Acta Informatica (2021) 58:263–279
https://doi.org/10.1007/s00236-021-00396-9

ORIGINAL ARTICLE

Reversible parallel communicating finite automata systems

Henning Bordihn[1] · György Vaszil[2]

Received: 3 May 2020 / Accepted: 16 January 2021

Abstract
We study the concept of reversibility in connection with parallel communicating systems of finite automata (PCFA in short). We define the notion of reversibility in the case of PCFA (also covering the non-deterministic case) and discuss the relationship of the reversibility of the systems and the reversibility of its components. We show that a system can be reversible with non-reversible components, and the other way around, the reversibility of the components does not necessarily imply the reversibility of the system as a whole. We also investigate the computational power of deterministic centralized reversible PCFA. We show that these very simple types of PCFA (returning or non-returning) can recognize regular languages which cannot be accepted by reversible (deterministic) finite automata, and that they can even accept languages that are not context-free. We also separate the deterministic and non-deterministic variants in the case of systems with non-returning communication. We show that there are languages accepted by non-deterministic centralized PCFA, which cannot be recognized by any deterministic variant of the same type.

Keywords Finite automata · Reversibility · Systems of parallel communicating automata

1 Introduction

Parallel communicating finite automata (PCFA) are systems of several finite state automata processing the same input word in an autonomous and synchronized way. In certain situations, automata of the system, depending on the state reached, may request the state of another automaton. The concept of PCFA has been introduced in [17] and further investigated in,

Research supported in part by Project No. K 120558, implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the K 16 funding scheme.

✉ Henning Bordihn
henning@cs.uni-potsdam.de

György Vaszil
vaszil.gyorgy@inf.unideb.hu

1 Institut für Informatik, Universität Potsdam, August-Bebel-Straße 89, 14482 Potsdam, Germany

2 Department of Computer Science, Faculty of Informatics, University of Debrecen, Kassai út 26, 4028 Debrecen, Hungary

e.g., [3–5,7]. The concept can be viewed as a simple model of computer networks, falling into the line of research as of, e.g., [6] or [11].

In PCFA, spontaneous transitions are allowed so that different automata of a system can read different symbols of the input word at the same time during computations. This is similar to multi-head finite automata [20] which are shown to be computationally equivalent to PCFA, see [17] and [7] for the non-deterministic case and [4] for the deterministic case. Inspired by parallel communicating grammar systems which have been investigated as string generating variants of the classroom model known from artificial intelligence [8], one distinguishes between the returning and non-returning working modes. In PCFA, these working modes refer to the state in which an automaton continues its computation after it has sent its state to another automaton upon request: In the returning mode, the automaton is set back to its initial state, while in the non-returning mode, it just keeps the state it has communicated to another automaton. An interesting special case is centralized PCFA, where only one designated automaton is allowed to send communication requests to other automata of the system. This models a particular, star-like network architecture with restricted power, leading to subclasses in the hierarchy of language families, strict in most cases [4].

In the present paper, the concept of reversibility is added to PCFA. Roughly, reversibility means that no information is lost during the computation. This is of interest due to the physical observation that loss of information involves heat dissipation [2,16]. In abstract models, a computation is reversible if every configuration has a unique predecessor configuration. Thus, in deterministic devices the computation step relation is injective. For example, reversible Turing machines have been investigated in [2] where it is shown that for every Turing machine, there is an equivalent reversible one. Many other automata models have been considered under the restriction of reversibility such as the massively parallel cellular automata, see, e.g., [18], pushdown automata [13], their input driven variants [15], two-way multi-head finite automata [1,19] and one-way multi-head finite automata [14]. In [10], reversible non-deterministic finite automata have been considered. Different aspects of reversibility in a setting of computing devices with a finite number of discrete internal states and a read-only input tape are addressed in [12]. This work also contains many further references about reversibility in automata models.

PCFA were also considered from the point of view of reversibility in [9] where the authors showed that parallel communicating systems of reversible (deterministic) finite automata are able to accept all regular languages. They also considered an intuitive notion of the reversibility of PCFA systems when they remarked that such a system is not necessarily reversible as a whole, although all of its components are reversible.

In the present paper, we first give a definition of the notion of reversibility for PCFA by formalizing the way the predecessor configurations are computed, particularly in the case when communication may have happened. We also discuss the relation between the reversibility of parallel communicating systems and the reversibility of their components, and then, we focus on the computational power of reversible systems. Throughout the paper, both the deterministic and non-deterministic variants are taken into consideration.

2 Definitions and basic properties

We assume that the reader is familiar with basic concepts and terminology of automata theory as it can be found, e.g., in [22]. We shall use the following notation. For a set S, its powerset is designated by 2^S and its cardinality by $|S|$. The symbols $\subseteq$ and $\subset$ are used for set inclusion and strict set inclusion, respectively. For a word w, its length is designated by $|w|$, $w(i)$ denotes the i-th symbol of w, for $1 \leq i \leq |w|$, and w^R is the reverse (mirror image) of w. The symbol λ is used for the empty word. Let Σ be an alphabet. The notation Σ^* is used for the set of all words over Σ including λ, while $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$.

Let $A = (S, \Sigma, s_0, \delta, F)$ be a non-deterministic finite automaton (NFA), where S is the finite set of states, Σ is the alphabet of input symbols, $s_0 \in S$ is the initial state, $\delta : S \times \Sigma \to 2^S$ is the transition function and $F \subseteq S$ is the set of accepting states. The automaton is deterministic if $|\delta(s, a)| = 1$ for all $s \in S$ and $a \in \Sigma$. A configuration of A is a pair in $S \times \Sigma^*$. A computation of A on input word w is a sequence of configurations $c_0, c_1, \ldots c_k$, for some $k \geq 0$ such that c_0 is the initial configuration (s_0, w), and $c_i = (s, v)$ if $c_{i-1} = (q, av)$ and $s \in \delta(q, a)$, for $1 \leq i \leq k$. We write $c_{i-1} \vdash c_i$, $1 \leq i \leq k$.

An NFA $A = (S, \Sigma, s_0, \delta, F)$ is *reversible* according to [10] if there is no internal state $s \in S$, such that $s \in \delta(p, a) \cap \delta(r, a)$ holds for some $a \in \Sigma$ and states $p, r \in S$ with $p \neq r$.

A *non-deterministic parallel communicating finite automata system of degree k* (PCFA(k)) is a tuple $\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_k, Q, \lhd)$ where Σ is the input alphabet, $A_i = (S_i, \Sigma, \delta_i, s_{0,i}, F_i)$ for each i, $1 \leq i \leq k$, is a non-deterministic finite automaton (the ith component of $\mathcal{A}$) with S_i being the set of states, $s_{0,i} \in S_i$ the initial state, $F_i \subseteq S_i$ the set of accepting states, and $\delta_i : S_i \times (\Sigma \cup \{\lambda, \lhd\}) \to 2^{S_i}$ the transition function. Further, $Q \subseteq \{q_1, q_2, \ldots, q_k\}$ is the set of query states, $Q \subseteq \bigcup_{i=1}^{k} S_i$, and $\lhd \notin \Sigma$ is the end-of-input symbol.

A *configuration* $c = (w, s_1, p_1, \ldots, s_k, p_k) \in \Sigma^* \times S_1 \times \mathbb{N} \times \ldots \times S_k \times \mathbb{N}$ of $\mathcal{A}$ represents the tape contents with input word $w \in \Sigma^*$, the current states $s_i \in S_i$, and the current positions $p_i \geq 1$ of the reading heads of the component automata A_i, $1 \leq i \leq k$, where $p_i > |w|$ means that A_i is scanning the end-of-input symbol $\lhd$. For a word w and an integer $i \geq 1$, let $w(i)$ denote the i-th symbol of w if $1 \leq i \leq |w|$, and $w(i) = \lhd$ if $i > |w|$.

An initial configuration is of the form $(w, s_{0,1}, 1, s_{0,2}, 1, \ldots, s_{0,k}, 1)$, that is, each component is in its initial state and scans the first symbol of an input word $w \in \Sigma^*$. The definition of the successor configuration relation $\vdash_A$ of A is defined as follows. (We may omit the subscript A in $\vdash_A$ if there is no risk of confusion.) For any $w \in \Sigma^*$,

$$(w, s_1, p_1, s_2, p_2, \ldots, s_k, p_k) \vdash_A (w, s'_1, p'_1, s'_2, p'_2, \ldots, s'_k, p'_k),$$

if either

1. *(Reading step)* $Q \cap \{s_1, s_2, \ldots, s_k\} = \emptyset$, $s'_i \in \delta_i(s_i, a_i)$ with $a_i \in \{w(p_i), \lambda\}$, and $p'_i = p_i$ if $a_i = \lambda$ and $p'_i = p_i + 1$ otherwise, $1 \leq i \leq k$, or
2. *(Communication step)* for all i, $1 \leq i \leq k$, we have $p'_i = p_i$ and, if $s_i = q_j$ with $q_j \in Q$ and $s_j \notin Q$, then $s'_i = s_j$; furthermore,
   (a) $s'_r = s_r$ for all the other r, $1 \leq r \leq k$ (*non-returning communication*) or, alternatively,
   (b) $s'_j = s_{0,j}$, and $s'_r = s_r$ for all the other r, $1 \leq r \leq k$ (*returning communication*).

In a reading step, all components are in non-query states and perform an ordinary (non-communicating) step independently (which may also be a λ-step). Notice that components with head positions greater than the input length will always scan the end-of-input symbol in non-λ reading steps. In a communication step, components in query states receive the requested states as long as the sender is not in a query state itself. This process is repeated until all requests are resolved, if possible. If the requests are cyclic, no successor configuration exists. As mentioned above, we distinguish *non-returning* communication, that is, the sender remains in its current state, and *returning* communication, that is, the sender is reset to its initial state. Both in returning and non-returning communication steps, the heads do not move. In what follows, every PCFA consistently work with either returning or non-returning communication steps; thus, we either have returning or non-returning systems.

A computation *halts* when the successor configuration is not defined for the current situation. In particular, this may happen when cyclic communication requests appear, or when the transition function of one component is not defined. (We regard the transition function as undefined whenever it maps to the empty set.) The language $L(\mathcal{A})$ accepted by a PCFA(k) $\mathcal{A}$ is precisely the set of words w such that there is some computation beginning with w on the input tape and halting with at least one component having an undefined transition function and being in an accepting state. Let $\vdash_A^*$ denote the reflexive and transitive closure of $\vdash_A$ and set $L(\mathcal{A}) = \{ w \in \Sigma^* \mid (w, s_{0,1}, 1, s_{0,2}, 1, \ldots, s_{0,k}, 1) \vdash_A^* (w, t_1, r_1, t_2, r_2, \ldots, t_k, r_k)$, for some i, $1 \leq i \leq k$, such that $t_i \in F_i$ and $\delta_i(t_i, w(r_i))$ is undefined $\}$.

If all components A_i are deterministic finite automata, that is, for all $s \in S_i$ and for all $a \in \Sigma$, the transition function $\delta_i(s, a)$ maps to a set of at most one state and is undefined whenever $\delta_i(s, \lambda)$ is defined, then the whole system is called *deterministic*, and we add the prefix D (to PCFA) to denote it. The absence or presence of an R in the type of the system denotes whether it works in *non-returning* or *returning* mode, respectively. Finally, if there is just one component, say A_1, that is allowed to query for states, that is, $S_i \cap Q = \emptyset$, for $2 \leq i \leq k$, then the system is said to be *centralized*. In this case, we refer to A_1 as the *master component* and add a C to the notation of the type of the system. Whenever the degree is missing, we mean systems of arbitrary degree. The *family of languages accepted* by devices of type X (with degree k) is denoted by $\mathscr{L}(X)$ $(\mathscr{L}(X(k)))$.

For the definition of reversibility of a PCFA, we introduce the following notion. A configuration $(w, t_1, r_1, t_2, r_2, \ldots, t_k, r_k)$ is *reachable* by a PCFA A, if there is a computation $(w, s_{0,1}, 1, s_{0,2}, 1, \ldots, s_{0,k}, 1) \vdash_A^* (w, t_1, r_1, t_2, r_2, \ldots, t_k, r_k)$.

Definition 1 A PCFA(k) is *reversible* if considering any reachable configurations $(w, s_1, p_1, \ldots, s_k, p_k)$ and $(w, s'_1, p'_1, \ldots, s'_k, p'_k)$, if there are computational steps

$$(w, t_1, r_1, \ldots, t_k, r_k) \vdash (w, s_1, p_1, \ldots, s_k, p_k)$$

and

$$(w, t'_1, r'_1, \ldots, t'_k, r'_k) \vdash (w, s'_1, p'_1, \ldots, s'_k, p'_k),$$

then the following holds:

- If $(s_1, \ldots, s_k) = (s'_1, \ldots, s'_k)$, then $p_i - r_i = p'_i - r'_i$ for all i, $1 \leq i \leq k$, and moreover,
- if $w(r_i) = w(r'_i)$ for all i with $p_i \neq r_i$, $1 \leq i \leq k$, then $(t_1, \ldots, t_k) = (t'_1, \ldots, t'_k)$.

Intuitively, if two reachable configurations cannot be distinguished by the states reached, then it is determined which positions have been increased in the previous computation step and which have not. This way one can identify the symbols which have been read in the previous computation step. If the read symbols are identical, then also the states of the two predecessor configurations have been the same.

In the definition, one might alternatively take *all* configurations into account instead of only reachable ones. Due to the motivation of the concept of reversibility, computations should be reversible that can be performed by the machine model. Therefore, we restrict to reachable configurations in the definition.

For all the language classes $\mathcal{L}(X\text{PCFA})$ and $\mathcal{L}(X\text{PCFA}(k))$ with $X \in \{\lambda, \text{R}, \text{C}, \text{DR}, \text{DC}, \text{DRC}\}$, we add the prefix "rev" in order to refer to the corresponding families of languages accepted by reversible PCFA.

Example 1 Consider the deterministic non-returning PCFA with two components (DCPCFA(2) in short) $\mathcal{A} = (\{a, b, \$, ¢\}, A_1, A_2, \{q_2\}, \lhd)$ where for $i \in \{1, 2\}$

$$A_i = (S_i, \{a, b, \$, ¢\}, \delta_i, s_{0,i}, F_i)$$

with

$$\begin{aligned} S_1 &= \{s_{0,1}, s_{\mathrm{acc}}, s_1, s_a, s_b, s_\$, s_\lhd, q_2\}, & F_1 &= \{s_{\mathrm{acc}}\},\\ S_2 &= \{s_{0,2}, s_1, s_a, s_b, s_\$, s_\lhd, s_f\}, & F_2 &= \emptyset, \end{aligned}$$

$$\begin{array}{rrr} \delta_1(s_{0,1}, \lambda) = q_2, & \delta_1(s_\$, ¢) = q_2, & \delta_1(s_\lhd, \$) = s_{\mathrm{acc}},\\ \delta_1(s_1, \lambda) = q_2, & \delta_1(s_a, a) = q_2, & \\ & \delta_1(s_b, b) = q_2, & \end{array}$$

and

$$\begin{array}{rrr} \delta_2(s_{0,2}, ¢) = s_1, & \delta_2(s_\$, a) = s_a, & \delta_2(s_\$, \lhd) = s_\lhd,\\ \delta_2(s_1, a) = s_1, & \delta_2(s_a, a) = s_a, & \delta_2(s_a, \lhd) = s_\lhd,\\ \delta_2(s_1, b) = s_1, & \delta_2(s_b, a) = s_a, & \delta_2(s_b, \lhd) = s_\lhd,\\ \delta_2(s_1, \$) = s_\$, & \delta_2(s_\$, b) = s_b, & \delta_2(s_\lhd, \lambda) = s_f.\\ & \delta_2(s_a, b) = s_b, & \\ & \delta_2(s_b, b) = s_b, & \end{array}$$

We show that $\mathcal{A}$ is reversible and it accepts the language $L(\mathcal{A}) = \{¢w\$w \mid w \in \{a, b\}^*\}$.

To see how $\mathcal{A}$ works, consider an initial configuration $(\alpha, s_{0,1}, 1, s_{0,2}, 1)$. Notice that $\alpha = ¢\alpha'$ or the second component cannot start reading the input which results in a non-accepting blocking configuration. Moreover, α' contains exactly one \$ symbol or the second component cannot reach the end-of-input symbol $\lhd$, so the first component (hence, the system) cannot enter the accepting state. This means that the input string must be of the form $¢w\$w'$ with $w, w' \in \{a, b\}^*$.

The transition relations are defined in such a way that the computation must start with

$$(¢w\$w', s_{0,1}, 1, s_{0,2}, 1) \vdash (¢w\$w', q_2, 1, s_1, 2) \vdash (¢w\$w', s_1, 1, s_1, 2) \vdash \ldots$$

and then

$$\ldots \vdash (¢w\$w', s_1, 1, s_1, k) \vdash (¢w\$w', q_2, 1, s_\$, k + 1) \vdash (¢w\$w', s_\$, 1, s_\$, k + 1)$$

where $¢w\$w'(k) = \$$, that is, the kth symbol of the input is the \$ symbol and it has just been read by the second component.

We can check that this phase of the computation is reversible by considering the following.

– If the states of the components are (s_1, s_1) or $(s_\$, s_\$)$, then the previous computational step must have been a communication step, so for any configuration $(¢w\$w', s_1, i, s_1, j)$ or $(¢w\$w', s_\$, i, s_\$, j)$ with $i, j \geq 1$, the previous configuration must have been $(¢w\$w', q_2, i, s_1, j)$ or $(¢w\$w', q_2, i, s_\$, j)$, respectively.

– If the states of the components are (q_2, s_1), then the previous computational step must have been a reading step where the second component made a non-λ-move. As the second component is in s_1, it has never sent a state from $\{s_\$, s_a, s_b\}$ to the first component by a communication step. Therefore, the first component must have entered state q_2 by a λ-move. If the symbol read by the second component was $¢$, then for any configuration $(¢w\$w', q_2, i, s_1, j)$ with $i \geq 1$, $j > 1$, the previous configuration must have been $(¢w\$w', s_{0,1}, i, s_{0,2}, j - 1)$ which is only reachable if it is the initial configuration $(¢w\$w', s_{0,1}, 1, s_{0,2}, 1)$. If the symbol read by the second component was a or b, then the previous configuration is $(¢w\$w', s_1, i, s_1, j - 1)$.

– Similarly, if the states of the components are $(q_2, s_\$)$, then the previous computational step must have been a reading step where the first component made a λ-move, and the second component made a non-λ-move. The symbol read by the second component must have been a \$, so the previous configuration for any $(¢w\$w', q_2, i, s_\$, j)$ with $i, j \geq 1$, is $(¢w\$w', s_1, i, s_1, j-1)$.

Notice that $i = 1$ in any reachable configuration of these forms.

Continuing the computation, we have

$$(¢w\$w', s_\$, 1, s_\$, k+1) \vdash (¢w\$w', q_2, 2, s_x, k+2) \vdash (¢w\$w', s_x, 2, s_x, k+2)$$

where $x \in \{a, b, \lhd\}$ depending on the $(k+1)$st symbol on the input tape. First, let $x \in \{a, b\}$. If the second symbol of the input is also x, the computation can continue with

$$(¢w\$w', s_x, 2, s_x, k+2) \vdash (¢w\$w', q_2, 3, s_{x'}, k+3) \vdash (¢w\$w', s_{x'}, 3, s_{x'}, k+3)$$

where $x' \in \{a, b, \lhd\}$ now depends on the $(k+2)$nd symbol on the input tape. Repeating similar steps, the system now is able to check whether $w = w'$ in the input string $¢w\$w'$. The first component can only read the jth symbol of w (which is the $(1+j)$th symbol on the tape), if it is the same as the jth symbol of w' (which is the $(k+j)$th symbol on the tape) that was read in the previous non-communication step by the second component and transferred in the previous communication step to the first component as the state s_x, $x \in \{a, b, \lhd\}$.

This phase of the computation is also reversible, as can be seen by the following.

– If the states of the components are (s_x, s_x) for some $x \in \{a, b, \lhd\}$, then the previous computational step must have been a communication step, so for any configuration $(¢w\$w', s_x, i, s_x, j)$ with $i, j > 1$, the previous configuration must have been $(¢w\$w', q_2, i, s_x, j)$.
– If the states of the components are (q_2, s_x) for some $x \in \{a, b, \lhd\}$, then the previous computational step must have been a reading step where both components made a non-λ-move. If the symbol read by the first component is $y \in \{a, b, \$\}$, then for any configuration $(¢w\$w', q_2, i, s_x, j)$ with $i, j > 1$, the previous configuration must have been $(¢w\$w', s_y, i-1, s_y, j-1)$. (Note that configurations of the form $(¢w\$w', s_x, i, s_{x'}, j)$ for $x \neq x'$ are not reachable by the system. Moreover, $j = k+i$ holds in every reachable configuration of this form.)

Now, we consider situations in which the second component reaches the end-of-input symbol $\lhd$. If the first component reaches the \$ symbol before the second component reaches the end-of-input symbol $\lhd$, then the system is blocked in a non-accepting state of the form $(¢w\$w', s_x, i, s_x, j)$ where $¢w\$w'(i) = \$$, so in order for the computation to be successful, the second component must reach the end-of-input symbol first through a computational step

$$(¢w\$w', s_x, 1+j, s_x, k+j+1) \vdash (¢w\$w', q_2, 1+j+1, s_\lhd, k+j+2),$$

followed by the communication

$$(¢w\$w', q_2, 1+j+1, s_\lhd, k+j+2) \vdash (¢w\$w', s_\lhd, 1+j+1, s_\lhd, k+j+2).$$

Now, if the first component is scanning an a or b symbol, then the computation is blocked in a non-accepting state. On the other hand, if the first component is scanning the \$ symbol (that is, if the input string is of the form $¢w\$w$), the system reaches the accepting configuration

$$(¢w\$w', s_{\mathrm{acc}}, j+3, s_f, k+j+2)$$

where the \$ symbol is on the kth position of the input string, that is, $j = k - 2$ for $|¢w\$w| = 2k - 2$.

The last step of the computation is also reversible, as having the current states (s_{acc}, s_f) implies that the two components made a non-λ-move and a λ-move, respectively, and the states of the previous configuration must have been $s_\lhd$ for both of them.

The next example shows that a similar language can also be accepted by returning systems.

Example 2 Consider the DRCPCFA(2) $\mathcal{A} = (\{a, b, \$_1, \$_2, ¢\}, A_1, A_2, \{q_2\}, \lhd)$ where

$$A_1 = (S_1, \{a, b, \$_1, \$_2, ¢\}, \delta_1, s_{0,1}, \{s_{\text{acc}}\})$$

with $S_1 = \{s_{0,1}, s'_{0,1}, s_1, s'_1, s_a, s_b, s_{\$_1}, s_¢, s_{\text{acc}}, q_2\}$ and

$$\begin{array}{rrr} \delta_1(s_{0,1}, \lambda) = s'_{0,1}, & \delta_1(s_1, a) = s_1, & \delta_1(s'_1, \lambda) = q_2, \\ \delta_1(s'_{0,1}, ¢) = s_1, & \delta_1(s_1, b) = s_1, & \delta_1(s_¢, \$_2) = q_2, \\ & \delta_1(s_1, \$_1) = s'_1, & \delta_1(s_a, a) = q_2, \\ & & \delta_1(s_b, b) = q_2, \\ & & \delta_1(s_{\$_1}, \lhd) = s_{\text{acc}}. \end{array}$$

Further,

$$A_2 = (S_2, \{a, b, \$_1, \$_2, ¢\}, \delta_2, s_{0,2}, \emptyset)$$

with $S_2 = \{s_{0,2}, s_a, s_b, s_{\$_1}, s_{\$_2}, s_¢\}$ and

$$\begin{array}{rrr} \delta_2(s_{0,2}, ¢) = s_¢, & \delta_2(s_{0,2}, a) = s_a, & \delta_2(s_{0,2}, \$_1) = s_{\$_1}, \\ \delta_2(s_¢, \lambda) = s_¢, & \delta_2(s_{0,2}, b) = s_b, & \delta_2(s_{0,2}, \$_2) = s_{\$_2}. \end{array}$$

We show that $\mathcal{A}$ is reversible and accepts the language $L(\mathcal{A}) = \{¢w\$_1\$_2w \mid w \in \{a, b\}^*\}$.

To see how $\mathcal{A}$ works, consider an initial configuration $(\alpha, s_{0,1}, 1, s_{0,2}, 1)$. Notice that $\alpha = ¢\alpha'$ or the first component cannot perform more than one computational step on the input, which results in a non-accepting blocking configuration of the form $(\alpha, s'_{0,1}, 1, s_x, 2)$ with $x \in \{a, b, \$_1, \$_2\}$. Such computational step is reversible as it can only be reached from the initial configuration according to the definition of δ_1. Moreover, the first component, thus the system, cannot read more than one ¢ symbol, so the leading ¢ is the only such symbol in the input string. Similarly, the first component must read exactly one $\$_1$ symbol, or it cannot reach the accepting state. As a consequence, α' must contain exactly one $\$_2$ symbol, since the first component may read $\$_2$ only after receiving state $s_¢$ from the second component, which can happen only once, as there is only one ¢ symbol in the input string and $\mathcal{A}$ works in returning mode. Note also that the $\$_1$, $\$_2$ symbols must be adjacent, or the first component cannot reach the accepting state. To see this, note that the first component initiates the first query after reading the symbol $\$_1$. Since leftmost input symbol is ¢, the second component must respond to this query by sending the state $s_¢$, and after receiving $s_¢$, the first component can only continue by reading a $\$_2$ symbol. This means that the input string must be of the form $¢\alpha'$ where α' contains exactly one occurrence of the substring $\$_1\$_2$.

The transition relations are defined in such a way that the computation must start with

$$(¢w\$_1\$_2w', s_{0,1}, 1, s_{0,2}, 1) \vdash (¢w\$_1\$_2w', s'_{0,1}, 1, s_¢, 2) \vdash (¢w\$_1\$_2w', s_1, 2, s_¢, 2)$$

and then

$$\begin{array}{l} (¢w\$_1\$_2w', s_1, 2, s_¢, 2) \vdash \ldots \vdash (¢w\$_1\$_2w', s_1, k, s_¢, 2) \vdash \\ (¢w\$_1\$_2w', s'_1, k+1, s_¢, 2) \vdash (¢w\$_1\$_2w', q_2, k+1, s_¢, 2) \vdash \end{array}$$

$$(¢w\$_1\$_2w', s_{¢}, k+1, s_{0,2}, 2)$$

where $¢w\$_1\$_2w'(k) = \$_1$, that is, the kth symbol of the input is the $\$_1$ symbol and it has just been read by the first component.

To see that this phase of the computation is reversible, consider the following.

– If the states of the components are $(s'_{0,1}, s_{¢})$, then the previous computational step must have been a reading step where the first component made a λ-move, and the second component made a non-λ-move, so for any configuration $(¢w\$_1\$_2w', s'_{0,1}, i, s_{¢}, j)$ with $i, j \geq 1$, the previous configuration must have been $(¢w\$_1\$_2w', s_{0,1}, i, s_{0,2}, j-1)$. (This actually must be the initial configuration of the system, since $s_{0,1}$ cannot be reached by the first component later during the computation.)
– If the states of the components are $(s_1, s_{¢})$, then the previous computational step must have been a reading step where the first component made a non-λ-move, and the second component made a λ-move (without changing the internal state). If the symbol read by the first component was $¢$, then the previous configuration must have been $(¢w\$_1\$_2w', s'_{0,1}, i-1, s_{¢}, j)$ for any configuration $(¢w\$_1\$_2w', s_1, i, s_{¢}, j), i, j > 1$. If the symbol read by the first component was a or b, then the previous configuration must have been $(¢w\$_1\$_2w', s_1, i-1, s_{¢}, j)$. (Notice that $j = 2$ in reachable configurations of these forms. Moreover, computational steps in which the first component has read a different symbol ($\$_1$, $\$_2$ or $\lhd$) are not possible.)
– If the states of the components are $(s'_1, s_{¢})$, then the previous computational step must have been a reading step where the first component made a non-λ-move, and the second component made a λ-move (without changing the internal state). The symbol read by the first component must have been $\$_1$, so the previous configuration must have been $(¢w\$_1\$_2w', s_1, i-1, s_{¢}, j)$ for any configuration $(¢w\$_1\$_2w', s'_1, i, s_{¢}, j), i, j > 1$.
– If the states of the components are $(q_2, s_{¢})$, then the previous computational step must have been a reading step where both components made λ-moves, what is seen as follows. The first component can enter state q_2 in a non-λ-move only from one of the states $s_{¢}$, s_a or s_b. But none of these states can be received by the first component in a communication step as long as the second component is in $s_{¢}$ (in reachable configurations). If the second component would perform a non-λ reading step when entering state $s_{¢}$, then it must read a $¢$ symbol while changing state $s_{0,2}$ to $s_{¢}$. This $¢$ symbol cannot be the very first symbol of the input string since the first component cannot reach state q_2 in the very first step. But it can also not be another occurrence of the $¢$ symbol as it has been argued further above. Therefore, we can conclude that the second component must have been in state $s_{¢}$ in the previous configuration, if reachable, which must have been $(¢w\$_1\$_2w', s'_1, i, s_{¢}, j)$ for any configuration $(¢w\$_1\$_2w', q_2, i, s_{¢}, j), i, j \geq 1$.
– If the states of the components are $(s_{¢}, s_{0,2})$, then the previous computational step must have been a communication step with the previous configuration $(¢w\$_1\$_2w', q_2, i, s_{¢}, j)$ for any $(¢w\$_1\$_2w', s_{¢}, i, s_{0,2}, j), i, j \geq 1$.

Continuing the computation, we have

$$(¢w\$_1\$_2w', s_{¢}, k+1, s_{0,2}, 2) \vdash (¢w\$_1\$_2w', q_2, k+2, s_x, 3) \vdash$$
$$(¢w\$_1\$_2w', s_x, k+2, s_{0,2}, 3)$$

where x depends on the second symbol on the input tape. Now, if $x \in \{a, b\}$ and the $(k+2)$nd symbol of the input is also x, the computation can continue with

$$(¢w\$_1\$_2w', s_x, k+2, s_{0,2}, 3) \vdash (¢w\$_1\$_2w', q_2, k+3, s_{x'}, 4) \vdash$$

$$(¢w\$_1\$_2w', s_{x'}, k+3, s_{0,2}, 4)$$

where x' now depends on the third symbol on the input tape. Repeating similar steps, the system is now able to check whether $w = w'$ in the input string $¢w\$_1\$_2w'$. The first component can only read the jth symbol of w' (which is the $(k+1+j)$th symbol on the tape), if it is the same as the jth symbol of w (which is the $(1+j)$th symbol on the tape) that was read in the previous non-communication step by the second component and transferred in the previous communication step to the first component as the state s_x, for $x \in \{a, b\}$.

This phase of the computation is also reversible, as can be seen by the following.

- If the states of the components are (q_2, s_x) for some $x \in \{a, b\}$ then the previous computational step must have been a reading step where both components made a non-λ-move. To see that the first component must also have made a non-λ-move, note that this configuration cannot be the first configuration of the computation where q_2 appears, since if this was the case, then the second component would be in state $s_¢$. This implies that the first component cannot enter the state q_2 with a λ-move from state s'_1, since it can only be in state s'_1 before any communication happened. If the symbol read by the first component is $y \in \{a, b\}$, then for any configuration $(¢w\$_1\$_2w', q_2, i, s_x, j)$ with $i, j > 1$, the previous configuration must have been $(¢w\$_1\$_2w', s_y, i-1, s_{0,2}, j-1)$.
- If the states of the components are $(s_x, s_{0,2})$ for some $x \in \{a, b\}$, then the previous computational step must have been a communication step, so for any configuration $(¢w\$_1\$_2w', s_x, i, s_{0,2}, j)$ with $i, j \geq 1$, the previous configuration must have been $(¢w\$_1\$_2w', q_2, i, s_x, j)$.

If the second component reaches the $\$_2$ symbol before the first component reaches the end-of-input $\lhd$, that is, the configuration reached would be of the form $(¢w\$_1\$_2w', s_{\$_1}, 2k, s_{0,2}, k+1)$ where $¢w\$_1\$_2w'(k+1) = \$_2$ and $¢w\$_1\$_2w'(2k) \neq \lhd$, then the system is blocked in non-accepting states, because the first component can continue from state $s_{\$_1}$ only when reading the end-of-input symbol. If the first component reaches the end-of-input symbol $\lhd$ before the second component reaches the $\$_2$ symbol, the system is blocked in $(¢w\$_1\$_2w', s_x, 2j-1, s_{0,2}, j)$ for some $x \in \{a, b\}$ and some $j > k+1$ which is not accepting. Note also that the communication steps before these blocking configurations are reversible.

Thus, in order for the computation to be successful, the two components must reach the end-of-input symbol $\lhd$ and the symbol $\$_2$ simultaneously through the computational steps

$$(¢w\$_1\$_2w', s_x, 2k-1, s_{0,2}, k) \vdash (¢w\$_1\$_2w', q_2, 2k, s_{\$_1}, k+1) \vdash$$
$$(¢w\$_1\$_2w', s_{\$_1}, 2k, s_{0,2}, k+1) \vdash (¢w\$_1\$_2w', s_{\mathrm{acc}}, 2k+1, s_{\$_2}, k+2).$$

These steps of the computation are also reversible, since

- having the current states $(q_2, s_{\$_1})$ implies that the previous step was a reading step in which both heads have moved on, because in reachable configurations, the first component has already passed beyond $\$_2$ ($k \geq 2$, so $2k > k+1$). The state of the first component must have been s_x with $x = \alpha(2k-1), x \in \{a, b\}$, and the state of the second component must have been $s_{0,2}$.
- Having the current states $(s_{\$_1}, s_{0,2})$ implies that the previous step was a communication step, so for any configurations $(¢w\$_1\$_2w', s_{\$_1}, i, s_{0,2}, j), i, j \geq 1$, the previous configuration must have been $(¢w\$_1\$_2w', q_2, i, s_{\$_1}, j)$.
- If the current states are $(s_{\mathrm{acc}}, s_{\$_2})$, the two components performed reading move in the previous computational steps, and the states of previous configuration must have been $s_{\$_1}$ and $s_{0,2}$, respectively.

As the components of PCFA are non-deterministic finite automata (NFA), we may consider the relationship of the reversibility of a PCFA and the reversibility of its components. Since in PCFA, there is an end-of-input symbol $\lhd$ and λ-steps are allowed in the component automata, we extend the definition of reversibility of NFA as it is given in [10] as follows: Let $\Sigma' = \Sigma \cup \{\lhd\}$. An NFA $A = (S, \Sigma, s_0, \delta, F)$ is reversible if

1. there is no internal state $s \in S$, such that $s \in \delta(p, a) \cap \delta(r, a)$ holds for some $a \in \Sigma'$ and states $p, r \in S$ with $p \neq r$;
2. there is no internal state $s \in S$, such that $s \in \delta(p, a) \cap \delta(r, \lambda)$ holds for some $a \in \Sigma' \cup \{\lambda\}$ and states $p, r \in S$ with $p \neq r$.

Note that the fact that systems of (deterministic) reversible automata are not necessarily reversible was already remarked in [9] without giving a formal definition of the notion of reversibility for PCFA systems. In the second part of the following proposition, we give a similar, but more simple demonstration than the one in [9].

Proposition 1 *1. There are reversible PCFA $\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_k, Q, \lhd)$ such that there is i, $1 \le i \le k$, where A_i is not reversible.*
2. There are PCFA $\mathcal{A} = (\Sigma, A_1, A_2, \ldots, A_k, Q, \lhd)$ such that the components A_i are reversible for all i, $1 \le i \le k$, and the system $\mathcal{A}$ is not reversible.

Proof To show that the first statement holds, we can use the reversible PCFA of Example 1. Obviously, neither A_1 nor A_2 are reversible components. Consider for example $\delta_2(s_\$, a) = \delta_2(s_a, a) = \delta_2(s_b, a) = s_a$.

To prove the second statement, consider the PCFA $\mathcal{A} = (\{a, b\}, A_1, A_2, \{q_2\}, \lhd)$ where the first component is $A_1 = (\{s_{0,1}, s_a, s_a', q_2\}, \{a, b\}, s_{0,1}, \delta_1, \emptyset)$ with

$$\begin{array}{lll} \delta_1(s_{0,1}, a) = q_2, & \delta_1(s_a, a) = s_a', & \delta_1(s_a', a) = s_a. \\ \delta_1(s_{0,1}, b) = s_{0,1}, & \delta_1(s_a, b) = q_2, & \end{array}$$

The second component is defined as $A_2 = (\{s_a, s_1, s_{\mathrm{acc}}\}, \{a, b\}, s_a, \delta_2, \{s_{\mathrm{acc}}\})$ with

$$\begin{array}{lll} \delta_2(s_a, a) = s_1, & \delta_2(s_1, a) = s_a, & \delta_2(s_a, \lhd) = s_{\mathrm{acc}}. \\ \delta_2(s_a, b) = s_1, & \delta_1(s_1, b) = s_a, & \end{array}$$

The reader may easily check that both components are reversible. To see that the system $\mathcal{A}$ is not reversible, consider, for example, the computation

$$(baaa, s_{0,1}, 1, s_a, 1) \vdash (baaa, s_{0,1}, 2, s_1, 2) \vdash (baaa, q_2, 3, s_a, 3) \vdash$$
$$(baaa, s_a, 3, s_a, 3) \vdash (baaa, s_a', 4, s_1, 4) \vdash (baaa, s_a, 5, s_a, 5) \vdash \ldots$$

If the states of the two components are (s_a, s_a), the previous computational step is not unique. (Therefore, the set of possible previous configurations contains more than one element.) More formally (see Definition 1), there are computational steps

$$(baaa, q_2, 3, s_a, 3) \vdash (baaa, s_a, 3, s_a, 3)$$

and

$$(baaa, s_a', 4, s_1, 4) \vdash (baaa, s_a, 5, s_a, 5)$$

which lead to the same pair of states (s_a, s_a), but $3 - 3 \neq 5 - 4$ (since the first one is a communication step, while the second is a non-λ reading step).

Notice that we used deterministic centralized systems of degree 2 in both cases.

3 Some results on the computational capacity

First, we prove that reversible deterministic PCFA are more powerful than reversible deterministic finite automata. It turns out that for accepting a non-reversible regular language with deterministic PCFA, communication is even unnecessary.

Theorem 1 *There is a language in* $\mathcal{L}(\text{revDCPCFA}(2)) \cap \mathcal{L}(\text{revDRCPCFA}(2))$ *that cannot be accepted by any reversible deterministic finite automaton.*

Proof First, we prove that for the language $L = \{01^k \mid k \geq 1\}$, there is no reversible deterministic finite state automaton. Let A be any deterministic finite automaton accepting L. Assume the number of states of A be n. Let

$$(q_0, 01^n) \vdash (p_0, 1^n) \vdash (p_1, 1^{n-1}) \vdash (p_2, 1^{n-2}) \vdash \ldots \vdash (p_{n-1}, 1) \vdash (p_n, \lambda)$$

be the accepting computation for 01^n, in A, where $n > 2$, which is seen as follows. If $q_0 = p_i$ for some i, $0 \leq i \leq n$, then the input word 1^{n-i} would be accepted. Hence, $q_0 \neq p_i$ for all i, $0 \leq i \leq n$. Furthermore, state p_0 is not accepting, while every state p_i with $i > 1$ is accepting. Thus, $p_0 \neq p_i$, for $1 \leq i \leq n$. Since A has n states, there are i and j, $1 \leq i, j \leq n$, with $i \neq j$, and $p_i = p_j$. Consider the smallest such i, that is, let i be the smallest number greater than 0 such that there is $j > i$ and $p_i = p_j$. Then, $\delta(p_{i-1}, 1) \cap \delta(p_{j-1}, 1) = p_i$ and $p_{i-1} \neq p_{j-1}$. Therefore, A is not reversible.

It is left to show that $L \in \mathcal{L}(\text{revDCPCFA}(2)) \cap \mathcal{L}(\text{revDRCPCFA}(2))$. For proving this, we consider the deterministic PCFA

$$\mathcal{A} = (\{0, 1\}, A_1, A_2, \emptyset, \triangleleft)$$

where $A_1 = (\{s_{0,1}, s_1, s_2\}, \{0, 1\}, s_{0,1}, \delta_1, \{s_2\})$ with

$$\begin{aligned} \delta_1(s_{0,1}, 0) &= s_1, \\ \delta_1(s_1, 1) = \delta_1(s_2, 1) &= s_2, \end{aligned}$$

and $A_2 = (\{s_{0,2}, p_1, p_2\}, \{0, 1\}, s_{0,2}, \delta_2, \emptyset)$ with

$$\begin{aligned} \delta_2(s_{0,2}, \lambda) &= p_1, \\ \delta_2(p_1, 0) = \delta_2(p_2, 1) &= p_2. \end{aligned}$$

For any 01^k, $k \geq 1$, the computation of $\mathcal{A}$ is as follows:

$$\begin{aligned} (01^k, s_{0,1}, 1, s_{0,2}, 1) &\vdash\ (01^k, s_1, 2, p_1, 1) \\ &\vdash\ (01^k, s_2, 3, p_2, 2) \\ &\vdash^{k-1} (01^k, s_2, k+2, p_2, k+1) \end{aligned}$$

This computation is accepting as $\delta_1(s_2, \triangleleft)$ is undefined and s_2 is an accepting state of A_1. As the set of query states is empty, one may consider the DPCFA $\mathcal{A}$ to be centralized and both returning and non-returning. It is left to prove that the PCFA $\mathcal{A}$ is reversible.

- If, in a configuration reached, the pair of states is (s_1, p_1), then the first component has made a move reading 0 from $s_{0,1}$, while the second component has made a λ-move coming from $s_{0,2}$, and the predecessor configuration must have been the initial one.
- If the states reached are (s_2, p_2), then both components made a non-λ-move. If the symbol read by the second component was 0, then, for any configuration $(01^k, s_2, i, p_2, j)$, the predecessor configuration must have been the configuration $(01^k, s_1, i-1, p_1, j-1)$. In fact, in reachable configurations, $i = 3$ and $j = 2$ must hold. If the symbol read by the

second component was 1, then the predecessor configuration of $(01^k, s_2, i, p_2, j)$ must have been $(01^k, s_2, i-1, p_2, j-1)$.
– Configurations with other states are not reachable.

Examples 1 and 2 show that the following theorem holds.

Theorem 2 *There are non-context-free languages in* $\mathcal{L}(\mathrm{revDCPCFA}(2))$ *and in* $\mathcal{L}(\mathrm{revDRCPCFA}(2))$.

Next, we study the relationship of deterministic and non-deterministic centralized systems. In the not necessarily reversible case, deterministic systems are strictly weaker both in returning and non-returning modes, see [4] for details. For reversible PCFA, we can show a similar result for the non-returning case.

Theorem 3 $\mathcal{L}(\mathrm{revDCPCFA}) \subset \mathcal{L}(\mathrm{revCPCFA})$.

Proof We show that $L_{\mathrm{nonpal}} = \{\,\$w \mid w \in \{a,b\}^+,\ w \neq w^R\,\} \in \mathcal{L}(\mathrm{revCPCFA})$. This is sufficient, since $\mathcal{L}(\mathrm{revDCPCFA}(k)) \subseteq \mathcal{L}(\mathrm{DCPCFA}(k)) \subseteq \mathcal{L}(k\text{-DFA})$, where $\mathcal{L}(k\text{-DFA})$ is the family of languages accepted by k-head deterministic finite automata as defined in, e.g., [20]. But $L_{\mathrm{nonpal}} \notin \mathcal{L}(k\text{-DFA})$ for any $k \geq 1$. To see this, recall that $\mathcal{L}(k\text{-DFA})$ is closed under complementation and under intersection with regular sets, thus, $L_{\mathrm{nonpal}} \in \mathcal{L}(k\text{-DFA})$ would imply that $\overline{L_{\mathrm{nonpal}}} \cap (\{\$\}\{a,b\}^*) = \{\$\}\{\,w \mid w \in \{a,b\}^*,\ w = w^R\,\} \in \mathcal{L}(k\text{-DFA})$, but this is not the case, as stated, for example, in [21].

To see that $L_{\mathrm{nonpal}} \in \mathcal{L}(\mathrm{revCPCFA})$, consider the centralized PCFA system

$$\mathcal{A} = (\{\$, a, b\}, A_1, A_2, A_3, \{q_2, q_3\}, \lhd)$$

where $A_1 = (\{s_{0,1}, s_a, s'_a, s_b, s'_b, s_{\mathrm{acc}}, q_2, q_3\}, \{\$, a, b\}, s_{0,1}, \delta_1, \{s_{\mathrm{acc}}\})$ with

$$\begin{aligned}
\delta_1(s_{0,1}, \lambda) &= \{s_{0,1}, q_2\},\\
\delta_1(s_x, \$) &= \{s'_x\} \text{ for } x \in \{a,b\},\\
\delta_1(s'_x, x) &= \{s'_x\} \text{ for } x \in \{a,b\},\\
\delta_1(s'_x, y) &= \{s'_x, q_3\} \text{ for } x, y \in \{a,b\},\ x \neq y.
\end{aligned}$$

The second component is $A_2 = (\{s_{0,2}, s_a, s_b, s_\lhd, s_f\}, \{a,b\}, s_{0,2}, \delta_2, \emptyset\})$ with

$$\begin{aligned}
\delta_2(s_{0,2}, \$) &= \{s_a, s_b\},\\
\delta_2(s_x, x) &= \{s_a, s_b, s_\lhd\} \text{ for } x \in \{a,b\},\\
\delta_2(s_\lhd, \lhd) &= \{s_f\},
\end{aligned}$$

and the third component is $A_3 = (\{s_{0,3}, s_{\mathrm{acc}}\}, \{a,b\}, s_{0,3}, \delta_3, \emptyset\})$ with

$$\begin{aligned}
\delta_3(s_{0,3}, x) &= \{s_{0,3}\} \text{ for } x \in \{\$, a, b\},\\
\delta_3(s_{0,3}, \lhd) &= \{s_{\mathrm{acc}}\}.
\end{aligned}$$

We show that this system is reversible and it accepts the language $L_{\mathrm{nonpal}} = \{\,\$w \mid w \in \{a,b\}^+,\ w \neq w^R\,\}$ To see how $\mathcal{A}$ works, consider first an initial configuration $(\alpha, s_{0,1}, 1, s_{0,2}, 1, s_{0,3}, 1)$. Notice that $\alpha = \$\alpha'$ for some $\alpha' \in (a+b)^*$ or the second component cannot read through the input. Note also that if $\$w$ is accepted, then $|w| \geq 2$. To see this, consider a computation on a (shorter) input of the form $\$x,\ x \in \{\lambda, a, b\}$. The first step of the computation of $\mathcal{A}$ is

$$(\$x, s_{0,1}, 1, s_{0,2}, 1, s_{0,3}, 1) \vdash (\$x, s, 1, s_y, 2, s_{0,3}, 2)$$

where $s \in \{s_{0,1}, q_2\},\ y \in \{a, b\}$. Note that this computational step is reversible, since the states imply that the first component made a λ-reading step with the previous state being $s_{0,1}$, while the other two components made non-λ-reading steps. Considering that the \$ symbol was read by these components, we can also deduce that their previous states were $s_{0,2}$ and $s_{0,3}$, respectively. Now, if $x = \lambda$ or $y \neq x$, then the system is in a non-accepting blocking configuration (or enters such a configuration with the following—also reversible—communication step) because of the definition of δ_2. Because of these observations, it is sufficient to consider

$$(\$x, s, 1, s_x, 2, s_{0,3}, 2) \tag{1}$$

with $x \in \{a, b\}$. If $s = s_{0,1}$, we have

$$(\$x, s_{0,1}, 1, s_x, 2, s_{0,3}, 2) \vdash (\$x, s, 1, s_y, 3, s_{0,3}, 3),$$

with $s \in \{s_{0,1}, q_2\},\ y \in \{a, b, \lhd\}$. If $y \neq \lhd$, then the system is in a non-accepting blocking configuration (or enters such a configuration with the following communication), so we must have

$$(\$x, s_{0,1}, 1, s_{\lhd}, 3, s_{0,3}, 3) \vdash (\$x, s, 1, s_f, 4, s_{\text{acc}}, 4),$$

with $s \in \{s_{0,1}, q_2\}$, or

$$(\$x, q_2, 1, s_{\lhd}, 3, s_{0,3}, 3) \vdash (\$x, s_{\lhd}, 1, s_{\lhd}, 3, s_{0,3}, 3).$$

In both of these cases, the system is in a non-accepting blocking configuration (or enters such a configuration with the following communication step), and it is not difficult to see that all the above steps are also reversible. On the other hand, if in (1) we have $s = q_2$, then

$$(\$x, q_2, 1, s_x, 2, s_{0,3}, 2) \vdash (\$x, s_x, 1, s_x, 2, s_{0,3}, 2) \vdash (\$x, s'_x, 2, s_y, 3, s_{0,3}, 3),$$

where $x \in \{a, b\},\ y \in \{a, b, \lhd\}$. These steps are reversible, as s_x at the first component implies that the previous step was a communication, and the states $(s'_x, s_y, s_{0,3})$ imply that all components performed non-λ-reading steps. Further, since the first component read the \$ symbol, its previous state must be s_x, while the reading of an $z \in \{a, b, \lhd\}$ by the second component implies that its previous state is s_z.

Now, if $y \neq \lhd$, then the system is in a non-accepting blocking configuration, so we must have

$$(\$x, s'_x, 2, s_{\lhd}, 3, s_{0,3}, 3) \vdash (\$x, s'_x, 3, s_f, 4, s_{\text{acc}}, 4)$$

which is also a non-accepting blocking configuration, since, e.g., the first component has no transition for reading the end-of-input symbol $\lhd$ in state s'_x. (Note that this last step is also reversible.)

Consider now the computations on an input $\$w,\ |w| \geq 2$. These computations must start with

$$(\$w, s_{0,1}, 1, s_{0,2}, 1, s_{0,3}, 1) \vdash (\$w, s, 1, s_x, 2, s_{0,3}, 2)$$

for some $s \in \{s_{0,1}, q_2\},\ x \in \{a, b\}$. If $s = q_2$, we get

$$(\$w, q_2, 1, s_x, 2, s_{0,3}, 2) \vdash (\$w, s_x, 1, s_x, 2, s_{0,3}, 2).$$

Similarly, if the first component enters the query state q_2 in a later step, we have

$$\begin{aligned}(\$w, s_{0,1}, 1, s_x, 2, s_{0,3}, 2) &\vdash \ldots \vdash (\$w, s_{0,1}, 1, s_{x'}, k-1, s_{0,3}, k-1) \vdash \\ &\vdash (\$w, q_2, 1, s_y, k, s_{0,3}, k) \vdash (\$w, s_y, 1, s_y, k, s_{0,3}, k)\end{aligned}$$

for $x', y \in \{a, b, \triangleleft\}$ and some head position k. In any of these cases, the system reaches a configuration

$$(\$w, s_x, 1, s_x, k, s_{0,3}, k)$$

where $x \in \{a, b, \triangleleft\}$ and $k \geq 2$. If $\$w(k) \neq x$, then this configuration is a non-accepting blocking one due to the definition of δ_2. Otherwise, such a configuration records the information that the kth symbol on the input tape is x (where k is "nondeterministically chosen"). This ends the first phase of the computation.

To see that this phase is reversible, consider the following.

- If the states of the components are $(s_{0,1}, s_x, s_{0,3})$ or $(q_2, s_x, s_{0,3})$, $x \in \{a, b, \triangleleft\}$, then the first component made a λ reading step, the second and the third non-λ reading steps. If the symbol read by the second component was a \$, then the previous configuration must have been $(\$w, s_{0,1}, 1, s_{0,2}, 1, s_{0,3}, 1)$, the initial configuration. If the second component read a symbol $y \in \{a, b\}$, then for any configuration $(\$w, s, 1, s_x, i, s_{0,3}, i)$ with $s \in \{s_{0,1}, q_2\}$, $i \geq 2$, the previous configuration must have been $(\$w, s_{0,1}, 1, s_y, i-1, s_{0,3}, i-1)$. (Note that in any reachable configuration, if the state of the first component is $s_{0,1}$ or q_2, then its head is on position 1.)
- If the states of the components are $(s_x, s_x, s_{0,3})$, for some $x \in \{a, b, \triangleleft\}$, then the previous computational step was a communication, so for any configuration $(\$w, s_x, 1, s_x, i, s_{0,3}, i), i \geq 2$, the previous configuration must have been $(\$w, q_2, 1, s_x, i, s_{0,3}, i)$. (Note that in any reachable configuration with states $(s_x, s_x, s_{0,3})$, the heads of the three components must be on positions $(1, i, i)$ for some $i \geq 2$.)

Consider now how the computation continues from the configuration

$$(\$w, s_x, 1, s_x, k, s_{0,3}, k)$$

recording the information that the kth symbol on the input tape is x for some $k \geq 2$ and $x \in \{a, b\}$. (If $\$w(k) \neq x$ or $x = \triangleleft$, then the computation cannot continue.) The next phase of the computation makes sure that w is not a palindrome by checking that the symbol at position $(|\$w| - (k - 2))$ of the input (the kth position counting from the end of the input) is different from x.

The computation continues by

$$\begin{aligned}&(\$w, s_x, 1, s_x, k, s_{0,3}, k) \vdash (\$w, s'_x, 2, s_y, k+1, s_{0,3}, k+1) \vdash \dots \\ &\dots \vdash (\$w, s'_x, 1+j, s_{y'}, k+j, s_{0,3}, k+j)\end{aligned}$$

where $y, y' \in \{a, b, \triangleleft\}$, $j \geq 1$. Now, due to the construction of δ_1, if in such a configuration the first component enters the communication state q_3, then the $(j+1)$st symbol of the input word is different from x. Thus, if the step

$$(\$w, s'_x, 1+j, s_{y'}, k+j, s_{0,3}, k+j) \vdash (\$w, q_3, 1+j+1, s, k+j+1, s', k+j+1)$$

is performed, then $\$w(j+1) \neq x$. Now, it remains to be seen that the accepting configuration can only be reached if $j+1 = (|\$w| - (k-2))$, that is, if w is a non-palindrome. In order to successfully terminate, the configurations above must be with $s_{y'} = s_\triangleleft$, $s = s_f$ and $s' = s_{\mathrm{acc}}$, then we get

$$\begin{aligned}&(\$w, q_3, 1+j+1, s_f, k+j+1, s_{\mathrm{acc}}, k+j+1) \vdash \\ &\qquad (\$w, s_{\mathrm{acc}}, 1+j+1, s_f, k+j+1, s_{\mathrm{acc}}, k+j+1)\end{aligned}$$

which is an accepting configuration. Since the position of the end-of-input symbol on the tape is the $(k+j)$th position, we have that $|\$w| = k+j-1$ which means that $(|\$w| - (k-2)) = j+1$, so the input must have been a non-palindrome.

If j has a different value as above, then either we have a non-accepting blocking configuration of the form

$$(\$w, s'_x, 1+j+1, s_f, k+j+1, s_{\mathrm{acc}}, k+j+1),$$

or we have

$$(\$w, q_3, 1+j+1, s_y, k+j+1, s_{0,3}, k+j+1)$$

for $y \in \{a, b, \lhd\}$, and then a non-accepting blocking configuration after the communication, namely

$$(\$w, s_{0,3}, 1+j+1, s_y, k+j+1, s_{0,3}, k+j+1).$$

From these considerations, we can see that $L(\mathcal{A}) = L_{\mathrm{nonpal}}$, what remains to be shown is the reversibility of the second phase of the computation. To this aim, consider the following.

- If the states of the components are $(s_{x'}, s_y, s_{0,3})$, then all three components made non-λ reading steps. If the symbol read by the first component was a \$, and the symbol read by the second component was $z \in \{a, b\}$, then for any configuration $(\$w, s_{x'}, 1+i, s_y, k+i, s_{0,3}, k+i)$, the previous configuration must have been $(\$w, s_x, i, s_z, k+i-1, s_{0,3}, k+i-1)$. If the symbol read by the first component was different from \$, then the previous configuration is $(\$w, s_{x'}, i, s_z, k+i-1, s_{0,3}, k+i-1)$.
- If the states of the components are $(q_3, s_f, s_{\mathrm{acc}})$, then all components performed non-λ reading steps. If the symbol read by the first component was $x \in \{a, b\}$, then for any configuration $(\$w, q_3, 1+i, s_f, k+i, s_{\mathrm{acc}}, k+i)$, the previous configuration must have been $(\$w, s'_y, i, s_\lhd, k+i-1, s_{0,3}, k+i-1)$ where $y \neq x$.
- If the states of the components are $(q_3, s_y, s_{0,3})$ for some $y \in \{a, b, \lhd\}$, then all components performed non-λ reading steps. If the symbol read by the first and the second components was x and z in $\{a, b\}$, respectively, then for any configuration $(\$w, q_3, 1+i, s_y, k+i, s_{0,3}, k+i)$, the previous configuration must have been $(\$w, s'_y, i, s_z, k+i-1, s_{0,3}, k+i-1)$ where $y \neq x$.
- If the states of the components are $(s'_x, s_f, s_{\mathrm{acc}})$, then all components performed non-λ reading steps. If the symbol read by the first component was \$, then for any configuration $(\$w, s'_x, 1+i, s_f, k+i, s_{\mathrm{acc}}, k+i)$, the previous configuration must have been $(\$w, s_x, i, s_\lhd, k+i-1, s_{0,3}, k+i-1)$. If the symbol read by the first component was different from \$, then the previous configuration must have been $(\$w, s'_x, i, s_\lhd, k+i-1, s_{0,3}, k+i-1)$.
- If the states of the components are $(s_{\mathrm{acc}}, s_f, s_{\mathrm{acc}})$ or $(s_{0,3}, s_x, s_{0,3})$ for some $x \in \{a, b, \lhd\}$, then the previous computational step was a communication. In this case, for any configuration $(\$w, s_{\mathrm{acc}}, 1+i, s_f, k+i, s_{\mathrm{acc}}, k+i)$ or $(\$w, s_{0,3}, 1+i, s_x, k+i, s_{0,3}, k+i)$, the previous configurations must have been $(\$w, q_3, 1+i, s_f, k+i, s_{\mathrm{acc}}, k+i)$ or $(\$w, q_3, 1+i, s_x, k+i, s_{0,3}, k+i)$, respectively.

4 Concluding remarks

In this paper, a definition of the concept of reversibility for systems of parallel communicating finite automata (PCFA) has been suggested. The definition covers also non-deterministic

PCFA. It was shown that there are reversible PCFA that have non-reversible components and there are non-reversible PCFA in which every component is a reversible finite state automaton. Examples demonstrate that the weakest types of (non-trivial) reversible PCFA, namely reversible deterministic centralized PCFA of degree 2, can recognize non-context-free languages, both in returning and in non-returning communication mode. Furthermore, there is a language which can be recognized with a reversible non-deterministic centralized PCFA in non-returning mode, but it cannot be accepted by any deterministic PCFA. The question whether or not the same statement is true also for reversible non-deterministic returning centralized PCFA is left open here. But there are many more questions regarding the computational power of reversible PCFA that are left for future research work, for instance:

– What is the relation between reversible centralized and non-centralized PCFA (deterministic or not)?
– What is the relation between reversible centralized returning and non-returning PCFA (deterministic or not)?
– What is the relation between reversible PCFA and reversible one-way multi-head finite automata (deterministic or not)?

As to the latter question, it is obvious that deterministic (non-deterministic) reversible one-way multi-head finite automata can simulate any type of reversible deterministic (non-deterministic) PCFA. It is to be examined whether the proofs showing the other direction (e.g., see [4]) can be adapted to the reversible case.

Moreover, several decidability problems should be investigated in future research on reversible PCFA, such as:

– Given a PCFA $\mathcal{A}$, is $\mathcal{A}$ reversible?
– Given a PCFA language L, can L be accepted by a reversible PCFA (of the same type)?

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Axelsen, H.B.: Reversible multi-head finite automata characterize reversible logarithmic space. In: Okhotin, A., Martin-Vide, C., Shapira, D. (eds.) Language and Automata Theory and Applications (LATA, 2012), Volume 7183 of LNCS, pp. 95–105. Springer, Berlin (2012)
2. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**, 525–532 (1973)
3. Bordihn, H., Kutrib, M., Malcher, A.: Undecidability and hierarchy results for parallel communicating finite automata. Int. J. Found. Comput. Sci. **22**, 1577–1592 (2011)
4. Bordihn, H., Kutrib, M., Malcher, A.: On the computational capacity of parallel communicating finite automata. Int. J. Found. Comput. Sci. **23**, 713–732 (2012)
5. Bordihn, H., Kutrib, M., Malcher, A.: Returning parallel communicating finite automata with communication bounds: hierarchies, decidabilities, and undecidabilities. Int. J. Found. Comput. Sci. **26**, 1101–1126 (2015)
6. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM **30**, 323–342 (1983)

7. Choudhary, A., Krithivasan, K., Mitrana, V.: Returning and non-returning parallel communicating finite automata are equivalent. RAIRO Inform. Théor. **41**, 137–145 (2007)
8. Csuhaj-Varjú, E., Dassow, J., Kelemen, J., Păun, Gh.: Grammar Systems: A Grammatical Approach to Distribution and Cooperation. Gordon and Breach, Yverdon (1984)
9. Ganguly, D., Chatterjee, K., Ray, K.S.: Parallel communicating one-way reversible finite automata system. ArXiv:1903.10428 (2019)
10. Holzer, M., Kutrib, M.: Reversible nondeterministic finite automata. In: Phillips, I., Rahaman, H. (eds.) Reversible Computing (RC 2017), Volume 10301 of LNCS, pp. 35–51. Springer, Berlin (2017)
11. Klemm, R.: Systems of communicating finite state machines as a distributed alternative to finite state machines. Ph.D. Thesis (Pennsylvania State University) (1996)
12. Kutrib, M.: Aspects of reversibility for classical automata. In: Calude, C.S., Freivalds, R., Kazuo, I. (eds.) Computing with New Resources, Volume 8808 of LNCS, pp. 83–98. Springer, Berlin (2014)
13. Kutrib, M., Malcher, A.: Reversible pushdown automata. J. Comput. Syst. Sci. **78**, 1814–1827 (2012)
14. Kutrib, M., Malcher, A.: One-way reversible multi-head finite automata. Theor. Comput. Sci. **682**, 149–164 (2017)
15. Kutrib, M., Malcher, A., Wendlandt, M.: When input-driven pushdown automata meet reversiblity. RAIRO Theor. Inf. Appl. **50**, 313–330 (2016)
16. Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5**, 18–191 (1961)
17. Martín-Vide, C., Mateescu, A., Mitrana, V.: Parallel finite automata systems communicating by states. Int. J. Found. Comput. Sci. **13**, 733–749 (2002)
18. Morita, K.: Reversible simulation of one-dimensional irreversible cellular automata. Theor. Comput. Sci. **148**, 157–163 (1995)
19. Morita, K.: Two-way reversible multi-head finite automata. Fund. Inf. **110**, 241–254 (2011)
20. Rosenberg, A.L.: On multi-head finite automata. IBM J. Res. Dev. **10**, 388–394 (1966)
21. Wagner, K., Wechsung, G.: Computational Complexity. Reidel, Dordrecht (1986)
22. Wood, D.: Theory of Computation. Wiley, New York (1987)