Monographs in Theoretical Computer Science
An EATCS Series

Kenichi Morita

Theory of Reversible Computing

Springer

Monographs in Theoretical Computer Science
An EATCS Series

Kenichi Morita

Theory of Reversible Computing

Springer

Kenichi Morita
Professor Emeritus, Hiroshima University
Hiroshima, Japan

ISSN 1431-2654 ISSN 2193-2069 (electronic)
Monographs in Theoretical Computer Science. An EATCS Series
ISBN 978-4-431-56604-5 ISBN 978-4-431-56606-9 (eBook)
DOI 10.1007/978-4-431-56606-9

Library of Congress Control Number: 2017957823

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer Japan KK
The registered company address is: Chiyoda First Bldg. East, 8-1 Nishi-Kanda Chiyoda-ku, 101-0065 Tokyo, Japan

To my parents, my wife, my children, and my grandchildren

Preface

A reversible computing system is a "backward deterministic" system such that every state of the system has at most one predecessor. Hence, there is no pair of distinct states that go to the same state. Though its definition is so simple, it is closely related to physical reversibility. The study of reversible computing originated from an investigation of energy dissipation in reversible and irreversible computing systems. Rolf Landauer investigated the relation between reversibility in computing and reversibility in physics in his paper "Irreversibility and heat generation in the computing process" (IBM J. Res. Dev., Vol. 5, pp. 183–191, 1961). He pointed out that an irreversible logical operation inevitably causes energy dissipation in the computing system. Since then, reversible computing has been studied in relation to physical reversibility. Besides the problem of energy dissipation in computing, it is important to know how reversibility can be effectively utilized in computing. This is because future computing devices will surely be implemented directly by physical phenomena in the nano-scale level, and reversibility is one of the fundamental microscopic physical laws of Nature. For this purpose, various models of reversible computing have been proposed and investigated till now.

In this book, reversible computing is studied from the standpoint of the theory of automata and computing. We deal with various reversible computing models belonging to several different levels, which range from a microscopic level to a macroscopic one. They are reversible physical models, reversible logic elements, reversible functional modules composed of logic elements, reversible computing systems such as Turing machines, cellular automata, and others. The purpose of this book is to clarify how computation can be carried out efficiently and elegantly in these reversible computing models. We shall see that even very simple reversible systems have computational universality in spite of the constraint of reversibility. We shall also see various reversible systems in different levels are related each other, i.e., a reversible system in a higher level can be constructed out of those in a lower level. Moreover, the construction methods are often very unique and different from those in the traditional methods. Thus, these computing models as well as the designing methods will give us new insights for future computing systems.

This book is not a comprehensive textbook on reversible computing, but describes mainly the results shown in the papers by myself and my colleagues, which were published between 1989 and 2017. In this book, readers will see how the world of reversible computing works, how reversible computing systems are constructed out of simple reversible primitives, how they are different from the traditional computing systems, and how they can be computationally universal. In fact, we shall see even very simple reversible systems have high capability of computing, and thus reversibility is not a constraint, but a useful property for computing.

This book consists of 14 chapters. Chapter 1 is an introduction. In Chaps. 2–4, reversible logic elements for constructing reversible machines are investigated. In Chaps. 5–8, reversible Turing machines, a standard model of reversible computing systems, are studied. In Chap. 9, some other reversible computing models are investigated. In Chaps. 10–14, reversible cellular automata, a spatiotemporal model of reversible dynamical systems, are studied.

There is no prerequisite knowledge to read this book besides some basics on logic, discrete mathematics and formal languages. But, it is preferable to have some knowledge on the theories of automata and computing. Fortunately, the framework of reversible computing itself is very simple. Therefore, in many cases, readers can easily understand the basic function and the structure of each such system. However, its behavior can be very complex even if the structure of the system is simple. Hence, sometimes, it becomes quite difficult to follow its behavior by using only paper and pencil. In some of these cases, readers can find files in the References that contain computer simulation results of such reversible systems.

More than 50 years have passed since Landauer's paper appeared. Thus, the history of reversible computing is relatively long. But, it is still developing, and there remain many problems to be investigated. Also, even at present, it is not so clear which results will become practically useful in the future. However, the world of reversible computing will lead readers to the new ways of thinking that cannot be found in the traditional design methodologies for computing systems. I hope the theory of reversible computing will stimulate readers' interest, and open new vistas for future computing systems.

Acknowledgments. This work would not have been accomplished without the help and encouragement given by many people in many countries. I would express my highest gratitude to all of them. In particular, I would express my special thanks to Prof. Toshio Mitsui and Prof. Kazuhiro Sugata who guided me in the early days of my research career in Osaka University. Frequent discussions with them led me to the research fields of automata theory, cellular automata, and finally reversible computing. Since then, I had many good colleagues and students in Hiroshima University, Yamagata University, Osaka University, and other places in the world. Working with them was a great pleasure for me. I am grateful to them, especially, Susumu Adachi, Andrew Adamatzky, Artiom Alhazov, Yoichi Fujii, Yoshifumi Gono, Masateru Harao, Takahiro Hori, Katsunobu Imai, Teijiro Isokawa, Chuzo Iwamoto, Atsushi Kanno, Hiroko Kato, Jia Lee, Maurice Margenstern, Genaro J. Martinez, Mitsuya Morimoto, Yuta Mukai, Noritaka Nishihara, Masanori Ogino,

Tsuyoshi Ogiro, Ferdinand Peper, Akihiko Shirasaki, Rei Suyama, Keiji Tanaka, Tsuyoshi Tanizawa, Yasuyuki Tojima, Ryoichi Ueno, Satoshi Ueno, Hiroshi Umeo, Yoshikazu Yamaguchi, and Yasunori Yamamoto for their helpful discussions, cooperation, and coauthoring the papers on reversible computing and related fields.

I also express my cordial thanks to the anonymous reviewer for his/her very careful reading and valuable comments. The manuscript was greatly improved by the detailed comments.

This work on reversible computing was supported by JSPS KAKENHI Grant Number JP15K00019.

September 2017 *Kenichi Morita*

Contents

Acronyms

CA	cellular automaton
CM	counter machine
CTS	cyclic tag system
CTSH	cyclic tag system with halting condition
ECA	elementary cellular automaton
ETPCA	elementary triangular partitioned cellular automaton
GoL	Game of Life
ID	instantaneous description
IDTM	irreversible deterministic Turing machine
INTM	irreversible nondeterministic Turing machine
$\mathscr{L}(\mathscr{A})$	class of languages accepted by the class of automata $\mathscr{A}$
LBA	linear-bounded automaton
MFA	multi-head finite automaton
m-TS	m-tag system
PCA	partitioned cellular automaton
RCA	reversible cellular automaton
RCM	reversible counter machine
RDTM	reversible deterministic Turing machine
RE	rotary element
RETPCA	reversible elementary triangular partitioned cellular automaton
RLEM	reversible logic element with memory
RMFA	reversible multi-head finite automaton
RNTM	reversible nondeterministic Turing machine
RPCA	reversible partitioned cellular automaton
RSM	reversible sequential machine
RTM	reversible Turing machine
SM	sequential machine
TM	Turing machine
TM($S(n)$)	$S(n)$ space-bounded Turing machine
UTM	universal Turing machine
URTM	universal reversible Turing machine

Chapter 1
Introduction

Abstract Reversible computing is a paradigm that has a close relation to physical reversibility. Since microscopic physical laws are reversible, and future computing devices will surely be implemented directly by physical phenomena in the nano-scale level, it is an important problem to know how reversibility can be effectively utilized in computing. Reversible computing systems are defined as systems for which each of their computational configurations has at most one predecessor. Hence, they are "backward deterministic" systems. Though the definition is thus rather simple, these systems reflect physical reversibility very well, and they are suited for investigating how computing systems can be realized in reversible physical environments. In this chapter, we argue reversibility in physics and computing, the significance of reversible computing, and the scope of this volume. Various models of reversible computing ranging from a microscopic level to a macroscopic one are dealt with from the viewpoint of the theory of automata and computing. Terminologies and notations on logic, mathematics, and formal languages used in this volume are also summarized.

Keywords reversibility in computing, reversibility in physics, reversible computing machine, reversible cellular automaton, reversible logic element

1.1 Reversibility in Physics and Computing

Reversibility is a notion that was argued originally in physics. It is known that microscopic physical laws are reversible in the sense that they are invariant under the time-reversal operation. For example, in classical mechanics, the same law holds for both positive and negative directions of time. It is also the case in quantum mechanics, where an evolution of a quantum system is described by a unitary operator. Since physical reversibility is thus one of the fundamental microscopic properties of Nature, it is an important problem to know how such a property can be effec-

K. Morita, *Theory of Reversible Computing*, Monographs in Theoretical Computer Science.
An EATCS Series, DOI 10.1007/978-4-431-56606-9_1

tively utilized in computing. This is because future computing devices will surely be implemented directly by physical phenomena in the nano-scale level.

Reversibility in computing is an analogous notion to physical reversibility, but its definition is rather simple. Reversible computing systems are systems for which each of their computational configurations has at most one predecessor. Hence, every computing process can be traced backward uniquely from the end to the start. In other words, they are backward deterministic systems. Though its definition is thus simple, we shall see later that these systems reflect physical reversibility very well.

Landauer [37] first argued the relation between reversibility in computing and reversibility in physics. He proposed *Landauer's principle* stating that any irreversible logical operation, such as erasure of a piece of information from memory, or a merge of two paths in a program, is associated with physical irreversibility in the macroscopic level, and hence it necessarily causes heat generation in the computing system. In particular, if one bit of information is completely erased in the system, at least $kT\ln 2$ of energy will be dissipated, where k is the Boltzmann constant, and T is the absolute temperature. If the computing system is reversible, then no such lower bound on energy consumption exists, and hence it leads to a possibility of dissipation-less computing system.

Today's computers are composed of electronic devices, and logical operations are realized by controlling the average behavior of a very large number of electrons. Thus, generally, they consume much more energy than $kT\ln 2$ per each primitive operation, even for reversible logical operations. They also require considerable amounts of energy for giving clock signals to the electronic circuits. Therefore, at present, $kT\ln 2$ is a negligible amount of energy in them. However, computing devices will be implemented in a much more microscopic level in the future. At that time, the lower bound $kT\ln 2$ will become critical. In addition, besides the problem of energy consumption, for the purpose of further miniaturization of computing systems, we have to find a way of directly utilizing microscopic physical phenomena for logical operations, where reversibility is one of the key features. Thus, investigating effective methods for using such properties of Nature in computing will give us a new insight into constructing future computers.

1.2 Significance of Reversible Computing

Reversible computing is a paradigm in which reversible computers are hierarchically constructed based on reversible physical phenomena and reversible operations. In the theory of reversible computing, there are several levels of computing models ranging from a microscopic level to a macroscopic one. In the bottom (i.e., microscopic) level, there are reversible physical models. Reversible logic elements are in the next level. Then, there are reversible logic circuits that work as functional modules in reversible systems. In the top level, there are various models of reversible computers. In this hierarchy, each system in a higher level can be constructed from systems in a lower level.

The objective of the study of reversible computing is to clarify how computation can be performed efficiently in reversible computers, how elegantly higher level reversible systems can be constructed from those in the lower level, and which kind of simple reversible primitives are universal and useful for constructing reversible computers. As we shall see in the following chapters, many reversible systems have high computing capability in spite of the strong constraint of reversibility. Furthermore, universal reversible computers can be composed of very simple reversible logic elements. In addition, some of the models are constructed in a very unique way, which cannot be seen in the traditional design theory of computing systems made of logic gates and memories.

A Turing machine (TM) is a standard model in the traditional theory of computing. A reversible Turing machine (RTM) is a backward deterministic TM, and is also useful in the theory of reversible computing. Lecerf first studied RTMs in the paper [40], where the halting problem and some other related decision problems on them were shown to be unsolvable like the case of general (i.e., irreversible) TMs. Later, Bennett [7] studied RTMs from the viewpoint of Landauer's principle. Note that it is easy to simulate an irreversible TM by an RTM by recording all the movements in a history tape. But, when the computation terminates, they are left as "garbage" information. Disposal of the garbage information is actually equivalent to erasure of the information, and hence it leads to energy dissipation. Bennett showed that it is possible to construct an RTM that simulates a given TM and leaves no garbage information on its tape when it halts (see Sect. 5.2.3). This result is important, because any computation can be performed in an efficient way with respect to energy consumption in an ideal situation.

After the work of Bennett, various reversible computing models ranging from microscopic to macroscopic ones have been proposed and investigated. In particular, reversible cellular automata, reversible logic elements and circuits, reversible physical models, and others were studied, and their relation to physical reversibility was argued.

A cellular automaton (CA) is a framework that can deal with spatiotemporal phenomena, and thus a reversible CA is an abstract model of a reversible spatial dynamical system. A CA is a system composed of an infinite number of identical finite automata called cells, which are placed and connected uniformly in a space. Hence, it is also suited for studying how complex phenomena appear from simple functions. Toffoli [63] investigated the relation between irreversible CAs and reversible CAs, and showed that an irreversible k-dimensional CA can be simulated by a $(k+1)$-dimensional reversible CA, and thus two-dimensional reversible CAs are computationally universal. Later, Margolus [41] proposed a very simple universal two-dimensional reversible CA. On the other hand, Morita and Harao [49] proved that reversible CAs can be universal even in the one-dimensional case. After that, it has been shown that there are various simple one- and two-dimensional universal reversible CAs (see Chaps. 11–13). Therefore, computational universality emerges even from a very primitive function of a cell.

Reversible logic elements are those whose operations are described by an injective (i.e., one-to-one) functions. An early study on reversible logic gates is found in

the paper of Petri [55]. Later, Toffoli [64, 65] studied them in the relation to physical reversibility. In particular, he proposed a universal reversible logic gate called a Toffoli gate. Then, Fredkin and Toffoli [25] introduced another universal reversible gate called a Fredkin gate, and showed that any logical function can be realized by a garbage-less circuit composed of it. On the other hand, Morita [45] proposed a reversible logic element with one-bit memory called a *rotary element* (RE) and showed any RTM can be concisely constructed by it in a very unique method (see Sect. 6.1). Thus, besides reversible gates, a reversible logic element with memory (RLEM), a kind of reversible sequential machine (RSM), is also useful in reversible computing. Note that today's computers are designed based on the well-known logic elements such as AND, OR, NOT, and some others. These operations, in particular, AND, OR and NOT, have been known since the era of ancient Greece (see e.g., [13]), and thus have a very long history. Since they were obtained from the analysis of thinking and reasoning processes performed by humans, it is easy for us to understand and use them. However, to investigate future computing systems, we should not be tied to old traditions, and we have to look for new bases and methods that are directly related to microscopic physical phenomena.

As for physical models of reversible computing, Fredkin and Toffoli [25] proposed an interesting model called the billiard ball model (BBM). It consists of idealized balls and reflectors. Logical operations are simulated by elastic collisions of moving balls. They showed any reversible logic circuit composed of Fredkin gates can be embedded in BBM. Of course, it can work only in an idealized situation, since it requires infinite precision on the sizes, positions and velocities of balls. However, it is a very insightful model for considering the relation between physical reversibility and computational reversibility.

As seen above, various interesting ideas have been proposed so far, and they opened new vistas in the theory of reversible computing. However, to understand it more deeply, we still have to find and develop new methodologies for it, which do not exist in the world of traditional computing. In order to do so, it is important not to consider reversibility as a "constraint", but to find a way of using it positively as a "useful property".

1.3 Scope of This Volume

In this volume, reversible computing is studied from the viewpoint of the theory of automata and computing. It mainly describes the results shown in the past studies by the author of this volume, his colleagues, and his former students. But, of course, other researchers' results related to them are also cited for completeness. Thus, it is not a very comprehensive book on reversible computing. Instead, it investigates various aspects of computational universality in reversible systems in detail.

This volume studies, in particular, the problems of how reversible machines can be designed, how computing can be carried out in a reversible machine, how simple universal reversible computers can be, which universal reversible logic elements

are useful for constructing reversible computers elegantly, and so on. The following chapters will give answers to these problems obtained so far. Although some of them may be improved in future research, they will provide good insights into reversible computing.

1.3.1 Organization of this book

The following chapters can be divided into three parts. The first part consists of Chaps. 2–4, in which reversible logic elements and circuits are studied. It also investigates the relation between reversible logic elements and a reversible physical model. The second part consists of Chaps. 5–9, in which reversible Turing machines (RTMs) and related models are studied. Thus, it deals with several reversible computing systems in the macroscopic level. But, it also argues how RTMs can be constructed out of reversible logic elements. The third part consists of Chaps. 10–14, in which reversible cellular automata (RCAs) are studied. Here, the problem of how complex phenomena, such as computational universality, appear from simple reversible operations is investigated. Hence, the framework of RCAs itself also connects between the microscopic and macroscopic levels. The details of each chapter are as follows.

In Chap. 2, a reversible logic element called a rotary element (RE) is given. Different from a reversible logic gate, it is defined as a two-state reversible sequential machine (RSM) with four input symbols and four output symbols. It is shown that any RSM can be compactly implemented by an RE. In this sense it is a universal logic element for constructing reversible machines. It is also shown that RE is simply realized in the billiard ball model (BBM), a kind of reversible physical model. In Chap. 3, two-state RLEMs are classified, and their universality is investigated. It is remarkable that *all* the non-degenerate RLEMs except only four two-symbol RLEMs are universal. In addition, three two-symbol RLEMs among four are proved to be non-universal. A systematic realization method of four-symbol RLEMs in the BBM is also shown. In Chap. 4, reversible logic gates and their circuits are studied. In particular, the Fredkin gate and its circuits are dealt with. Their basic properties, and their relation to RE are given. Furthermore, a method of constructing a garbage-less circuit out of Fredkin gates that simulates a given RSM is shown.

In Chap. 5, a reversible Turing machine (RTM) is defined, and its basic properties are shown. First, universality of a garbage-less RTM proved by Bennett [7] is explained. Then, simplification methods of RTMs, i.e., reducing the numbers of tapes, symbols, and states of an RTM, are shown. In Chap. 6, constructing methods of RTMs out of reversible logic elements are studied. It is shown that any RTM can be realized concisely as a circuit composed only of REs, or other two-state RLEMs. The methods are very different from the conventional ones that use logic gates and memory elements. In Chap. 7, universal RTMs (URTMs), which are RTMs that can simulate any TM, are studied. Here, it is investigated how we can obtain URTMs with small numbers of states and symbols. By simulating cyclic

tag systems, a kind of universal string rewriting systems proposed by Cook [14], several small URTMs are constructed. In Chap. 8, memory limited computing in RTMs is studied. In particular, it is argued how reversibility and determinism affect the computational power of space-bounded TMs. In Chap. 9, several models of reversible machines other than RTMs are studied. It is shown that a reversible counter machine with only two counters is computationally universal. Equivalence of a reversible two-way multi-head finite automaton and an irreversible one with the same number of heads is also proved.

In Chap. 10, reversible cellular automata (RCAs) are studied. Basic properties of RCAs and design methods are shown. Here, the framework of partitioned CAs (PCAs) is given for making it easy to design RCAs. In Chap. 11, universality of one-dimensional RCAs is investigated, and several RCAs that simulate RTMs and cyclic tag systems are constructed. In Chap. 12, two models of universal two-dimensional 16-state RCAs that can simulate Fredkin gates are given. A universal 81-state RCA in which any reversible two-counter machine can be simulated by their *finite* configurations is also shown. In Chap. 13, CAs on the triangular tessellation called the elementary triangular partitioned cellular automata (ETPCAs) are studied. There are 256 ETPCAs, and among them there are 36 reversible ETPCAs. In spite of the extreme simplicity of their local transition functions, they have very rich computing capabilities, and it is shown that ten reversible ETPCAs are computationally universal. In Chap. 14, self-reproduction in RCAs is studied. Self-reproducing CAs were first studied by von Neumann [51]. Later, Langton [39] proposed a simplified framework for self-reproducing CAs. Here, it is shown that self-reproduction of Langton's type is possible in two- and three-dimensional RCAs.

1.3.2 Related studies and references

Here, we give some remarks on related studies and references of reversible computing, though they are not exhaustive.

As described in Sect. 1.1, Landauer [37] investigated the relation between physical reversibility and logical reversibility. After that, several studies on reversible computing from the physical or thermodynamic viewpoint appeared [7, 8, 9, 11, 31]. Feynman also argued reversible computing from the standpoint of physics [21, 22], and his idea opened the way to quantum computing [19, 20]. Since evolution of a quantum computing system is described by a unitary operator, reversible computing is closely related to quantum computing (see, e.g., [27]). Various models of quantum computing, such as quantum Turing machines [6, 12, 17, 53], quantum finite automata [32], and quantum cellular automata [4, 5, 68], have been proposed. These quantum computing systems can be considered as generalizations of reversible ones. Both reversible computing and quantum computing are research fields where we are looking for microscopic phenomena that can be directly used as primitive operations for computing. DNA computing and molecular computing (e.g., [1, 57, 70, 71]) also have the same objectives on such a point.

So far, many kinds of reversible computing models were proposed and their properties were investigated. Bennett [10] studied time/space trade-offs in reversible Turing machines (RTMs), which is related to garbage generation and its reversible erasure in RTMs. On the other hand, Lange, McKenzie and Tapp [38] showed that any irreversible TM can be simulated by an RTM without increasing the memory space, though its computing time grows exponentially. Note that a simpler simulation method is given in Sect. 8.2.2. There are also studies on one-way reversible finite automata [3, 56], two-way reversible finite automata [32], reversible pushdown automata [33], one-way reversible multi-head finite automata [34], and so on. In Chap. 9, reversible counter machines, and two-way reversible multi-head finite automata will be investigated. The paper by Vitányi [67] gives a survey on reversible computing based on RTMs.

As for the study on cellular automata (CAs), there is a classical literature [51] on von Neumann's works that investigate construction-universality as well as computational universality of CAs. The reference [30] is a general survey on CAs, and [18, 52] are on universality of CAs. Reversible CAs (RCAs) have been studied since the early stage of their history [28, 44, 50, 59], but they were called *injective CAs* at that time. There are survey papers by Kari [29], Toffoli and Margolus [66], and Morita [46, 47, 48] on RCAs and related topics.

We now give an additional remark on reversibility in computing. It is known that microscopic physical laws are time-reversal-symmetric (see, e.g., [36]), and thus the same laws hold also for the negative direction of time. But, since reversible computing systems are simply defined as backward deterministic ones, the backward transition rules may not be the same as the forward ones. Gajardo, Kari and Moreira [26] introduced the notion of time-symmetry on CAs. It is an interesting property, and is much closer to physical reversibility. This notion is also defined for other machines by Kutrib and Worsch [35]. However, here we do not use this definition because of the following reason. In many cases, reversible computing machines are constructed hierarchically. Namely, reversible machines are implemented as a reversible logic circuit consisting of reversible logic elements, and the elements are realized in a reversible physical model. In this hierarchy, only the physical model is given as a system whose evolution is time-reversal-symmetric. Therefore, the whole computing system can be embedded in a time-reversal-symmetric system in the bottom level, even if it is not so in the higher level.

Since this volume investigates reversible systems from the standpoint of theory of computing, there are topics that are not dealt with. Among them, it is an important problem how a reversible computer can be realized in hardware. So far, there have been several interesting attempts such as implementation of reversible logic circuits as electrically controlled switches [42], c-MOS implementation of reversible logic gates and circuits [16], and adiabatic circuits for reversible computer [23, 24]. Also, recently, studies on synthesis of reversible and quantum logic circuits have been extensively done [2, 15, 43, 58, 60, 61, 62, 69]. However, ultimately, reversible logic elements and computing systems should be implemented at the atomic or molecular level. Although finding such solutions is very difficult, it is a challenging problem left for future investigations. Studies on more practical architectures for reversible

computers, and on reversible programming languages are also interesting subjects that are not dealt with in this volume. On reversible programming languages and software, see, e.g., [54, 72, 73]. In the research field of reversible computing, there will still be many interesting problems to study, and therefore unique ideas and novel methodologies are sought.

1.4 Terminology and Notations

In this section, we explain basic terminology and notations on logic, mathematics, and formal languages used in this book.

First, we give notations on logic. Here, P, P_1 and P_2 are propositions, x is a variable, and $P(x)$ is a predicate with a free variable x.

$\neg P$ Negation of P
$P_1 \vee P_2$ Disjunction (logical OR) of P_1 and P_2
$P_1 \wedge P_2$ Conjunction (logical AND) of P_1 and P_2
$P_1 \Rightarrow P_2$ P_1 implies P_2
$P_1 \Leftrightarrow P_2$ P_1 if and only if P_2
$\forall x(P(x))$ For all x, $P(x)$ holds
$\exists x(P(x))$ There exists x such that $P(x)$ holds

When describing logical functions of logic gates, and combinatorial logic circuits, operations of NOT (negation), logical OR, and logical AND are expressed by $\overline{a}$, $a+b$, and $a \cdot b$, respectively, instead of $\neg$, $\vee$, and $\wedge$. Exclusive OR (XOR) is denoted by $a \oplus b$. Here, a and b are Boolean variables with a value 0 (false) or 1 (true).

Notations and symbols on set theory are as follows, where S, S_1 and S_2 are sets, a is an object, x is a variable, and $P(x)$ is a predicate with a free variable x.

$\emptyset$ The empty set
$a \in S$ a is an element of S
$S_1 \subseteq S_2$ S_1 is a subset (not necessarily a proper subset) of S_2
$S_1 \subset S_2$ S_1 is a proper subset of S_2
$S_1 \cup S_2$ The union of S_1 and S_2
$S_1 \cap S_2$ The intersection of S_1 and S_2
$S_1 - S_2$ The difference of S_1 and S_2
$S_1 \times S_2$ The Cartesian product of S_1 and S_2 (note that $S \times S$ is denoted by S^2)
2^S The power set of S
$|S|$ The number of elements in S
$\mathbb{N}$ The set of all natural numbers (including 0)
$\mathbb{Z}$ The set of all integers
$\mathbb{Z}_+$ The set of all positive integers
$\mathbb{R}$ The set of all real numbers
$\{x \mid P(x)\}$ The set of all elements x that satisfy $P(x)$

A *singleton* is a set that has exactly one element.

Next, terminology on relations and mappings (functions) is given. Let S_1 and S_2 be sets. If $R \subseteq S_1 \times S_2$, then R is called a *(binary) relation*. Generally, let $S_1, \ldots, S_n$ be sets, and if $R \subseteq S_1 \times \cdots \times S_n$, then R is called an *n-ary relation*. For the case $R \subseteq S \times S\ (= S^2)$, we define R^n $(n \in \mathbb{N})$ recursively as follows: $R^0 = \{(x,x) \mid x \in S\}$, and $R^{i+1} = \{(x,y) \mid \exists z \in S\ ((x,z) \in R \wedge (z,y) \in R^i)\}$ $(i \in \mathbb{N})$. Then, R^* and R^+ are defined below.

R^* The *reflexive and transitive closure* of the relation R, i.e., $R^* = \bigcup_{i=0}^{\infty} R^i$
R^+ The *transitive closure* of the relation R, i.e., $R^+ = \bigcup_{i=1}^{\infty} R^i$

A relation $f \subseteq S_1 \times S_2$ is called a *partial mapping* (or *partial function*) from S_1 to S_2, if it satisfies

$$\forall x \in S_1\ \forall y_1, y_2 \in S_2\ (((x,y_1) \in f) \wedge ((x,y_2) \in f) \Rightarrow (y_1 = y_2)),$$

which means that for each $x \in S_1$ there exists at most one $y \in S_2$ such that $(x,y) \in f$. A partial mapping f is called a *mapping* (or *function*) from S_1 to S_2, if it further satisfies

$$\forall x \in S_1\ \exists y \in S_2\ ((x,y) \in f).$$

It is also called a *total mapping* (or *total function*). A partial mapping f from S_1 to S_2 is denoted by $f : S_1 \to S_2$, where the sets S_1 and S_2 are called the *domain* and the *codomain* of f, respectively. As usual, $(x,y) \in f$ is denoted by $y = f(x)$. Note that if $(x,y) \notin f$ for all $y \in S_2$, then $f(x)$ is undefined. The notation $x \mapsto f(x)$ indicates x maps to $f(x)$.

Let f and g be total mappings such that $f : A_1 \to B$, $g : A_2 \to B$, and $A_1 \subseteq A_2$. If $\forall x \in A_1 (g(x) = f(x))$ holds, then g is called an *extension* of f, and f is called a *restriction* of g to A_1, which is denoted by $g|_{A_1}$.

A partial mapping $f : S_1 \to S_2$ is called *injective* if

$$\forall x_1, x_2 \in S_1\ \forall y \in S_2\ ((f(x_1) = y) \wedge (f(x_2) = y) \Rightarrow (x_1 = x_2)).$$

A partial mapping $f : S_1 \to S_2$ is called *surjective* if

$$\forall y \in S_2\ \exists x \in S_1\ (f(x) = y).$$

A partial mapping $f : S_1 \to S_2$ that is both injective and surjective is called *bijective*. If a total mapping $f : S_1 \to S_2$ is injective (surjective, or bijective, respectively), then it is called an *injection* (*surjection*, or *bijection*).

Let $f : S_1 \to S_2$ be an injection. The *inverse partial mapping* of f is denoted by $f^{-1} : S_2 \to S_1$, and is defined as follows.

$$\forall x \in S_1\ \forall y \in S_2\ (f(x) = y \Leftrightarrow f^{-1}(y) = x)$$

Hence, $f^{-1}(f(x)) = x$ holds for all $x \in S_1$, and f^{-1} is an injective partial mapping. Note that, for $y_0 \in S_2$, if there is no $x \in S_1$ such that $f(x) = y_0$, then $f^{-1}(y_0)$ is

undefined. If f is a bijection, then f^{-1} is totally defined, and thus called the *inverse mapping* of f, which is also a bijection.

Notations on formal languages are given below. A nonempty finite set of symbols is called an *alphabet*. Let Σ be an alphabet. A finite sequence of symbols $a_1 \cdots a_n$ ($n \in \mathbb{N}$) taken from Σ is called a *string* (or *word*) over the alphabet Σ. The *concatenation* of strings w_1 and w_2 is denoted by $w_1 \cdot w_2$ (usually $\cdot$ is omitted). The length of a string w is denoted by $|w|$. Hence, if $w = a_1 \cdots a_n$, then $|w| = n$. We denote the *empty string* (i.e., the string of length 0) by λ. The *reversal* of a string w is denoted by w^{R}. Thus, if $w = a_1 \cdots a_n$, then $w^{\mathrm{R}} = a_n \cdots a_1$. For a symbol a, we use a^n to denote the string consisting of n repetitions of a ($n \in \mathbb{N}$). We define the set of strings Σ^n ($n \in \mathbb{N}$) recursively as follows: $\Sigma^0 = \{\lambda\}$, and $\Sigma^{i+1} = \{aw \mid a \in \Sigma \ \wedge \ w \in \Sigma^i\}$ ($i \in \mathbb{N}$). Then, Σ^* and Σ^+ are defined below.

Σ^* The set of all strings over Σ including λ, i.e., $\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$
Σ^+ The set of all strings over Σ of positive length, i.e., $\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i$

Let Σ_1 and Σ_2 be alphabets. A *string homomorphism* is a mapping $\varphi : \Sigma_1^* \to \Sigma_2^*$ that satisfy the following: $\varphi(\lambda) = \lambda$, $\varphi(a) \in \Sigma_2^*$ for all $a \in \Sigma_1$, and $\varphi(aw) = \varphi(a)\varphi(w)$ for all $a \in \Sigma_1$ and $w \in \Sigma_1^*$.

A subset of Σ^* is called a *(formal) language* over the alphabet Σ. Let L, L_1 and L_2 be languages over Σ. The *concatenation* of L_1 and L_2 is defined by $L_1 \cdot L_2 = \{w_1 w_2 \mid w_1 \in L_1 \ \wedge \ w_2 \in L_2\}$. We define the language L^n recursively in a similar manner as in Σ^n: $L^0 = \{\lambda\}$, and $L^{i+1} = L \cdot L^i$ ($i \in \mathbb{N}$). Then, L^* and L^+ are as follows: $L^* = \bigcup_{i=0}^{\infty} L^i$, and $L^+ = \bigcup_{i=1}^{\infty} L^i$.

In the later chapters, we define some automata as acceptors of languages to investigate their capability. For this purpose, we use the notation $\mathscr{L}(\mathscr{A})$ to denote the *class of languages* accepted by the class of automata $\mathscr{A}$, i.e., $\mathscr{L}(\mathscr{A}) = \{L \mid L \text{ is accepted by some } A \in \mathscr{A}\}$ (see, e.g., Sect. 8.1.4).

References

1. Adleman, L.M.: Molecular computation of solutions to combinatorial problems. Science **266**, 1021–1024 (1994). doi:10.1126/science.7973651
2. Al-Rabadi, A.N.: Reversible Logic Synthesis. Springer (2004). doi:10.1007/978-3-642-18853-4
3. Angluin, D.: Inference of reversible languages. J. ACM **29**, 741–765 (1982). doi:10.1145/322326.322334
4. Arrighi, P., Grattage, J.: Intrinsically universal n-dimensional quantum cellular automata. J. Comput. Syst. Sci. **78**, 1883–1898 (2012). doi:10.1016/j.jcss.2011.12.008
5. Arrighi, P., Grattage, J.: Partitioned quantum cellular automata are intrinsically universal. Natural Computing **11**, 13–22 (2012). doi:10.1007/s11047-011-9277-6
6. Benioff, P.: The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. J. Statist. Phys. **22**, 563–591 (1980). doi:10.1007/BF01011339
7. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**, 525–532 (1973). doi:10.1147/rd.176.0525

8. Bennett, C.H.: The thermodynamics of computation — a review. Int. J. Theoret. Phys. **21**, 905–940 (1982). doi:10.1007/BF02084158
9. Bennett, C.H.: Notes on the history of reversible computation. IBM J. Res. Dev. **32**, 16–23 (1988). doi:10.1147/rd.321.0016
10. Bennett, C.H.: Time/space trade-offs for reversible computation. SIAM J. Comput. **18**, 766–776 (1989). doi:10.1137/0218053
11. Bennett, C.H., Landauer, R.: The fundamental physical limits of computation. Sci. Am. **253**, 38–46 (1985). doi:10.1038/scientificamerican0785-48
12. Bernstein, E., Vazirani, U.V.: Quantum complexity theory. SIAM J. Comput. **26**, 1411–1473 (1997). doi:10.1137/S0097539796300921
13. Bocheński, J.M.: Ancient Formal Logic. North-Holland, Amsterdam (1951)
14. Cook, M.: Universality in elementary cellular automata. Complex Syst. **15**, 1–40 (2004)
15. De Vos, A.: Reversible Computing: Fundamentals, Quantum Computing, and Applications. Wiley-VCH (2010). doi:10.1002/9783527633999
16. De Vos, A., Desoete, B., Adamski, A., Pietrzak, P., Sibinski, M., Widerski, T.: Design of reversible logic circuits by means of control gates. In: Proc. PATMOS 2000 (eds. D. Soudris, P. Pirsch, E. Barke), LNCS 1918, pp. 255–264 (2000). doi:10.1007/3-540-45373-3_27
17. Deutsch, D.: Quantum theory, the Church-Turing principle and the universal quantum computer. Proc. R. Soc. Lond. A **400**, 97–117 (1985). doi:10.1098/rspa.1985.0070
18. Durand-Lose, J.O.: Cellular automata, universality of. In: Encyclopedia of Complexity and Systems Science (eds. R.A. Meyers, et al.), pp. 901–913. Springer (2009). doi:10.1007/978-0-387-30440-3_59
19. Feynman, R.P.: Simulating physics with computers. Int. J. Theoret. Phys. **21**, 467–488 (1982). doi:10.1007/BF02650179
20. Feynman, R.P.: Quantum mechanical computers. Opt. News **11**, 11–46 (1985). doi:10.1364/ON.11.2.000011
21. Feynman, R.P.: Feynman lectures on computation (eds., A.J.G. Hey and R.W. Allen). Perseus Books, Reading, Massachusetts (1996)
22. Feynman, R.P.: The computing machines in the future (Nishina Memorial Lecture in Tokyo, 1985). In: Lect. Notes Phys. **746**, pp. 99–113 (2008). doi:10.1007/978-4-431-77056-5_6
23. Frank, M.P.: Reversibility for efficient computing. Ph.D. thesis, MIT (1999)
24. Frank, M.P., Vieri, C., Ammer, M.J., Love, N., Margolus, N.H., Knight, T.E.: A scalable reversible computer in silicon. In: Unconventional Models of Computation (eds. C.S. Calude, J. Casti and M.J. Dinneen), pp. 183–200. Springer (1998)
25. Fredkin, E., Toffoli, T.: Conservative logic. Int. J. Theoret. Phys. **21**, 219–253 (1982). doi:10.1007/BF01857727
26. Gajardo, A., Kari, J., Moreira, M.: On time-symmetry in cellular automata. J. Comput. Syst. Sci. **78**, 1115–1126 (2012). doi:10.1016/j.jcss.2012.01.006
27. Gruska, J.: Quantum Computing. McGraw-Hill, London (1999)
28. Hedlund, G.A.: Endomorphisms and automorphisms of the shift dynamical system. Math. Syst. Theory **3**, 320–375 (1969). doi:10.1007/BF01691062
29. Kari, J.: Reversible cellular automata. In: Proc. DLT 2005 (eds. C. de Felice, A. Restivo), LNCS 3572, pp. 57–68 (2005). doi:10.1007/11505877_5
30. Kari, J.: Theory of cellular automata: a survey. Theoret. Comput. Sci. **334**, 3–33 (2005). doi:10.1016/j.tcs.2004.11.021
31. Keyes, R.W., Landauer, R.: Minimal energy dissipation in logic. IBM J. Res. Dev. **14**, 152–157 (1970). doi:10.1147/rd.142.0152
32. Kondacs, A., Watrous, J.: On the power of quantum finite state automata. In: Proc. 36th FOCS, pp. 66–75. IEEE (1997). doi:10.1109/SFCS.1997.646094
33. Kutrib, M., Malcher, A.: Reversible pushdown automata. J. Comput. Syst. Sci. **78**, 1814–1827 (2012). doi:10.1016/j.jcss.2011.12.004
34. Kutrib, M., Malcher, A.: One-way reversible multi-head finite automata. In: Proc. RC 2012 (eds. R. Glück, T. Yokoyama), LNCS 7581, pp. 14–28 (2013). doi:10.1007/978-3-642-36315-3_2

35. Kutrib, M., Worsch, T.: Time-symmetric machines. In: Proc. RC 2013 (eds. G.W. Dueck, D.M. Miller), LNCS 7948, pp. 168–181 (2013). doi:10.1007/978-3-642-38986-3_14
36. Lamb, J., Roberts, J.: Time-reversal symmetry in dynamical systems: A survey. Physica D **112**, 1–39 (1998). doi:10.1016/S0167-2789(97)00199-1
37. Landauer, R.: Irreversibility and heat generation in the computing process. IBM J. Res. Dev. **5**, 183–191 (1961). doi:10.1147/rd.53.0183
38. Lange, K.J., McKenzie, P., Tapp, A.: Reversible space equals deterministic space. J. Comput. Syst. Sci. **60**, 354–367 (2000). doi:10.1006/jcss.1999.1672
39. Langton, C.G.: Self-reproduction in cellular automata. Physica D **10**, 135–144 (1984). doi:10.1016/0167-2789(84)90256-2
40. Lecerf, Y.: Machines de Turing réversibles — récursive insolubilité en $n \in \mathbf{N}$ de l'équation $u = \theta^n u$, où θ est un isomorphisme de codes. Comptes Rendus Hebdomadaires des Séances de L'académie des Sciences **257**, 2597–2600 (1963)
41. Margolus, N.: Physics-like model of computation. Physica D **10**, 81–95 (1984). doi:10.1016/0167-2789(84)90252-5
42. Merkle, R.C.: Reversible electronic logic using switches. Nanotechnology **4**, 20–41 (1993). doi:10.1088/0957-4484/4/1/002
43. Miller, D.M., Maslov, D., Dueck, G.W.: A transformation based algorithm for reversible logic synthesis. In: Proc. Design Automation Conference, pp. 318–323 (2003). doi:10.1109/DAC.2003.1219016
44. Moore, E.F.: Machine models of self-reproduction. Proc. Symposia in Applied Mathematics, Am. Math. Soc. **14**, 17–33 (1962). doi:10.1090/psapm/014/9961
45. Morita, K.: A simple reversible logic element and cellular automata for reversible computing. In: Proc. MCU 2001 (eds. M. Margenstern, Y. Rogozhin), LNCS 2055, pp. 102–113 (2001). doi:10.1007/3-540-45132-3_6
46. Morita, K.: Reversible computing and cellular automata — A survey. Theoret. Comput. Sci. **395**, 101–131 (2008). doi:10.1016/j.tcs.2008.01.041
47. Morita, K.: Computation in reversible cellular automata. Int. J. of General Systems **41**, 569–581 (2012). doi:10.1080/03081079.2012.695897
48. Morita, K.: Reversible cellular automata. In: Handbook of Natural Computing (eds. G. Rozenberg, T. Bäck, J.N. Kok), pp. 231–257. Springer (2012). doi:10.1007/978-3-540-92910-9_7
49. Morita, K., Harao, M.: Computation universality of one-dimensional reversible (injective) cellular automata. Trans. IEICE Japan **E72**, 758–762 (1989)
50. Myhill, J.: The converse of Moore's Garden-of-Eden theorem. Proc. Am. Math. Soc. **14**, 658–686 (1963). doi:10.2307/2034301
51. von Neumann, J.: Theory of Self-reproducing Automata (ed. A.W. Burks). The University of Illinois Press, Urbana (1966)
52. Ollinger, N.: Universalities in cellular automata. In: Handbook of Natural Computing, pp. 189–229. Springer (2012). doi:10.1007/978-3-540-92910-9_6
53. Peres, A.: Reversible logic and quantum computers. Phys. Rev. A **32**, 3266–3276 (1985). doi:10.1103/PhysRevA.32.3266
54. Perumalla, K.S.: Introduction to Reversible Computing. CRC Press (2014)
55. Petri, C.A.: Grundsätzliches zur Beschreibung diskreter Prozesse. In: Proc. 3rd Colloquium über Automatentheorie (eds. W. Händler, E. Peschl, H. Unger), Birkhäuser Verlag, pp. 121–140 (1967). doi:10.1007/978-3-0348-5879-3_10
56. Pin, J.E.: On reversible automata. In: Proc. LATIN '92 (ed. I. Simon), LNCS 583, pp. 401–416 (1992). doi:10.1007/BFb0023844
57. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing. Springer (1998). doi:10.1007/978-3-662-03563-4
58. Rice, J.E.: An introduction to reversible latches. The Computer J. **51**, 700–709 (2008). doi:10.1093/comjnl/bxm116
59. Richardson, D.: Tessellations with local transformations. J. Comput. Syst. Sci. **6**, 373–388 (1972). doi:10.1016/S0022-0000(72)80009-6
60. Saeedi, M., Markov, I.L.: Synthesis and optimization of reversible circuits - a survey. ACM Comput. Surv. **45**, 21 (2013). doi:10.1145/2431211.2431220

61. Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Synthesis of reversible logic circuits. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems **22**, 710–722 (2003). doi:10.1109/TCAD.2003.811448
62. Thapliyal, H., Ranganathan, N.: Design of reversible sequential circuits optimizing quantum cost, delay, and garbage outputs. ACM Journal on Emerging Technologies in Computing Systems **6**, 14:1–14:31 (2010). doi:10.1145/1877745.1877748
63. Toffoli, T.: Computation and construction universality of reversible cellular automata. J. Comput. Syst. Sci. **15**, 213–231 (1977). doi:10.1016/S0022-0000(77)80007-X
64. Toffoli, T.: Reversible computing. In: Automata, Languages and Programming (eds. J.W. de Bakker, J. van Leeuwen), LNCS 85, pp. 632–644 (1980). doi:10.1007/3-540-10003-2_104
65. Toffoli, T.: Bicontinuous extensions of invertible combinatorial functions. Math. Syst. Theory **14**, 12–23 (1981). doi:10.1007/BF01752388
66. Toffoli, T., Margolus, N.: Invertible cellular automata: a review. Physica D **45**, 229–253 (1990). doi:10.1016/0167-2789(90)90185-R
67. Vitányi, P.M.B.: Time, space, and energy in reversible computing. In: Conf. Computing Frontiers, pp. 435–444 (2005). doi:10.1145/1062261.1062335
68. Watrous, J.: On one-dimensional quantum cellular automata. In: Proc. FOCS, pp. 528–537 (1995). doi:10.1109/SFCS.1995.492583
69. Wille, R., Drechsler, R.: Towards a Design Flow for Reversible Logic. Springer (2010). doi:10.1007/978-90-481-9579-4
70. Winfree, E.: Self-healing tile sets. In: Nanotechnology: Science and Computation, pp. 55–78. Springer (2006). doi:10.1007/3-540-30296-4_4
71. Winfree, E., Liu, F., Wenzler, L.A., Seeman, N.C.: Design and self-assembly of two-dimensional DNA crystals. Nature **394**, 539–544 (1998). doi:10.1038/28998
72. Yokoyama, T.: Reversible computation and reversible programming languages. Electron. Notes in Theoret. Comput. Sci. **253**, 71–81 (2010). doi:10.1016/j.entcs.2010.02.007
73. Yokoyama, T., Axelsen, H.B., Glück, R.: Fundamentals of reversible flowchart languages. Theoret. Comput. Sci. (2015). doi:10.1016/j.tcs.2015.07.046

Chapter 2
Reversible Logic Elements with Memory

Abstract A reversible logic element with memory (RLEM) is presented as a logical primitive for constructing reversible computing systems. In the conventional design theory of logic circuits, logic gates, which are elements without memory, are mainly used as logical primitives. On the other hand, in the case of reversible computing, RLEMs are also useful as logical primitives. This is because reversible machines can be constructed easily and concisely by a simple RLEM. In addition, the design method using RLEMs is very different from those in the traditional theory of logic circuits based on logic gates. Thus RLEMs give new vistas and suggest various possibilities in the theory of reversible computing. Here, we present a typical 2-state 4-symbol RLEM called a rotary element (RE), and investigate how it is realized in the billiard ball model, a reversible physical model of computing, and how reversible sequential machines can be constructed from it.

Keywords reversible logic element with memory, rotary element, reversible sequential machine, billiard ball model

2.1 Logical Primitives for Reversible Computers

A *logic element* is a primitive for composing logic circuits by which computing systems are implemented. There are two types of logic elements: one without memory, which is usually called a *logic gate*, and one with memory. In the traditional design theory of logic circuits, logic gates such as AND, OR, NOT, NAND, and others are used as primitives, and thus logic gates are dealt with separately from memory elements such as flip-flops. In the theory of reversible logic circuits, also, logic gates have been treated as the main primitives for circuit design, and there has been much research on them. However, as we shall see in the following sections and chapters, a *reversible logic element with memory* (RLEM) is also useful for composing various models of reversible computing systems. In particular, we shall see that even very simple RLEMs have universality, and we can compose reversible se-

K. Morita, *Theory of Reversible Computing*, Monographs in Theoretical Computer Science.
An EATCS Series, DOI 10.1007/978-4-431-56606-9_2

quential machines, reversible Turing machines, and others rather easily from them. In addition, these machines are constructed in a very different manner from those in the traditional design theory of logic circuits. Since RLEMs thus give new vistas and suggest various possibilities in the theory of reversible computing, we mainly discuss RLEMs in this book.

A reversible logic element is one whose operation is described by an injection. For example, a NOT gate is reversible, since it realizes the injective logical function $f_{\mathrm{NOT}} : \{0,1\} \to \{0,1\}$ where $f_{\mathrm{NOT}}(0) = 1$ and $f_{\mathrm{NOT}}(1) = 0$. On the other hand, NAND is irreversible, since it realizes the non-injective logical function $f_{\mathrm{NAND}} : \{0,1\}^2 \to \{0,1\}$ where $f_{\mathrm{NAND}}(0,0) = 1$, $f_{\mathrm{NAND}}(0,1) = 1$, $f_{\mathrm{NAND}}(1,0) = 1$, and $f_{\mathrm{NAND}}(1,1) = 0$. It is, of course, possible to implement reversible computing models, such as reversible Turing machines, by irreversible logic elements like NAND gates and flip-flops. But, such an implementation is almost meaningless because of the following reason. One of the objectives of the study of reversible computing is to find an efficient method of implementing reversible machines by reversible logic elements, and to find an elegant way of realizing reversible logic elements by physically reversible phenomena. Hence, our final goal is to realize a reversible computer in a reversible physical system in an efficient way. Therefore, the important point of the study of reversible logic elements is to find a key element that lies in the intermediate level between the levels of abstract machines and physical systems. Namely, the problem is which reversible logic element is useful for constructing reversible machines, and is easily realizable in a reversible physical system. We investigate this problem based mainly on RLEMs in this book. In fact, there are several interesting RLEMs from which reversible machines are constructed very simply. We shall also see that they are concisely implemented in the billiard ball model, a kind of a reversible physical model of computing.

In this chapter, we introduce a particular reversible logic element with 1-bit memory called a *rotary element* (RE) [5], since its operation is very easily understood. We shall see that the RE can be concisely realized in the billiard ball model, and it is powerful enough for constructing reversible sequential machines.

2.2 Reversible Logic Element with Memory (RLEM)

We first define a sequential machine (SM) and its reversible version, since a reversible logic element with memory (RLEM) is a kind of reversible sequential machine (RSM). An SM given here is a finite automaton with an output port as well as an input port, which is often called an SM of Mealy type.

Definition 2.1. A *sequential machine* (SM) is a system defined by $M = (Q, \Sigma, \Gamma, \delta)$, where Q is a finite set of internal states, Σ and Γ are finite sets of input and output symbols, and $\delta : Q \times \Sigma \to Q \times \Gamma$ is a move function. If δ is injective, M is called a *reversible sequential machine* (RSM). Note that if M is reversible, then $|\Sigma| \leq |\Gamma|$ must hold. It is also called a $|Q|$-state $|\Gamma|$-symbol RSM.