Theoretical Informatics and Applications
Theoret. Informatics Appl. **35** (2001) 239–258

NUMBER-CONSERVING REVERSIBLE CELLULAR AUTOMATA AND THEIR COMPUTATION-UNIVERSALITY

Kenichi Morita[1] and Katsunobu Imai[1]

Abstract. We introduce a new model of cellular automaton called a one-dimensional number-conserving partitioned cellular automaton (NC-PCA). An NC-PCA is a system such that a state of a cell is represented by a triple of non-negative integers, and the total (*i.e.*, sum) of integers over the configuration is conserved throughout its evolving (computing) process. It can be thought as a kind of modelization of the physical conservation law of mass (particles) or energy. We also define a reversible version of NC-PCA, and prove that a reversible NC-PCA is computation-universal. It is proved by showing that a reversible two-counter machine, which has been known to be universal, can be simulated by a reversible NC-PCA.

Mathematics Subject Classification. 68Q80, 68Q05.

1. Introduction

Recently, various kinds of interesting computing models which directly reflect laws of nature have been proposed and investigated. Among others, quantum computing, DNA computing, reversible computing, etc. have been extensively studied. A reversible computer is a system such that its transition function of the whole state is a one-to-one mapping (injection), hence, roughly speaking, it is a backward deterministic system. It is a kind of model reflecting physical reversibility, and has been known to be very important when studying inevitable power dissipation in a computing process [3, 4]. In spite of the constraint of reversibility, such a system has rich ability of computing. Bennett first showed computation-universality of a reversible Turing machine [2]. A reversible cellular automaton

Keywords and phrases: Cellular automata, reversibility, conservation law, universality.

[1] Hiroshima University, Faculty of Engineering, Higashi-Hiroshima 739-8527, Japan;
e-mail: {morita, imai}@iec.hiroshima-u.ac.jp

(CA) has also been studied extensively, and several versions of universality results have been shown [7–11, 13, 14].

Conservation of mass or energy is also an important physical law as well as reversibility. Fredkin and Toffoli [4] proposed Conservative Logic, a kind of logic circuit theory, that models both reversibility and conservation law of physics, and showed its universality. In this system, each primitive logic gate must satisfy the constraints of reversibility (*i.e.*, its logical function is an injection), and conservation of bits (*i.e.*, the total number of logical value "1"s is conserved between its input and output). Also for cellular automata, several universal models that are both reversible and bit-conserving have been known [7, 8, 10].

In this paper, we define a new model of cellular automaton (CA) called a one-dimensional number-conserving partitioned cellular automaton (NC-PCA), which generalize the notion of bit-conserving CA. In an NC-PCA, each cell is partitioned into three parts, *i.e.*, left, center, and right parts, and the state of each part is represented by a non-negative integer (thus, the state of a cell is represented by a triple of non-negative integers). The next state of a cell is determined by the present states of the right part of the left-neighboring cell, the center part of this cell, and the left part of the right-neighboring cell (not depending on the whole state of the three cells). The total number is conserved during the local transition, hence the total number over a configuration is also conserved throughout the evolving process.

Related to this model, a few other models in which each cell state is represented by a non-negative integer have been known: a totalistic CA [1] and a sand pile model [5,6]. In [1], a CA with a simple totalistic rule (but not necessarily number-conserving) has been shown to be universal. In [5,6], a kind of an automata system having a specific type of number-conserving rules are studied.

Here, we investigate the computing ability of an NC-PCA, and its reversible version. We show that an NP-PCA is computation universal even if it is reversible. This strengthens the previous result that a one-dimensional reversible CA (not necessarily a number-conserving one) is computation-universal [9]. We prove it by showing that a reversible two-counter machine, which has been known to be universal [12], can be simulated by a reversible NC-PCA.

2. Number-conserving partitioned cellular automata

In order to define a one-dimensional number-conserving partitioned cellular automaton (NC-PCA), we first give a definition of a partitioned cellular automaton (PCA) that has been introduced to design a reversible cellular automaton [9].

Definition 2.1. A *deterministic one-dimensional three-neighbor partitioned cellular automaton* (PCA) is a system defined by

$$A = (\mathbf{Z}, (L, C, R), g, (\tilde{q}_L, \tilde{q}_C, \tilde{q}_R)),$$

where $\mathbf{Z}$ is the set of all integers at which cells are placed, L, C and R are non-empty finite sets of states of left, center, and right parts of a cell, $g : R \times C \times L \to L \times C \times R$

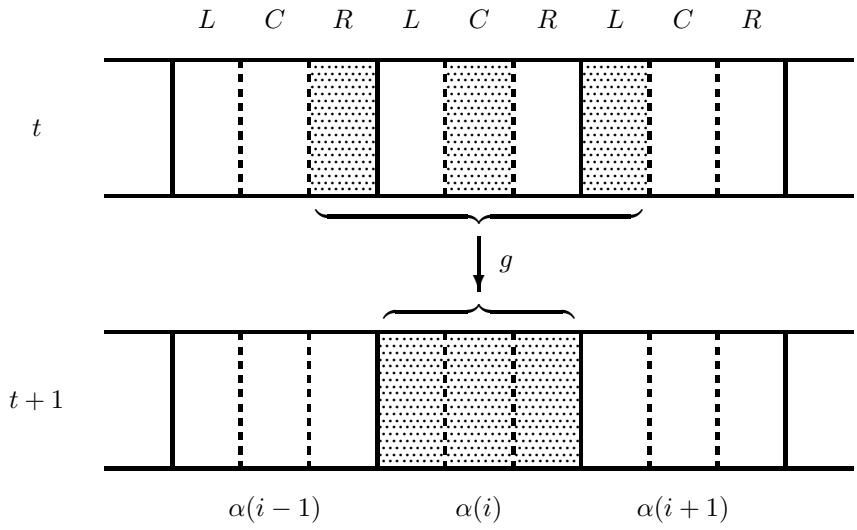

FIGURE 1. The local transition function g of a PCA A.

is a local function, and $(\tilde{q}_L, \tilde{q}_C, \tilde{q}_R) \in L \times C \times R$ is a quiescent state that satisfies $g(\tilde{q}_R, \tilde{q}_C, \tilde{q}_L) = (\tilde{q}_L, \tilde{q}_C, \tilde{q}_R)$.

A *configuration* over the set $Q = L \times C \times R$ is a mapping $\alpha : \mathbf{Z} \to Q$. Let $\mathrm{Conf}(Q)$ denote the set of all configurations over Q, *i.e.*, $\mathrm{Conf}(Q) = \{\alpha \mid \alpha : \mathbf{Z} \to Q\}$. A *quiescent configuration* is the one such that all the cells are in the quiescent states $(\tilde{q}_L, \tilde{q}_C, \tilde{q}_R)$.

Let $\mathrm{pro}_L : Q \to L$ is a projection function such that $\mathrm{pro}_L(l, c, r) = l$ for all $(l, c, r) \in Q$. Projection functions $\mathrm{pro}_C : Q \to C$, and $\mathrm{pro}_R : Q \to R$ are also defined similarly. The *global function* $G : \mathrm{Conf}(Q) \to \mathrm{Conf}(Q)$ of A is defined as follows.

$$\forall x \in \mathbf{Z} : \ G(\alpha)(x) = g(\mathrm{pro}_R(\alpha(x-1)), \mathrm{pro}_C(\alpha(x)), \mathrm{pro}_L(\alpha(x+1))).$$

Figure 1 shows how the local function g is applied to each cell. In the following, an equation $g(r, c, l) = (l', c', r')$ is called a *rule* of A, and write it by

$$[r, c, l] \to [l', c', r'].$$

We regard the local function g as the set of such rules for convenience.

Next, we define the notion of reversibility for PCAs.

Definition 2.2. Let $A = (\mathbf{Z}, (L, C, R), g, (\tilde{q}_L, \tilde{q}_C, \tilde{q}_R))$ be a PCA. We say A is *globally reversible* iff its global function G is one-to-one, and *locally reversible* iff its local function g is one-to-one.

It is easy to prove the following proposition on PCA, which has been shown in [9].

Proposition 2.3. *Let A be a PCA. A is globally reversible iff it is locally reversible.*

By Proposition 2.3, a globally or locally reversible PCA is called simply "reversible" and denoted by RPCA. By this, if we want to construct a reversible CA, it is sufficient to give a PCA whose local function g is one-to-one. This makes it easy to design a reversible CA.

When we design a one-to-one local function g, it is sufficient to define it only on a subset of $R \times C \times L$ that are needed to perform a given task. Because, we can always find a one-to-one extension from a given partial function provided that the latter function is one-to-one on the subset.

We now give a definition of a number-conserving PCA. As in the case of a reversible CA, it is also convenient to use the framework of a PCA. Because, the the notion of number-conservation can be expressed by a simple constraint on a local function of a PCA.

Definition 2.4. Let $A = (\mathbf{Z}, (\mathbf{N}_m, \mathbf{N}_m, \mathbf{N}_m), g, (0, k, 0))$ be a PCA, where $\mathbf{N}_m$ denotes the set of integers $\{0, 1, \cdots, m-1, m\}$, and $k(\leq m)$ is a non-negative integer. A is called a *one-dimensional number-conserving partitioned cellular automaton* (NC-PCA), iff it satisfies the following condition: for all (r, c, l), $(l', c', r') \in \mathbf{N}_m^3$, if $g(r, c, l) = (l', c', r')$, then $r + c + l = l' + c' + r'$.

A *reversible NC-PCA* is also defined similarly, and denoted by NC-RPCA.

Example 2.5. A simple example of an NC-RPCA:

$$A_1 = (\mathbf{Z}, \mathbf{N}_2^3, g_1, (0, 0, 0)).$$

The local function g_1 contains the following rules.

[0,0,0] → [0,0,0]	[1,1,0] → [0,0,2]	[2,1,0] → [2,1,0]
[0,1,0] → [0,1,0]	[0,1,1] → [2,0,0]	[0,1,2] → [0,1,2]
[1,0,0] → [0,0,1]	[2,0,0] → [1,1,0]	
[0,0,1] → [1,0,0]	[0,0,2] → [0,1,1]	

We can verify that each rule satisfies the constraint of number-conservation. It is also easy to see that the right-hand side of each rule differs from those of the others, hence A_1 is reversible. Figure 2 shows an example of its transitions of configurations, where each number is represented by the number of particles. We can observe that single "flying particle" goes back and forth between the "walls" made also of particles. Each time the flying particle collides a wall, the latter is shifted by one cell.

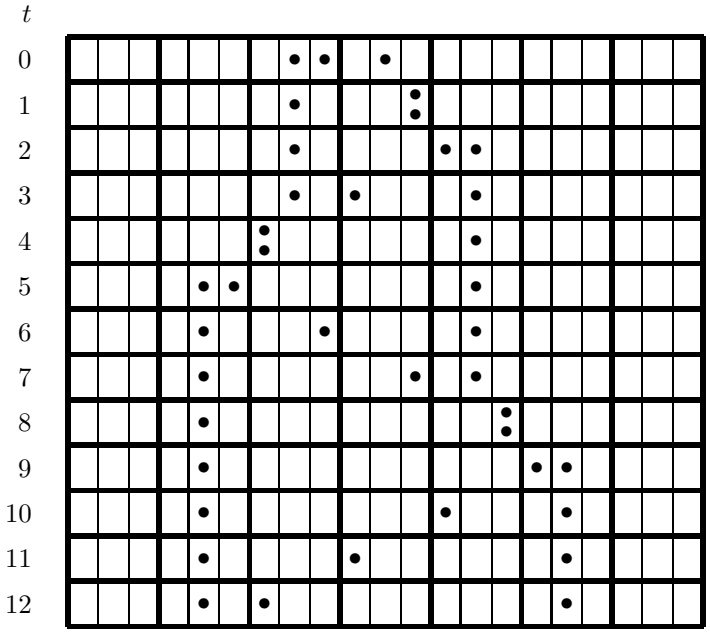

FIGURE 2. Behavior of the NC-RPCA A_1.

3. Universality of an NC-RPCA

In this section, we show that for any reversible two-counter machine there is an NC-RPCA that simulates it. Since a reversible two-counter machines has been known to be computation-universal [12], we can conclude that an NC-RPCA is also universal.

In [12] a counter machine (CM) is defined as a kind of multi-tape Turing machine whose heads are read-only ones and whose tapes are all blank except the leftmost squares as shown in Figure 3 (P is a blank symbol). This definition is convenient for giving the notion of reversibility on a CM.

Definition 3.1. A *k-counter machine* (CM(k)) is a system

$$M = (k, Q, \delta, q_0, q_f),$$

where k is the number of tapes (or counters), Q is a nonempty finite set of internal states, $q_0 \in Q$ is an initial state, and $q_f \in Q$ is a final (halting) state. M uses $\{Z, P\}$ as a tape alphabet. δ is a move relation which is a subset of $(Q \times \{0, 1, \cdots, k-1\} \times \{Z, P\} \times Q) \cup (Q \times \{0, 1, \cdots, k-1\} \times \{-, 0, +\} \times Q)$ (where "$-$", "0", and "$+$" denote left-shift, no-shift, and right-shift of a head, respectively). Tapes are one-way (rightward) infinite. The leftmost squares of the tapes contain the symbol "Z"s, and all the other squares contain "P"s (Z and P stand for "zero" and "positive").

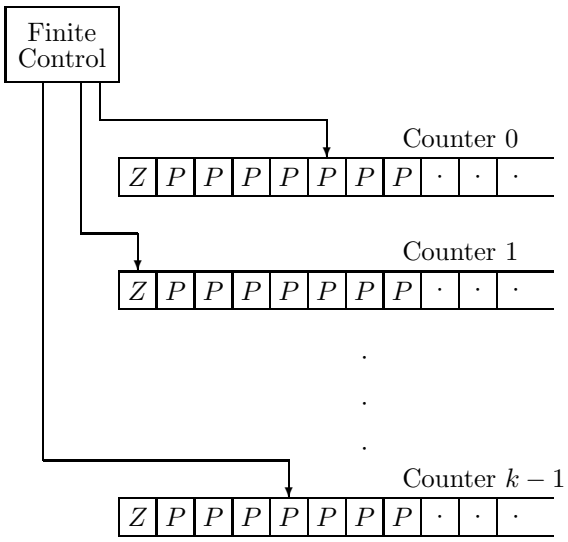

FIGURE 3. A k-counter machine (CM(k)).

Each element of δ is called a *quadruple*, and is either of the form

$$[q, i, s, q'] \quad \text{or} \quad [q, i, d, q'],$$

where $q, q' \in Q,\ i \in \{0, 1, \cdots, k-1\},\ s \in \{Z, P\},\ d \in \{-, 0, +\}$. The quadruple $[q, i, s, q']$ means that if M is in the state q and the i-th head is reading the symbol s then change the state into q'. It is used to test whether the contents of a counter are zero or positive. On the other hand, $[q, i, d, q']$ means that if M is in the state q then shift the i-th head to the direction d and change the state into q'. It is used to increment or decrement a counter by one (or make no change if $d = 0$).

Definition 3.2. An *instantaneous description* (ID) of a CM(k) $M = (k, Q, \delta, q_0, q_f)$ is a $(k+1)$-tuple

$$(q, n_0, n_1, \cdots, n_{k-1}) \in Q \times \mathbf{N}^k,$$

where $\mathbf{N} = \{0, 1, \cdots\}$. It represents that M is in the state q and the counter i keeps n_i (we assume the position of the leftmost square of a tape is 0). The transition relation $\vdash_M$ over IDs of M is defined as follows:

$$\begin{aligned}&(q, n_0, \cdots, n_{i-1}, n_i, n_{i+1}, \cdots, n_{k-1})\\ \vdash_M\ &(q', n_0, \cdots, n_{i-1}, n_i', n_{i+1}, \cdots, n_{k-1})\end{aligned}$$

holds iff one of the following conditions (1–5) is satisfied.

(1) $[q, i, Z, q'] \in \delta$ and $n_i = n'_i = 0$.
(2) $[q, i, P, q'] \in \delta$ and $n_i = n'_i > 0$.
(3) $[q, i, -, q'] \in \delta$ and $n_i - 1 = n'_i$.
(4) $[q, i, 0, q'] \in \delta$ and $n_i = n'_i$.
(5) $[q, i, +, q'] \in \delta$ and $n_i + 1 = n'_i$.

We denote reflexive and transitive closure of $\vdash_M$ by $\vdash_M^*$, and n-step transition by $\vdash_M^n$ $(n = 0, 1, \cdots)$.

Definition 3.3. Let $M = (k, Q, \delta, q_0, q_f)$ be a CM(k), and

$$\alpha_1 = [p_1, i_1, x_1, p'_1] \text{ and } \alpha_2 = [p_2, i_2, x_2, p'_2]$$

be two distinct quadruples in δ. We say α_1 and α_2 *overlap in domain* iff the following holds, where $D = \{-, 0, +\}$.

$$p_1 = p_2 \ \wedge \ [i_1 \neq i_2 \ \vee \ x_1 = x_2 \ \vee \ x_1 \in D \ \vee \ x_2 \in D].$$

We say α_1 and α_2 *overlap in range* iff the following holds.

$$p'_1 = p'_2 \ \wedge \ [i_1 \neq i_2 \ \vee \ x_1 = x_2 \ \vee \ x_1 \in D \ \vee \ x_2 \in D].$$

A quadruple α is called *deterministic* (*reversible*, respectively) iff there is no other quadruple in δ which overlaps in domain (range) with α. M is called *deterministic* (*reversible*, respectively) iff every quadruple in δ is deterministic (reversible). A reversible CM(k) is denoted by RCM(k).

For example, the following pair

$$[q_1, 2, P, q_3] \text{ and } [q_4, 2, +, q_3]$$

overlaps in range, while the pair

$$[q_1, 2, Z, q_3] \text{ and } [q_4, 2, P, q_3]$$

does not. As seen from this definition, every ID of a deterministic (reversible, respectively) CM(k) has at most one ID that immediately follows (precedes) it. Hereafter, we consider only deterministic reversible and deterministic irreversible CM(k)s.

It has been known that an RCM(2) is computation-universal [12].

Proposition 3.4. [12] *For any Turing machine T, there is a deterministic RCM(2) M that simulates T.*

We need the following lemma to prove Theorem 3.6.

Lemma 3.5. *For any deterministic CM(2) $M = (2, Q, \delta, q_0, q_f)$, there is a deterministic CM(2) $M' = (2, Q', \delta, q'_0, q'_f)$ that simulates M satisfying the following*

conditions: (i) the initial state q'_0 never appears as the fourth element of a quadruple in δ' (hence it appears only at time 0). (ii) If M is reversible then M' is also reversible.

Proof. In the case M is irreversible, it is very easy to construct such M' by adding a new initial state to Q. So, we consider the reversible case. In [12], a construction method of a reversible CM(2) M_2 that simulates a given CM(k) M_1 ($k = 1, 2, \cdots$) (that is not necessarily reversible) has been shown. By checking the construction method shown in [12], we can verify that M_2 satisfies the above condition (i), provided that M_1 also satisfies it. Hence the Lemma holds. □

Theorem 3.6. *For any deterministic CM(2) M, there is a deterministic NC-PCA A that simulates M satisfying the following condition: if M is reversible then A is also reversible.*

Proof. Without loss of generality, we assume that the state set of M is $Q = \{q_0, q_1, \cdots, q_{m-1}\}$, and the initial and final states are q_0, and q_{m-1}, respectively. Hence,

$$M = (2, \{q_0, q_1, \cdots, q_{m-1}\}, \delta, q_0, q_{m-1}).$$

Further assume that q_0 never appears as the fourth element of a quadruple in δ (by Lem. 3.5). Let Inc_j, Dec_j, Nop, and Test_j be the sets of states defined as follows ($j \in \{0, 1\}$).

$$\begin{aligned}
\mathrm{Inc}_j &= \{p \mid [p, j, +, q] \in \delta \text{ for some } q \in Q\} \\
\mathrm{Dec}_j &= \{p \mid [p, j, -, q] \in \delta \text{ for some } q \in Q\} \\
\mathrm{Nop} &= \{p \mid [p, j, 0, q] \in \delta \text{ for some } j \in \{0, 1\}, \text{ and } q \in Q\} \\
\mathrm{Test}_j &= \{p \mid [p, j, s, q] \in \delta \text{ for some } s \in \{Z, P\}, \text{ and } q \in Q\}.
\end{aligned}$$

These sets stand for instructions of "increment the counter j", "decrement the counter j", "no-operation", and "test if the counter j is zero or positive". It is easy to see that Inc_j, Dec_j, Nop, and Test_j are pairwise disjoint (for example, $\mathrm{Inc}_0 \cap \mathrm{Inc}_1 = \emptyset$, $\mathrm{Dec}_1 \cap \mathrm{Nop} = \emptyset$, etc.), since M is deterministic.

We now construct an NC-PCA A that simulates M. Each part of a cell of A keeps a number at most $m + 18$, and the quiescent state is $(0, 0, 0)$. Thus,

$$A = (\mathbf{Z}, \mathbf{N}^3_{m+18}, g, (0, 0, 0)).$$

Before defining the local function g, we fix a coding method of an ID of M by a configuration of A. First, we assign an "operation code" to each state of M by the following function $\gamma : Q \to \{0, 2, 4, 6, 7\}$.

$$\gamma(q) = \begin{cases} 2 & \text{if } q \in \mathrm{Inc}_0 \\ 4 & \text{if } q \in \mathrm{Dec}_0 \\ 6 & \text{if } q \in \mathrm{Inc}_1 \\ 7 & \text{if } q \in \mathrm{Dec}_1 \\ 0 & \text{otherwise.} \end{cases}$$

Note that the states in $\mathrm{Nop} \cup \mathrm{Test}_0 \cup \mathrm{Test}_1$ and the halting states have the same operation code 0. We then define a coding function $\varphi : Q \times \mathbf{N}^2 \to \mathrm{Conf}(\mathbf{N}^3_{m+18})$, which maps each ID of M to a configuration of A. Let $I = (q_i, n_0, n_1)$ be an ID of M. A configuration $\varphi(I)$ is computed by the following procedure.

begin
  $\alpha :=$ the quiescent configuration;
  $\mathrm{pro}_L(\alpha(0)) := i + 10$;
  $\mathrm{pro}_C(\alpha(0)) := (m + 16) - (i + 10) - \gamma(q_i)$;
  $\mathrm{pro}_R(\alpha(0)) := \gamma(q_i)$;
  for each $j \in \{0, 1\}$ **do**
    if $n_j = 0$ and $q_i \in \mathrm{Inc}_j$ **then** $\mathrm{pro}_R(\alpha(0)) := \mathrm{pro}_R(\alpha(0)) + 2^j$
    else $\mathrm{pro}_C(\alpha(n_j)) := \mathrm{pro}_C(\alpha(n_j)) + 2^j$;
  $\varphi(I) := \alpha$
end.

For example, the configuration $\varphi(q_i, 2, 0)$ such that $q_i \in \mathrm{Inc}_1$ is shown in Figure 4.

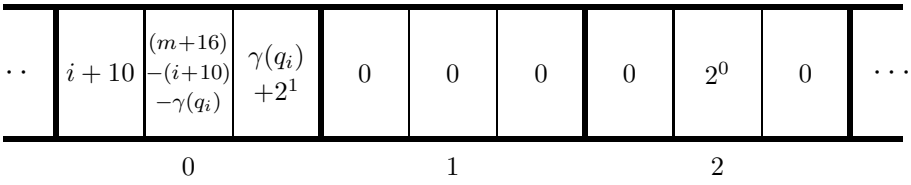

FIGURE 4. The configuration $\varphi(q_i, 2, 0)$ such that $q_i \in \mathrm{Inc}_1$.

Each configuration $\varphi(I)$ has the number $m + 19$ in total. The number n_j kept by the counter j ($j \in \{0, 1\}$) is recorded by putting the number 2^j at the center part of the cell n_j (at the right part of the cell 0, if $n_j = 0$ and the current state is in Inc_j). The number 2^j is called a *counter marker*. The remaining $m + 16$ particles are used to record the state of M, and to execute operations on counters.

In what follows, each state in $\{1, 2, \cdots, m+9\}$ appearing in the left or the right part (not in the center part) of a cell is called a *signal*. Signals in $\{10, 11, \cdots, m+9\}$ are called *state signals*, and are used to record the current state of M. State signals are kept by the cells 0 and -1 (they go back and forth between these cells). Signals in $\{2, 4, 6, 7\}$ are called *operation signals*, which are used to execute increment/decrement operations. Each of the four operation signals sometimes carry a counter marker to move it to the right- or left-neighboring cell. At that time, these signals "2", "4", "6", and "7" temporarily become "3", "5", "8", and "9", respectively. The signal "1" is a special one called an *initial/final signal* (it will be explained later).

We now define the local function g of A as follows:

1. Rules for the cases where no signal exists:
For each $x \in \mathbf{N}_{m+18}$, include the following rule in g.

$$[\ \ 0,\ \ x,\ \ 0\ \ ] \ \to\ [\ \ 0,\ \ x,\ \ 0\ \ ]. \tag{1}$$

2. Rules for state signals:
For each $x \in \{10, 11, \cdots, m+9\}$, include the following rule in g.

$$[\ \ 0,\ \ 0,\ \ x\ \ ] \ \to\ [\ \ 0,\ \ 0,\ \ x\ \ ]. \tag{2}$$

3. Rules for the increment operation:
For each $j \in \{0,1\}$ and $c \in \{0,1\}$, include the following rules in g ($\oplus$ denotes the addition in mod 2).

$$\begin{array}{lcl}
[\ \ 2+4j,\ c\cdot 2^{j\oplus 1},\ 0\ \ ] & \to & [\ \ 0,\ c\cdot 2^{j\oplus 1},\ 2+4j\ \ ] \quad (3.1)\\
[\ \ 2+4j,\ 2^j+c\cdot 2^{j\oplus 1},\ 0\ \ ] & \to & [\ \ 0,\ c\cdot 2^{j\oplus 1},\ 2+4j+2^j\ \ ] \quad (3.2)\\
[\ \ 2+4j+2^j,\ c\cdot 2^{j\oplus 1},\ 0\ \ ] & \to & [\ \ 2+4j,\ 2^j+c\cdot 2^{j\oplus 1},\ 0\ \ ] \quad (3.3)\\
[\ \ 0,\ c\cdot 2^{j\oplus 1},\ 2+4j\ \ ] & \to & [\ \ 2+4j,\ c\cdot 2^{j\oplus 1},\ 0\ \ ]. \quad (3.4)
\end{array}$$

4. Rules for the decrement operation:
For each $j \in \{0,1\}$ and $c \in \{0,1\}$, include the following rules in g.

$$\begin{array}{lcl}
[\ \ 4+3j,\ c\cdot 2^{j\oplus 1},\ 0\ \ ] & \to & [\ \ 0,\ c\cdot 2^{j\oplus 1},\ 4+3j\ \ ] \quad (4.1)\\
[\ \ 4+3j,\ 2^j+c\cdot 2^{j\oplus 1},\ 0\ \ ] & \to & [\ \ 4+3j+2^j,\ c\cdot 2^{j\oplus 1},\ 0\ \ ] \quad (4.2)\\
[\ \ 0,\ c\cdot 2^{j\oplus 1},\ 4+3j+2^j\ \ ] & \to & [\ \ 4+3j,\ 2^j+c\cdot 2^{j\oplus 1},\ 0\ \ ] \quad (4.3)\\
[\ \ 0,\ c\cdot 2^{j\oplus 1},\ 4+3j\ \ ] & \to & [\ \ 4+3j,\ c\cdot 2^{j\oplus 1},\ 0\ \ ]. \quad (4.4)
\end{array}$$

5. Rules for waiting for the completion of the increment/decrement operation:
For each $[q_i, j, d, q_k] \in \delta$ such that $d \in \{+,-\}$, and for each $c \in \{0,1\}$, include the following rule in g.

$$\begin{array}{l}
[\ \ i+10,\ \ (m+16)-(i+10)-\gamma(q_i)+c\cdot 2^{j\oplus 1},\ \ 0\ \ ] \ \to\\
\qquad [\ \ i+10,\ \ (m+16)-(i+10)-\gamma(q_i)+c\cdot 2^{j\oplus 1},\ \ 0\ \ ]. \quad (5)
\end{array}$$

6. Rules for simulating M's state transition from a state in Inc_j:
For each $[q_i, j, +, q_k] \in \delta$ and $c \in \{0,1\}$, if $q_k \notin \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$\begin{array}{l}
[\ \ i+10,\ \ (m+16)-(i+10)-\gamma(q_i)+c\cdot 2^{j\oplus 1},\ \ \gamma(q_i)\ \ ] \ \to\\
\qquad [\ \ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^{j\oplus 1},\ \ \gamma(q_k)\ \ ]. \quad (6.1)
\end{array}$$

For each $[q_i, j, +, q_k] \in \delta$ and $c \in \{0,1\}$, if $q_k \in \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$\begin{array}{l}
[\ \ i+10,\ \ (m+16)-(i+10)-\gamma(q_i)+c\cdot 2^{j\oplus 1},\ \ \gamma(q_i)\ \ ] \ \to\\
\qquad [\ \ k+10,\ \ (m+16)-(k+10)-\gamma(q_k),\ \ \gamma(q_k)+c\cdot 2^{j\oplus 1}\ \ ]. \quad (6.2)
\end{array}$$

7. Rules for simulating M's state transition from a state in Dec_j:
For each $[q_i, j, -, q_k] \in \delta$ and $c, c' \in \{0,1\}$, if $q_k \notin (\mathrm{Inc}_j \cup \mathrm{Inc}_{j\oplus 1})$, then include the following rule in g.

$$\begin{array}{l}[\ \ i+10,\ \ (m+16)-(i+10)-\gamma(q_i)+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_i)+c\cdot 2^j\ \ ]\ \ \rightarrow \\ \quad [\ \ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^j+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_k)\ \ ].\end{array} \tag{7.1}$$

For each $[q_i, j, -, q_k] \in \delta$ and $c, c' \in \{0,1\}$, if $q_k \in \mathrm{Inc}_j$, then include the following rule in g.

$$\begin{array}{l}[\ \ i+10,\ \ (m+16)-(i+10)-\gamma(q_i)+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_i)+c\cdot 2^j\ \ ]\ \ \rightarrow \\ \quad [\ \ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_k)+c\cdot 2^j\ \ ].\end{array} \tag{7.2}$$

For each $[q_i, j, -, q_k] \in \delta$ and $c, c' \in \{0,1\}$, if $q_k \in \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$\begin{array}{l}[\ \ i+10,\ \ (m+16)-(i+10)-\gamma(q_i)+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_i)+c\cdot 2^j\ \ ]\ \ \rightarrow \\ \quad [\ \ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^j,\ \ \gamma(q_k)+c'\cdot 2^{j\oplus 1}\ \ ].\end{array} \tag{7.3}$$

8. Rules for simulating M's state transition from a state in Nop:
For each $[q_i, j, 0, q_k] \in \delta$ and $c, c' \in \{0,1\}$, if $q_k \notin (\mathrm{Inc}_0 \cup \mathrm{Inc}_1)$, then include the following rule in g.

$$\begin{array}{l}[\ \ i+10,\ \ (m+16)-(i+10)+c\cdot 2^0+c'\cdot 2^1,\ \ 0\ \ ]\ \ \rightarrow \\ \quad [\ \ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^0+c'\cdot 2^1,\ \ \gamma(q_k)\ \ ].\end{array} \tag{8.1}$$

For each $[q_i, j, 0, q_k] \in \delta$ and $c, c' \in \{0,1\}$, if $q_k \in \mathrm{Inc}_j$ for some j, then include the following rule in g.

$$\begin{array}{l}[\ \ i+10,\ \ (m+16)-(i+10)+c\cdot 2^j+c'\cdot 2^{j\oplus 1},\ \ 0\ \ ]\ \ \rightarrow \\ \quad [\ \ k+10,\ \ (m+16)-(k+10)+\gamma(q_k)+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_k)+c\cdot 2^j\ \ ].\end{array} \tag{8.2}$$

9. Rules for simulating M's state transition from a state in Test_j:
For each $[q_i, j, Z, q_k] \in \delta$ and $c \in \{0,1\}$, if $q_k \notin (\mathrm{Inc}_j \cup \mathrm{Inc}_{j\oplus 1})$, then include the following rule in g.

$$\begin{array}{l}[\ \ i+10,\ \ (m+16)-(i+10)+2^j+c\cdot 2^{j\oplus 1},\ \ 0\ \ ]\ \ \rightarrow \\ \quad [\ \ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+2^j+c\cdot 2^{j\oplus 1},\ \ \gamma(q_k)\ \ ].\end{array} \tag{9.1}$$

For each $[q_i, j, Z, q_k] \in \delta$ such that and $c \in \{0,1\}$, if $q_k \in \mathrm{Inc}_j$, then include the following rule in g.

$$\begin{array}{l}[\ \ i+10,\ \ (m+16)-(i+10)+2^j+c\cdot 2^{j\oplus 1},\ \ 0\ \ ]\ \ \rightarrow \\ \quad [\ \ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^{j\oplus 1},\ \ \gamma(q_k)+2^j\ \ ].\end{array} \tag{9.2}$$

For each $[q_i, j, Z, q_k] \in \delta$ and $c \in \{0,1\}$, if $q_k \in \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$[\ i+10,\ \ (m+16)-(i+10)+2^j+c\cdot 2^{j\oplus 1},\ \ 0\ ] \ \rightarrow \\ [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+2^j,\ \ \gamma(q_k)+c\cdot 2^{j\oplus 1}\ ]. \qquad (9.3)$$

For each $[q_i, j, P, q_k] \in \delta$ and $c \in \{0,1\}$, if $q_k \notin \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$[\ i+10,\ \ (m+16)-(i+10)+c\cdot 2^{j\oplus 1},\ \ 0\ ] \ \rightarrow \\ [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k)+c\cdot 2^{j\oplus 1},\ \ \gamma(q_k)\ ]. \qquad (9.4)$$

For each $[q_i, j, P, q_k] \in \delta$ and $c \in \{0,1\}$, if $q_k \in \mathrm{Inc}_{j\oplus 1}$, then include the following rule in g.

$$[\ i+10,\ \ (m+16)-(i+10)+c\cdot 2^{j\oplus 1},\ \ 0\ ] \ \rightarrow \\ [\ k+10,\ \ (m+16)-(k+10)-\gamma(q_k),\ \ \gamma(q_k)+c\cdot 2^{j\oplus 1}\ ]. \qquad (9.5)$$

10. Rules for the initial/final signal:
Include the following rules in g.

$$[\ 1,\ 0,\ 0\ ] \rightarrow [\ 0,\ 0,\ 1\ ] \qquad (10.1)$$
$$[\ 0,\ 0,\ 1\ ] \rightarrow [\ 1,\ 0,\ 0\ ]. \qquad (10.2)$$

For each $c, c' \in \{0,1\}$, if $q_0 \notin (\mathrm{Inc}_0 \cup \mathrm{Inc}_1)$, then include the following rule in g.

$$[\ 1,\ \ (m+15)+c\cdot 2^0+c'\cdot 2^1,\ \ 0\ ] \ \rightarrow \\ [\ 10,\ \ (m+16)-10-\gamma(q_0)+c\cdot 2^0+c'\cdot 2^1,\ \ \gamma(q_0)\ ]. \qquad (10.3)$$

For each $c, c' \in \{0,1\}$, if $\exists j\,(q_0 \in \mathrm{Inc}_j)$, then include the following rule in g.

$$[\ 1,\ \ (m+15)+c\cdot 2^j+c'\cdot 2^{j\oplus 1},\ \ 0\ ] \ \rightarrow \\ [\ 10,\ \ (m+16)-10-\gamma(q_0)+c'\cdot 2^{j\oplus 1},\ \ \gamma(q_0)+c\cdot 2^j\ ]. \qquad (10.4)$$

For each $c, c' \in \{0,1\}$, include the following rule in g.

$$[\ (m+9),\ \ (m+16)-(m+9)+c\cdot 2^0+c'\cdot 2^1,\ \ 0\ ] \ \rightarrow \\ [\ 1,\ \ (m+15)+c\cdot 2^0+c'\cdot 2^1,\ \ 0\ ]. \qquad (10.5)$$

We now explain how each operation of M can be simulated by the rules of A. Although the simulation method itself is a rather straight-forward one, the above rules are designed so that the condition "if M is reversible, so is A" holds.

(a) Execution of an increment operation $[q_i, j, +, q_k]$:

The operation signals "2" and "6" are used for the increment of the counters 0 and 1, respectively. Such a signal is generated at the cell 0, and travels rightward until it meets a corresponding counter marker 2^0 or 2^1. The signal shifts the

t	0	0	2	0	0	0	0	2	0	0	1	0	0	0	0
$t+1$	0	0	0	0	0	2	0	2	0	0	1	0	0	0	0
$t+2$	0	0	0	0	0	0	0	2	2	0	1	0	0	0	0
$t+3$	0	0	0	0	0	0	0	2	0	0	0	3	0	0	0
$t+4$	0	0	0	0	0	0	0	2	0	0	0	0	2	1	0
$t+5$	0	0	0	0	0	0	0	2	0	2	0	0	0	1	0
$t+6$	0	0	0	0	0	0	2	2	0	0	0	0	0	1	0
$t+7$	0	0	0	2	0	0	0	2	0	0	0	0	0	1	0

FIGURE 5. Performing an increment operation to the counter 0.

counter marker to the right by one cell, and then goes back to the cell 0. This operation can be performed by the rules (3.1–3.4) (strictly speaking, they are "rule schemes"). Figure 5 shows an example of this process.

When the operation signal returns to the cell 0, the state transition from q_i to q_k in M is simulated by the rule (6.1) or (6.2) (depending on whether $q_k \in \mathrm{Inc}_{j\oplus 1}$ or not) in A. Figure 6 (the case $n_j > 0$) and Figure 7 (the case $n_j = 0$) show examples of the whole execution processes. The rules (1, 2), and (5) are also used during this operation.

(b) Execution of a decrement operation $[q_i, j, -, q_k]$:

The operation signals "4" and "7" are used for the decrement of the counter 0 and 1, respectively. The shifting operation of a counter marker is similar to the case of the increment operation, and is be performed by the rules (4.1–4.4). Figure 8 shows an example of this process. An examples of the whole execution process of $[q_i, j, -, q_k]$ is shown in Figure 9. The rules (1, 2, 5), and (7.1–7.3) are also used besides (4.1–4.4).

(c) Execution of a no-operation $[q_i, j, 0, q_k]$:

A no-operation simply changes the state of M. The rules (1, 2), and (8.1–8.2) are used for it. Figure 10 shows an execution process of $[q_i, j, 0, q_k]$.

(d) Execution of a test-if-zero/positive operations $[q_i, j, Z, q_k]$ and $[q_i, j, P, q_\ell]$:

The operations $[q_i, j, Z, q_k]$ and $[q_i, j, P, q_\ell]$ are performed by the rules (9.1–9.3), and (9.4, 9.5), respectively (the rules (1) and (2) are also used). Figure 11 and Figure 12 show examples of the execution processes of $[q_i, j, Z, q_k]$ and $[q_i, j, P, q_\ell]$, respectively. Note that which rule group (9.1–9.3) or (9.4, 9.5) is used is determined whether the center part of cell 0 contains the term 2^j.

(e) Rules for the initial/final signal:

The rules (10.1–10.5) are the ones for the initial/final signal. When M halts in the final state, the signal "1" is generated by the rule (10.5), and this signal travels

t	0	0	0	$i+10$	$\begin{matrix}(m+16)\\ -(i+10)\\ -\gamma(q_i)\end{matrix}$	$\gamma(q_i)$	0	2^j	0	0	0	0
$t+1$	0	0	$i+10$	0	$\begin{matrix}(m+16)\\ -(i+10)\\ -\gamma(q_i)\end{matrix}$	0	0	0	$\begin{matrix}\gamma(q_i)\\ +2^j\end{matrix}$	0	0	0
$t+2$	0	0	0	$i+10$	$\begin{matrix}(m+16)\\ -(i+10)\\ -\gamma(q_i)\end{matrix}$	0	0	0	0	$\gamma(q_i)$	2^j	0
$t+3$	0	0	$i+10$	0	$\begin{matrix}(m+16)\\ -(i+10)\\ -\gamma(q_i)\end{matrix}$	0	$\gamma(q_i)$	0	0	0	2^j	0
$t+4$	0	0	0	$k+10$	$\begin{matrix}(m+16)\\ -(k+10\\ -\gamma(q_k)\end{matrix}$	$\gamma(q_k)$	0	0	0	0	2^j	0

FIGURE 6. Execution of an increment operation $[q_i, j, +, q_k]$ for the case $n_j > 0$.

t	0	0	0	$i+10$	$\begin{matrix}(m+16)\\ -(i+10)\\ -\gamma(q_i)\end{matrix}$	$\begin{matrix}\gamma(q_i)\\ +2^j\end{matrix}$	0	0	0	0	0	0
$t+1$	0	0	$i+10$	0	$\begin{matrix}(m+16)\\ -(i+10)\\ -\gamma(q_i)\end{matrix}$	0	$\gamma(q_i)$	2^j	0	0	0	0
$t+2$	0	0	0	$k+10$	$\begin{matrix}(m+16)\\ -(k+10\\ -\gamma(q_k)\end{matrix}$	$\gamma(q_k)$	0	2^j	0	0	0	0

FIGURE 7. Execution of an increment operation $[q_i, j, +, q_k]$ for the case $n_j = 0$.

leftward indefinitely by the rule (10.2). Note that these rules are not necessary for the simulation itself. But, by them, the contents of the counters (*i.e.*, the final result) are kept unchanged even after the computation of M terminates. Symmetrically to this, by the rules (10.1, 10.3), and (10.4), we can go backward before the initial computational configuration of M.

By above, we can see that A correctly simulates M step by step. It is easy to verify that each rule conserves the total number between left- and right-hand sides, and hence A is an NC-PCA.

t	0	0	7	0	0	0	0	0	0	0	3	0	0	0	0
$t+1$	0	0	0	0	0	7	0	0	0	0	3	0	0	0	0
$t+2$	0	0	0	0	0	0	0	0	7	0	3	0	0	0	0
$t+3$	0	0	0	0	0	0	0	0	0	9	1	0	0	0	0
$t+4$	0	0	0	0	0	0	7	2	0	0	1	0	0	0	0
$t+5$	0	0	0	7	0	0	0	2	0	0	1	0	0	0	0
$t+6$	7	0	0	0	0	0	0	2	0	0	1	0	0	0	0

FIGURE 8. Performing a decrement operation to the counter 1.

t	0	0	0	$i+10$	$\begin{matrix}(m+16)\\-(i+10)\\-\gamma(q_i)\end{matrix}$	$\gamma(q_i)$	0	0	0	0	2^j	0
$t+1$	0	0	$i+10$	0	$\begin{matrix}(m+16)\\-(i+10)\\-\gamma(q_i)\end{matrix}$	0	0	0	$\gamma(q_i)$	0	2^j	0
$t+2$	0	0	0	$i+10$	$\begin{matrix}(m+16)\\-(i+10)\\-\gamma(q_i)\end{matrix}$	0	0	0	0	$\begin{matrix}\gamma(q_i)\\+2^j\end{matrix}$	0	0
$t+3$	0	0	$i+10$	0	$\begin{matrix}(m+16)\\-(i+10)\\-\gamma(q_i)\end{matrix}$	0	$\gamma(q_i)$	2^j	0	0	0	0
$t+4$	0	0	0	$k+10$	$\begin{matrix}(m+16)\\-(k+10\\-\gamma(q_k)\end{matrix}$	$\gamma(q_k)$	0	2^j	0	0	0	0

FIGURE 9. Execution of a decrement operation $[q_i, j, -, q_k]$.

Now, we show that the following statement holds: If M is reversible, so is A. Assume M is reversible. It suffices to show that each of the above rules has a different right-hand side from those of the other rules. First, we can easily verify that rules (1, 2, 3.x, 4.x, 10.1, 10.2), and (10.5) satisfy this constraint by simply comparing their right-hand sides with other rules. The rules (10.3), and (10.4) are also so. Because the state q_0 does not appear as the forth element of a quadruple in δ, hence the right-hand sides of these rules never matches those of (6.x, 7.x, 8.x), and (9.x), as well as the others.

t	0	0	0	$i+10$	$\begin{matrix}(m+16)\\ -(i+10)\end{matrix}$	0	0	0	0	0	0	0
$t+1$	0	0	$i+10$	0	$\begin{matrix}(m+16)\\ -(i+10)\end{matrix}$	0	0	0	0	0	0	0
$t+2$	0	0	0	$k+10$	$\begin{matrix}(m+16)\\ -(k+10)\\ -\gamma(q_k)\end{matrix}$	$\gamma(q_k)$	0	0	0	0	0	0

FIGURE 10. Execution of a no-operation $[q_i, j, 0, q_k]$.

t	0	0	0	$i+10$	$\begin{matrix}(m+16)\\ -(i+10)\\ +2^j\end{matrix}$	0	0	0	0	0	0	0
$t+1$	0	0	$i+10$	0	$\begin{matrix}(m+16)\\ -(i+10)\\ +2^j\end{matrix}$	0	0	0	0	0	0	0
$t+2$	0	0	0	$k+10$	$\begin{matrix}(m+16)\\ -(k+10)\\ -\gamma(q_k)\\ +2^j\end{matrix}$	$\gamma(q_k)$	0	0	0	0	0	0

FIGURE 11. Execution of a test-if-zero operation $[q_i, j, Z, q_k]$ for the case $n_j = 0$.

t	0	0	0	$i+10$	$\begin{matrix}(m+16)\\ -(i+10)\end{matrix}$	0	0	0	0	0	0	0
$t+1$	0	0	$i+10$	0	$\begin{matrix}(m+16)\\ -(i+10)\end{matrix}$	0	0	0	0	0	0	0
$t+2$	0	0	0	$\ell+10$	$\begin{matrix}(m+16)\\ -(\ell+10)\\ -\gamma(q_\ell)\end{matrix}$	$\gamma(q_\ell)$	0	0	0	0	0	0

FIGURE 12. Execution of a test-if-positive operation $[q_i, j, P, q_\ell]$ for the case $n_j > 0$.

Next, we consider the rules (5), which correspond to the quadruple $[q_i, j, d, q_k] \in \delta$ such that $d \in \{+, -\}$. The state q_i may appear as the fourth element of the other quadruples. But, since $\gamma(q_i) > 0$ (because $q_i \in (\mathrm{Inc}_0 \cup \mathrm{Inc}_1 \cup \mathrm{Dec}_0 \cup \mathrm{Dec}_1)$), the right-hand sides of the rules (5) never match those of (6.x, 7.x, 8.x), and (9.x).

We finally verify that rules (6.x, 7.x, 8.x), and (9.x) satisfy the reversibility constraint. Let Inc$'$, Dec$'$, Nop$'$, and Test$'$ be the sets of states of M defined as follows.

$$\begin{array}{lcl}
\mathrm{Inc}' & = & \{q_k \mid [q_i, j, +, q_k] \in \delta \text{ for some } j \in \{0,1\}, \text{ and } q_i \in Q\} \\
\mathrm{Dec}' & = & \{q_k \mid [q_i, j, -, q_k] \in \delta \text{ for some } j \in \{0,1\}, \text{ and } q_i \in Q\} \\
\mathrm{Nop}' & = & \{q_k \mid [q_i, j, 0, q_k] \in \delta \text{ for some } j \in \{0,1\}, \text{ and } q_i \in Q\} \\
\mathrm{Test}' & = & \{q_k \mid [q_i, j, s, q_k] \in \delta \text{ for some } j \in \{0,1\}, s \in \{Z, P\}, \text{ and } q_i \in Q\}.
\end{array}$$

For each $q_k \in (\mathrm{Inc}' \cup \mathrm{Dec}' \cup \mathrm{Nop}')$ there is exactly one quadruple containing q_k as the fourth element, since M is reversible (thus the quadruple is of the form $[q_i, j, d, q_k]$ ($q_i \in Q, j \in \{0,1\}$, $d \in \{-, 0, +\}$)). Hence, the rules in (6.x, 7.x), and (8.x) corresponding to this quadruple have different right-hand sides from the others. Next, for each $q_k \in \mathrm{Test}'$, there are at most two quadruples containing q_k as the fourth element since M is reversible. They are of the form $[q_i, j, s, q_k]$ ($q_i \in Q, j \in \{0,1\}, s \in \{Z, P\}$). If there is only one, the rules in (9.x) corresponding to this quadruple satisfy the reversibility constraint as above. In the case there are two, they must be of the forms $[q_i, j, Z, q_k]$ and $[q_i', j', P, q_k]$, because M is reversible. We can see the rules in (9.1–9.3), and (9.4, 9.5) corresponding to the two rules have mutually different right-hand sides, because the center part of the cell 0 should be different between two cases of the contents of the counter j. They also differs from the other rules.

By above, each rule in g has different right-hand side from the others, and thus we can conclude that if M is reversible, A is also reversible. □

Example 3.7. Consider a deterministic RCM(2) $M_2 = (2, Q, \{Z, P\}, \delta, q_0, q_6)$ having the following quadruples as δ.

$$\begin{array}{llll}
[q_0, & 1, & Z, & q_1] \\
[q_1, & 0, & Z, & q_6] \\
[q_1, & 0, & P, & q_2] \\
[q_2, & 0, & -, & q_3]
\end{array}
\qquad
\begin{array}{llll}
[q_3, & 1, & +, & q_4] \\
[q_4, & 1, & +, & q_5] \\
[q_5, & 1, & P, & q_1]
\end{array}$$

M_2 performs the computation $(q_0, n, 0) \overset{*}{\underset{M_2}{\vdash}} (q_6, 0, 2n)$ for any $n(= 0, 1, \cdots)$. An NC-RPCA A_2 constructed from M_2 by the method given in Theorem 3.6 is as follows.

$$A_2 = (\mathbf{Z}, \mathbf{N}_{25}^3, g_2, (0,0,0)).$$

The local function g_2 is defined by the following set of rules, and a simulation process of $(q_0, 2, 0) \overset{12}{\underset{M_2}{\vdash}} (q_6, 0, 4)$ is shown in Figure 13.

Rules by (1) ($x \in \{0, 1, \cdots, 25\}$):

$[\,0,\ x,\ 0\,] \to [\,0,\ x,\ 0\,]$

Rules by (2) ($y \in \{10, 11, \cdots, 16\}$):

$[\,0,\ 0,\ y\,] \to [\,0,\ 0,\ y\,]$

Rules by (3.1):

$[\,2,\ 0,\ 0\,] \to [\,0,\ 0,\ 2\,]$
$[\,2,\ 2,\ 0\,] \to [\,0,\ 2,\ 2\,]$
$[\,6,\ 0,\ 0\,] \to [\,0,\ 0,\ 6\,]$
$[\,6,\ 1,\ 0\,] \to [\,0,\ 1,\ 6\,]$

Rules by (3.2):

$[\,2,\ 1,\ 0\,] \to [\,0,\ 0,\ 3\,]$
$[\,2,\ 3,\ 0\,] \to [\,0,\ 2,\ 3\,]$
$[\,6,\ 2,\ 0\,] \to [\,0,\ 0,\ 8\,]$
$[\,6,\ 3,\ 0\,] \to [\,0,\ 1,\ 8\,]$

Rules by (3.3):

$[\,3,\ 0,\ 0\,] \to [\,2,\ 1,\ 0\,]$
$[\,3,\ 2,\ 0\,] \to [\,2,\ 3,\ 0\,]$
$[\,8,\ 0,\ 0\,] \to [\,6,\ 2,\ 0\,]$
$[\,8,\ 1,\ 0\,] \to [\,6,\ 3,\ 0\,]$

Rules by (3.4):

$[\,0,\ 0,\ 2\,] \to [\,2,\ 0,\ 0\,]$
$[\,0,\ 2,\ 2\,] \to [\,2,\ 2,\ 0\,]$
$[\,0,\ 0,\ 6\,] \to [\,6,\ 0,\ 0\,]$
$[\,0,\ 1,\ 6\,] \to [\,6,\ 1,\ 0\,]$

Rules by (4.1):

$[\,4,\ 0,\ 0\,] \to [\,0,\ 0,\ 4\,]$
$[\,4,\ 2,\ 0\,] \to [\,0,\ 2,\ 4\,]$
$[\,7,\ 0,\ 0\,] \to [\,0,\ 0,\ 7\,]$
$[\,7,\ 1,\ 0\,] \to [\,0,\ 1,\ 7\,]$

Rules by (4.2):

$[\,4,\ 1,\ 0\,] \to [\,5,\ 0,\ 0\,]$
$[\,4,\ 3,\ 0\,] \to [\,5,\ 2,\ 0\,]$
$[\,7,\ 2,\ 0\,] \to [\,9,\ 0,\ 0\,]$
$[\,7,\ 3,\ 0\,] \to [\,9,\ 1,\ 0\,]$

Rules by (4.3):

$[\,0,\ 0,\ 5\,] \to [\,4,\ 1,\ 0\,]$
$[\,0,\ 2,\ 5\,] \to [\,4,\ 3,\ 0\,]$
$[\,0,\ 0,\ 9\,] \to [\,7,\ 2,\ 0\,]$
$[\,0,\ 1,\ 9\,] \to [\,7,\ 3,\ 0\,]$

Rules by (4.4):

$[\,0,\ 0,\ 4\,] \to [\,4,\ 0,\ 0\,]$
$[\,0,\ 2,\ 4\,] \to [\,4,\ 2,\ 0\,]$
$[\,0,\ 0,\ 7\,] \to [\,7,\ 0,\ 0\,]$
$[\,0,\ 1,\ 7\,] \to [\,7,\ 1,\ 0\,]$

Rules by (5) for $[q_2, 0, -, q_3]$:

$[\,12,\ 7,\ 0\,] \to [\,12,\ 7,\ 0\,]$
$[\,12,\ 9,\ 0\,] \to [\,12,\ 9,\ 0\,]$

Rules by (5) for $[q_3, 1, +, q_4]$:

$[\,13,\ 4,\ 0\,] \to [\,13,\ 4,\ 0\,]$
$[\,13,\ 5,\ 0\,] \to [\,13,\ 5,\ 0\,]$

Rules by (5) for $[q_4, 1, +, q_5]$:

$[\,14,\ 3,\ 0\,] \to [\,14,\ 3,\ 0\,]$
$[\,14,\ 4,\ 0\,] \to [\,14,\ 4,\ 0\,]$

Rules by (6.1) for $[q_3, 1, +, q_4]$:

$[\,13,\ 4,\ 6\,] \to [\,14,\ 3,\ 6\,]$
$[\,13,\ 5,\ 6\,] \to [\,14,\ 4,\ 6\,]$

Rules by (6.1) for $[q_4, 1, +, q_5]$:

$[\,14,\ 3,\ 6\,] \to [\,15,\ 8,\ 0\,]$
$[\,14,\ 4,\ 6\,] \to [\,15,\ 9,\ 0\,]$

Rules by (7.3) for $[q_2, 0, -, q_3]$:

$[\,12,\ 7,\ 4\,] \to [\,13,\ 4,\ 6\,]$
$[\,12,\ 7,\ 5\,] \to [\,13,\ 5,\ 6\,]$
$[\,12,\ 9,\ 4\,] \to [\,13,\ 4,\ 8\,]$
$[\,12,\ 9,\ 5\,] \to [\,13,\ 5,\ 8\,]$

Rules by (9.1) for $[q_0, 1, Z, q_1]$:

$[\,10,\ 15,\ 0\,] \to [\,11,\ 14,\ 0\,]$
$[\,10,\ 16,\ 0\,] \to [\,11,\ 15,\ 0\,]$

Rules by (9.1) for $[q_1, 0, Z, q_6]$:

$[\,11,\ 13,\ 0\,] \to [\,16,\ 8,\ 0\,]$
$[\,11,\ 15,\ 0\,] \to [\,16,\ 10,\ 0\,]$

Rules by (9.4) for $[q_1, 0, P, q_2]$:

$[\,11,\ 12,\ 0\,] \to [\,12,\ 7,\ 4\,]$
$[\,11,\ 14,\ 0\,] \to [\,12,\ 9,\ 4\,]$

Rules by (9.4) for $[q_5, 1, P, q_1]$:

$[\,15,\ 8,\ 0\,] \to [\,11,\ 12,\ 0\,]$
$[\,15,\ 9,\ 0\,] \to [\,11,\ 13,\ 0\,]$

Rules by (10.1) and (10.2):

$[\,1,\ 0,\ 0\,] \to [\,0,\ 0,\ 1\,]$
$[\,0,\ 0,\ 1\,] \to [\,1,\ 0,\ 0\,]$

Rules by (10.3):

$[\,1,\ 22,\ 0\,] \to [\,10,\ 13,\ 0\,]$
$[\,1,\ 23,\ 0\,] \to [\,10,\ 14,\ 0\,]$
$[\,1,\ 24,\ 0\,] \to [\,10,\ 15,\ 0\,]$
$[\,1,\ 25,\ 0\,] \to [\,10,\ 16,\ 0\,]$

Rules by (10.4):

$[\,16,\ 7,\ 0\,] \to [\,1,\ 22,\ 0\,]$
$[\,16,\ 8,\ 0\,] \to [\,1,\ 23,\ 0\,]$
$[\,16,\ 9,\ 0\,] \to [\,1,\ 24,\ 0\,]$
$[\,16,\ 10,\ 0\,] \to [\,1,\ 25,\ 0\,]$

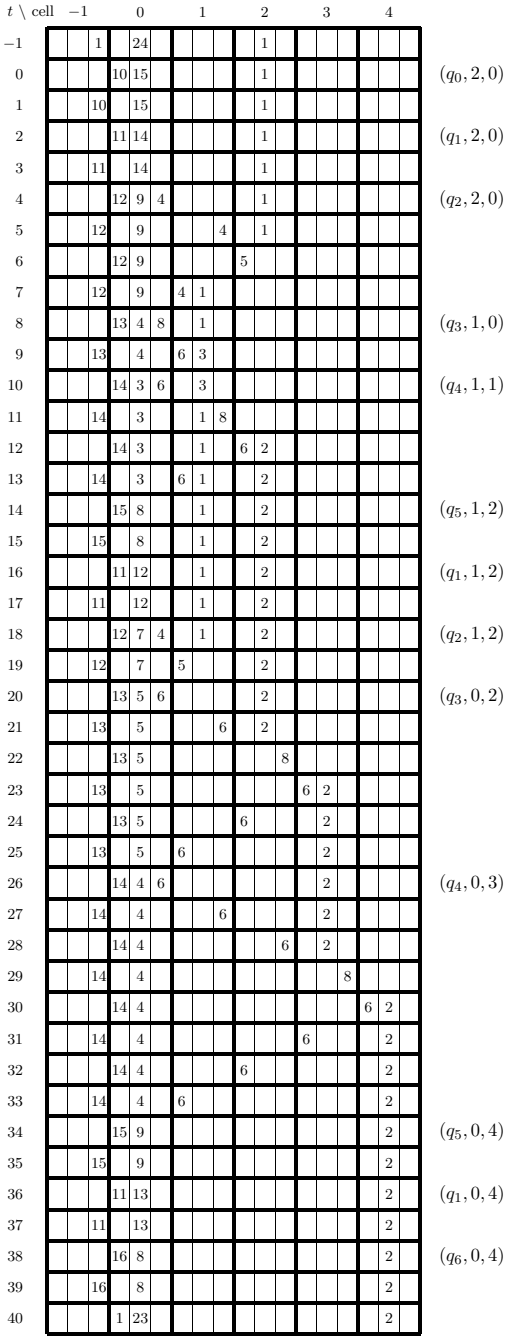

FIGURE 13. A simulation process of an RCM(2) M_2 by an NC-RPCA A_2.

From Lemma 3.6 and Proposition 3.4, universality of an NC-RPCA is concluded.

Corollary 3.8. *An NC-RPCA is computation-universal.*

4. Concluding remarks

In this paper, we proved that an NC-RPCA can simulate a reversible two-counter machine, hence it is computation-universal. In [9], universality of an RCA (not necessarily number-conserving) has been shown by simulating a one-tape reversible Turing machine by an RCA. It is also possible to simulate a one-tape reversible Turing machine by an NC-RPCA by using a similar technique as in this paper. But, if we employ a simulation method in which the contents of each tape square are stored in each cell, then the quiescent state of the NC-RPCA should be $(0, m, 0)$ for some $m > 0$ rather than $(0, 0, 0)$.

Acknowledgements. The authors are grateful to the referees for their valuable comments. This work was supported in part by Grant-in-Aid for Scientific Research (C) No. 12680353 from JSPS, Electric Technology Research Foundation of Chugoku, and Kayamori Foundation of Information Science Advancement.

References

[1] J. Albert and K. Culik II, A simple universal cellular automaton and its one-way and totalistic version. *Complex Systems* **1** (1987) 1-16.
[2] C.H. Bennett, Logical reversibility of computation. *IBM J. Res. Dev.* **17** (1973) 525-532.
[3] C.H. Bennett, Notes on the history of reversible computation. *IBM J. Res. Dev.* **32** (1988) 16-23.
[4] E. Fredkin and T. Toffoli, Conservative logic. *Int. J. Theoret. Phys.* **21** (1982) 219-253.
[5] E. Goles, Sand pile automata. *Ann. Inst. H. Poincaré* **56** (1992) 75-90.
[6] E. Goles and M. Margenstern, Sand pile as a universal computer. *Int. J. Modern Physics C* **7** (1996) 113-122.
[7] K. Imai and K. Morita, A computation-universal two-dimensional 8-state triangular reversible cellular automaton. *Theoret. Comput. Sci.* (in press).
[8] N. Margolus, Physics-like model of computation. *Physica D* **10** (1984) 81-95.
[9] K. Morita and M. Harao, Computation universality of one-dimensional reversible (injective) cellular automata. *Trans. IEICE Japan* **E-72** (1989) 758-762.
[10] K. Morita and S. Ueno, Computation-universal models of two-dimensional 16-state reversible cellular automata. *IEICE Trans. Inf. & Syst.* **E75-D** (1992) 141-147.
[11] K. Morita, Computation-universality of one-dimensional one-way reversible cellular automata. *Inform. Process. Lett.* **42** (1992) 325-329.
[12] K. Morita, Universality of a reversible two-counter machine. *Theoret. Comput. Sci.* **168** (1996) 303-320.
[13] T. Toffoli, Computation and construction universality of reversible cellular automata. *J. Comput. Syst. Sci.* **15** (1977) 213-231.
[14] T. Toffoli and N. Margolus, Invertible cellular automata: A review. *Physica D* **45** (1990) 229-253.

Communicated by M. Ito and Z. Ésik.
Received August 27, 1999. Accepted August 21, 2001.