages are realtime OCA languages [7] and therefore realtime SVOCA languages. Additionally, the family of realtime SVOCA languages is closed under intersection. So, if the family $\mathscr{L}_{rt}$(SVOCA) would be closed under homomorphisms, it would contain every recursively enumerable language. Due to the time bound to realtime this is a contradiction. □

Proposition 12. *The family of languages accepted by realtime SVOCA is closed under inverse homomorphisms.*

The closure of $\mathscr{L}_{rt}$(SVOCA) with respect to Kleene star and non-erasing homomorphisms are not known. They are settled for nondeterministic devices since, basically, for iteration it is sufficient to guess the the positions in the array at which words are concatenated, and for non-erasing homomorphism it is sufficient to guess the pre-image of the input. However, self-verifying devices have to reject explicitly if the input does not belong to the language. It seems that they have to 'know' that all choices either do not lead to accepting computations or are 'wrong.'

Table 1. Closure properties of the language family $\mathscr{L}_{rt}$(SVOCA) in comparison with the family $\mathscr{L}_{rt}$(OCA), where h_λ denotes λ-free homomorphisms.

Family	—	$\cup$	$\cap$	R	$\cdot$	$*$	h_λ	h	h^{-1}
$\mathscr{L}_{rt}$(SVOCA)	Yes	Yes	Yes	Yes	Yes	**?**	**?**	No	Yes
$\mathscr{L}_{rt}$(OCA)	Yes	Yes	Yes	Yes	No	No	No	No	Yes

4.3 Decidability Questions

Now we turn to decidability questions. The membership problem is decidable for realtime SVOCA languages since the family is effectively included in the deterministic context-sensitive languages. However, even realtime OCA can accept the so-called valid computations of Turing machines. These are languages of encodings of accepting Turing machine computations (see [18] for details or [15] for a survey). Hence many of the not even semi-decidable problems for Turing machines can be reduced to realtime OCA (see [18] for details or [15] for a survey). The following theorem is from [15].

Theorem 13. *For any language family that effectively contains $\mathscr{L}_{rt}(OCA)$ the problems emptiness, universality, finiteness, infiniteness, context-freeness, and regularity are not semidecidable.*

So, we have the following consequences.

Corollary 14. *The problems emptiness, universality, finiteness, infiniteness, inclusion, equivalence, regularity, and context-freeness are not semidecidable for realtime SVOCA.*

Finally, we turn to the problem to decide whether a given realtime nondeterministic one-way cellular automaton is self-verifying or not.

Theorem 15. *Given a realtime deterministic one-way cellular automaton M, it is not semidecidable whether or not M is an SVOCA.*

Proof. Let M be a realtime OCA with accepting states F. An equivalent realtime SVOCA M' is constructed (Lemma 2). Next, M' is modified by adding a new input symbol (and neutral state) $\boxminus$ and new states $\ominus \in F_-$ and $\oplus \in F_+$. The transition functions are modified such that a cell in state $\boxminus$ in the first step nondeterministically can either change to $\ominus$ and remain in that state forever or to stay in $\boxminus$ unless its right neighbor is in an accepting state. In the latter case, the cell changes from state $\boxminus$ to $\oplus$ and stays in that state from then on.

We claim that M' is self-verifying if and only if $L(M')$ is empty. If $L(M')$ is empty, none of its cells will ever enter an accepting state. So, a cell that is in state $\ominus$ remains in $\ominus$ and, thus, will not give a contradictory answer. On the other hand, if there is $w \in L(M')$, then on input $\boxminus w$ by the choices of the leftmost cell there is a rejecting computation, but an accepting one as well. Therefore, in this case, M' is not self-verifying. If it were semidecidable whether a realtime OCA is self-verifying then one could semidecide emptiness contradicting Corollary 14. □

By Lemma 2 any deterministic CA with a time-computable time complexity can effectively be made self-verifying. But it is non-semidecidable whether it already *is* self-verifying. That generalizes immediately to nondeterministic cellular automata. However, Lemma 2 does not since an input may induce accepting as well as non-accepting computations, which would become rejecting. In fact, it is an open problem whether the family of realtime one-way nondeterministic cellular automata is closed under complementation or not.

References

1. Bucher, W., Čulik II, K.: On real time and linear time cellular automata. RAIRO Inform. Théor. **18**, 307–325 (1984)
2. Buchholz, T., Klein, A., Kutrib, M.: On interacting automata with limited nondeterminism. Fundam. Inform. **52**, 15–38 (2002)
3. Buchholz, T., Kutrib, M.: On time computability of functions in one-way cellular automata. Acta Inform. **35**, 329–352 (1998)
4. Choffrut, C., Čulik II, K.: On real-time cellular automata and trellis automata. Acta Inform. **21**, 393–407 (1984)
5. Chomsky, N.: Context-free grammars and pushdown storage. Tech report, QPR 65, Massachusetts Institute of Technology (1962)
6. Duriš, P., Hromkovič, J., Rolim, J.D.P., Schnitger, G.: Las Vegas versus determinism for one-way communication complexity, finite automata, and polynomial-time computations. In: Reischuk, R., Morvan, M. (eds.) STACS 1997. LNCS, vol. 1200, pp. 117–128. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0023453
7. Dyer, C.R.: One-way bounded cellular automata. Inf. Control **44**, 261–281 (1980)
8. Fernau, H., Kutrib, M., Wendlandt, M.: Self-verifying pushdown automata. In: Freund, R., Mráz, F., Průša, D. (eds.) Non-Classical Models of Automata and Applications (NCMA 2017), vol. 329, pp. 103–117. Austrian Computer Society, Vienna (2017). books@ocg.at
9. Fischer, P.C., Kintala, C.M.R.: Real-time computations with restricted nondeterminism. Math. Syst. Theory **12**, 219–231 (1979)
10. Ginsburg, S., Greibach, S.A., Harrison, M.A.: One-way stack automata. J. ACM **14**, 389–418 (1967)
11. Hromkovic, J., Schnitger, G.: On the power of Las Vegas for one-way communication complexity, OBDDs, and finite automata. Inf. Comput. **169**, 284–296 (2001)
12. Hromkovic, J., Schnitger, G.: Nondeterministic communication with a limited number of advice bits. SIAM J. Comput. **33**, 43–68 (2003)
13. Ibarra, O.H., Kim, S.M., Moran, S.: Sequential machine characterizations of trellis and cellular automata and applications. SIAM J. Comput. **14**, 426–447 (1985)
14. Jirásková, G., Pighizzini, G.: Optimal simulation of self-verifying automata by deterministic automata. Inf. Comput. **209**, 528–535 (2011)
15. Kutrib, M.: Cellular automata and language theory. In: Meyers, R. (ed.) Encyclopedia of Complexity and System Science, pp. 800–823. Springer, Heidelberg (2009). https://doi.org/10.1007/978-0-387-30440-3
16. Kutrib, M.: Non-deterministic cellular automata and languages. Int. J. Gen. Syst. **41**, 555–568 (2012)
17. Kutrib, M., Worsch, Th.: Self-verifying cellular automata. Technical report, 1803, Universität Gießen (2018). http://www.informatik.uni-giessen.de/reports/Report1803.pdf
18. Malcher, A.: Descriptional complexity of cellular automata and decidability questions. J. Autom. Lang. Comb. **7**, 549–560 (2002)
19. Smith III, A.R.: Real-time language recognition by one-dimensional cellular automata. J. Comput. Syst. Sci. **6**, 233–253 (1972)
20. Umeo, H., Morita, K., Sugata, K.: Deterministic one-way simulation of two-way real-time cellular automata and its related problems. Inf. Process. Lett. **14**, 158–161 (1982)