Chapter 10
Two Small Universal Reversible Turing Machines

Kenichi Morita

Abstract We study the problem of constructing small universal Turing machines (UTMs) under the constraint of *reversibility*, which is a property closely related to physical reversibility. Let URTM(m,n) denote an m-state n-symbol universal reversible Turing machine (URTM). Then, the problem is to find URTM(m,n) with small m and n. So far, several kinds of small URTMs have been given. Here, we newly construct two small URTMs. They are URTM(13,7) and URTM(10,8) that can simulate cyclic tag systems, a kind of universal string rewriting systems proposed by Cook. We show how these URTMs can be designed, and compare them with other existing URTMs.

10.1 Introduction

A universal Turing machine (UTM) is a TM that can simulate any TM. Turing himself showed it is possible to construct such a machine [21]. Since then, there have been many researches on UTMs, in particular, on finding small UTMs. If we write an m-state n-symbol UTM by UTM(m,n), then the problem is to find UTM(m,n) with small values of m and n. This problem attracted many researchers, and various small UTMs have been presented till now (see e.g., a survey paper [23]). In the early stage of this study, a direct simulation method of TMs was employed. Later, an indirect method of simulating universal systems that are much simpler than TMs was proposed to construct very small UTMs. Minsky [7] presented a method of simulating 2-tag systems, which are universal string rewriting systems, and gave a UTM that has seven states and four symbols. After that, such an indirect simulation method has mainly been used to give small UTMs. Rogozhin [20] designed small UTMs for many pairs of m and n. They are UTM(24,2), UTM(10,3), UTM(7,4), UTM(5,5), UTM(4,6), UTM(3,10), and UTM(2,18) that also simulate 2-tag systems. Some of these results were improved later. Kudlek and Rogozhin [6] gave UTM(3,9) that simulates 2-tag

K. Morita (✉)
Hiroshima University, Higashi-hiroshima 739-8527, Japan
e-mail: km@hiroshima-u.ac.jp

© Springer International Publishing Switzerland 2017

A. Adamatzky (ed.), *Advances in Unconventional Computing*,
Emergence, Complexity and Computation 22,
DOI 10.1007/978-3-319-33924-5_10

systems, and Neary and Woods [18] constructed UTM(15,2), UTM(9,3), UTM(6,4), and UTM(5,5) that simulate bi-tag systems.

Here, we study the problem of constructing small universal reversible Turing machines (URTMs). *Reversible computing* is a paradigm of computing that reflects physical reversibility, one of the fundamental microscopic properties of physical systems. It is thus related to quantum computing, since the evolution of a quantum system is reversible. A reversible Turing machine (RTM) is a standard model in the theory of reversible computing. In fact, it was shown by Bennett [2] that for any (irreversible) TM, there is an RTM that simulates the former.

Roughly speaking, an RTM is a "backward deterministic" TM, where each computational configuration has at most one predecessor (a precise definition will be given in the next section). Although its definition is simple, it has a close relation to reversible physical systems. It has been shown that any RTM can be implemented as a circuit composed of reversible logic element with 1-bit memory very simply [8, 11, 14]. It is also known that a reversible logic element with 1-bit memory can be realized in the billiard ball model (BBM) [9, 17]. BBM is an idealized mechanical model of a reversible physical system proposed by Fredkin and Toffoli [5], where computing is carried out by collisions of balls and reflectors. Hence, the whole system of an RTM can be embedded in such a reversible physical model.

So far, there have been several researches on small URTMs. Let URTM(m,n) denote an m-state n-symbol URTM. Morita and Yamaguchi [15] first constructed URTM(17,5) that simulates cyclic tag systems, which are another kind of universal string rewriting systems proposed by Cook [4]. Axelsen and Glück [1] studied a different type of URTM that computes all computable injective functions, but their objective was not finding a small URTM. Later, Morita [9, 13] constructed URTM(15,6), URTM(24,4), and URTM(32,3), which also simulate cyclic tag systems. On the other hand, it is in general difficult to design simple URTMs with only two symbols, or with a very small number of states. As for a 2-symbol URTM, we can use a general procedure for converting a many-symbol RTM into a 2-symbol RTM [16]. By this, we obtain URTM(138,2) [13]. In [10], methods for converting an m-state n-symbol RTM into 4-state $(2mn + n)$-symbol RTM, and 3-state $O(m^2n^3)$-state RTM were given. Applying these methods to URTM(17,5) and URTM(32,3), we obtain URTM(4,175) and URTM(3,36654), respectively. Note that, in Sect. 10.3, we newly give URTM(10,8). Applying the conversion method to it, we have URTM(4,168) that is slightly simpler than URTM(4,175).

In this chapter, we give two new URTMs, and explain how we can design small URTMs. They are URTM(13,7), and URTM(10,8), which again simulate cyclic tag systems. In Sect. 10.2, we give basic definitions on RTMs, 2-tag systems (2-TSs), and cyclic tag systems with halting condition (CTSHs). We also show how a CTSH can simulate a 2-TS. In Sect. 10.3, we construct URTM(13,7) and URTM(10,8). In Sect. 10.4, we compare these two URTMs with other small URTMs, and summarize the results.

10.2 Reversible Turing Machines and Tag Systems

In this section, we give definitions and basic properties on reversible Turing machines, m-tag systems, and cyclic tag systems.

10.2.1 Reversible Turing Machines

There are two kinds of formulations for reversible Turing machines. They are the quadruple formulation [2], and the quintuple formulation [9]. They can be easily converted each other keeping reversibility. Here, we use the quintuple formulation, because the number of states of a Turing machine of this form can be about a half of that in the quadruple form. Also, most classical universal Turing machines are given in the quintuple form.

Definition 1 A *one-tape Turing machine* (TM) in the quintuple form is defined by

$$T = (Q, S, q_0, s_0, \delta),$$

where Q is a non-empty finite set of states, S is a non-empty finite set of symbols, q_0 is an initial state ($q_0 \in Q$), s_0 is a special blank symbol ($s_0 \in S$). δ is a move relation, which is a subset of $(Q \times S \times S \times \{-, +\} \times Q)$. The symbols "$-$", and "$+$" are shift directions of the head, which stand for "left-shift", and "right-shift", respectively. Each element of δ is a quintuple of the form $[p, s, s', d, q]$. It means if T reads the symbol s in the state p, then writes s', shifts the head to the direction d, and goes to the state q.

In the above definition, final states that halt for any symbol are not specified according to the designing convention of universal Turing machines (in other words, final states are not counted in the number of states of a universal Turing machine). Also note that δ is defined as a "relation" rather than a "function", This is because determinism and reversibility will be defined almost symmetrically by this definition (but, they are slightly asymmetric since the head-shift operation is performed after the read/write operation).

Let $w \in S^*$, $q \in Q$, and $h \in \{0, 1, \ldots, |w| - 1\}$. A triplet $[w, q, h]$ is called a *computational configuration* (or simply a *configuration*) of $T = (Q, S, q_0, s_0, \delta)$. The configuration $[w, q, h]$ means that the tape contains w (all the other squares of the tape have the blank symbol s_0), the state is q, and the head position is at the hth symbol of w (the position of the leftmost symbol of w is the 0-th). In the following, we use such an expression to write a configuration of T.

Determinism and reversibility of TM is defined as follows. T is called a *deterministic TM* iff the following holds for any pair of distinct quintuples $[p_1, s_1, s_1', d_1, q_1]$ and $[p_2, s_2, s_2', d_2, q_2]$ in δ.

$$\text{If } p_1 = p_2, \text{ then } s_1 \neq s_2$$

T is called a *reversible TM* iff the following holds for any pair of distinct quintuples $[p_1, s_1, s_1', d_1, q_1]$ and $[p_2, s_2, s_2', d_2, q_2]$ in δ.

$$\text{If } q_1 = q_2, \text{ then } s_1' \neq s_2' \ \wedge \ d_1 = d_2$$

The above is called the *reversibility condition*. It is easy to see that if T is reversible, then there is at most one reversely applicable quintuple to each configuration, and thus every configuration of T has at most one predecessor.

In the following, we consider only deterministic (irreversible or reversible) TMs, and thus the word "deterministic" is omitted. Hence, by a "reversible TM" (RTM), we mean a deterministic reversible TM.

10.2.2 m-Tag Systems

A tag system is a string rewriting system originally proposed by Post [19], and an m-tag system ($m = 1, 2, \ldots$) is a variant of it. In the m-tag system, rewriting of strings is performed in the following way. Let $\alpha = a_1 \ldots a_n$ be a string over an alphabet A. If the system has a production rule $a_1 \to b_1 \ldots b_k$ and $n \geq m$, then we can obtain a new string $a_{m+1} \ldots a_n b_1 \ldots b_k$. Namely, if the first symbol of α is a_1 and $|\alpha| \geq m$, then remove the leftmost m symbols, and append the string $b_1 \ldots b_k$ at the right end of it as shown in Fig. 10.1. Repeating this procedure, we can obtain new strings successively. If we reach a string β to which there is no applicable production rule, or $|\beta| < m$, then the rewriting process terminates.

We now define m-tag systems based on the definition by Rogozhin [20].

Definition 2 An *m-tag system* (m-TS) is defined by $T = (m, A, P)$, where m is a positive integer, A is a finite alphabet, and $P : A \to A^* \cup \{\mathrm{h}alt\}$ is a mapping that gives a set of production rules (we assume $\mathrm{h}alt \notin A$). Let $a \in A$. If $P(a) = b_1 \ldots b_k \in A^*$, we write it by $a \to b_1 \ldots b_k$, and call it a *production rule* of T. If $P(a) = \mathrm{h}alt$, then a is called a *halting symbol*. We usually write P as the set of production rules: $\{a \to P(a) \mid a \in A \ \wedge \ P(a) \neq \mathrm{h}alt\}$.

The *transition relation* $\underset{T}{\Rightarrow}$ on A^* is defined as follows. For any $a_1, \ldots, a_m$, $a_{m+1}, \ldots, a_n, b_1, \ldots, b_k \in A$ such that $n \geq m$,

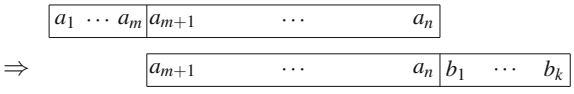

Fig. 10.1 Rewriting in an m-TS. If there is a production rule $a_1 \to b_1 \ldots b_k$ and $n \geq m$, then the first m symbols are removed, and the string $b_1 \ldots b_k$ is appended at the *right end*. If a_1 is a halting symbol or $n < m$, then the rewriting process terminates

$$a_1 \dots a_m a_{m+1} \dots a_n \underset{T}{\Rightarrow} a_{m+1} \dots a_n b_1 \dots b_k \quad \text{iff} \quad a_1 \to b_1 \dots b_k \in P.$$

When there is no ambiguity, we use $\Rightarrow$ instead of $\underset{T}{\Rightarrow}$. Let $\alpha \in A^*$. By the above definition of $\Rightarrow$, if the first symbol of α is a halting symbol, or $|\alpha| < m$, then there is no $\alpha' \in A^*$ that satisfy $\alpha \Rightarrow \alpha'$. Such α is called a *halting string* or a *final string*. The reflexive and transitive closure of $\Rightarrow$ is denoted by $\overset{*}{\Rightarrow}$. Let $\alpha_i \in A^*$ ($i \in \{0, 1, \dots, n\}$, $n \in \mathbb{N}$). We say $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_n$ is a *complete computing process* of T starting from α_0 if α_n is a halting string.

In [3, 7], it is shown that for any TM there is a 2-TS that simulates the TM. Hence, the class of 2-TS is computationally universal.

Theorem 1 ([3, 7]) *For any one-tape two-symbol TM, there is a 2-TS that simulates the TM.*

10.2.3 Cyclic Tag Systems

A cyclic tag system (CTS) is a variant of a tag system proposed by Cook [4]. He used CTS to prove computational universality of the elementary cellular automaton of rule 110. Since CTS has two kinds of symbols, we fix its alphabet as $\{Y, N\}$. In CTS, there are k ($= 1, 2, \dots$) production rules $Y \to w_0, Y \to w_1, \dots, Y \to w_{k-1}$, which are used one by one cyclically in this order. More precisely, the pth production rule $Y \to w_p$ is applicable at time t, if $p = t \bmod k$. If the first symbol of the string at time t is Y, then it is removed, and w_p is appended at the end of the string. On the other hand, if the first symbol is N, then it is removed, and nothing is appended. Hence, we assume that the production rule $N \to \lambda$ is always applicable. Figure 10.2 shows this process. In the following, we write the set of production rules as $(w_0, \dots, w_{k-1})$, since the left-hand side of each production rule is always Y. CTSs are simpler than m-TSs because of the following reasons: they have only two kinds of symbols Y and N, production rules are used one by one in the specified order (hence there is no need of table lookup), and a string is appended only when the first symbol is Y.

In the original definition of a CTS in [4], the notion of halting was not defined explicitly. In fact, it halts only if the string becomes the empty string λ. Hence, the final configuration of simulated TM cannot be retrieved from the halting string (i.e., λ) of a CTS. Therefore, when we use CTS as an intermediate system for making a UTM, then some halting mechanism should be incorporated. Though there will be several ways of defining the notion of halting, we use the method employed in [15], which is given in the following definition.

Definition 3 A *cyclic tag system with halting condition* (CTSH) is a system defined by

$$C = (k, (halt, w_1, \dots, w_{k-1})),$$

Fig. 10.2 Rewriting in a cyclic tag system at time t. Here, we assume its cycle length is k, and the pth production rule is $y \to w_p$, where $p = t \bmod k$. If the first symbol of the string at time t is Y, then it is removed, and w_p is appended at the *right end*. If the first symbol is N, then it is removed, and nothing is appended

where $k \in \mathbb{Z}_+$ is the length of a cycle, and $(w_1, \ldots, w_{k-1}) \in (\{Y, N\}^*)^{k-1}$ is a $(k-1)$-tuple of *production rules*. A pair (v, m) is an *instantaneous description* (ID) of C, where $v \in \{Y, N\}^*$ and $m \in \{0, \ldots, k-1\}$. m is called the *phase* of the ID. The transition relation $\underset{C}{\Rightarrow}$ is defined below. For any $v \in \{Y, N\}^*$, $m, m' \in \{0, \ldots, k-1\}$,

$$
\begin{aligned}
(Yv, m) &\underset{C}{\Rightarrow} (vw_m, m') \text{ iff } (m \neq 0) \ \wedge \ (m' = m + 1 \bmod k), \\
(Nv, m) &\underset{C}{\Rightarrow} (v, m') \quad \text{iff } m' = m + 1 \bmod k.
\end{aligned}
$$

If there is no ambiguity, we use $\Rightarrow$ instead of $\underset{C}{\Rightarrow}$. By the definition of $\Rightarrow$, we can see that, for any $v \in \{Y, N\}^*$ and $m \in \{0, \ldots, k-1\}$, IDs $(Yv, 0)$ and (λ, m) have no successor ID. Hence, an ID of the form $(Yv, 0)$ or (λ, m) is called a *halting ID*. Let $v_i \in \{Y, N\}^*, m_i \in \{0, \ldots, k-1\}$ $(i \in \{0, 1, \ldots, n\}, n \in \mathbb{N})$. We say $(v_0, m_0) \Rightarrow (v_1, m_1) \Rightarrow \cdots \Rightarrow (v_n, m_n)$ is a *complete computing process* of C starting from an *initial string* v if $(v_0, m_0) = (v, 0)$ and (v_n, m_n) is a halting ID. Here, v_n is called a *final string*. The reflexive and transitive closure of $\Rightarrow$ is denoted by $\overset{*}{\Rightarrow}$. An n-step transition is denoted by $\overset{n}{\Rightarrow}$.

We give a simple example of a CTSH $\hat{C}$ in Example 1. In Sect. 10.3 it will be used to explain how constructed URTM simulate CTSHs.

Example 1 Consider a CTSH $\hat{C} = (3, (halt, NY, NNY))$. A complete computing process of $\hat{C}$ starting from the initial string NYY is as follows.

$$
\begin{aligned}
&(NYY, 0) \quad \Rightarrow (YY, 1) \quad \Rightarrow (Y\,NY, 2) \\
\Rightarrow\ &(NY\,NNY, 0) \Rightarrow (Y\,NNY, 1) \Rightarrow (NNY\,NY, 2) \\
\Rightarrow\ &(NY\,NY, 0) \quad \Rightarrow (Y\,NY, 1) \quad \Rightarrow (NY\,NY, 2) \\
\Rightarrow\ &(Y\,NY, 0)
\end{aligned}
$$

The last ID $(YNY, 0)$ is a halting ID, and YNY is the final string. □

We now show that any 2-TS can be simulated by a CTSH. Thus, from Theorem 1, the class of CTSHs is computationally universal. The proof method is due to Cook [4] except that halting of CTSH is properly managed here.

Theorem 2 *For any 2-TS T, we can construct a CTSH C that simulates T.*

Proof Let $T = (2, A, P)$. We define A_{N} and A_{H} as follows: $A_{\mathrm{N}} = \{a \mid P(a) \neq \mathrm{h}alt\}$ and $A_{\mathrm{H}} = \{a \mid P(a) = \mathrm{h}alt\}$. They are the sets of non-halting symbols, and halting symbols, respectively. Thus, $A = A_{\mathrm{N}} \cup A_{\mathrm{H}}$. We denote $A_{\mathrm{N}} = \{a_1, \ldots, a_n\}$ and $A_{\mathrm{H}} = \{b_0, \ldots, b_{h-1}\}$. Let $k = \max\{n, \lceil \log_2 h \rceil\}$. Let $\mathrm{b}in_k : \{0, \ldots, 2^k - 1\} \to \{N, Y\}^k$ be the function that maps an integer j $(0 \leq j \leq 2^k - 1)$ to the k-bit binary number represented by N and Y, where N and Y stand for 0 and 1, respectively. For example, $\mathrm{b}in_4(12) = YYNN$. Now, we define a coding function $\varphi : A^* \to \{N, Y\}^*$. It is a string homomorphism that satisfies the following.

$$\begin{aligned} \varphi(a_i) &= N^i Y N^{k-i} \ (1 \leq i \leq n) \\ \varphi(b_i) &= Y\mathrm{b}in_k(i) \ (0 \leq i \leq h-1) \end{aligned}$$

Namely, each symbol in A is coded into a string of length $k + 1$ over $\{N, Y\}$. Now, the CTSH C that simulates T is given as follows.

$$\begin{aligned} C &= (2k+2, (\mathrm{h}alt, w_1, \ldots, w_{2k+1})) \\ w_i &= \begin{cases} \varphi(P(a_i)) & (1 \leq i \leq n) \\ \lambda & (n+1 \leq i \leq 2k+1) \end{cases} \end{aligned}$$

Let $s_1 \cdots s_m$ be a string over A, where $s_j \in A$ $(j \in \{1, \ldots, m\})$. In C, it is represented by $(\varphi(s_1 \cdots s_m), 0)$. First, consider the case $s_1 = a_i$ for some $a_i \in A_{\mathrm{N}}$. Thus, in T, $s_1 \cdots s_m \underset{T}{\Rightarrow} s_3 \cdots s_m P(a_i)$ holds. Since $\varphi(s_1 \cdots s_m) = N^i Y N^{k-i} \varphi(s_2)\varphi(s_3 \cdots s_m)$, this transition is simulated by C in $2k + 2$ steps as below.

$$\begin{aligned} & (N^i Y N^{k-i} \varphi(s_2)\varphi(s_3 \cdots s_m), 0) \\ \overset{i}{\underset{C}{\Rightarrow}} & (Y N^{k-i} \varphi(s_2)\varphi(s_3 \cdots s_m), i) \\ \underset{C}{\Rightarrow} & (N^{k-i} \varphi(s_2)\varphi(s_3 \cdots s_m)\varphi(P(a_i)), i+1) \\ \overset{k-i}{\underset{C}{\Rightarrow}} & (\varphi(s_2)\varphi(s_3 \cdots s_m)\varphi(P(a_i)), k+1) \\ \overset{k+1}{\underset{C}{\Rightarrow}} & (\varphi(s_3 \cdots s_m P(a_i)), 0) \end{aligned}$$

Second, consider the case $s_1 = b_i$ for some $b_i \in A_{\mathrm{H}}$. In this case, $s_1 \cdots s_m$ is a halting string in T. Since $\varphi(s_1 \cdots s_m) = Y\mathrm{b}in_k(i)\varphi(s_2 \cdots s_m)$, the ID $(Y\mathrm{b}in_k(i)$ $\varphi(s_2 \cdots s_m), 0)$ is also a halting ID in C.

By above, if

$$\alpha_0 \underset{T}{\Rightarrow} \alpha_1 \underset{T}{\Rightarrow} \cdots \underset{T}{\Rightarrow} \alpha_{l-1} \underset{T}{\Rightarrow} \alpha_l$$

is a complete computing process of T, then it is simulated by

$$(\varphi(\alpha_0), 0) \overset{2k+2}{\underset{C}{\Rightarrow}} (\varphi(\alpha_1), 0) \overset{2k+2}{\underset{C}{\Rightarrow}} \cdots \overset{2k+2}{\underset{C}{\Rightarrow}} (\varphi(\alpha_{l-1}), 0) \overset{2k+2}{\underset{C}{\Rightarrow}} (\varphi(\alpha_l), 0),$$

which is a complete computing process of C. □

10.3 Constructing Small Universal Reversible Turing Machines

In this section, we give URTM(13,7) and URTM(10,8). If codes (descriptions) of a CTSH C and an initial string $\alpha_0 \in \{Y, N\}^*$ are given, each of these URTMs simulates the rewriting process of C from the initial ID $(\alpha_0, 0)$ step by step until C halts. The URTM U has a one-way infinite tape, and keeps the codes of C and an ID of C as shown in Fig. 10.3. The production rules of C are stored in the left-side segment of the tape. Initially, the segment of "removed symbols" on the tape is empty, and the initial string α_0 is kept in the segment of "current string". To indicate the border between the removed symbols and the current string, different kinds of symbols are used for the removed ones, and for the leftmost one of the current string (or, temporarily pointed by the head). Each time the leftmost symbol of the current string is removed by a rewriting in C, this border is shifted to the right by one square. Thus, if the ID of C is (α, m), then α is stored in the segment of the current string. The phase m of the ID is recorded by putting a "phase marker", which is also a specified symbol of U, at the mth production rule of C on the tape. If the first symbol of the current string is Y and $m > 0$, then the right-hand side w_m of the mth production rule $Y \to w_m$ is appended at the right end of the current string. If the first symbol is N, then nothing is appended. In both cases, the phase marker is moved to the position of the next production rule. If C enters a halting ID, then U halts.

10.3.1 13-State 7-Symbol URTM

We first give URTM(13,7) U_{13_7}. It is defined as follows.

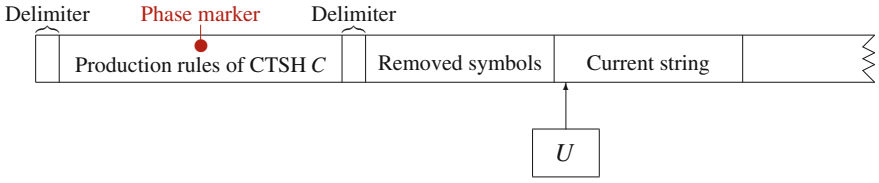

Fig. 10.3 A configuration of a URTM U that simulates a CTSH C

$$U_{13\_7} = (Q_{13\_7}, \{b, y, n, Y, N, *, \$\}, q_{begin}, b, \delta_{13\_7})$$
$$Q_{13\_7} = \{q_{begin}, q_{case\_y\_1}, q_{case\_y\_2}, q_{case\_y\_n}, q_{copy\_start}, q_{copy\_y\_1}, q_{copy\_y\_2}, q_{copy\_y\_3}, q_{copy\_n\_1}, q_{copy\_n\_2}, q_{copy\_n\_3}, q_{copy\_end}, q_{cycle\_end}\}$$

The move relation δ_{13_7} is described in Table 10.1. It contains 57 quintuples. In this table, "halt" means that the simulated CTSH halts with an ID $(Yv, 0)$ for some $v \in \{Y, N\}^*$, while "null" means that it halts with an ID (λ, m) for some $m \in \mathbb{N}$. We can verify that U_{13_7} satisfies the reversibility condition by a careful inspection of δ_{13_7}. It was also verified by a computer program. Note that, if reversibility is not required, then, for example, the states $q_{case\_y\_2}$ and $q_{copy\_y\_2}$ could be merged to reduce the number of states. However, since there are quintuples $[q_{case\_y\_2}, y, y, +, q_{case\_y\_2}]$, and

Table 10.1 The move relation δ_{13_7} of U_{13_7}

	b	y	n	
q_{begin}	(*null*)	$Y, -, q_{case\_y\_1}$	$N, -, q_{case\_y\_n}$	
$q_{case\_y\_1}$	(*halt*)	$y, -, q_{case\_y\_1}$	$n, -, q_{case\_y\_1}$	
$q_{case\_y\_2}$		$y, +, q_{case\_y\_2}$	$n, +, q_{case\_y\_2}$	
$q_{case\_y\_n}$	$*, -, q_{copy\_start}$	$y, -, q_{case\_y\_n}$	$n, -, q_{case\_y\_n}$	
q_{copy_start}	$b, +, q_{cycle\_end}$	$b, +, q_{copy\_y\_1}$	$b, +, q_{copy\_n\_1}$	
$q_{copy\_y\_1}$	$y, +, q_{copy\_y\_2}$	$y, +, q_{copy\_y\_1}$	$n, +, q_{copy\_y\_1}$	
$q_{copy\_y\_2}$	$b, -, q_{copy\_y\_3}$			
$q_{copy\_y\_3}$	$y, -, q_{copy\_start}$	$y, -, q_{copy\_y\_3}$	$n, -, q_{copy\_y\_3}$	
$q_{copy\_n\_1}$	$n, +, q_{copy\_n\_2}$	$y, +, q_{copy\_n\_1}$	$n, +, q_{copy\_n\_1}$	
$q_{copy\_n\_2}$	$b, -, q_{copy\_n\_3}$			
$q_{copy\_n\_3}$	$n, -, q_{copy\_start}$	$y, -, q_{copy\_n\_3}$	$n, -, q_{copy\_n\_3}$	
q_{copy_end}		$y, +, q_{copy\_end}$	$n, +, q_{copy\_end}$	
q_{cycle_end}		$y, +, q_{cycle\_end}$	$n, +, q_{cycle\_end}$	
	Y	N	$*$	$\$$
q_{begin}				
$q_{case\_y\_1}$			$*, +, q_{case\_y\_2}$	$\$, -, q_{case\_y\_1}$
$q_{case\_y\_2}$	$Y, -, q_{case\_y\_n}$			$\$, +, q_{case\_y\_2}$
$q_{case\_y\_n}$			$*, -, q_{case\_y\_n}$	$\$, -, q_{case\_y\_n}$
q_{copy_start}			$b, +, q_{copy\_end}$	
$q_{copy\_y\_1}$	$Y, +, q_{copy\_y\_1}$	$N, -, q_{copy\_y\_3}$	$*, +, q_{copy\_y\_1}$	$\$, +, q_{copy\_y\_1}$
$q_{copy\_y\_2}$				
$q_{copy\_y\_3}$	$Y, -, q_{copy\_y\_3}$		$*, -, q_{copy\_y\_3}$	$\$, -, q_{copy\_y\_3}$
$q_{copy\_n\_1}$	$Y, +, q_{copy\_n\_1}$	$N, -, q_{copy\_n\_3}$	$*, +, q_{copy\_n\_1}$	$\$, +, q_{copy\_n\_1}$
$q_{copy\_n\_2}$				
$q_{copy\_n\_3}$	$Y, -, q_{copy\_n\_3}$		$*, -, q_{copy\_n\_3}$	$\$, -, q_{copy\_n\_3}$
q_{copy_end}	$y, +, q_{begin}$	$n, +, q_{begin}$	$*, +, q_{copy\_end}$	$\$, +, q_{copy\_end}$
q_{cycle_end}			$*, +, q_{cycle\_end}$	$\$, -, q_{copy\_start}$

$[q_{copy\_y\_1}, b, y, +, q_{copy\_y\_2}]$, they cannot be merged without violating the reversibility condition.

We now give a string homomorphism $\varphi_1 : \{Y, N\}^* \to \{y, n\}^*$ as follows: $\varphi_1(Y) = y,\ \varphi_1(N) = n$. Note that φ_1 simply converts the uppercase Y and N into lower case y and n. Let $C = (k, (halt, w_1, \ldots, w_{k-1}))$ be an arbitrary CTSH, and $v_0 \in \{Y, N\}^*$ be an initial string. Then the initial tape for U_{13_7} is as follows, where \$ and the leftmost b are used as delimiters (see Fig. 10.3). Here, w^{R} denotes the reversal of the string w.

$$b\ \varphi_1(w_{k-1}^{\mathrm{R}}) * \cdots * \varphi_1(w_2^{\mathrm{R}}) * \varphi_1(w_1^{\mathrm{R}}) * b\ \$\ \varphi_1(v_0)\ b$$

In the case of CTSH $\hat{C}$ with $v_0 = NYY$ in Example 1, the initial tape for it is $b\,ynn * yn * b\,\$\ nyy\,b$. Snapshots of computation of U_{13_7} is as below. It simulates the complete computing process of $\hat{C}$: $(NYY, 0) \Rightarrow (YY, 1) \Rightarrow (YNY, 2) \Rightarrow (NYNNY, 0) \Rightarrow (YNNY, 1) \Rightarrow (NNYNY, 2) \Rightarrow (NYNY, 0) \Rightarrow (YNY, 1) \Rightarrow (NYNY, 2) \Rightarrow (YNY, 0)$. In each computational configuration of U_{13_7}, the head position is also indicated by the underline.

$$
\begin{aligned}
t = 0 &: [\ b\,y\,n\,n * y\,n * b\,\$\,\underline{n}\,y\,y\,b,\ q_{begin},\ 10\ ] \\
7 &: [\ b\,y\,n\,n * y\,n\,b * \$\,n\,\underline{y}\,y\,b,\ q_{begin},\ 11\ ] \\
8 &: [\ b\,y\,n\,n * y\,n\,b * \$\,\underline{n}\,\overline{Y}\,y\,b,\ q_{case\_y\_1},\ 10\ ] \\
18 &: [\ b\,y\,n\,n * y\,\underline{n} * * \$\,n\,Y\,y\,b,\ q_{copy\_start},\ 6\ ] \\
19 &: [\ b\,y\,n\,n * y\,b\,\underline{*} * \$\,n\,Y\,y\,b,\ q_{copy\_n\_1},\ 7\ ] \\
25 &: [\ b\,y\,n\,n * y\,b * * \$\,n\,Y\,y\,\underline{b},\ q_{copy\_n\_1},\ 13\ ] \\
26 &: [\ b\,y\,n\,n * y\,b * * \$\,n\,Y\,y\,n\,\underline{b},\ q_{copy\_n\_2},\ 14\ ] \\
27 &: [\ b\,y\,n\,n * y\,b * * \$\,n\,Y\,y\,\underline{n}\,b,\ q_{copy\_n\_3},\ 13\ ] \\
35 &: [\ b\,y\,n\,n * \underline{y}\,n * * \$\,n\,Y\,y\,n\,b,\ q_{copy\_start},\ 5\ ] \\
36 &: [\ b\,y\,n\,n * \overline{b}\,\underline{n} * * \$\,n\,Y\,y\,n\,b,\ q_{copy\_y\_1},\ 6\ ] \\
44 &: [\ b\,y\,n\,n * b\,n * * \$\,n\,Y\,y\,n\,\underline{b},\ q_{copy\_y\_1},\ 14\ ] \\
45 &: [\ b\,y\,n\,n * b\,n * * \$\,n\,Y\,y\,n\,y\,\underline{b},\ q_{copy\_y\_2},\ 15\ ] \\
46 &: [\ b\,y\,n\,n * b\,n * * \$\,n\,Y\,y\,n\,\underline{y}\,b,\ q_{copy\_y\_3},\ 14\ ] \\
56 &: [\ b\,y\,n\,n\,\underline{*}\,y\,n * * \$\,n\,Y\,y\,n\,y\,b,\ q_{copy\_start},\ 4\ ] \\
57 &: [\ b\,y\,n\,n\,b\,\underline{y}\,n * * \$\,n\,Y\,y\,n\,y\,b,\ q_{copy\_end},\ 5\ ] \\
64 &: [\ b\,y\,n\,n\,b\,y\,n * * \$\,n\,y\,\underline{y}\,n\,y\,b,\ q_{begin},\ 12\ ] \\
174 &: [\ \underline{b}\,y\,n\,n * y\,n * * \$\,n\,y\,Y\,n\,y\,n\,n\,y\,b,\ q_{copy\_start},\ 0\ ] \\
175 &: [\ b\,\underline{y}\,n\,n * y\,n * * \$\,n\,y\,Y\,n\,y\,n\,n\,y\,b,\ q_{cycle\_end},\ 1\ ] \\
184 &: [\ b\,y\,n\,n * y\,n * \underline{*} \$\,n\,y\,Y\,n\,y\,n\,n\,y\,b,\ q_{copy\_start},\ 8\ ] \\
185 &: [\ b\,y\,n\,n * y\,n * b\,\underline{\$}\,n\,y\,Y\,n\,y\,n\,n\,y\,b,\ q_{copy\_end},\ 9\ ] \\
291 &: [\ b\,y\,n\,n\,b\,y\,n * * \$\,n\,y\,y\,n\,y\,\underline{n}\,n\,y\,n\,y\,b,\ q_{begin},\ 15\ ] \\
292 &: [\ b\,y\,n\,n\,b\,y\,n * * \$\,n\,y\,y\,n\,\underline{y}\,N\,n\,y\,n\,y\,b,\ q_{case\_y\_n},\ 14\ ] \\
303 &: [\ b\,y\,n\,\underline{n} * y\,n * * \$\,n\,y\,y\,n\,\overline{y}\,N\,n\,y\,n\,y\,b,\ q_{copy\_start},\ 3\ ] \\
315 &: [\ b\,y\,n\,b * y\,n * * \$\,n\,y\,y\,n\,y\,\underline{N}\,n\,y\,n\,y\,b,\ q_{copy\_n\_1},\ 15\ ] \\
316 &: [\ b\,y\,n\,b * y\,n * * \$\,n\,y\,y\,n\,\underline{y}\,N\,n\,y\,n\,y\,b,\ q_{copy\_n\_3},\ 14\ ] \\
665 &: [\ b\,y\,n\,n * y\,n * b\,\$\,n\,y\,y\,n\,\overline{y}\,n\,n\,y\,n\,\underline{y}\,n\,y\,b,\ q_{begin},\ 19\ ] \\
676 &: [\ b\,y\,n\,n * y\,n * \underline{b}\,\$\,n\,y\,y\,n\,y\,n\,n\,y\,n\,\overline{Y}\,n\,y\,b,\ q_{case\_y\_1},\ 8\ ]
\end{aligned}
$$

We explain how U_{13_7} simulates CTSH by this example. Production rules of $\hat{C}$ is basically expressed by the string $ynn*\,yn**$. However, to indicate the phase m of an ID (v, m), the mth $*$ from the right is altered into b (where the rightmost $*$ is the 0-th). This b is used as a "phase marker". Namely, $ynn*\,yn*\,b$, $ynn*\,yn\,b\,*$, and $ynn\,b\,yn**$ indicate the phase is 0, 1, and 2, respectively. Hence, in the configuration at $t = 0$, the string $ynn*\,yn*\,b$ is given on the tape. To the right of the production rules the initial string nyy is given. Between them, there is a delimiter \$ that is not rewritten into another symbol throughout the computation. In the state q_{begin}, the leftmost symbol of the current string is pointed by the head. Then, it is changed to the uppercase letter Y or N to indicate the leftmost position of the current string.

The state q_{begin} (appearing at time $t = 0, 7, 64, 291$, and 665) reads the first symbol y or n of the current string, and temporarily changes it into Y or N, respectively. Depending on the read symbol y or n, U_{13_7} goes to either $q_{case\_y\_1}$ ($t = 8$), or $q_{case\_y\_n}$ ($t = 292$). If the read symbol is b, U_{13_7} halts, because it means the string is null. If the symbol is y, U_{13_7} performs the following operations (the case n is explained in the next paragraph). By the state $q_{case\_y\_1}$ ($t = 8$), the URTM moves leftward to find the delimiter \$, and then visits the left-neighboring square by $q_{case\_y\_1}$. If it reads b, then it halts ($t = 676$), because the phase is 0. If otherwise, U_{13_7} returns to the delimiter \$. Then, using $q_{case\_y\_2}$, U_{13_7} goes to the state $q_{case\_y\_n}$, and moves leftward to find the phase marker b that indicates the position of the next production rule. By the state q_{copy_start} ($t = 18$, and 35) U_{13_7} starts to copy each symbol of the production rule. If U_{13_7} reads a symbol y ($t = 18$) (or n ($t = 35$), respectively), then it shifts the marker b to this position, and goes to $q_{copy\_y\_1}$ ($t = 36$) (or $q_{copy\_n\_1}$ ($t = 19$)) to attach the symbol at the end of the string to be rewritten. On the other hand, if it reads symbol $*$ in q_{copy_start} ($t = 56$), then it goes to q_{copy_end} ($t = 57$), which mean the end of execution of a production rule, and thus it starts to read the next symbol in the rewritten string ($t = 64$). Likewise, if it reads symbol b in q_{copy_start} ($t = 174$), then it goes to q_{cycle_end} ($t = 175$), which mean the end of one cycle, and thus the phase is set to 0 ($t = 185$). The state $q_{copy\_y\_1}$ is for moving rightward to find the first b that is to the right of the current string ($t = 44$), and rewrites it into y ($t = 45$). The states $q_{copy\_y\_2}$ and $q_{copy\_y\_3}$ ($t = 46$) are for returning to the marker position and for repeating the copying procedure. $q_{copy\_n\_1}, \ldots, q_{copy\_n\_3}$ are for copying the symbol n, which are similar to the case of y ($t = 19, 25, 26, 27$).

On the other hand, if U_{13_7} reads a symbol n in the state q_{begin} ($t = 291$), then it enters the state $q_{case\_y\_n}$ ($t = 292$), and tries to copy symbols as in the case of y. At $t = 303$ it starts to copy a symbol n in q_{copy_start}. However, since it finds a symbol N in the state $q_{copy\_n\_1}$ ($t = 315$), it enters the state $q_{copy\_n\_3}$ ($t = 316$) without attaching the symbol n at the right end. By above, the phase marker is finally shifted to the next production rule without copying the symbols of the current production rule.

Repeating the above procedure, U_{13_7} simulates a given CTSH step by step, and halts in the state $q_{case\_y\_1}$ reading the symbol b if the CTSH halts in an ID with phase 0. If the string of the CTSH becomes null, U_{13_7} halts in the state q_{begin} reading the symbol b. In the above example, U_{13_7} halts at $t = 676$, and the final string YNY of $\hat{C}$ is obtained at as a suffix of the string (excluding the last blank symbol b) starting from the symbol Y.

10.3.2 10-State 8-Symbol URTM

Next, we give URTM(10,8) U_{10_8}. It is defined as below.

$$U_{10\_8} = (Q_{10\_8}, \{b, y, n, n', Y, N, *, \$\}, q_{begin}, b, \delta_{10\_8})$$
$$Q_{10\_8} = \{q_{begin}, q_{case\_y\_1}, q_{case\_y\_2}, q_{case\_y\_n}, q_{copy\_start}, q_{copy\_y\_1}, q_{copy\_y\_2}, q_{copy\_n\_1}, q_{copy\_n\_2}, q_{copy\_end}\}$$

Table 10.2 shows the move relation δ_{10_8}. It contains 61 quintuples. Reversibility of U_{10_8} is verified by a careful checking of δ_{10_8}. It was also checked by a computer program. The URTM U_{10_8} is constructed by modifying U_{13_7} in the previous subsection. Thus, the initial tape of U_{10_8} is just the same as that of U_{13_7}. Furthermore, the simulation time for a given CTSH is also the same.

The difference between U_{13_7} and U_{10_8} is as follows. First, the removed symbols from the string are indicated by the uppercase letters Y and N. Second, if the leftmost symbol of the current string is n, then it is temporarily changed to n', which is a newly added symbol in U_{10_8}. Third, if the current phase is m, then the symbols of the

Table 10.2 The move relation δ_{10_8} of U_{10_8}

	b	y	n	n'
q_{begin}	(*null*)	$y, -, q_{case\_y\_1}$	$n', -, q_{case\_y\_n}$	
$q_{case\_y\_1}$	(*halt*)			
$q_{case\_y\_2}$	$b, -, q_{copy\_y\_2}$	$y, -, q_{case\_y\_n}$		
$q_{case\_y\_n}$	$*, -, q_{copy\_start}$			
q_{copy_start}	$b, +, q_{begin}$	$b, +, q_{copy\_y\_1}$	$b, +, q_{copy\_n\_1}$	
$q_{copy\_y\_1}$	$y, +, q_{case\_y\_2}$	$y, +, q_{copy\_y\_1}$	$n, +, q_{copy\_y\_1}$	$n', -, q_{copy\_y\_2}$
$q_{copy\_y\_2}$	$Y, -, q_{copy\_start}$	$y, -, q_{copy\_y\_2}$	$n, -, q_{copy\_y\_2}$	
$q_{copy\_n\_1}$	$n, +, q_{copy\_end}$	$y, +, q_{copy\_n\_1}$	$n, +, q_{copy\_n\_1}$	$n', -, q_{copy\_n\_2}$
$q_{copy\_n\_2}$	$N, -, q_{copy\_start}$	$y, -, q_{copy\_n\_2}$	$n, -, q_{copy\_n\_2}$	
q_{copy_end}	$b, -, q_{copy\_n\_2}$	$Y, +, q_{begin}$		$N, +, q_{begin}$
	Y	N	$*$	$\$$
q_{begin}	$y, +, q_{begin}$	$n, +, q_{begin}$	$*, +, q_{begin}$	$\$, -, q_{copy\_start}$
$q_{case\_y\_1}$	$Y, -, q_{case\_y\_1}$	$N, -, q_{case\_y\_1}$	$*, +, q_{case\_y\_2}$	$\$, -, q_{case\_y\_1}$
$q_{case\_y\_2}$	$Y, +, q_{case\_y\_2}$	$N, +, q_{case\_y\_2}$		$\$, +, q_{case\_y\_2}$
$q_{case\_y\_n}$	$Y, -, q_{case\_y\_n}$	$N, -, q_{case\_y\_n}$	$*, -, q_{case\_y\_n}$	$\$, -, q_{case\_y\_n}$
q_{copy_start}			$b, +, q_{copy\_end}$	
$q_{copy\_y\_1}$	$Y, +, q_{copy\_y\_1}$	$N, +, q_{copy\_y\_1}$	$*, +, q_{copy\_y\_1}$	$\$, +, q_{copy\_y\_1}$
$q_{copy\_y\_2}$	$Y, -, q_{copy\_y\_2}$	$N, -, q_{copy\_y\_2}$	$*, -, q_{copy\_y\_2}$	$\$, -, q_{copy\_y\_2}$
$q_{copy\_n\_1}$	$Y, +, q_{copy\_n\_1}$	$N, +, q_{copy\_n\_1}$	$*, +, q_{copy\_n\_1}$	$\$, +, q_{copy\_n\_1}$
$q_{copy\_n\_2}$	$Y, -, q_{copy\_n\_2}$	$N, -, q_{copy\_n\_2}$	$*, -, q_{copy\_n\_2}$	$\$, -, q_{copy\_n\_2}$
q_{copy_end}	$Y, +, q_{copy\_end}$	$N, +, q_{copy\_end}$	$*, +, q_{copy\_end}$	$\$, +, q_{copy\_end}$

production rules $w_1, \ldots, w_{m-1}$ are changed into the uppercase letters. By above, the states $q_{copy\_y\_2}$ and $q_{case\_y\_2}$ in U_{13_7} can be merged into one state without violating the reversibility condition. Likewise, the states $q_{copy\_n\_2}$ and q_{copy_end} in U_{13_7} can be merged into one state. Hence, in U_{10_8}, the old state $q_{copy\_y\_2}$ ($q_{copy\_n\_2}$, respectively) is removed, and the old state $q_{copy\_y\_3}$ ($q_{copy\_n\_3}$) is renamed to $q_{copy\_y\_2}$ ($q_{copy\_n\_2}$). Furthermore, the states q_{begin} and q_{cycle_end} in U_{13_7} can be merged into one state. Therefore, in U_{13_7}, q_{cycle_end} is removed. By above, the number of states of U_{10_8} is reduced to 10.

Snapshots of computation process of U_{10_8} for the CTSH $\hat{C}$ with the initial string *NYY* is as below.

$$
\begin{aligned}
t = 0 &: [\ b\,y\,n\,n\,*\,y\,n\,*\,b\,\$\,\underline{n}\,y\,y\,b,\ q_{begin},\ 10\ ] \\
7 &: [\ b\,y\,n\,n\,*\,y\,n\,b\,*\ \$\,N\underline{y}\,y\,b,\ q_{begin},\ 11\ ] \\
8 &: [\ b\,y\,n\,n\,*\,y\,n\,b\,*\ \$\,\underline{N}\underline{y}\,y\,b,\ q_{case\_y\_1},\ 10\ ] \\
18 &: [\ b\,y\,n\,n\,*\,y\,\underline{n}\,*\,*\,\$\,N\underline{y}\,y\,b,\ q_{copy\_start},\ 6\ ] \\
19 &: [\ b\,y\,n\,n\,*\,y\,b\,\underline{*}\,*\ \$\,N\underline{y}\,y\,b,\ q_{copy\_n\_1},\ 7\ ] \\
25 &: [\ b\,y\,n\,n\,*\,y\,b\,*\,*\,\$\,N\underline{y}\,y\,\underline{b},\ q_{copy\_n\_1},\ 13\ ] \\
26 &: [\ b\,y\,n\,n\,*\,y\,b\,*\,*\,\$\,N\underline{y}\,y\,n\,\underline{b},\ q_{copy\_end},\ 14\ ] \\
27 &: [\ b\,y\,n\,n\,*\,y\,b\,*\,*\,\$\,N\underline{y}\,y\,\underline{n}\,b,\ q_{copy\_n\_2},\ 13\ ] \\
35 &: [\ b\,y\,n\,n\,*\,\underline{y}\,N\,*\,*\,\$\,N\underline{y}\,y\,n\,b,\ q_{copy\_start},\ 5\ ] \\
36 &: [\ b\,y\,n\,n\,*\,b\,\underline{N}\,*\,*\,\$\,N\underline{y}\,y\,n\,b,\ q_{copy\_y\_1},\ 6\ ] \\
44 &: [\ b\,y\,n\,n\,*\,b\,N\,*\,*\,\$\,N\underline{y}\,y\,n\,\underline{b},\ q_{copy\_y\_1},\ 14\ ] \\
45 &: [\ b\,y\,n\,n\,*\,b\,N\,*\,*\,\$\,N\underline{y}\,y\,n\,\underline{y}\,\underline{b},\ q_{case\_y\_2},\ 15\ ] \\
46 &: [\ b\,y\,n\,n\,*\,b\,N\,*\,*\,\$\,N\underline{y}\,y\,n\,\underline{y}\,b,\ q_{copy\_y\_2},\ 14\ ] \\
56 &: [\ b\,y\,n\,n\,\underline{*}\,Y\,N\,*\,*\,\$\,N\underline{y}\,y\,n\,\underline{y}\,b,\ q_{copy\_start},\ 4\ ] \\
57 &: [\ b\,y\,n\,n\,b\,\underline{Y}\,N\,*\,*\,\$\,N\underline{y}\,y\,n\,\underline{y}\,b,\ q_{copy\_end},\ 5\ ] \\
64 &: [\ b\,y\,n\,n\,b\,\underline{Y}\,N\,*\,*\,\$\,N\underline{Y}\,\underline{y}\,n\,\underline{y}\,b,\ q_{begin},\ 12\ ] \\
174 &: [\ \underline{b}\,Y\,N\,N\,*\,Y\,N\,*\,*\,\$\,N\underline{Y}\,\underline{y}\,n\,\underline{y}\,n\,n\,\underline{y}\,b,\ q_{copy\_start},\ 0\ ] \\
175 &: [\ b\,\underline{Y}\,N\,N\,*\ Y\,N\,*\,*\,\$\,N\underline{Y}\,\underline{y}\,n\,\underline{y}\,n\,n\,\underline{y}\,b,\ q_{begin},\ 1\ ] \\
184 &: [\ b\,y\,n\,n\,*\,y\,n\,*\,\underline{*}\,\$\,N\underline{Y}\,\underline{y}\,n\,\underline{y}\,n\,n\,\underline{y}\,b,\ q_{copy\_start},\ 8\ ] \\
185 &: [\ b\,y\,n\,n\,*\,y\,n\,*\,b\,\underline{\$}\,N\underline{Y}\,\underline{y}\,n\,\underline{y}\,n\,n\,\underline{y}\,b,\ q_{copy\_end},\ 9\ ] \\
291 &: [\ b\,y\,n\,n\,b\,Y\,N\,*\,*\,\$\,NYYN\underline{Y}\,\underline{n}\,n\,y\,n\,y\,b,\ q_{begin},\ 15\ ] \\
292 &: [\ b\,y\,n\,n\,b\,Y\,N\,*\,*\,\$\,NYYN\underline{Y}\,\underline{n'}\,n\,y\,n\,y\,b,\ q_{case\_y\_n},\ 14\ ] \\
303 &: [\ b\,y\,n\,\underline{n}\,*\,Y\,N\,*\,*\,\$\,NYYNYn'\,n\,y\,n\,y\,b,\ q_{copy\_start},\ 3\ ] \\
315 &: [\ b\,y\,n\,b\,*\,Y\,N\,*\,*\,\$\,NYYNY\underline{n'}\,n\,y\,n\,y\,b,\ q_{copy\_n\_1},\ 15\ ] \\
316 &: [\ b\,y\,n\,b\,*\,Y\,N\,*\,*\,\$\,NYYN\underline{Y}\,\underline{n'}\,n\,y\,n\,y\,b,\ q_{copy\_n\_2},\ 14\ ] \\
665 &: [\ b\,y\,n\,n\,*\,y\,n\,*\,b\,\$\,NYYNYNNYN\underline{y}\,\underline{n}\,y\,b,\ q_{begin},\ 19\ ] \\
676 &: [\ b\,y\,n\,n\,*\,y\,n\,*\,\underline{b}\,\$\,NYYNYNNYN\underline{y}\,\underline{n}\,y\,b,\ q_{case\_y\_1},\ 8\ ]
\end{aligned}
$$

Comparing the above computational configurations with the ones of U_{13_7}, we can see that, e.g., at time $t = 45$ the state $q_{case\_y\_2}$ is used instead of $q_{copy\_y\_2}$, and at time $t = 175$ the state q_{begin} is used instead of q_{cycle_end}. However, the essentially the same operation as in U_{13_7} is performed at each step, and thus at time $t = 676$ the final string *YNY* is obtained.

10.4 Comparison with Other Small URTMs

Besides URTM(13,7) and URTM(10,8), which are constructed here, several URTMs that simulates CTSH have been given in [9, 13, 15]. They are URTM(15,6), URTM(17,5), URTM(24,4), and URTM(32,3).

On the other hand, it is generally difficult to design an RTM that has only two symbols, or a very small number of states. To obtain an RTM with a small number of states, general procedures for converting a given many-state RTM into a 4-state and 3-state RTMs are given in [10], though the number of symbols of the resulting RTMs becomes very large.

Theorem 3 ([10]) *For any one-tape m-state n-symbol RTM T, we can construct a one-tape 4-state* $(2mn + n)$*-symbol RTM* $\tilde{T}$ *that simulates T.*

Theorem 4 ([10]) *For any one-tape m-state n-symbol RTM T, we can construct a one-tape 3-state RTM* $\hat{T}$ *with* $O(m^2n^3)$*-symbols that simulates T.*

Applying the method of Theorem 3 to URTM(10,8), we obtain URTM(4,168). Likewise, by the method of Theorem 4, we obtain URTM(3, 36654) from URTM(32,3).

To construct a 2-symbol URTM, we can use a method of converting a many-symbol RTM into a 2-symbol RTM shown in [16]. In particular, the following lemma is shown in [13] to convert a 4-symbol RTM to a 2-symbol RTM.

Lemma 1 ([13]) *For any one-tape m-state 4-symbol RTM T, we can construct a one-tape* m'*-state 2-symbol RTM* $T^{\dagger}$ *that simulates T such that* $m' \leq 6m$*.*

By this method, we can obtain URTM(138,4) from URTM(24,4) [13].

These results are summarized as follows.

- URTM(3,36654) with 37936 quintuples [10]
- URTM(4,168) with 381 quintuples [10]
- URTM(10,8) with 61 quintuples
- URTM(13,7) with 57 quintuples
- URTM(15,6) with 62 quintuples [9]
- URTM(17,5) with 67 quintuples [15]
- URTM(24,4) with 82 quintuples [13]
- URTM(32,3) with 82 quintuples [13]
- URTM(138,2) with 220 quintuples [13]

We can see URTM(10,8) has the minimum value of $m \times n$ among the above URTM(m, n)'s. On the other hand, URTM(13,7) has the smallest number of quintuples among them. The pairs of numbers of states and symbols of these URTMs, as well as the smallest UTMs so far known, are plotted in Fig. 10.4.

The products of the numbers of states and symbols of URTM(10,8), URTM(13,7), URTM(15,6), URTM(17,5), URTM(24,4), and URTM(32,3) are all less than 100, and thus relatively small. However, those of URTM(3,36654), URTM(4,168), and URTM(138,2), which are converted from the above ones, are very large, and it is not

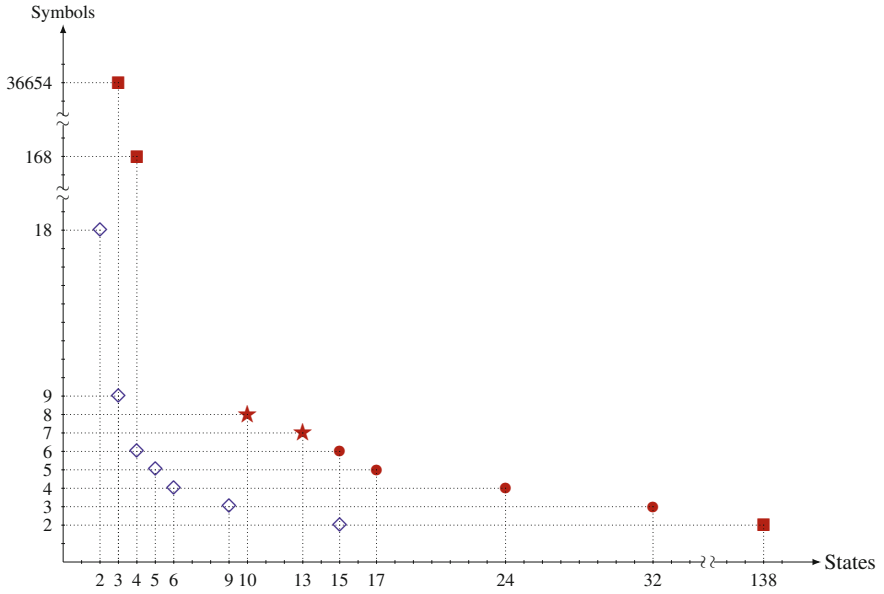

Fig. 10.4 State-symbol plotting of small URTMs and UTMs. ★ shows URTMs newly given in this paper that simulate cyclic tag systems. ● indicates URTMs given in [9, 13, 15] that simulate cyclic tag systems. ■ indicates URTMs converted from other URTMs. ◇ shows UTMs given in [6, 18, 20] that simulate 2-tag systems or bi-tag systems

known whether there are much smaller URTMs. Also, small URTM(m,n)'s such that $5 \leq m \leq 9$ have not yet been constructed till now.

Examples of computing processes of the nine URTMs listed above were simulated by a computer program. Animation-like figures of the computer simulation results, as well as description files of the URTMs, are available in [12].

10.5 Concluding Remarks

We studied the problem of constructing small URTMs, which are universal TMs that satisfy the reversibility condition. For this, we used a method of simulating cyclic tag systems with halting condition. In this way, we newly obtained URTM(13,8), and URTM(10,8).

Woods and Neary [22] proved that both cyclic tag systems, and 2-tag systems can simulate TMs in polynomial time, and thus the small UTMs of Minsky [7], Rogozhin [20], Kudlek and Rogozhin [6], Neary and Woods [18], and others can simulate TMs efficiently. Although we did not discuss time complexity of the URTMs in detail, it is easy to see that the URTMs given here simulate cyclic tag systems in polynomial time. Therefore, these URTMs also simulate TMs in polynomial time.

In this study, we used a method of simulating cyclic tag systems with halting condition to construct small URTMs. However, it is not known whether there are

better methods other than it. Also, it is not known whether a 2-state URTM exists. Since there have been only several researches on small URTMs so far, there seems much room for improvement, and thus they are left for the future study.

Acknowledgments This work was supported by JSPS KAKENHI Grant Number 15K00019.

References

1. Axelsen, H.B., Glück, R.: A simple and efficient universal reversible Turing machines. In: Proceedings of the LATA 2011, LNCS, vol. 6638, pp. 117–128 (2011). doi:10.1007/978-3-642-21254-3_8
2. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**, 525–532 (1973). doi:10.1147/rd.176.0525
3. Cocke, J., Minsky, M.: Universality of tag systems with P = 2. J. Assoc. Comput. Mach. **11**, 15–20 (1964). doi:10.1145/321203.321206
4. Cook, M.: Universality in elementary cellular automata. Complex Syst. **15**, 1–40 (2004)
5. Fredkin, E., Toffoli, T.: Conserv. Log. Int. J. Theoret. Phys. **21**, 219–253 (1982). doi:10.1007/BF01857727
6. Kudlek, M., Rogozhin, Y.: A universal Turing machine with 3 states and 9 symbols. In: Proceedings of the DLT 2001, LNCS, vol. 2295, pp. 311–318 (2002). doi:10.1007/3-540-46011-X_27
7. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice-Hall, Englewood Cliffs (1967)
8. Morita, K.: A simple reversible logic element and cellular automata for reversible computing. In: Proceedings of the MCU 2001, LNCS 2055, pp. 102–113 (2001). doi:10.1007/3-540-45132-3_6
9. Morita, K.: Reversible computing and cellular automata – A survey. Theoret. Comput. Sci. **395**, 101–131 (2008). doi:10.1016/j.tcs.2008.01.041
10. Morita, K.: Reversible Turing machines with a small number of states. In: Proceedings of the NCMA 2014, pp. 179–190 (2014). Slides with figures of computer simulation: Hiroshima University Institutional Repository, http://ir.lib.hiroshima-u.ac.jp/00036075
11. Morita, K.: Constructing reversible Turing machines by reversible logic element with memory. In: Adamatzky, A. (ed.) Automata, Computation, Universality, pp. 127–138. Springer-Verlag (2015). doi:10.1007/978-3-319-09039-9_6. Slides with figures of computer simulation: Hiroshima University Institutional Repository, http://ir.lib.hiroshima-u.ac.jp/00029224
12. Morita, K.: Constructing small universal reversible Turing machines (slides with figures of computer simulation). Hiroshima University Institutional Repository (2015). http://ir.lib.hiroshima-u.ac.jp/00036736
13. Morita, K.: Universal reversible Turing machines with a small number of tape symbols. Fundam. Inform. **138**, 17–29 (2015). doi:10.3233/FI-2015-1195
14. Morita, K., Suyama, R.: Compact realization of reversible Turing machines by 2-state reversible logic elements. In: Proceedings of the UCNC 2014, LNCS, vol. 8553, pp. 280–292 (2014). doi:10.1007/978-3-319-08123-6_23. Slides with figures of computer simulation: Hiroshima University Institutional Repository, http://ir.lib.hiroshima-u.ac.jp/00036076
15. Morita, K., Yamaguchi, Y.: A universal reversible Turing machine. In: Proceedings of the MCU 2007, LNCS, vol. 4664, pp. 90–98 (2007). doi:10.1007/978-3-540-74593-8_8
16. Morita, K., Shirasaki, A., Gono, Y.: A 1-tape 2-symbol reversible Turing machine. Trans. IEICE Japan **E-72**, 223–228 (1989)
17. Mukai, Y., Morita, K.: Realizing reversible logic elements with memory in the billiard ball model. Int. J. Unconv. Comput. **8**, 47–59 (2012)

18. Neary, T., Woods, D.: Four small universal Turing machines. Fundamenta Informaticae **91**, 123–144 (2009). doi:10.3233/FI-2009-0036
19. Post, E.L.: Formal reductions of the general combinatorial decision problem. Am. J. Math. **65**, 197–215 (1943). doi:10.2307/2371809
20. Rogozhin, Y.: Small universal Turing machines. Theoret. Comput. Sci. **168**, 215–240 (1996). doi:10.1016/S0304-3975(96)00077-1
21. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proc. Lond. Math. Soc. Ser. 2 **42**, 230–265 (1936)
22. Woods, D., Neary, T.: On the time complexity of 2-tag systems and small universal Turing machines. In: Proceedings of the 47th Symposium on Foundations of Computer Science, pp. 439–446 (2006). doi:10.1109/FOCS.2006.58
23. Woods, D., Neary, T.: The complexity of small universal Turing machines: a survey. Theoret. Comput. Sci. **410**, 443–450 (2009). doi:10.1016/j.tcs.2008.09.051