Language Recognition by Reversible Partitioned Cellular Automata

Kenichi Morita(✉)

Hiroshima University, Higashi-Hiroshima 739–8527, Japan
km@hiroshima-u.ac.jp

Abstract. We investigate the language accepting capability of one-dimensional reversible partitioned cellular automata (RPCAs). It is well known that bounded cellular automata (CAs) are equivalent to deterministic linear-bounded automata (DLBAs) in their language accepting capability. Here, we prove RPCAs are also equivalent to them by showing a construction method of an RPCA that simulates a given DLBA. Thus, the reversibility constraint does not decrease the ability of PCAs.

1 Introduction

One-dimensional cellular automata (CAs) as language acceptors have been extensively studied until now, and fast recognition algorithms as well as their properties have been investigated (see, e.g., a survey [1]). Smith [8] showed deterministic CAs whose space is bounded by the input length are equivalent to deterministic linear-bounded automata (DLBA) in their accepting capability if computing time is not bounded. On the other hand, Kutrib and Malcher [2] studied reversible CA acceptors, and derived basic properties of real-time ones.

In this paper, we study how the constraint of reversibility affects the accepting capability of bounded CAs in the case computing time is not bounded. For this purpose, we consider the following two sub-problems. The first one is how a DLBA is converted into an equivalent reversible DLBA (RDLBA). The second one is how an RDLBA is simulated by a reversible bounded CA. For the first problem, Lange, McKenzie and Tapp [3] showed that a DLBA can be simulated by an RDLBA. However, their method is complex, and it is difficult to give a practical procedure of conversion. Here, we give a much simpler conversion method based on the one shown in [6]. For the second problem, we use the framework of a deterministic partitioned cellular automaton (PCA), since it makes design of reversible CAs easier. In [4,7], it is shown that a reversible Turing machine (RTM) is simulated by a reversible PCA (RPCA). However, there, the configuration size of the RPCA was not bounded, and thus a new technique is required to simulate an RDLBA in a bounded RPCA. Here, we propose a formulation of an RPCA acceptor, and give a conversion method of an RDLBA into an RPCA that simulates the former in the cellular space whose working space is always bounded by the input length plus 2. By above, any given DLBA can be converted into an RPCA acceptor that simulates the former. Hence, the

T. Isokawa et al. (Eds.): AUTOMATA 2014, LNCS 8996, pp. 106–120, 2015.
DOI: 10.1007/978-3-319-18812-6_9

language accepting capability of bounded PCAs does not decrease, even if the reversibility constraint is added.

2 Reversible Partitioned Cellular Automaton (RPCA)

Definition 1. A 1-dimensional 3-neighbor deterministic partitioned cellular automaton (PCA) *an acceptor of a language is defined by* $P = ((L, C, R), f, (\#, \#, \#), r_s, \Sigma, A)$. *Here,* L, C,*and* R *are nonempty finite sets of states of left, center, and right parts of a cell, and thus the state set of a cell is* $Q = L \times C \times R$. *A mapping* $f : Q \to Q$ is a *local function*, $(\#, \#, \#) \in Q$ *is a quiescent state that satisfies* $f(\#, \#, \#) = (\#, \#, \#)$, $r_s \in R$ is a *start state*, $\Sigma \subset C$ *is an input alphabet, and* $A \subset L$ *is a set of* accepting states.

Let $p_L : Q \to L$ be the projection such that $p_L(l, c, r) = l$ for all $(l, c, r) \in Q$. The projections $p_C : Q \to C$ and $p_R : Q \to R$ are also defined similarly. Let $\mathrm{Conf}(Q)$ be the set of all configurations over Q, i.e., $\mathrm{Conf}(Q) = \{\alpha \mid \alpha : \mathbb{Z} \to Q\}$, where $\mathbb{Z}$ is the set of all integers. The *global function* $F : \mathrm{Conf}(Q) \to \mathrm{Conf}(Q)$ of P induced by f is defined as the one that satisfies the following:

$$\forall \alpha \in \mathrm{Conf}(Q), \forall x \in \mathbb{Z}\ (F(\alpha)(x) = f(p_L(\alpha(x+1)), p_C(\alpha(x)), p_R(\alpha(x-1)))).$$

Let F^t denote the operation of applying F repeatedly t times ($t = 0, 1, \ldots$).

In a PCA P, the next state of each cell is determined by the present state of the left part of the right-neighboring cell, the center part of this cell, and the right part of the left-neighboring cell. Note that a state in L (R, respectively) can be regarded as a "signal" to the left-neighboring (right-neighboring) cell. An equation $f(l, c, r) = (l', c', r')$, where $(l, c, r), (l', c', r') \in Q$, is called a *rule* of P.

Definition 2. Let $P = ((L, C, R), f, (\#, \#, \#), r_s, \Sigma, A, N)$ *be a* PCA. P *is called locally reversible iff the local function* f *is injective, and called globally reversible iff the global function* F *induced by* f *is injective.*

Proposition 1. *[7] Any PCA* P *is locally reversible iff it is globally reversible.*

As stated in Proposition 1, local and global reversibility are equivalent in PCAs. Hence, in the following, we shall design a locally reversible PCA to obtain a globally reversible one, and it is simply called a *reversible* PCA (RPCA).

In the following, we assume a PCA (or RPCA) P satisfies the condition (P1) below so that the number of non-quiescent cells does not exceeds $n + 2$ throughout a computation process, where n is the length of an input.

(P1) If a cell of P is in the state # in the center part, then it bounces any signal l (r, respectively) from the right (left), and sends back a signal r' (l') to the right (left): $\forall l \in L, \exists r' \in R\ (f(l, \#, \#) = (\#, \#, r'))$ and $\forall r \in R, \exists l' \in L\ (f(\#, \#, r) = (l', \#, \#))$.

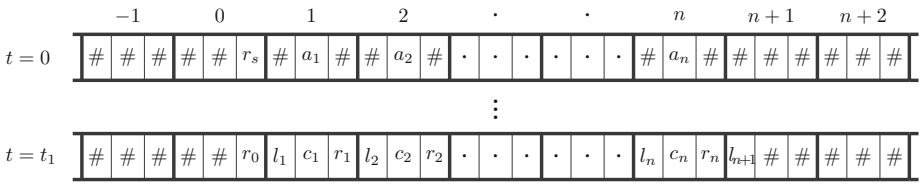

Fig. 1. An initial configuration α_w $(t = 0)$ of a PCA with an input $w = a_1 a_2 \cdots a_n$, and an accepting configuration $(t = t_1)$ where $p_L(F^{t_1}(\alpha_w)(1)) = l_1 \in A$.

Definition 3. Let $P = ((L, C, R), f, (\#, \#, \#), r_s, \Sigma, A, N)$ *be a*PCA,*and* $w = a_1 a_2 \cdots a_n \in \Sigma^n$ $(n = 1, 2, \ldots)$ *be an input word. The configuration* α_w *defined below is called an* initial configuration of P *with* w.

$$\alpha_w(x) = \begin{cases} (\#, \#, r_s) & \text{if } x = 0 \\ (\#, a_x, \#) & \text{if } 1 \le x \le n \\ (\#, \#, \#) & \text{if } x < 0 \text{ or } x > n \end{cases}$$

We say$w$ *is accepted by* P *if* $\exists t_1 > 0$ $(p_L(F^{t_1}(\alpha_w)(1)) \in A)$. The language *accepted by* P is: $L(P) = \{w \in \Sigma^* \mid \exists t_1 > 0 \ (p_L(F^{t_1}(\alpha_w)(1)) \in A)\}$.

An initial configuration and an accepting configuration are illustrated in Fig. 1. Here, we assume an infinite array of cells. But, since the condition (P1) holds, only $n+2$ cells are used in a computation. In the case of usual CA acceptors border cells at the positions 0 and $n+1$ do not change their states (see [1]). But, in PCA acceptors, the right part of the left border cell, and the left part of the right border cell may change their states.

3 Reversible Linear-Bounded Automaton (RLBA)

Lange, McKenzie and Tapp [3] showed that the complexity class of deterministic space $S(n)$ is equal to that of reversible space $S(n)$. From this, we obtain equivalence of a deterministic linear-bounded automaton (DLBA) and a reversible DLBA (RDLBA) by letting $S(n) = n$. But, their method is complex, and it is difficult to get a concrete description of the RDLBA. Here, we show a simpler method of converting a DLBA into an RDLBA based on the method in [6].

In this paper, a linear-bounded automaton (LBA) is defined as a 2-track LBA shown in Fig. 2 rather than a standard 1-track LBA, because in the proof of Lemma 2, which gives a method of converting an irreversible LBA to a reversible one, it is required that an input word is kept unchanged throughout its computation. It is easy to see a 2-track LBA can simulate a 1-track LBA, and vice versa by a straightforward method, and thus they are equivalent.

Definition 4. A 2-track linear-bounded automaton *(LBA) consists of a finite-state control, a read-write head, and a tape with an input track and a storage*

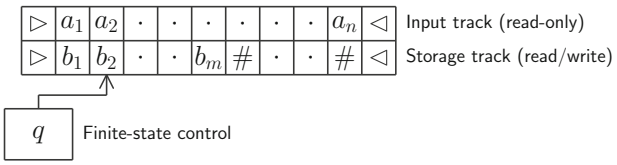

Fig. 2. A 2-track linear-bounded automaton (LBA).

track (Fig. 2). It is defined by $M = (Q, \Sigma, \Gamma, \delta, \triangleright, \triangleleft, \#, q_0, A, N)$. *$Q$ is a nonempty finite set of states, Σ is a nonempty finite set of input symbols for the input track, and Γ is a nonempty finite set of storage symbols for the storage track. $\triangleright$ and $\triangleleft$ are left and right endmarkers of each track of the tape such that* $\{\triangleright, \triangleleft\} \cap (\Sigma \cup \Gamma) = \emptyset$, and $\# \in \Gamma$ *is a blank symbol for the storage track.* $q_0 (\in Q)$ *is the initial state, and* $A (\subset Q)$ and $N (\subset Q)$ *are sets of accepting and non-accepting states that satisfy* $A \cap N = \emptyset$. δ *is a subset of* $(Q \times ((\Sigma \times \Gamma^2 \cup \{[\triangleright, [\triangleright, \triangleright]], [\triangleleft, [\triangleleft, \triangleleft]]\}) \cup \{-1, 0, +1\}) \times Q)$ *that determines the transition relation on M' s configurations. Here,* $-1, 0,$ *and* $+1$ *stand for left-shift, no-shift, and right-shift of the head, respectively. In what follows, we also use* $-$ and $+$ *instead of* -1 *and* $+1$ *for simplicity. Note that only reading is permitted on the input track, while both reading and writing are allowed on the storage track.*

Each element $r = [p, x, q] \in \delta$ is called a *rule* of M in the triple form, where $x = [a, [b, c]] \in \Sigma \times \Gamma^2 \cup \{[\triangleright, [\triangleright, \triangleright]], [\triangleleft, [\triangleleft, \triangleleft]]\}$ or $x = d \in \{-, 0, +\}$. A rule of the form $[p, [a, [b, c]], q]$ is called a *read-write rule*, and means if M is in the state p and reads an input symbol a and a storage symbol b, then rewrites the latter to c, and enters the state q. Here, $\triangleright$ and $\triangleleft$ should not be rewritten to any other symbol. A rule of the form $[p, d, q]$ is called a *shift rule*, and means if M is in the state p, then shifts the head to the direction d, and enters the state q.

Let $q \in Q$, $w \in \Sigma^*$, $v \in \Gamma^*$ such that $|v| = |w|$, and $h \in \{0, 1, \ldots, |w| + 1\}$. Then, $[\triangleright w \triangleleft, \triangleright v \triangleleft, q, h]$ is called a *computational configuration* (or simply a *configuration*) of M with an input w. It means that the contents of the input track and the storage track are w and v, the finite-state control is in the state q, and the head position is h, where the position of $\triangleright$ is 0.

Let S be a set of symbols. A partial function $s : S^+ \times \mathbb{N} \to S$ is defined as follows, where $\mathbb{N}$ is the set of all non-negative integers. If $x = x_0 x_1 \cdots x_{n-1}$ $(x_i \in S)$, then $s(x, j) = x_j$ for $0 \leq j < n$, and $s(x, j)$ is undefined for $j \geq n$. Hence, $s(x, j)$ gives the j-th symbol of x, where the leftmost symbol is the 0-th.

The *transition relation* $\vdash_M$ between a pair of configurations $[\triangleright w \triangleleft, \triangleright v \triangleleft, q, h]$ and $[\triangleright w \triangleleft, \triangleright v' \triangleleft, q', h']$ is defined as follows.

$$[\triangleright w \triangleleft, \triangleright v \triangleleft, q, h] \vdash_M [\triangleright w \triangleleft, \triangleright v' \triangleleft, q', h'] \text{ iff (1) or (2) holds.}$$

(1) $[q, [s(\triangleright w \triangleleft, h), [s(\triangleright v \triangleleft, h), s(\triangleright v' \triangleleft, h')]], q'] \in \delta \ \wedge \ h' = h \ \wedge$
$\forall j \, (0 \leq j \leq |v| + 1 \ \wedge \ j \neq h \ \Rightarrow \ s(\triangleright v' \triangleleft, j) = s(\triangleright v \triangleleft, j))$

(2) $[q, h' - h, q'] \in \delta \ \wedge \ v' = v$

The reflexive and transitive closure, and the transitive closure of the relation $\vdash_M$ is denoted by $\vdash_M^*$ and $\vdash_M^+$, respectively. A configuration $[\triangleright w\triangleleft, \triangleright \#^{|w|}\triangleleft, q_0, 0]$ is called an *initial configuration* with an input $w \in \Sigma^*$. A configuration C is called a *halting configuration* if there is no configuration C' such that $C \vdash_M C'$.

We say $w \in \Sigma^*$ is *accepted* by M if $[\triangleright w\triangleleft, \triangleright \#^{|w|}\triangleleft, q_0, 0] \vdash_M^* [\triangleright w\triangleleft, \triangleright v\triangleleft, q, h]$ for some $q \in A$, $v \in \Gamma^{|w|}$, and $h \in \{0, 1, \ldots, |w|+1\}$. The *language* accepted by M is the set of all words accepted by M, and denoted by $L(M)$.

$$L(M) = \{w \in \Sigma^* \mid [\triangleright w\triangleleft, \triangleright \#^{|w|}\triangleleft, q_0, 0] \vdash_M^* [\triangleright w\triangleleft, \triangleright v\triangleleft, q, h] \text{ for some } q \in A,\ v \in \Gamma^{|w|},\ \text{and } h \in \{0, 1, \ldots, |w|+1\}\ \}$$

The set N of non-accepting states is not used in the definition of acceptance. But, it is convenient to specify it for the later construction of reversible LBAs.

Definition 5. *An LBA* $M = (Q, \Sigma, \Gamma, \delta, \triangleright, \triangleleft, \#, q_0, A, N)$ *is called a* deterministic LBA *(DLBA) iff the following* determinism condition *holds.*

$$\begin{array}{l}\forall\, r_1 = [p, x, q] \in \delta,\ \forall\, r_2 = [p', x', q'] \in \delta : \\ \quad (r_1 \neq r_2\ \wedge\ p = p') \Rightarrow\ (x \notin \{-, 0, +\}\ \wedge\ x' \notin \{-, 0, +\}\ \wedge \\ \quad \forall\, [a, [b, c]], [a'[b', c']] \in \Sigma \times \Gamma^2 \cup \{[\triangleright, [\triangleright, \triangleright]], [\triangleleft, [\triangleleft, \triangleleft]]\} \\ \quad\quad (x = [a, [b, c]]\ \wedge\ x' = [a', [b', c']]\ \Rightarrow\ [a, b] \neq [a', b']))\end{array}$$

It means that for any two distinct rules r_1 and r_2 *in* δ*, if the present states* p *and* p' *are the same, then they are both read-write rules, and the pairs of the input symbols and the read storage symbols* $[a, b]$ *and* $[a', b']$ *are different.*

Definition 6. *An LBA* $M = (Q, \Sigma, \Gamma, \delta, \triangleright, \triangleleft, \#, q_0, A, N)$ *is called a* reversible LBA *(RLBA) iff the following* reversibility condition *holds.*

$$\begin{array}{l}\forall\, r_1 = [p, x, q] \in \delta,\ \forall\, r_2 = [p', x', q'] \in \delta : \\ \quad (r_1 \neq r_2\ \wedge\ q = q') \Rightarrow\ (x \notin \{-, 0, +\}\ \wedge\ x' \notin \{-, 0, +\}\ \wedge \\ \quad \forall\, [a, [b, c]], [a'[b', c']] \in \Sigma \times \Gamma^2 \cup \{[\triangleright, [\triangleright, \triangleright]], [\triangleleft, [\triangleleft, \triangleleft]]\} \\ \quad\quad (x = [a, [b, c]]\ \wedge\ x' = [a', [b', c']]\ \Rightarrow\ [a, c] \neq [a', c']))\end{array}$$

It means that for any two distinct rules r_1 *and* r_2 *in* δ*, if the next states* q *and* q' *are the same, then they are both read-write rules and the pairs of the input symbols and the written storage symbols* $[a, c]$ *and* $[a', c']$ *are different.*

A rule $[p, x, q]$ is called a *deterministic rule* (*reversible rule*, respectively), if there is no rule $[p', x', q']$ such that the pair $([p, x, q], [p', x', q'])$ violates the determinism (reversibility) condition.

A reversible and deterministic LBA is denoted by RDLBA. In the following, we consider only DLBAs and RDLBAs. From the definition, it is easily seen that if M is deterministic, then for every configuration C of M there is at most one configuration C' such that $C \vdash_M C'$. Likewise, if M is reversible, then for every configuration C of M there is at most one configuration C' such that $C' \vdash_M C$.

We define a *computation graph* $G_{M,w} = (V, E)$ of M with an input $w \in \Sigma^*$ as follows. Let $\mathrm{Conf}(M, w)$ be the set of all configurations of M with w, i.e.,

$\mathrm{Conf}(M,w) = \{[\triangleright w\triangleleft, \triangleright v\triangleleft, q, h] \mid q \in Q \wedge v \in \Gamma^{|w|} \wedge h \in \{0,1,\ldots,|w|+1\}\}$. The set $V(\subset \mathrm{Conf}(M,w))$ of nodes is the smallest set that contains the initial configuration $[\triangleright w\triangleleft, \triangleright \#^{|w|}\triangleleft, q, h]$, and satisfies the following condition: $\forall C_1, C_2 \in \mathrm{Conf}(M,w)\,((C_1 \in V \wedge (C_1 \vdash_M C_2 \vee C_2 \vdash_M C_1)) \Rightarrow C_2 \in V)$. Namely, V is the set of all configurations connected to the initial configuration, and is finite. The set E of directed edges is: $E = \{(C_1, C_2) \mid C_1, C_2 \in V \wedge C_1 \vdash_M C_2\}$. If M is deterministic, then outdegree of each node in V is either 0 or 1, where a node of outdegree 0 corresponds to a halting configuration. On the other hand, if M is reversible, then indegree of each node in V is either 0 or 1.

In the following, we assume, without loss of generality, any given DLBA $M = (Q, \Sigma, \Gamma, \delta, \triangleright, \triangleleft, \#, q_0, A, N)$ satisfies the following conditions (C1)–(C6) for the later convenience. In fact, M is easily modified so that it satisfies them.

(C1) The initial state q_0 does not appear as the third component of a rule in δ: $\forall [q, x, q'] \in \delta \;\; (q' \neq q_0)$.

(C2) M performs read-write and shift operations alternately. Hence, Q is written as $Q = Q_{\mathrm{rw}} \cup Q_{\mathrm{sf}}$ for some Q_{rw} and Q_{sf} such that $Q_{\mathrm{rw}} \cap Q_{\mathrm{sf}} = \emptyset$, and δ satisfies the following condition:
$\forall\, [p, x, q] \in \delta$
$((x \in \Sigma \times \Gamma^2 \cup \{[\triangleright, [\triangleright, \triangleright]], [\triangleleft, [\triangleleft, \triangleleft]]\} \;\Rightarrow\; p \in Q_{\mathrm{rw}} \wedge q \in Q_{\mathrm{sf}}) \wedge$
$(x \in \{-, 0, +\} \;\Rightarrow\; p \in Q_{\mathrm{sf}} \wedge q \in Q_{\mathrm{rw}}))$
We can easily modify M so that it satisfies the above condition by adding new states to it. Each element of Q_{rw} and Q_{sf} is called a *read-write state* and a *shift state*, respectively. We further assume $q_0 \in Q_{\mathrm{rw}}$, and $A \cup N \subset Q_{\mathrm{sf}}$, though each state in $A \cup N$ makes no further move as in (C3).

(C3) Every state in $A \cup N$ is a halting state in Q_{sf}, and vice versa:
$\forall q \in Q \;\; (q \in A \cup N \;\Leftrightarrow\; q \in Q_{\mathrm{sf}} \wedge \neg\exists [q, x, q'] \in \delta)$.

(C4) If M reads a left (right, respectively) endmarker, then in the next step the shift direction of the head is to the right (left):
$\forall p, r \in Q_{\mathrm{rw}},\; \forall q \in Q_{\mathrm{sf}},$
$\forall [a, [b, c]] \in (\Sigma \times \Gamma^2 \cup \{[\triangleright, [\triangleright, \triangleright]], [\triangleleft, [\triangleleft, \triangleleft]]\}),\; \forall d \in \{-, 0, +\}$
$([p, [a, [b, c]], q], [q, d, r] \in \delta \;\Rightarrow (a = \triangleright \;\Rightarrow\; b = c = \triangleright \wedge d = +) \wedge$
$(a = \triangleleft \;\Rightarrow\; b = c = \triangleleft \wedge d = -))$
Likewise, if M reads a left (right, respectively) endmarker, then in the previous step the shift direction of the head is to the left (right):
$\forall p, r \in Q_{\mathrm{sf}},\; \forall q \in Q_{\mathrm{rw}},$
$\forall [a, [b, c]] \in (\Sigma \times \Gamma^2 \cup \{[\triangleright, [\triangleright, \triangleright]], [\triangleleft, [\triangleleft, \triangleleft]]\}),\; \forall d \in \{-, 0, +\}$
$([r, d, q], [q, [a, [b, c]], p] \in \delta \;\Rightarrow (a = \triangleright \;\Rightarrow\; b = c = \triangleright \wedge d = -) \wedge$
$(a = \triangleleft \;\Rightarrow\; b = c = \triangleleft \wedge d = +))$

(C5) Just after M starts to move, it confirms the storage track contains only blank symbols #s. It is done by replacing the rule $[q_0, [\triangleright, [\triangleright, \triangleright]], q]$ by
$[q_0, [\triangleright, [\triangleright, \triangleright]], q_{0,1}]$, $[q_{0,1}, +, q_{0,2}]$, $[q_{0,2}, [a, [\#, \#]], q_{0,1}]$,
$[q_{0,2}, [\triangleleft, [\triangleleft, \triangleleft]], q_{0,3}]$, $[q_{0,3}, -, q_{0,4}]$, $[q_{0,4}, [a, [\#, \#]], q_{0,3}]$, $[q_{0,4}, [\triangleright, [\triangleright, \triangleright]], q]$.

Here, $q_{0,1}, q_{0,2}, q_{0,3}, q_{0,4}$ are new states, and the rules $[q_{0,2}, [a, [\#, \#]], q_{0,1}]$ and $[q_{0,4}, [a, [\#, \#]], q_{0,3}]$ are added for each $a \in \Sigma$. Note that there is only one rule that has q_0 as the first component since (C1) and (C2) hold.

We define the following functions to give the condition (C6) below: prev-rw : $Q_{\rm rw} \to 2^{Q_{\rm sf}\times\{-,0,+\}}$, prev-sf : $Q_{\rm sf} \times (\Sigma\times\Gamma \cup \{[\rhd,\rhd],[\lhd,\lhd]\}) \to 2^{Q_{\rm rw}\times(\Gamma\cup\{\rhd,\lhd\})}$, $\deg_{\rm rw} : Q_{\rm rw} \to \mathbb{N}$, and $\deg_{\rm sf} : Q_{\rm sf} \times (\Sigma\times\Gamma \cup \{[\rhd,\rhd],[\lhd,\lhd]\}) \to \mathbb{N}$ as follows, where $Q_{\rm rw}$ and $Q_{\rm sf}$ are the sets given in (C2).

$$\begin{aligned}
\text{prev-rw}(q) &= \{[p,d] \mid p \in Q_{\rm sf} \ \wedge\ d \in \{-,0,+\} \ \wedge\ [p,d,q] \in \delta\} \\
\text{prev-sf}(q,a,c) &= \{[p,b] \mid p \in Q_{\rm rw} \ \wedge\ b \in (\Gamma \cup \{\rhd,\lhd\}) \ \wedge\ [p,[a,[b,c]],q] \in \delta\} \\
\deg_{\rm rw}(q) &= |\text{prev-rw}(q)| \\
\deg_{\rm sf}(q,a,c) &= |\text{prev-sf}(q,a,c)|
\end{aligned}$$

Assume M is in the configuration $[\rhd w\lhd, \rhd v\lhd, q, h]$. If q is a read-write state (shift state, respectively), then $\deg_{\rm rw}(q)$ ($\deg_{\rm sf}(q, s(\rhd w\lhd, h), s(\rhd v\lhd, h))$) gives the total number of previous configurations of $[\rhd w\lhd, \rhd v\lhd, q, h]$. Each element $[p,d] \in$ prev-rw(q) ($[p,b] \in$ prev-sf$(q, s(\rhd w\lhd, h), s(\rhd v\lhd, h))$, respectively) gives a previous state and a shift direction (a previous state and a previous storage symbol). If M is an RDLBA, then $\deg_{\rm rw}(q) \le 1$ and $\deg_{\rm sf}(q,a,c) \le 1$ hold for any $q \in Q$, and $(a,c) \in (\Sigma\times\Gamma \cup \{[\rhd,\rhd],[\lhd,\lhd]\})$.

(C6) M satisfies $\deg_{\rm rw}(q) \le 1$ for all $q \in Q_{\rm rw}$. If otherwise, we modify M as follows. If there is a pair of shift rules $[p,d_1,q]$ and $[p',d_2,q]$ in δ, then add a new state q' in Q, remove $[p',d_2,q]$ from δ, and add rules $[p',d_2,q']$ and $[q',[a,[b,c]],r]$ for each $[q,[a,[b,c]],r] \in \delta$. Hence, $\deg_{\rm sf}(r,a,c)$ increases, but $\deg_{\rm rw}(q)$ decreases. Repeat this procedure until no such pair exits.

We first show Lemma 1 stating that an RDLBA always halts. It is proved in a similar manner to the case of a reversible multi-head finite automaton [5].

Lemma 1. *Let $M = (Q,\Sigma,\Gamma,\delta,\rhd,\lhd,\#,q_0,A,N)$ be an RDLBA that satisfies (C1). Then, it eventually halts for any input $w \in \Sigma^*$.*

Proof. Let $C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \cdots$ be a computation of M starting from the initial configuration $C_0 = [\rhd w\lhd, \rhd\#^{|w|}\lhd, q_0, 0]$ with an input $w \in \Sigma^*$. First, we show M never loops for any w. Assume, on the contrary, it loops, i.e., there exists a pair of integers (i,j) such that $0 \le i < j$ and $C_i = C_j$. Let (i_0,j_0) be the pair such that i_0 is the least integer among such (i,j)-pairs. By the condition (C1), there is no configuration C_{-1} that satisfy $C_{-1} \vdash_M C_0$. Hence, $C_0 \ne C_{i_0} = C_{j_0}$, and thus $i_0 > 0$. Therefore $C_{i_0-1} \ne C_{j_0-1}$. But, since $C_{i_0-1} \vdash_M C_{i_0}$, $C_{j_0-1} \vdash_M C_{j_0}$, and $C_{i_0} = C_{j_0}$ hold, it contradicts the assumption M is reversible. Therefore, M never loops. On the other hand, the total number of configurations reachable from C_0 is bounded by $|Q| \cdot |\Gamma|^{|w|} \cdot (|w|+2)$. Hence, M halts for any input w. □

We now give a method of converting a DLBA to an RDLBA in Lemma 2.

Lemma 2. *Let $M = (Q,\Sigma,\Gamma,\delta,\rhd,\lhd,\#,q_0,A,N)$ be a DLBA. We can construct an RDLBA $M^\dagger = (Q^\dagger,\Sigma,\Gamma,\delta^\dagger,\rhd,\lhd,\#,q_0,\{\hat{q}_0^{\rm b}\},\{q_0^{\rm b}\})$ such that the following holds, and thus $L(M^\dagger) = L(M)$.*

$$\begin{aligned}
\forall w \in \Sigma^* \ (&(w \in L(M) \ \Rightarrow\ [\rhd w\lhd, \rhd\#^{|w|}\lhd, q_0, 0] \ \vdash^+_{M^\dagger}\ [\rhd w\lhd, \rhd\#^{|w|}\lhd, \hat{q}_0^{\rm b}, 0]\,) \wedge \\
&(w \notin L(M) \ \Rightarrow\ [\rhd w\lhd, \rhd\#^{|w|}\lhd, q_0, 0] \ \vdash^+_{M^\dagger}\ [\rhd w\lhd, \rhd\#^{|w|}\lhd, q_0^{\rm b}, 0]\,))
\end{aligned}$$

Proof. In our construction, $M^\dagger$ traverses the computation graph $G_{M,w}$ from the initial configuration of M with an input w to find an accepting one as shown in Fig. 3. We assume M satisfies the conditions (C1)–(C6). We further assume the sets Q, and $\Gamma \cup \{\rhd, \lhd\}$ are totally ordered, and the elements of the set prev-sf(q, a, c) is sorted by these orders. Thus, we express it by an ordered list as below. Note that since M satisfies (C6), $|\text{prev-rw}(q)| \leq 1$ holds for all $q \in Q_{\text{rw}}$.

$$\text{prev-sf}(q, a, c) = [[p_1, b_1], \ldots, [p_k, b_k]], \text{ where } k = \deg_{\text{sf}}(q, a, c)$$

Then, $Q^\dagger = \{q, \hat{q}, q^{\text{b}}, \hat{q}^{\text{b}} \mid q \in Q\}$, and $\delta^\dagger$ is defined as below.

$$\begin{aligned}
\delta^\dagger = {} & \delta_1 \cup \cdots \cup \delta_4 \cup \hat{\delta}_1 \cup \cdots \cup \hat{\delta}_4 \cup \delta_A \cup \delta_N \\
\delta_1 = {} & \{\, [q^{\text{b}}, [a, [c, b_1]], p_1^{\text{b}}],\ [p_1, [a, [b_1, b_2]], p_2^{\text{b}}], [p_2, [a, [b_2, b_3]], p_3^{\text{b}}], \ldots, \\
& \quad [p_{k-1}, [a, [b_{k-1}, b_k]], p_k^{\text{b}}],\ [p_k, [a, [b_k, c]], q] \mid \\
& \quad q \in Q_{\text{sf}} \ \wedge\ (a, c) \in (\Sigma \times \Gamma \cup \{(\rhd, \rhd), (\lhd, \lhd)\}) \ \wedge\ \deg_{\text{sf}}(q, a, c) \geq 1 \\
& \quad \wedge\ \text{prev-sf}(q, a, c) = [[p_1, b_1], \ldots, [p_k, b_k]], \text{ where } k = \deg_{\text{sf}}(q, a, c) \,\} \\
\delta_2 = {} & \{\, [q^{\text{b}}, -d, p^{\text{b}}],\ [p, d, q] \mid q \in Q_{\text{rw}} \wedge \text{prev-rw}(q) = [[p, d]] \,\} \\
\delta_3 = {} & \{\, [q^{\text{b}}, [a, [c, c]], q] \mid q \in Q_{\text{sf}} - (A \cup N) \\
& \quad \wedge\ (a, c) \in (\Sigma \times \Gamma \cup \{(\rhd, \rhd), (\lhd, \lhd)\}) \ \wedge\ \deg_{\text{sf}}(q, a, c) = 0 \,\} \\
\delta_4 = {} & \{\, [q, [a, [b, b]], q^{\text{b}}] \mid q \in Q_{\text{rw}} - \{q_0\} \\
& \quad \wedge\ (a, b) \in (\Sigma \times \Gamma \cup \{(\rhd, \rhd), (\lhd, \lhd)\}) \ \wedge\ \neg\exists c \exists p\, ([q, [a, [b, c]], p] \in \delta) \,\} \\
\hat{\delta}_i = {} & \{\, [\hat{p}, x, \hat{q}] \mid [p, x, q] \in \delta_i \,\}\ (i = 1, \ldots, 4) \\
\delta_A = {} & \{\, [q, 0, \hat{q}^{\text{b}}] \mid q \in A \,\} \cup \{\, [\hat{q}, 0, q^{\text{b}}] \mid q \in A \,\} \\
\delta_N = {} & \{\, [q, 0, q^{\text{b}}] \mid q \in N \,\} \cup \{\, [\hat{q}, 0, \hat{q}^{\text{b}}] \mid q \in N \,\}
\end{aligned}$$

$Q^\dagger$ has four types of states. They are of the forms $q, \hat{q}, q^{\text{b}}$ and $\hat{q}^{\text{b}}$. The states without a superscript (i.e., q and $\hat{q}$) are for forward computation, while those with a superscript "b" (i.e., q^{b} and $\hat{q}^{\text{b}}$) are for backward computation. The states with "$\hat{}$" (i.e., $\hat{q}$ and $\hat{q}^{\text{b}}$) are the ones indicating that an accepting configuration of M was found in the process of traversal, while those without "$\hat{}$" (i.e., q and q^{b}) are for indicating no accepting configuration has been found so far.

$\delta_1, \ldots, \delta_4$ are the sets of rules for the states without "$\hat{}$", and $\hat{\delta}_1, \ldots, \hat{\delta}_4$ are the ones of corresponding rules for the states with "$\hat{}$". δ_1 and $\hat{\delta}_1$ are for searching the graph $G_{M,w}$ at a shift state of M. See, for example, the node with a shift state q_3 in Fig. 3 (a). By the rules in δ_1 and $\hat{\delta}_1$, the graph $G_{M,w}$ is searched by the states of $M^\dagger$ from $\hat{q}_3^{\text{b}}$ to $\hat{q}_5^{\text{b}}$, from $\hat{q}_5$ to $\hat{q}_0^{\text{b}}$, from q_0 to q_6^{b}, and from q_6 to q_3. δ_2 and $\hat{\delta}_2$ are for searching $G_{M,w}$ at a read-write states of M. For example, see the node with a read-write state q_1 in Fig. 3 (a). By these rules the graph is searched from $\hat{q}_1^{\text{b}}$ to $\hat{q}_3^{\text{b}}$, and from q_3 to q_1. δ_3 and $\hat{\delta}_3$ are for turning the direction of search from backward to forward in $G_{M,w}$ for a shift state. See, for example, the node with the shift state q_9 in Fig. 3 (a), where the state of $M^\dagger$ changes from $\hat{q}_9^{\text{b}}$ to $\hat{q}_9$. δ_4 and $\hat{\delta}_4$ are for turning the direction from forward to backward in for halting configuration with a read-write state. There is no example of this type in Fig. 3. But, if the configuration with q_2 were such a one, then the state of $M^\dagger$ changes from q_2 to q_2^{b}. δ_A (δ_N, respectively) is for turning the search direction from forward to backward for accepting (non-accepting) states. In addition, each rule in δ_A makes $M^\dagger$ change the state from a one without "$\hat{}$" to the corresponding one with "$\hat{}$". Note that the sets of rules

$\{[\hat{q}, 0, q^{\mathrm{b}}] \mid q \in A\} \subset \delta_A$ and $\{[\hat{q}, 0, \hat{q}^{\mathrm{b}}] \mid q \in N\} \subset \delta_N$ are not used to simulate M, but for keeping symmetry between the states with "$\hat{}$" and those without "$\hat{}$".

We can verify $M^\dagger$ is deterministic and reversible. For example, consider the rules in δ_1. Since prev-sf$(q, a, c) = [[p_1, b_1], \ldots, [p_k, b_k]]$ $(k = \deg_{\mathrm{sf}}(q, a, c) \geq 1)$, there are rules $[p_1, [a, [b_1, c], q]], [p_2, [a, [b_2, c], q]], \ldots, [p_k, [a, [b_k, c], q]]$ in δ of M. First, $[q^{\mathrm{b}}, [a, [c, b_1]], p_1^{\mathrm{b}}] \in \delta_1$ is a deterministic rule, because it is the sole rule of the form $[q^{\mathrm{b}}, [a, [c, x]], y^{\mathrm{b}}]$ (for some $x \in \Gamma \cup \{\rhd, \lhd\}$ and $y \in Q$) for the combination (q, a, c). It is also a reversible rule, since $[p_1, [a, [b_1, c], q]] \in \delta$ is a deterministic rule. Second, $[p_i, [a, [b_i, b_{i+1}]], p_{i+1}^{\mathrm{b}}] \in \delta_1$ $(i = 1, \ldots, k-1)$ is deterministic, since $[p_i, [a, [b_i, c], q]] \in \delta$ is deterministic. It is reversible, since $[p_{i+1}, [a, [b_{i+1}, c], q]] \in \delta$ is deterministic. Third, $[p_k, [a, [b_k, c]], q] \in \delta_1$ is deterministic, since $[p_k, [a, [b_k, c], q]] \in \delta$ is deterministic. It is also reversible, since it is the sole rule of the form $[x, [a, [y, c]], q] \in \delta_1$ (for some $x \in Q$ and $y \in \Gamma \cup \{\rhd, \lhd\}$) for the combination (q, a, c). It is also easy to verify that other rules in $\delta^\dagger$ are deterministic and reversible.

We can also verify that the constructed $M^\dagger$ also satisfies the conditions (C1)–(C6) except (C4) (since there are rules of the form $[p, 0, q]$ in δ_A and δ_N). For example, (C2) can be verified from the following fact: if $q, \hat{q} \in Q_{\mathrm{rw}}^\dagger$ $(q, \hat{q} \in Q_{\mathrm{sf}}^\dagger$, respectively), then $q^{\mathrm{b}}, \hat{q}^{\mathrm{b}} \in Q_{\mathrm{sf}}^\dagger$ $(q^{\mathrm{b}}, \hat{q}^{\mathrm{b}} \in Q_{\mathrm{rw}}^\dagger)$.

Now, consider the case where M finally halts in a configuration C_{h}. Then $G_{M,w}$ becomes a finite tree with the root C_{h}. Given the input w, $M^\dagger$ starts to search $G_{M,w}$. As explained above, from each node, $M^\dagger$ visits all of its child nodes one after another, and thus $M^\dagger$ will perform a depth-first search of a tree (Fig. 3 (a)). Note that the search starts not from the root of the tree but from the leaf node $[\rhd w \lhd, \rhd \#^{|w|} \lhd, q_0, 0]$. Since each node of $G_{M,w}$ is identified by the configuration of M of the form $[\rhd w \lhd, \rhd v \lhd, q, h]$, it is easy for $M^\dagger$ to keep it by the configuration of $M^\dagger$ itself.

If $M^\dagger$ enters an accepting state of M, say q_{a}, which is the root of the tree while traversing the tree, then $M^\dagger$ goes to the state $\hat{q}_{\mathrm{a}}^{\mathrm{b}}$ by a rule in δ_A, and continues the depth-first search. After that, $M^\dagger$ uses the states of the form $\hat{q}$ and $\hat{q}^{\mathrm{b}}$ indicating that the input w should be accepted. $M^\dagger$ will eventually reach the initial configuration of M by its configuration $[\rhd w \lhd, \rhd \#^{|w|} \lhd, \hat{q}_0^{\mathrm{b}}, 0]$. Thus, $M^\dagger$ halts and accepts the input.

If $M^\dagger$ enters a halting state of M other than the accepting states, then by a rule in $\delta_N \cup \delta_4$, and then by rules in $\delta_1 \cup \delta_2 \cup \delta_3$ it continues the depth-first search without entering a state of the form $\hat{q}$. Also in this case, $M^\dagger$ will finally reach the initial configuration of M by its configuration $[\rhd w \lhd, \rhd \#^{|w|} \lhd, q_0^{\mathrm{b}}, 0]$. Thus, $M^\dagger$ halts and rejects the input.

We can see $M^\dagger$ halts either in the configuration $[\rhd w \lhd, \rhd \#^{|w|} \lhd, \hat{q}_0^{\mathrm{b}}, 0]$ or $[\rhd w \lhd, \rhd \#^{|w|} \lhd, q_0^{\mathrm{b}}, 0]$ by the following reasons. First, $M^\dagger$ does not halt in a state other than $\hat{q}_0^{\mathrm{b}}$ and q_0^{b}, since $\delta^\dagger$ is so designed that $M^\dagger$ continues the traversal at any node of $G_{M,w}$ such that M's state is not q_0. Second, $M^\dagger$ does not halt in a configuration $[\rhd w' \lhd, \rhd v \lhd, \hat{q}_0^{\mathrm{b}}, h]$ or $[\rhd w' \lhd, \rhd v \lhd, q_0^{\mathrm{b}}, h]$ for some $w' \in \Sigma^*$ such that $w' \neq w$, $v \in \Gamma^{|w'|}$ and $h \in \{0, \ldots, |w'| + 1\}$, since input symbols are not rewritten. Note that, if M rewrites input symbols, then $G_{M,w}$

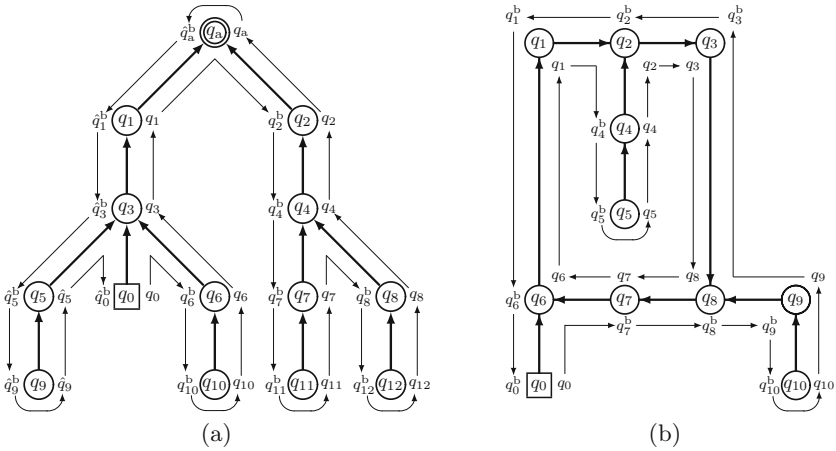

Fig. 3. Examples of computation graphs $G_{M,w}$ of an DLBA M. Each node represents a configuration of M, though only a state of the finite-state control is written in a circle. The node labeled by q_0 represents the initial configuration of M. An RDLBA $M^\dagger$ traverses these graphs along thin arrows using its configurations, and finally halts. (a) A case M halts in an accepting state q_a. (b) A case M loops forever.

may have two or more initial configurations, and thus $M^\dagger$ does not traverse $G_{M,w}$ entirely even if it is a tree. Third, $M^\dagger$ does not halt in a configuration $[\triangleright w\triangleleft, \triangleright v\triangleleft, \hat{q}_0^\mathrm{b}, h]$ or $[\triangleright w\triangleleft, \triangleright v\triangleleft, q_0^\mathrm{b}, h]$ for some $v \in \Gamma^{|w|}$ and $h \neq 0$. Since the initial configuration of M is $[\triangleright w\triangleleft, \triangleright \#^{|w|}\triangleleft, q_0, 0]$, and (C1) holds, we can assume there is no rule of the form $[q_0, [a, [b, c]], q]$ with $[a, [b, c]] \neq [\triangleright, [\triangleright, \triangleright]]$ in δ. Hence, $[\triangleright w\triangleleft, \triangleright v\triangleleft, q_0, h]$ with $h \neq 0$ is not a node of $G_{M,w}$. Fourth, the case that $M^\dagger$ halts in $[\triangleright w\triangleleft, \triangleright v\triangleleft, \hat{q}_0^\mathrm{b}, 0]$ or $[\triangleright w\triangleleft, \triangleright v\triangleleft, q_0^\mathrm{b}, 0]$ for some $v \neq \#^{|w|}$ is also inhibited. If it starts from $[\triangleright w\triangleleft, \triangleright v\triangleleft, q_0, 0]$ with $v \neq \#^{|w|}$, it halts in $q_{0,2}$, because of (C5). Hence, $[\triangleright w\triangleleft, \triangleright v\triangleleft, q_0, 0]$ with $v \neq \#^{|w|}$ is not a node of $G_{M,w}$.

Next, consider the case where M enters a loop. Then $G_{M,w}$ is not a tree, but a finite graph (Fig. 3 (b)). In this case, since there is no accepting configuration in $G_{M,w}$, $M^\dagger$ never reaches an accepting state of M no matter how $M^\dagger$ visits the nodes of $G_{M,w}$ (it may not visit all the nodes of $G_{M,w}$). Thus, $M^\dagger$ uses only the states without "$\hat{}$". Since M satisfies the condition (C1), $M^\dagger$ eventually halts by Lemma 1. By the same argument as in the case $G_{M,w}$ is a tree, $M^\dagger$ must halt in the configuration $[\triangleright w\triangleleft, \triangleright \#^{|w|}\triangleleft, q_0^\mathrm{b}, 0]$. By above, the theorem holds. □

Example 1. Consider a DLBA M_p that accepts all well-formed parentheses.

$$
\begin{aligned}
M_\mathrm{p} &= (Q, \{\ (,\ )\ \}, \{\#, \mathrm{x}\}, \delta, \triangleright, \triangleleft, \#, q_0, \{q_\mathrm{a}\}, \{q_\mathrm{r}\}) \\
Q &= \{q_0, q_{0,1}, q_{0,2}, q_{0,3}, q_{0,4}, q_1, q_2, q_3, q_4, q_5, q_6, q_\mathrm{a}, q_\mathrm{r}\} \\
\delta &= \{\, [q_0, [\triangleright, [\triangleright, \triangleright]], q_{0,1}],\ [q_{0,1}, +, q_{0,2}],\ [q_{0,2}, [\ (, [\#, \#]], q_{0,1}],\ [q_{0,2}, [\ ), [\#, \#]], q_{0,1}], \\
&\quad [q_{0,2}, [\triangleleft, [\triangleleft, \triangleleft]], q_{0,3}],\ [q_{0,3}, -, q_{0,4}],\ [q_{0,4}, [\ (, [\#, \#]], q_{0,3}],\ [q_{0,4}, [\ ), [\#, \#]], q_{0,3}], \\
&\quad [q_{0,4}, [\triangleright, [\triangleright, \triangleright]], q_1],\ [q_1, +, q_2],\ [q_2, [\ (, [\#, \#]], q_1],\ [q_2, [\ (, [\mathrm{x}, \mathrm{x}]], q_1], \\
&\quad [q_2, [\ ), [\mathrm{x}, \mathrm{x}]], q_1],\ [q_2, [\ ), [\#, \mathrm{x}]], q_3],\ [q_2, [\triangleleft, [\triangleleft, \triangleleft]], q_5],\ [q_3, -, q_4], \\
&\quad [q_4, [\ (, [\mathrm{x}, \mathrm{x}]], q_3],\ [q_4, [\ ), [\mathrm{x}, \mathrm{x}]], q_3],\ [q_4, [\ (, [\#, \mathrm{x}]], q_1],\ [q_4, [\triangleright, [\triangleright, \triangleright]], q_\mathrm{r}], \\
&\quad [q_5, -, q_6],\ [q_6, [\ (, [\mathrm{x}, \mathrm{x}]], q_5],\ [q_6, [\ ), [\mathrm{x}, \mathrm{x}]], q_5],\ [q_6, [\ (, [\#, \#]], q_\mathrm{r}], \\
&\quad [q_6, [\triangleright, [\triangleright, \triangleright]], q_\mathrm{a}]\ \}
\end{aligned}
$$

If an input $w \in \{\ (,\ )\ \}^*$ is given, $M_{\rm p}$ first checks if the condition (C5) holds by the states $q_{0,1}, q_{0,2}, q_{0,3}$, and $q_{0,4}$. Next, it scans the input to the right to find the leftmost ")" using the states q_1 and q_2, and mark it by "x" on the storage track. Then, it scans the input to the left to find the corresponding "(", and also mark it by "x" by the states q_3 and q_4. $M_{\rm p}$ repeats this procedure until all ")"s are marked. Note that already marked parentheses are ignored. It finally checks if no unmatched parenthesis exists by the states q_5 and q_6. $M_{\rm p}$ is irreversible, since the pairs $([q_2, [\,(, [\mathrm{x}, \mathrm{x}]], q_1]$, $[q_4, [\,(, [\#, \mathrm{x}]], q_1])$ and $([q_2, [\,)], [\#, \mathrm{x}]], q_3]$, $[q_4, [\,)], [\mathrm{x}, \mathrm{x}]], q_3])$ violate the condition of Definition 6. We can see it satisfies (C1)–(C6). Examples of its computation are as below.

$$[\rhd(\,)\lhd, \rhd\#\#\lhd, q_0, 0] \mid\!\!\frac{29}{M_{\rm p}} [\rhd(\,)\lhd, \rhd\mathrm{xx}\lhd, q_{\rm a}, 0]$$
$$[\rhd(\,(\,)\,(\,)\,)\lhd, \rhd\#\#\#\#\#\#\lhd, q_0, 0] \mid\!\!\frac{85}{M_{\rm p}} [\rhd(\,(\,)\,(\,)\,)\lhd, \rhd\mathrm{xxxxxx}\lhd, q_{\rm a}, 0]$$
$$[\rhd(\,(\,)\,(\,)\lhd, \rhd\#\#\#\#\#\lhd, q_0, 0] \mid\!\!\frac{55}{M_{\rm p}} [\rhd(\,(\,)\,(\,)\lhd, \rhd\#\mathrm{xxxx}\lhd, q_{\rm r}, 1]$$

An RDLBA $M_{\rm p}^\dagger$ that simulates $M_{\rm p}$ obtained by the method in Lemma 2 is:

$$M_{\rm p}^\dagger = (Q^\dagger, \{\ (,\ )\ \}, \{\#, \mathrm{x}\}, \delta^\dagger, \rhd, \lhd, \#, q_0, \{\hat{q}_0^{\rm b}\}, \{q_0^{\rm b}\}),$$

where $Q^\dagger = \{q, \hat{q}, q^{\rm b}, \hat{q}^{\rm b} \mid q \in Q\}$. $\delta^\dagger$ has 152 rules, and is not described here.

Examples of computing processes of $M_{\rm p}^\dagger$ are as follows.

$$[\rhd(\,)\lhd, \rhd\#\#\lhd, q_0, 0] \mid\!\!\frac{71}{M_{\rm p}^\dagger} [\rhd(\,)\lhd, \rhd\#\#\lhd, \hat{q}_0^{\rm b}, 0]$$
$$[\rhd(\,(\,)\,(\,)\,)\lhd, \rhd\#\#\#\#\#\#\lhd, q_0, 0] \mid\!\!\frac{739}{M_{\rm p}^\dagger} [\rhd(\,(\,)\,(\,)\,)\lhd, \rhd\#\#\#\#\#\#\lhd, \hat{q}_0^{\rm b}, 0]$$
$$[\rhd(\,(\,)\,(\,)\lhd, \rhd\#\#\#\#\#\lhd, q_0, 0] \mid\!\!\frac{239}{M_{\rm p}^\dagger} [\rhd(\,(\,)\,(\,)\lhd, \rhd\#\#\#\#\#\lhd, q_0^{\rm b}, 0]$$

4 Simulating RDLBA by RPCA

Lemma 3. *For any DLBA $M = (Q, \Sigma, \Gamma, \delta, \rhd, \lhd, \#, q_0, A, N)$, we can construct an RPCA P_M such that $L(P_M) = L(M)$.*

Proof. Let $M^\dagger = (Q^\dagger, \Sigma, \Gamma, \delta^\dagger, \rhd, \lhd, \#, q_0, \{\hat{q}_0^{\rm b}\}, \{q_0^{\rm b}\})$ be the RDLBA converted from M by the method given in Lemma 2. Here, we design P_M so that it simulates $M^\dagger$. The simulation method is based on the one given in [4,7], but here P_M should be constructed so that it satisfies the condition (P1).

From the method shown in Lemma 2 it is easy to see that $M^\dagger$ also satisfies the condition (C2) as well as M. Let $Q_{\rm rw}^\dagger$ and $Q_{\rm sf}^\dagger$ be the sets of read-write states and shift states, respectively, where $Q^\dagger = Q_{\rm rw}^\dagger \cup Q_{\rm sf}^\dagger$, and $Q_{\rm rw}^\dagger \cap Q_{\rm sf}^\dagger = \emptyset$. Let $Q_-^\dagger, Q_0^\dagger$, and $Q_+^\dagger$, which are subsets of $Q_{\rm rw}^\dagger$, be as follows: $Q_-^\dagger = \{q \mid \exists\, p\, ([p, -, q] \in \delta^\dagger)\}$, $Q_0^\dagger = \{q \mid \exists\, p\, ([p, 0, q] \in \delta^\dagger)\}$, and $Q_+^\dagger = \{q \mid \exists\, p\, ([p, +, q] \in \delta^\dagger)\}$. Since $M^\dagger$ satisfies the reversibility condition and (C1), $Q_-^\dagger, Q_0^\dagger$, and $Q_+^\dagger$ are mutually disjoint, and $(Q_-^\dagger \cup Q_0^\dagger \cup Q_+^\dagger) \cap \{q_0, \hat{q}_0\} = \emptyset$. Note that, here we assume there

is no "useless" state in Q of M that never appears as the third component of a rule except q_0. Thus, $Q^\dagger_{\rm rw} = Q^\dagger_- \cup Q^\dagger_0 \cup Q^\dagger_+ \cup \{q_0, \hat{q}_0\}$ holds.

P_M is defined as follows.

$$
\begin{aligned}
&P_M = ((L, C, R), f, (\#, \#, \#), r_{\rm s}, \Sigma, \{\hat{l}_{\rm h}\})\\
&L = Q^\dagger_- \ \cup\ \{\#\}\\
&C = \Sigma \ \cup\ \Sigma{\times}(\Gamma - \{\#\}) \ \cup\ \Sigma{\times}\Gamma{\times}(Q^\dagger_0 \cup Q^\dagger_{\rm sf} - \{q^{\rm b}_0, \hat{q}^{\rm b}_0\}) \ \cup\ \{\#\}\\
&R = Q^\dagger_+ \ \cup\ \{\#\}\\
&r_{\rm s}(\in R) \text{ is the state such that } \exists p\,([q_0, [\rhd, [\rhd, \rhd]], p], [p, +, r_{\rm s}] \in \delta^\dagger).\\
&\hat{l}_{\rm h}(\in L) \text{ is the state such that } [\hat{l}_{\rm h}, [\rhd, [\rhd, \rhd]], \hat{q}^{\rm b}_0] \in \delta^\dagger.
\end{aligned}
$$

Note that $r_{\rm s}$ is the state of $M^\dagger$ that appears two steps after q_0, and $\hat{l}_{\rm h}$ is the state that appears just before $\hat{q}^{\rm b}_0$.

The local function $f : L{\times}C{\times}R \to L{\times}C{\times}R$ is defined as follows. Here, the notation $[a, b]$ in (b), (d), (e) and (g) represents the combination of symbols as it is, if $b \neq \#$. But, $[a, \#]$ (i.e., in the case $b = \#$) stands for the symbol $a \in \Sigma$. This is only for simplifying the description of the local function f.

1. Rules of P_M for the case a cell does not change its state.
   (a) For each $a \in (\Sigma \cup \Sigma{\times}(\Gamma - \{\#\}) \cup \{\#\})$, $f(\#, a, \#) = (\#, a, \#)$.
2. Rules of P_M for simulating shift rules of $M^\dagger$.
   (b) For each $p \in Q^\dagger_{\rm sf}$, $q \in Q^\dagger_-$, and $(a, b) \in \Sigma{\times}\Gamma$,
   if $[p, -, q] \in \delta^\dagger$, then $f(\#, [a, b, p], \#) = (q, [a, b], \#)$.
   (c) For each $p \in Q^\dagger_{\rm sf}$, $q \in Q^\dagger_0$, and $(a, b) \in \Sigma{\times}\Gamma$,
   if $[p, 0, q] \in \delta^\dagger$, then $f(\#, [a, b, p], \#) = (\#, [a, b, q], \#)$.
   (d) For each $p \in Q^\dagger_{\rm sf}$, $q \in Q^\dagger_+$, and $(a, b) \in \Sigma{\times}\Gamma$,
   if $[p, +, q] \in \delta^\dagger$, then $f(\#, [a, b, p], \#) = (\#, [a, b], q)$.
3. Rules of P_M for simulating read-write rules of $M^\dagger$.
   (e) For each $p \in Q^\dagger_-$, $q \in Q^\dagger_{\rm sf}$, and $(a, b, c) \in \Sigma{\times}\Gamma^2$,
   if $[p, [a, [b, c]], q] \in \delta^\dagger$, then $f(p, [a, b], \#) = (\#, [a, c, q], \#)$.
   (f) For each $p \in Q^\dagger_0$, $q \in Q^\dagger_{\rm sf}$, and $(a, b, c) \in \Sigma{\times}\Gamma^2$,
   if $[p, [a, [b, c]], q] \in \delta^\dagger$, then $f(\#, [a, b, p], \#) = (\#, [a, c, q], \#)$.
   (g) For each $p \in Q^\dagger_+$, $q \in Q^\dagger_{\rm sf}$, and $(a, b, c) \in \Sigma{\times}\Gamma^2$,
   if $[p, [a, [b, c]], q] \in \delta^\dagger$, then $f(\#, [a, b], p) = (\#, [a, c, q], \#)$.
4. Rules of P_M for simulating the movements of $M^\dagger$ at the left and the right endmarkers. Here, $H^\dagger = A \cup N \cup \hat{A} \cup \hat{N}$, where $\hat{A} = \{\hat{q} \mid q \in A\}$ and $\hat{N} = \{\hat{q} \mid q \in N\}$. By the rules in (h) and (i) ((j) and (k), respectively), two (four) steps of $M^\dagger$'s movements are simulated by one step of P_M.
   (h) For each $p_1 \in Q^\dagger_-$, $p_2 \in Q^\dagger_{\rm sf} - H^\dagger$, and $p_3 \in Q^\dagger_+$,
   if $[p_1, [\rhd, [\rhd, \rhd]], p_2]$, $[p_2, +, p_3] \in \delta^\dagger$, then $f(p_1, \#, \#) = (\#, \#, p_3)$.
   (i) For each $p_1 \in Q^\dagger_+$, $p_2 \in Q^\dagger_{\rm sf} - H^\dagger$, and $p_3 \in Q^\dagger_-$,
   if $[p_1, [\lhd, [\lhd, \lhd]], p_2]$, $[p_2, -, p_3] \in \delta^\dagger$, then $f(\#, \#, p_1) = (p_3, \#, \#)$.
   (j) For each $p_1 \in Q^\dagger_-$, $p_2 \in H^\dagger$, $p_3 \in Q^\dagger_0$, $p_4 \in Q^\dagger_{\rm sf} - H^\dagger$, and $p_5 \in Q^\dagger_+$,
   if $[p_1, [\rhd, [\rhd, \rhd]], p_2]$, $[p_2, 0, p_3]$, $[p_3, [\rhd, [\rhd, \rhd]], p_4]$, $[p_4, +, p_5] \in \delta^\dagger$,
   then $f(p_1, \#, \#) = (\#, \#, p_5)$.

(k) For each $p_1 \in Q_+^\dagger$, $p_2 \in H^\dagger$, $p_3 \in Q_0^\dagger$, $p_4 \in Q_{\rm sf}^\dagger - H^\dagger$, and $p_5 \in Q_-^\dagger$, if $[p_1, [\lhd, [\lhd, \lhd]], p_2]$, $[p_2, 0, p_3]$, $[p_3, [\lhd, [\lhd, \lhd]], p_4]$, $[p_4, -, p_5] \in \delta^\dagger$, then $f(\#, \#, p_1) = (p_5, \#, \#)$.

5. Rules of P_M for the cases $M^\dagger$ halts. Since the RPCA P_M cannot halt, here we set f to generate the signals q_0 and $\hat{q}_0$ by the signals $l_{\rm h}$ and $\hat{l}_{\rm h}$. By these rules, P_M finally goes back to the initial configuration, and repeats its computation indefinitely. However, note that, any P_M necessarily goes back to the initial configuration whatever the injection f is.
   (l) $f(\hat{l}_{\rm h}, \#, \#) = (\#, \#, \hat{r}_{\rm s})$, and $f(l_{\rm h}, \#, \#) = (\#, \#, r_{\rm s})$.

Though f is defined only on a subset of $L \times C \times R$ by (a)–(l), we can verify it is injective on this set, since $M^\dagger$ is reversible. From this partial function we can easily make an injective total function f by appropriately determining undefined values of f. Hence, P_M is an RPCA.

If an input $w \in \Sigma^*$ is given, P_M starts its computation from the initial configuration α_w (in Definition 3). Then, P_M simulates $M^\dagger$ step by step by the rules (b)–(g). Movements of $M^\dagger$ at the left and right border cells are simulated by (h)–(k). Hence, $\exists t_1 > 0$ $(p_L(F^{t_1}(\alpha_w)(1)) = \hat{l}_{\rm h})$ holds iff $w \in L(M^\dagger)$. Thus, $L(P_M) = L(M^\dagger) = L(M)$ is concluded. □

Let $\mathcal{L}(\mathcal{A})$ denote the class of languages accepted by the class of acceptors $\mathcal{A}$. From Lemmas 2 and 3, and the fact that PCAs can be easily simulated by DLBAs (since the condition (P1) is assumed), the following theorem is obtained.

Theorem 1. $\mathcal{L}(\text{RPCA}) = \mathcal{L}(\text{PCA}) = \mathcal{L}(\text{RDLBA}) = \mathcal{L}(\text{DLBA})$.

Example 2. Consider the DLBA $M_{\rm p}$ in Example 1. An RPCA $P_{M_{\rm p}}$ such that $L(P_{M_{\rm p}}) = L(M_{\rm p})$ is given below. $P_{M_{\rm p}}$ simulates the RDLBA $M_{\rm p}^\dagger$ constructed by the method shown in Lemma 3.

$$P_{M_{\rm p}} = ((L, C, R), f, (\#, \#, \#), q_{0,2}, \{(\,,\,)\}, \{\hat{q}_{0,1}^{\rm b}\})$$
$$L = Q_-^\dagger \cup \{\#\}$$
$$C = \{(\,,\,)\} \cup \{(\,,\,)\} \times \{\mathrm{x}\} \cup \{(\,,\,)\} \times \{\mathrm{x}, \#\} \times (Q_0^\dagger \cup Q_{\rm sf}^\dagger - \{q_0^{\rm b}, \hat{q}_0^{\rm b}\}) \cup \{\#\}$$
$$R = Q_+^\dagger \cup \{\#\}$$

In $M_{\rm p}^\dagger$, $|Q^\dagger| = 52$, $|Q_{\rm rw}^\dagger| = |Q_{\rm sf}^\dagger| = 26$, $|Q_-^\dagger| = 10$, $|Q_0^\dagger| = 4$, and $|Q_+^\dagger| = 10$. Therefore, $|L| = 11, |C| = 117$, and $|R| = 11$. Hence, the number of states of a cell, and that of rules of $P_{M_{\rm p}}$ are both 14157. However, the number of rules that are actually used to simulate $M_{\rm p}^\dagger$ is only 203. Here, we omit to describe f, but from Fig. 4 we can observe how the rules are applied.

Figure 4 shows an example of a computing process of $P_{M_{\rm p}}$ with the input $w = (\,)$. $P_{M_{\rm p}}$ accepts the input at time $t = 56$. Note that the initial state q_0, and the accepting state $\hat{q}_0^{\rm b}$ of $M_{\rm p}^\dagger$ do not appear in $P_{M_{\rm p}}$, since a few steps of $M_{\rm p}^\dagger$ at the left and right endmarkers are simulated by one step of $P_{M_{\rm p}}$. From $t = 57$ to 113, P_M performs essentially the same computing process as the one from $t = 0$ to 56, except that the states with "ˆ" and those without "ˆ" are swapped. At time $t = 114$, $P_{M_{\rm p}}$ becomes the initial configuration again, and

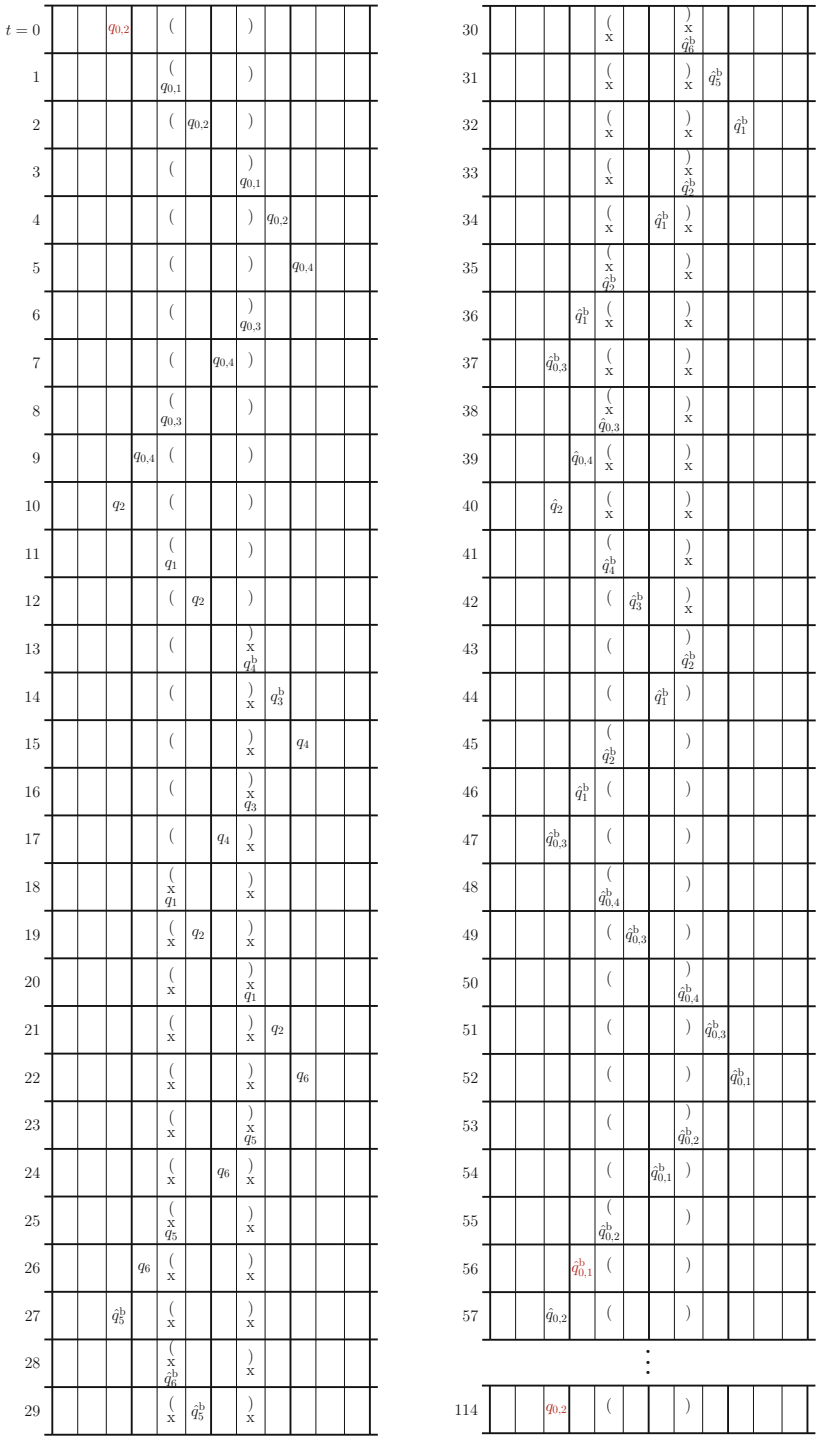

Fig. 4. A computing process of RPCA P_{M_p} with the input $w = (\,)$. It is accepted at $t = 56$, since $\hat{q}^{\mathrm{b}}_{0,1}$ is an accepting state. Here, the state # is indicated by a blank.

repeats the computing process infinitely many times. Note that, even if the rules of (l) in Lemma 3 are not included, P_{M_p} will eventually go back to the initial configuration, since it is reversible and (P1) holds (in such a case, generally, after a very large number of time steps).

5 Concluding Remarks

In this paper, we showed that the language accepting capability of PCAs is equal to that of deterministic linear-bounded automata, even if reversibility constraint is added (Theorem 1). This result is for the case computing time is not limited. It is left for the future study to characterize the capability of RPCAs for the case time is limited, e.g., in polynomial time, linear time, or real time.

Acknowledgement. This work was supported by JSPS KAKENHI Grant Number 24500017.

References

1. Kutrib, M.: Cellular automata and language theory. In: Meyers, B. (ed.) Encyclopedia of Complexity and System Science, pp. 800–823. Springer-Verlag, Berlin (2009)
2. Kutrib, M., Malcher, A.: Fast reversible language recognition using cellular automata. Inform. Comput. **206**, 1142–1151 (2008)
3. Lange, K.J., McKenzie, P., Tapp, A.: Reversible space equals deterministic space. J. Comput. Syst. Sci. **60**, 354–367 (2000)
4. Morita, K.: Simulating reversible Turing machines and cyclic tag systems by one-dimensional reversible cellular automata. Theoret. Comput. Sci. **412**, 3856–3865 (2011)
5. Morita, K.: Two-way reversible multi-head finite automata. Fundamenta Informaticae **110**(1–4), 241–254 (2011)
6. Morita, K.: A deterministic two-way multi-head finite automaton can be converted into a reversible one with the same number of heads. In: Glück, R., Yokoyama, T. (eds.) RC 2012. LNCS, vol. 7581, pp. 29–43. Springer, Heidelberg (2013)
7. Morita, K., Harao, M.: Computation universality of one-dimensional reversible (injective) cellular automata. Trans. IEICE Jpn. **E72**, 758–762 (1989)
8. Smith III, A.: Real-time language recognition by one-dimensional cellular automata. J. Comput. Syst. Sci. **6**, 233–253 (1972)