THE TRANSACTIONS OF THE IEICE, VOL. E 73, NO. 6 JUNE 1990

PAPER

A Simple Construction Method of a Reversible Finite Automaton out of Fredkin Gates, and Its Related Problem

Kenichi MORITA†, *Member*

SUMMARY A reversible finite automaton (RFA) is a backward deterministic automaton, i.e., it can uniquely retrace its move sequence if the inverse sequence of its outputs is given. In this paper, we show a simple method to construct an RFA from Fredkin gates, which are reversible and bit-conserving logic gates, and unit wires (unit delays). The resulting circuit obtained by this method is "garbage-less" in the sense that it has no inputs to which constants must be supplied nor outputs from which garbage signals are put out. We also show that a one-dimensional reversible partitioned cellular automaton, which are known to be computation universal, can be constructed from Fredkin gates and unit wires as a closed (thus garbage-less) infinite circuit.

1. Introduction

A reversible computing system is a backward deterministic system, in which every state of the whole system has at most one predecessor and thus its computation path can be uniquely retraced. Since Bennett[1] proved that "any" computation can be performed by a reversible Turing machine, reversible systems have been extensively studied by many researchers (Ref. (3) describes its history). There are various kinds of reversible systems (and various levels of reversibility) such as reversible Turing machines, reversible cellular automata, reversible logic circuits, reversible physical models of computation, and so on. Reversible systems provide us a "thermodynamical viewpoint" to a computation, and are very inportant when considering inevitable power dissipation in a computation theoretically[1]-[5].

"Conservative logic" introduced by Fredkin and Toffoli[5] is a framework for dealing with logic circuits made from reversible and bit-conserving logic gates (especially the ones called Fredkin gates) and unit wires (unit delays). They showed the universality of Fredkin gates and unit wires. That is, any (non-conservative) logic circuit can be embedded in a circuit made from Fredkin gates and unit wires by allowing the (unlimited) supply of constant signals to some inputs and the discharge of garbage (i.e., useless) signals from some outputs. They also proposed the "billiard ball model" of computation, in which computations are carried out by elastic collisions of balls, and showed that every conservative logic circuit can be realized by this model.

Although the (ideal) billiard ball model itself never dissipates energy in the computation, the supply of constants and the discharge of garbage signals in the circuit correspond to the supply of balls (supply of power) and the discard of balls (radiation of heat). Therefore, it is desired to construct a conservative logic circuit which has neither constant inputs nor garbage outputs. Such a circuit will be called "garbage-less".

It has been known that some kinds of systems can be realized as a garbage-less conservative logic circuit. Fredkin and Toffoli[5] showed that any combinational logic circuit is embeddable in an "almost" garbage-less conservative logic circuit. Morita[6] showed that a tape unit of a Turing machine is embeddable in a garbage-less circuit. On the other hand, it seems impossible to realize a general irreversible sequential machine (or finite automaton) as a garbage-less circuit without adding a stack in which the move history of the machine is stored. However, as for a reversible sequential machine, we can suspect that it can be realized as a simple garbage-less circuit. But its explicit construction method has not been given yet.

In this paper, we define a reversible finite automaton (RFA), and show that, for such an automaton, there is a simple and systematic realization method of a garbage-less conservative logic circuit. To do this, several building modules are designed using Fredkin gates and unit wires. Then these modules are combined to construct a given RFA in a systematic way.

RFA is a special type of finite automata, and might be considered to have very weak computing ability. However, it can be used as a building unit of a universal computing system. Previously Morita et al.[7] introduced a one-dimensional partitioned cellular automaton (PCA), which is a kind of cellular automaton, and showed that reversible PCA can simulate any Turing machine, especially a universal Turing machine. Here we show a construction method of a reversible PCA from Fredkin gates and unit wires. Since each cell of a reversible PCA is essentially an RFA, it is designed in a similar manner as in RFA. The entire system of PCA is obtained by connecting infinitely many copies of

Manuscript received September 22, 1989.

† The author is with the Faculty of Engineering, Yamagata University, Yonezawa-shi, 992 Japan.

the cell in a line. The resulting circuit is a closed (thus garbage-less) infinite circuit. Thus, we can obtain a closed infinite conservative logic circuit in which a universal Turing machine can be simulated.

2. A Reversible Finite Automaton and Conservative Logic

In this section, several definitions concerning a reversible finite automaton and conservative logic are given.

2.1 A Reversible Finite Automaton

A deterministic reversible finite automaton (RFA) is a system defined by

$$M=(Q, S, f, q_0),$$

where

(1) Q is a non-empty finite set of states,

(2) S is a non-empty finite set of symbols,

(3) $f: Q\times S\rightarrow Q\times S$ is a move function which is an injection (one-to-one mapping), and

(4) q_0 is an initial state ($q_0\in Q$).

M uses S as both its input and output alphabets. Thus, $f(q, s)=(q', s')$ means that if M reads the symbol s in state q, M outputs the symbol s' and transits its state into q'.

Note that an RFA may be regarded as a string translator which computes a mapping $S^*\rightarrow S^*$. However, we do not discuss this feature of RFA here, since we are now concerning the problem how RFAs can be constructed from Fredkin gates.

2.2 Conservative Logic

Conservative logic was introduced by Fredkin and Toffoli[5] to deal with logic circuits made from reversible and bit-conserving logic elements.

A logic gate with n inputs and n outputs is called reversible iff its logical function $\{0, 1\}^n\rightarrow\{0, 1\}^n$ is one-to-one, and is called bit-conserving iff the number of "1"s in the input agrees with the number of "1"s in the output. A reversible and bit-conserving logic gate is simply called a conservative gate.

Fredkin gate is a 3-input 3-output conservative gate shown in Fig. 1, and its function can be intuitively captured as in Fig. 2.

A unit wire is a unit-delay element shown in Fig. 3. In conservative logic circuits, it is used not only as a memory but also as a transmission wire between gates. In the following, series composition of n unit wires is abbreviated as shown in Fig. 4.

A conservative logic circuit is a circuit made from conservative gates and unit wires satisfying the following conditions.

(1) Fan-out of an output is not allowed (If fan-out is needed, a conservative logic circuit realizing fan-out function should be constructed[5].).

(2) Any output of a gate must be connected to the input of a unit wire (i.e., two gates cannot be directly connected without inserting unit wires).

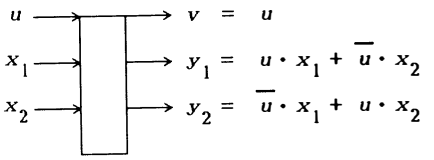


Fig. 1 A Fredkin gate.

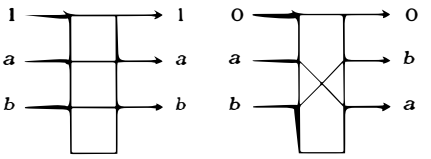


Fig. 2 The function of a Fredkin gate.

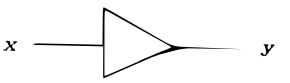


Fig. 3 A unit wire ($y(t)=x(t-1)$).

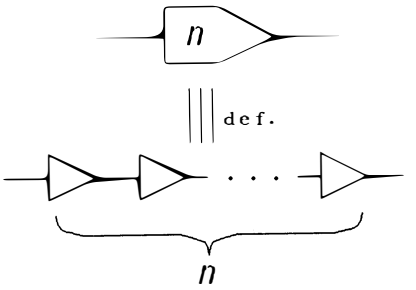


Fig. 4 An abbreviated notation of the series composition of n unit wires.

Let C be a conservative logic circuit. A line (i.e., input or output line) of a gate or a unit wire in C is called open if it is not connected to any other line in C. To an open input line an input signal or a constant must be given, and from an open output line an output signal or a garbage is put out. A circuit C is called semi-closed if it has no open lines except the lines to which (true) input or output signals are given. Furthermore, if C has no open lines at all, it is called closed.

A semi-closed (and of course, closed) circuit is "garbage-less" in the sense that it has no output lines putting out useless signals nor input lines to which we must supply constant signals.

3. Constructing an RFA out of Fredkin Gates

3.1 Building Modules

In order to facilitate the construction of RFAs, we first design several building modules using Fredkin gates and unit wires.

To represent a symbol or a state of an RFA in a circuit (or a module), we use a "bundle" of decoded lines. That is, for a given set of symbols (or states) $X=\{x_1, \cdots, x_n\}$ we prepare a bundle of n signal lines in the circuit. The i-th line in the bundle uniquely corresponds to the symbol x_i and also denoted by x_i. If the value of the i-th line is 1, we regard the symbol x_i is occurring in the RFA (therefore, just one of the lines $x_1, \cdots, x_n$ must be 1).

(1) Switch Module: SWITCH (n)

The switch module SWITCH (n), which is schematically represented as in Fig. 5, branches the input bundle $b_1, \cdots, b_n$ to the output bundle $ab_1, \cdots, ab_n$ or $\bar{a}b_1, \cdots, \bar{a}b_n$ according to the input signal a. Designing this module is the key point of the construction of RFA. Here, we pose the following constraints to its inputs.

(i) If $a=1$, then just one of the inputs $b_1, \cdots, b_n$ is 1.

(ii) $ac=0$.

These constraints make the construction of this module easier. Figure 6 shows the circuit SWITCH(3) made from Fredkin gates and unit wires. It is easily verified that this circuit has the desired function. It is easy to draw the circuit of SWITCH (n) for general n. The input-output delay in SWITCH (n) is $2n$.

Note that, in the circuit of SWITCH(3), there are three 0-inputs into which we must supply 0s, and one 0-output which always puts out 0s. These four lines are the open lines other than the ones for true input or output signals. Handling method of these open lines is described in Sect. 3.2.

(2) Delay Module: d-DELAY (n)

The delay module d-DELAY (n) simply consists of n lines of d-unit delays as shown in Fig. 7.

(3) Decoder Module: DECODER (m, n)

DECODER (m, n) decodes the combination of values of two input bundles with m lines and n lines. The decoded result is put out to the bundle with mn lines. The schematic representation is shown in Fig. 8. This module can be built from switch modules and delay modules as shown in Fig. 9. It is clear that this circuit works as a decoder because of the function of SWITCH (n). Its delay is $2mn$.

(4) Permutation Module: PERM (f)

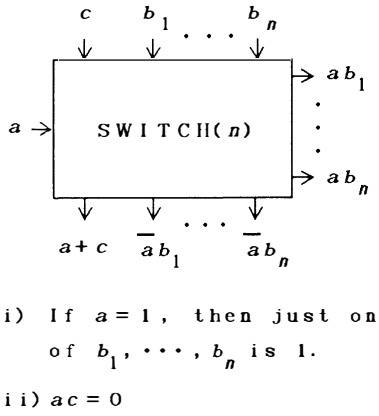


Fig. 5 Schematic representation of SWITCH(n) and the constraints to its inputs.

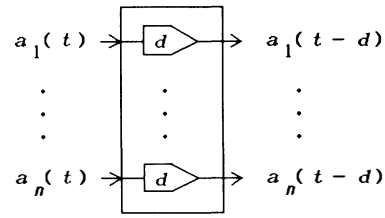


Fig. 7 Delay module d-DELAY(n).

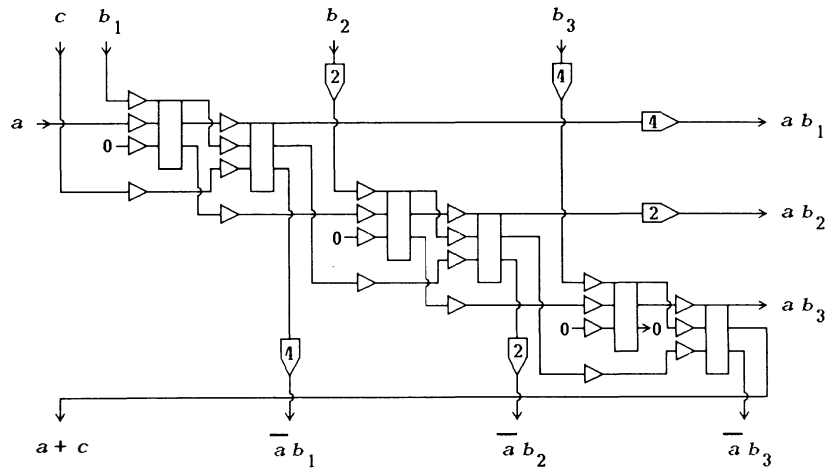


Fig. 6 The circuit of SWITCH(3).

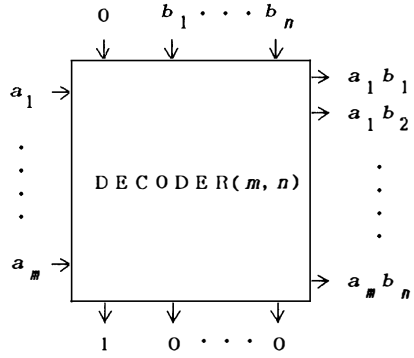


Fig. 8 Schematic representation of DECODER(m, n).

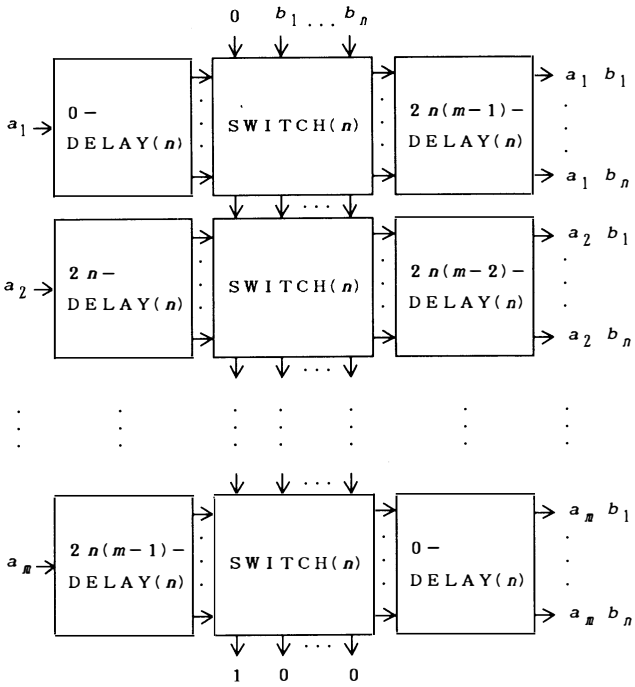


Fig. 9 Decoder module DECODER(m, n).

Let $f : \{1, \cdots, n\} \rightarrow \{1, \cdots, n\}$ be an injection (permutation). PERM(f) realizes this permutation as shown in Fig. 10.

Next, the notion of an inverse circuit (or module) is given. Let C be a conservative logic circuit. The inverse circuit C^{-1} is a conservative logic circuit obtained by inverting the inputs and outputs of each gate and each unit wire in C. For example, Figure 11 shows (the mirror image of) the inverse module of SWITCH (3). As mentioned in Ref. (5), if the circuit C realizes a combinational logic function f (with some delays), then C^{-1} realizes the inverse function f^{-1}. Thus, for example, DECODER $(m, n)^{-1}$ works as a kind of encoder.

3.2 Construction of an RFA

We now show a construction method of an RFA using the modules described above.

Let

$M=(Q, S, f, q_1)$,

be a given RFA. We assume the sets Q and S are as

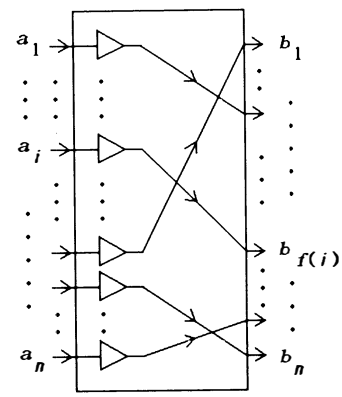


Fig. 10 Permutation module PERM(f).

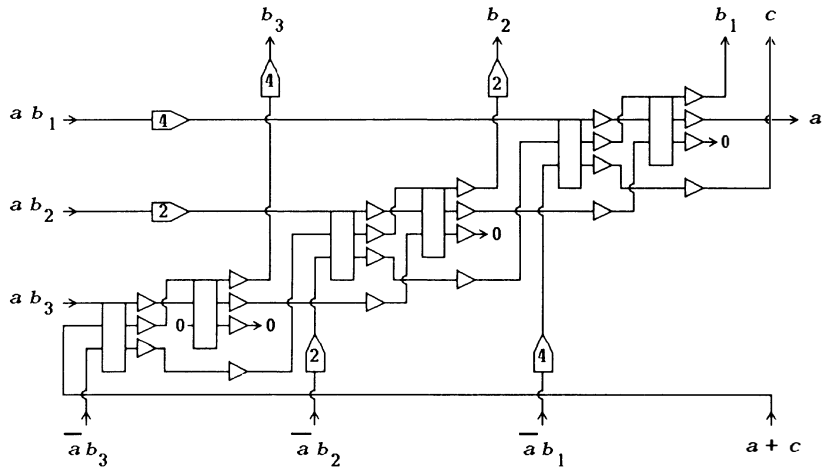


Fig. 11 SWITCH(3)$^{-1}$, the inverse module of SWITCH(3).

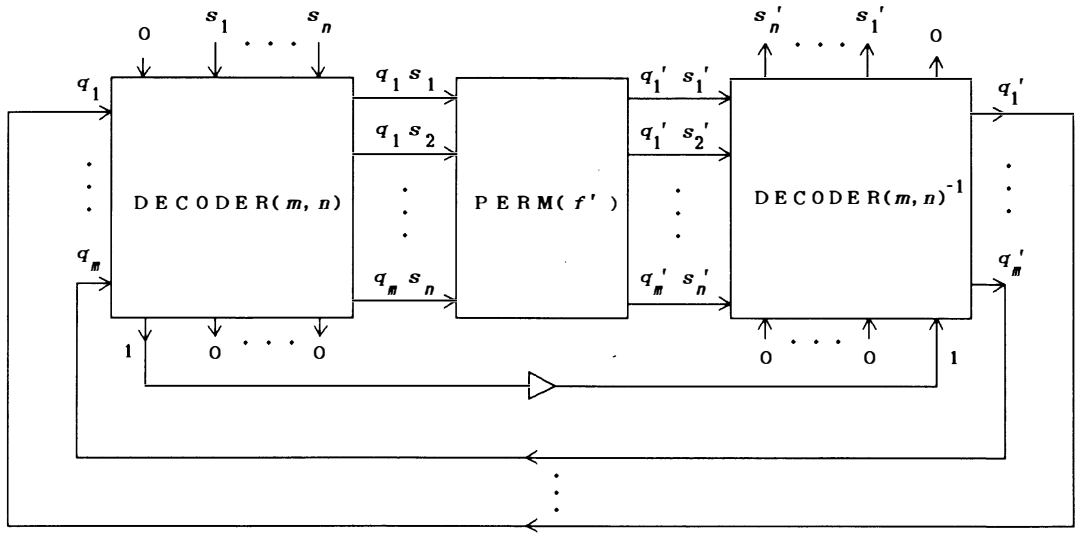


Fig. 12 A conservative logic circuit which realizes the RFA M.

follows.

$$Q=\{q_1, \cdots, q_m\}$$

$$S=\{s_1, \cdots, s_n\}$$

Let $f' : \{1, \cdots, mn\} \rightarrow \{1, \cdots, mn\}$ be the injection defined from the injection f as follows. For all $h, j \in \{1, \cdots, m\}$, and $i, k \in \{1, \cdots, n\}$,

$$f'((h-1)n+i)=(j-1)n+k$$

iff

$$f(q_h, s_i)=(q_j, s_k)$$

Then the RFA M is realized as a circuit shown in Fig. 12.

Assume the present state q_h and the input symbol s_i are given to the input bundles of DECODER (m, n). The combination of signals q_h and s_i is decoded by it. Then, PERM(f') computes the decoded (i. e., combined) signal of the next state q_j and the output symbol s_k. DECODER$(m, n)^{-1}$ separates the signals q_j and s_k and outputs to the appropriate lines. By this, one step movement of M is simulated.

Note that the whole circuit for M contains many 0-inputs and 0-outputs which are open. However, since DECODER$(m, n)^{-1}$ is the inverse of DECODER(m, n) and PERM(f') contains neither 0-input nor 0-output, the numbers of 0-inputs and 0-outputs are the same. Thus, by connecting each 0-output to some 0-input, we can cancel all these open lines. Thus, we can obtain the following theorem.

[Theorem 1] For any RFA $M=(Q, S, f, q_0)$, we can construct a semi-closed circuit which realizes M from Fredkin gates and unit wires (provided that input and output symbols in S are given through the bundles of decoded lines).

4. Constructing a Kind of Reversible Cellular Automata out of Fredkin Gates

In this section, we show that one-dimensional reversible partitioned cellular automaton can be constructed from Fredkin gates and unit wires as a closed circuit by applying the above construction method of an RFA.

A deterministic one-dimensional partitioned cellular automaton (PCA) is a system defined by

$$P=(\boldsymbol{Z}, L, C, R, f_P),$$

where

(1) $\boldsymbol{Z}$ is the set of all integers,

(2) L is a non-empty finite set of left internal states of each cell,

(3) C is a non-empty finite set of center internal states of each cell,

(4) R is a non-empty finite set of right internal states of each cell, and

(5) $f_P : R \times C \times L \rightarrow L \times C \times R$ is a mapping called a local function.

A configuration of P is a mapping

$$c : \boldsymbol{Z} \rightarrow L \times C \times R.$$

The set of all configurations over $L \times C \times R$ is denoted by $\mathrm{Conf}(L \times C \times R)$. Let "LEFT" ("CENTER", "RIGHT", respectively) be the projection function which picks out the left (center, right) element of a triple in $L \times C \times R$. The global function

$$F_P : \mathrm{Conf}(L \times C \times R) \rightarrow \mathrm{Conf}(L \times C \times R)$$

determined by P is defined as follows.

$$F_P(c)(i)$$

$$=f_P(\mathrm{RIGHT}(c(i-1)), \mathrm{CENTER}(c(i)),$$

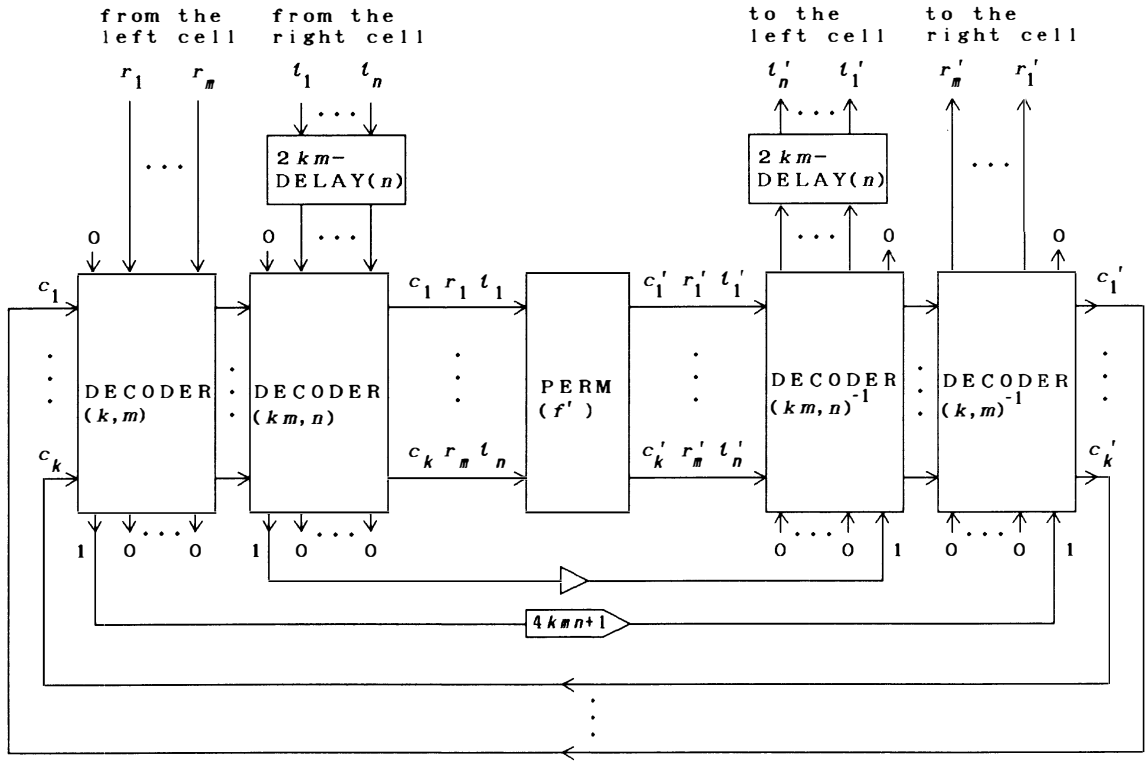


Fig. 13 A conservative logic circuit which realizes one cell of the PCA P.

$$\mathrm{LEFT}(c(i+1))) \qquad (i \in \boldsymbol{Z})$$

We say P is globally reversible iff F_P is an injection. We say P is locally reversible iff the local function f_P is an injection.

Intuitively speaking, each cell of PCA is partitioned into three parts (i. e., left, center, and right parts) and the next state of each cell is determined depending on the right part of the left cell, the center part of the center cell, and the left part of the right cell.

In Ref. (7), the following result is shown.

[Proposition 1] Let $P=(\boldsymbol{Z}, L, C, R, f_P)$ be a PCA. P is globally reversible iff P is locally reversible.

Thus, a globally or locally reversible PCA may be called simply "reversible" PCA.

We now give a closed conservative logic circuit which realizes a reversible PCA. Let $P=(\boldsymbol{Z}, L, C, R, f_P)$ be a given reversible PCA. Each cell of P can be regarded as a kind of RFA with two symbol sets R and L. Thus the cell of P can be realized in a similar manner as in RFA except that P has two kinds of alphabets. We assume C, R, and L are as follows.

$$C=\{c_1, \cdots, c_k\}$$

$$R=\{r_1, \cdots, r_m\}$$

$$L=\{l_1, \cdots, l_n\}$$

Let $f' : \{1, 2, \cdots, kmn\} \to \{1, 2, \cdots, kmn\}$ be the injection defined from the injection f_P as follows. For all $h, x \in \{1, \cdots, k\}$, $i, y \in \{1, \cdots, m\}$, and $j, z \in \{1, \cdots, n\}$,

$$f'((h-1)mn+(i-1)n+j)=(x-1)mn+(y-1)n+z$$

iff

$$f_P(r_i, c_h, l_j)=(l_z, c_x, r_y).$$

Figure 13 shows the circuit realizing the cell of P. By putting infinitely many copies of this circuit in a line, and appropriately connecting the input and the output lines of each circuit with those of its neighbors as noted in Fig. 13, we can obtain a closed circuit which realizes the whole system P.

By this, we obtain the following theorem.

[Theorem 2] For any reversible PCA $P=(\boldsymbol{Z}, L, C, R, f_P)$, we can construct a cellular structure closed circuit which realizes P, from Fredkin gates and unit wires.

In Ref. (7), (an equivalent result to) the following proposition is shown.

[Proposition 2] For any Turing machine T, there is a reversible PCA P which simulates T.

From Theorem 2 and Proposition 2, the following corollary is derived.

[Corollary 1] For any Turing machine T, we can construct a cellular structure closed circuit which simulates T, from Fredkin gates and unit wires.

5. Concluding Remarks

In this paper, we proposed a simple constructing method of an RFA from Fredkin gates and unit wires.

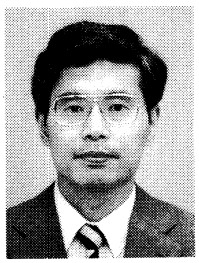
The circuit obtained by this method is semi-closed (i. e., garbage-less). The key point of this construction is to design a "switch module", from which a kind of decoder or encoder module is easily constructed.

The investigation of the property of an RFA as a string translator also seems interesting, and is left to the future study.

We also showed that a one-dimensional reversible PCA can be constructed from Fredkin gates and unit wires as a closed infinite circuit. Since one-dimensional reversible PCA is known to be computation universal, there exists a closed one-dimensional cellular structure conservative logic circuit in which any computation can be performed. If such a circuit is further realized by a billiard ball model, any computation can be performed by the autonomous movements of balls from some initial configuration without supplying or discarding balls from/to the outer world.

References

(1) C. H. Bennett: "Logical reversibility of computation", IBM J. Res. & Dev., **17**, pp. 525-532 (1973).
(2) C. H. Bennett: "The thermodynamics of computation—a review", Int. J. of Theoretical Physics, **21**, 12, pp. 905-940 (1982).
(3) C. H. Bennett: "Notes on the history of reversible computation", IBM J. Res. & Dev., **32**, 1, pp. 16-23 (1988).
(4) C. H. Bennett and R. Landauer: "The fundamental Physical limits of computation", Scientific American, **253**, 1, pp. 38-46 (July 1985).
(5) E. Fredkin and T. Toffoli: "Conservative logic", Int. J. of Theoretical Physics, **21**, 3/4, pp. 219-253 (1982).
(6) K. Morita: "Construction of a cellular structure memory unit out of Fredkin's reversible and conservative logic gate", Trans. IEICE, **J69-D**, 6, pp. 860-867 (June 1986).
(7) K. Morita and M. Harao: "Computation universality of one-dimensional reversible (injective) cellular automata", Trans. IEICE, **E72**, 6, pp. 758-762 (June 1988).



Kenichi Morita was born in Osaka, on March 30, 1949. He received the B. E., M. E., and Dr. E. degrees from Osaka University in 1971, 1973, and 1978, respectively. From 1974 to 1987, he was a Research Associate of the Faculty of Engineering Science, Osaka University. Since 1987 he has been an Associate Professor of the Faculty of Engineering, Yamagata University. He has been engaged in the research of automata theory, computational complexity, formal language theory, and logic systems for knowledge and language processing. Dr. Morita is a member of Information Processing Society of Japan, LA Symposium, Mathematical Linguistic Society of Japan, Japanese Society for Artificial Intelligence, ACM, and ACL.

Erratum

Fig. 9 should be replaced with the figure below.