Chapter 3
Artificial Chemistry and Molecular Network

Hideaki Suzuki

Abstract This chapter focuses on artificial chemistry, a research approach for constructing life-like systems in artificial environments, and presents its fundamental concepts and system design requirements. Based on this discussion, we move on to evaluate typical artificial chemistry systems: We propose 13 conditions necessary for emergent evolution and three topological conditions that must be met by the rules for transport of symbols. We also introduce the concept of a molecular network, which emulates the spatial relationship of the molecules. With hard-sphere random-walk simulations, we show seven topological conditions that the molecular network must satisfy. In the latter half of this chapter, we feature the example of network artificial chemistry (NAC), in which a molecular network is used for spatial representation, and present some of the research results derived since this was first proposed. We begin by introducing a model wherein a cluster formed by folding a node chain functions as an active machine. We then overview studies on rewiring rules for weak edges, which form the basis for network dynamics. We discuss the merits and demerits of the method expressing spatial constraints derived from the network energy, then introduce models devised to circumvent the difficulties: a model that implements active functions (programs) in the nodes and an improved model that implements the programs in agents to allow them to move within the network. The improved model, called "program-flow computing," is expected to be refined into a new computing model.

Keywords Artificial chemistry · Chemical reaction · Molecular interaction · Self-organization

H. Suzuki (✉)
National Institute of Information and Communications Technology, Kobe, Japan
e-mail: hsuzuki@nict.go.jp

H. Sawai (ed.), *Biological Functions for Information and Communication Technologies,* Studies in Computational Intelligence, 320,
DOI: 10.1007/978-3-642-15102-6_3, © Springer-Verlag Berlin Heidelberg 2011

3.1 Introduction

Each of the cells that make up our body is a gigantic machine in which several hundreds of millions of molecules (excluding water) interact in exquisite fashion in diverse biological processes, including genetic and metabolic activities. These machines are highly autonomous, adaptable, robust, reliable, and self-repairing, traits artificial machines have yet to approach. In recent years, researchers across a wide range of fields have pursued studies combining biology, chemistry, physics, and informatics to explore the design principles and mechanisms of the cell. Ambitious attempts in this area based on informatics include systems biology, which constructs detailed models of an entire living cell and performs simulations consuming vast computational resources [6, 27, 28, 36]. Nevertheless, even if we could acquire infinite computational resources and copy all the events and processes that occur in a living cell to computer code, then run simulations, would we really understand the "design principle" of a cell? Would we understand the diverse useful properties of the cells given above?

At the Bio-algorithms Project (http://www-karc.nict.go.jp/BA/index_J.html) of the Kobe Advanced ICT Research Center, the National Institute of Information and Communications Technology (NICT), our approach to research emphasizes a "bottom-up approach" (a constructive approach) (Fig. 3.1). This approach diverges significantly from the techniques used to create the detailed model described above. The goal is to build a computational and communication device with the properties of a living creature. We investigate the details of cell activity at the molecular level and build models of information processing and learning. The

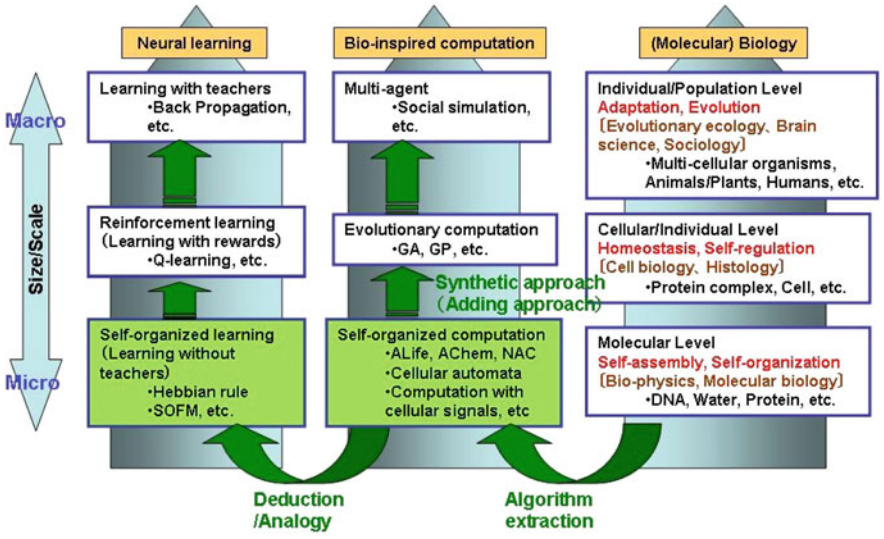

Fig. 3.1 Conceptual diagram of the bottom-up approach

resulting model may be less fine-grained than a comprehensive model that describes every physical and chemical property of the molecules in question (at exorbitant computational cost), and yet is required to be detailed enough to reflect molecular logic and to exhibit desirable properties of bio-molecules such as self-assembly and self-organization. By constructing such models and assessing their suitability, we seek to better understand the essence of biological properties at the molecular level and perhaps to propose new (non-von Neumann) information processing and communication models.

In order to present one example of this constructive approach, this chapter focuses on a research field known as artificial chemistry and provides a brief overview. After describing the basic concepts and requirements that must be met by artificial chemistry designs, we focus on NAC, which describes intermolecular interactions with mathematical graphs, and introduce the history of research in this area. In Sect. 3.7, we discuss future prospects for this area of study.

3.2 Artificial Chemistry

3.2.1 Basic Elements of Design in Artificial Chemistry

Artificial life is among several historical efforts to capture the essence of life via a constructive approach [3]. In brief, artificial life seeks to introduce the principles of biological evolution into computing or other artificial systems and to perform simulations that shed light on the nature of life using a constructive approach, observing life-like behaviors that emerge in the simulations and applying them to design engineering systems. Artificial chemistry has recently emerged as a subfield whose goal is to model the conditions under which bio-chemical reactions and chemical evolution occurred on the primeval Earth, eventually leading to emergent phenomena such as cell evolution and genetic encoding [9, 11, 14, 74]. In the design of self-organizing computational systems, the ideal model would be a chemical reaction system, confirmed to undergo self-organization or self-assembly and to produce complex structures and functions, composed of elemental processes whose mechanisms can be identified down to the level of quantum mechanics. Such a system could be used to design or evaluate models more accurate and reliable than the simplified systems (i.e., conventional artificial life) available now, which rely so heavily on human intuition.

In 2003, Suzuki et al. considered the environment required for such a system of artificial chemistry, distilling the essential characteristics to the five factors listed in Fig. 3.2 [61].

Here, of course, the elementary symbols are compared to bio-molecules in the narrow sense. In more generic artificial chemistry systems, the symbols might be compared to particles in the broad sense, including bases comprising DNA and RNA and other atomic clusters. The rules for reaction and transport can also be broadly

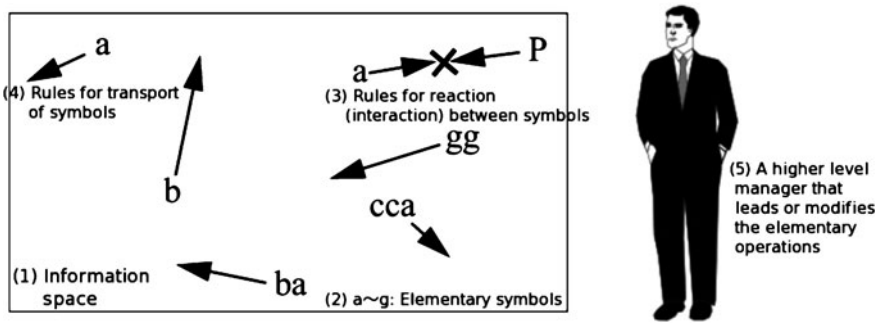

Fig. 3.2 Elements comprising the artificial chemistry framework. Elementary symbols such as *a* and *b* move and react with each other in a "rectangular" information space according to the rules

defined. For example, cellular automata [29, 30, 52, 53], a representative model for artificial life, may also be regarded as an artificial chemistry system in a broader sense. In this case, elementary symbols are the internal states of the automata, the information space is the cell space, and the rules governing transport and reactions are defined by the transition rules for the automata. In another example, Tierra [50, 51], a well-known core memory-based artificial life system, elementary symbols are machine instructions, the information space is the core memory space, rules governing transport and reactions are the computational operations defined for the machine instructions, and the higher-level manager is represented by external programs that cause mutations and kill individual programs. We also note the need to minimize or eliminate any interventions of a higher-level manager in artificial life/artificial chemistry models in which we take a bottom-up approach, allowing elements to react with each other as freely as possible.

3.2.2 Requirements for Artificial Chemistry System's Design: From the Perspective of Evolution and Emergence

Artificial chemistry is a research approach that emulates chemical evolution on the primeval Earth and generates emergent phenomena. The scenario for chemical evolution [33] currently posited in the life sciences states that primitive life emerged through several major events that happened in the ancient sea: the creation of polymers such as amino acids and proteins, the establishment of metabolic reactions, membrane formation, the creation of genetic materials such as RNA or DNA, and the establishment of a scheme for the translation of genetic information. It exceeds the author's capabilities and the scope of this chapter to reproduce or verify this scenario. Nevertheless, to design a better artificial chemistry system, we must set the five above-mentioned elements appropriately while considering chemical evolution.

In order to meet these requirements, in this section, we enumerate the qualitative conditions needed to generate the major events in the chemical evolution from the informatics perspective. Since the actual scenario of chemical evolution on Earth has yet to be revealed, the conditions specified in this section cannot be sufficient conditions (sufficient in the sense that emergent evolution will occur each time the conditions are met). Still, they likely constitute, at minimum, necessary conditions, or conditions without which emergent evolution cannot occur. We use the conditions obtained here to evaluate later various artificial chemistry system designs (Sect. 3.4).

Of the 13 conditions for emergence described below, Conditions (1)–(4) involve symbols and their reaction rules. Conditions (5) and (6) involve spatial structures. Conditions (7)–(11) involve rules of transport for symbols and compartments. Of these, Conditions (7)–(9) involve passive transport and conversion, which increase entropy, while Conditions (10) and (11) involve active transport, which decreases entropy. Finally, Conditions (12) and (13) are related to the gigantic symbol complex.

Emergence Condition (1): The total amount of symbols must be conserved as a resource.

Resources for the evolution of life fall into three general categories: energy, space, and matter. Of these, energy is a consumable resource, while space and matter are recyclable. A near-inexhaustible stream of energy is provided by the sun from outside the ecosystem; the ecosystem is a mere consumer of this resource. On the other hand, the total amount of space and matter are limited, and living creatures recycle both constantly. In particular, the total amount of matter (atoms) is all but perfectly conserved in chemical reactions, the elemental processes of biological activity, which makes it possible to recycle matter. What would happen if matter was not conserved and could be created or annihilated in the elemental processes of chemical reactions?

In order to answer this question, here we consider Tierra, the system of artificial life described earlier. In Tierra, machine instructions constitute the matter (symbol) resource. Tierra can allocate instructions freely to other locations in memory (space) by copying or by other operations. As a result, when Tierra operates, the memory quickly fills with instructions (matter), and the system is no longer able to function. In order to prevent this, Tierra applies a rule that results in the death of individuals—in other words, regular memory deallocation—through higher-level routines. Ideally, as this example indicates, the total amount of resources should be conserved for the evolution system to continue operating.

Emergence Condition (2): The system must be able to store and replicate an unlimited amount of genetic information.

In 1997, Szathmáry and Maynard-Smith [80] classified the information appearing, transmitted, and replicated in the initial stages of chemical evolution into three categories: Fixed Information, Compositional Information, and Programed Information. Fixed Information refers to information like molecular orientation, stored and replicated in processes like crystal growth. Compositional

Information refers to compositional distributions (which define cell types) of molecules in cells replicated in primitive cell division. Programed Information refers to the digital information encoded in the base sequences of the so-called informational macromolecules, including RNA and DNA.

Of course, the process of evolution generates a complex of individuals and diverse populations from mutations generated as characteristics are passed from parents to offspring. Evolution cannot occur if the information inherited from the parent to the offspring in this genetic mechanism cannot change or is significantly constrained. A system of artificial evolution also requires the presence and replication of unlimited information in the same manner as Programed Information in biological evolution.

Emergence Condition (3): An activator symbol (group) must exist that allows modification of symbols with specificity.

In addition to conditions governing genetic information, a cell, as a phenotype, requires another important function: a catalytic function with specificity. In a living cell, tens of thousands of different types of chemical reactions progress simultaneously in a single solution, and these reactions must occur in a time-specific manner at the required locations or the cell cannot function as an overall entity. Living creatures achieve this through enzymes (proteins), which are catalysts with specificity. We can also say that the set of these enzymes specifies the cell phenotype. An artificial evolution system must also provide such functions through its symbols or symbol complexes.

Emergence Condition (4): A translator from the genotype to the phenotype must exist, and the translation relationship must be expressed as a phenotype.

When we consider the above two conditions—the unlimited amount of genetic information and the activator symbol—from the evolutionary perspective, it becomes clear that we need some mechanism that relates the information to the activator. Living creatures cannot evolve without this mechanism, which maps mutational changes in genetic information to modifications in biological functions.

This relationship (normally called the translation relationship) plays an important role not just in living creatures, but in self-reproducing automata [84], explored by John von Neumann in his later years. Column 1 below describes von Neumann's self-reproducing automata in detail. One conclusion derived from this self-reproducing automata is this: If the translation relationship is not a fixed relationship prepared externally but established by the automata themselves as phenotypes, the translation relationship itself changes and is optimized during the process of evolution, resulting in an evolutionary system capable of even greater adaptability. In living creatures, this translation relationship is achieved by a set of molecules called transfer ribonucleic acid (aminoacyl tRNA) and by ribosomes. This protein family is yet another phenotype created from DNA and optimized through the process of evolution [32]. As in this biological translation system, an artificial system must have a mechanism for translation from the genotype to the phenotype, the mechanism itself being implemented as a phenotype.

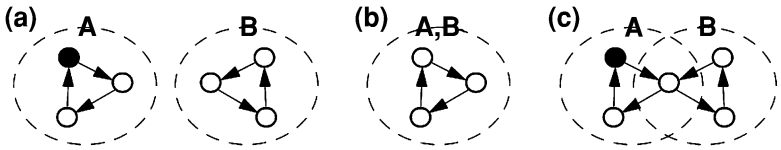

Fig. 3.3 Relationship between replicator species groups. **a** Groups A and B share no molecular species. **b** Groups A and B consist of identical molecular species. **c** Groups A and B share some, but not all molecular species. The *open* and *filled circles* are molecular species. The *arrows* indicate relationships whereby the molecular species at the tail of the arrow promotes the replication of the molecular species at the head of the arrow. The *filled circles* in **a** and **c** indicate molecular species with slightly increased rates of replication

Emergence Condition (5): Walls must be able to form and maintain compartment structures.

Up to the present, we have used the term *cell* without a clear definition. Here, we clarify the meaning of a cell or compartment. Compartmentalization and its grouping of the symbols have roughly two meanings.

First, consider a group of replicators floating in the sea—a solvent. Generally speaking, molecular replication is achieved through the collaborative action of multiple molecular species, not just one. Figure 3.3 shows the relationships between two molecular species groups, A and B. Assume here that Group A is slightly more efficient than Group B under a compartmentalized or non-compartmentalized environment. If Groups A and B have no relationship to each other or are identical, as in Fig. 3.3a, b, the presence or the absence of compartments has no effect on the subsequent relationship between Groups A and B. In Case (a), Group A will increase reproduction efficiency independent of Group B, with or without compartments. In Case (b), Groups A and B generate no differences. The problem arises in Case (c), when Groups A and B share some but not all molecular species. In this case, if the molecular species indicated by the filled circles improves the efficiency of Group A slightly, the molecular species common to Groups A and B are replicated more efficiently. Consequently, the replication efficiency of Group B increases slightly, whereas the increase in the replication efficiency of Group A declines slightly. On the other hand, if Groups A and B are enclosed in separate compartments, Group A can increase its efficiency without interference from Group B; as a result the efficiency in the compartment containing Group A increases dramatically compared to Case (a). As this example shows, compartmentalization can lead to large differences between molecular species groups and promote selection among them, which is a significant advantage in the unfolding of chemical evolution.

Another advantage of compartmentalization is observed in the situation in which the proliferation of replicators or cells has progressed to such an extent that the space is occupied by replicators or cells and resource molecules have begun to run short. In order for chemical evolution to continue from this point, the groups of molecular species must acquire the ability to catabolize (decompose) other molecular species into smaller molecules. Under these conditions, the presence of

compartments affects the outcome of the evolution of molecular species; if the system lacks compartments and all molecular species groups coexist in the same solution field, a particular replicator species group decomposes the molecules produced by its own proliferation (its children, so to speak). Thus, the resulting total proliferation rate does not readily increase. If the replicator species groups are enclosed in compartments, on the other hand, decomposition (predation) may begin after the molecules formed by their proliferation are completely separated by compartment division. This separation of the target of proliferation and decomposition by compartmentalization can increase proliferation rates and confer another advantage in chemical evolution. Emergence Condition (7), discussed later, guarantees to a certain extent that the compartment targeted for predation will differ from the compartment divided.

Emergence Condition (6): It must be possible for symbol operations—or at least the replication of the compartment—to change the effective size of a compartment.

This condition is based on the observation that the effective size of a complex cell (compartment)—defined as the number of molecules (symbols) contained—is larger than a simple cell. Since the process of evolutionary emergence involves the creation of larger, more complex cells through several major events, the size of a compartment in the information space prepared in a system of artificial chemistry must also be structurally modifiable. This susceptibility to modification should be achieved at either of the two levels described below.

The first short-term susceptibility to modification is enabled by the condition that the number of symbols in a compartment can change without generational change. Various artificial chemistry spaces previously proposed can be classified as rigid (solid) or elastic (liquid) spaces. Another symbol can be freely inserted into two arbitrary symbols in an elastic space, but not in a rigid space. An example of an elastic space is a lattice space [58, 59], in which, the lattice points are not fixed and another lattice point can be inserted between two lattice points. Another is a lattice space in which the lattice points are fixed, but the number of symbols belonging to each lattice point can be changed freely [38–40]. For either approach, an elastic space can achieve short-term susceptibility to modification.

The second longer-term susceptibility to modification is enabled by the possibility of change in the effective size of a compartment when the compartment is replicated from parent to offspring. For example, in the case of Tierra, although the core memory space is rigid, the size of a compartment (creature) can change due to transcription errors during replication from the parent to the children, making the compartment size susceptible to modification.

Emergence Condition (7): The compartments must gradually mingle with each other over time due to the effects of randomization.

As is well-known, we eukaryotes have cells (compartment structures) with highly structured internal forms comprising organelles such as the nucleus and mitochondria and numerous liposomes (bags made of lipid bilayer membranes) involved in endocytosis (absorption), exocytosis (egestion), and the transport of

molecules floating in the cell. Cases requiring more speed rely on transport along microtubules (rails) by motor proteins, but most cases depend on the slower mode of diffusion. In primitive cells lacking active transport systems, such passive transport is responsible for transporting all substances. For an artificial chemistry system too, therefore, a similar environment allowing the passive transport of compartments must be present.

The mingling (passive transport) of the compartments is also crucial for the emergence of patterns of interaction among individuals, such as predation, parasitism, and symbiosis, which are basic ecosystem activities. When cells or individuals occupy the space after repeated division and proliferation, offspring tend to occupy the space near the parents. When we observe spatial distributions after proliferation, individuals closer in pedigree are positioned nearby, while those distant in pedigree are positioned at a distance. If this tendency continues for an extended period, predation (to take one example) becomes highly disadvantageous to gene survival, targeting as it does the individuals (close relatives) located nearby. In addition, parasitism and symbiosis would not readily occur, since these relationships involve different species. In reality, in marine- or land-based environments, diffusion and randomization caused by wave or wind action would result in a mingled distribution of individuals.

Emergence Condition (8): Signal transmission by symbol transport must be possible.

Another outcome of passive transport (mingling) is signal transmission via the passive transport of symbols. We know that the various schemes of intercellular and intracellular signal transmission in living creatures are based on molecular diffusion, including for example, the well-known gene regulatory network formed by molecular interactions between DNA and switching proteins such as enhancers and repressors. Despite the difficulties associated with the straightforward implementation of symbol diffusion, particularly in spaces such as a core, an artificial chemistry system must provide some means of passive transport of symbols to allow emergent interactions between symbols. Chapter 2 of this book considers this condition in greater detail from a slightly different perspective, from which signal transmission by molecular transport is called molecular communication.

Emergence Condition (9): Some randomizing operation must be able to change the symbols or allow new combinations of symbols to form.

The last condition required as an outcome of passive transport is the creation of new functions by randomization. Consider the metabolic reactions established in a living cell. The chemical reaction velocity theory (see Column 2) always specifies the rate of molecular participation in a reaction at any given point in time. This means that the remaining molecules remain as a reaction reservoir from which molecules can readily be recruited if another reaction becomes possible. Thus, in a bio-molecular system, randomization and super redundancy serve as the driving forces for generating new reactions and functions.

Condition (9) requires equivalent properties of an artificial chemistry system. In most cases, a function in artificial chemistry is expressed as a symbol or a combination (arrangement) of symbols. If the arrangement (order) of symbols is not randomized by diffusion or mutation and is fixed permanently, new functions cannot be tested in the system. An artificial chemistry system must therefore have a mechanism that modifies the symbols and combinations of symbols, through randomization.

Emergence Condition (10): It must be possible to transport a symbol (or a symbol set) selectively according to the rules of transport or through the actions of activator symbol(s).

A biological system is nothing more or less than a physical and chemical system, the dynamics of which are specified by the direction in which the free energy, $F = E - TS$, decreases (where E is energy, T is absolute temperature, and S is entropy). According to this equation, if T is sufficiently high, the system shifts only in directions that increase S. However, if T is low, the system may reduce S to reduce E to achieve a total reduction in F. In this manner, at a certain temperature, the system can form orders (decrease S) according to the free energy minimization criteria. This is the driving force of creation of orders in living creatures. From cell formation and division to cell digestion and egestion, most primitive functions and orders can be emulated by assuming appropriate energy models for the molecules and applying the minimization principle. Examples include the chemotons proposed by Gánti [16] and simulations of the formation and division of artificial compartments by Ono et al. [38, 40] The transport of molecules based on energy minimization as such can be regarded as a means of active transport that makes indirect use of energy.

On the other hand, the cells of most present-day higher organisms have evolved an active transport system that uses energy more directly through the molecules known as ATP. Recently, Hirokawa et al. [20] have shown that a living cell has a network of microtubules, a system of rails, along which motor proteins carry loads (molecules in vesicles, in most cases) to destinations at various speeds. This transport system allows the cell to transport specific molecules to specific destinations in a timely manner. Lacking this mechanism, cells within higher organisms could not maintain or divide themselves.

Based on the above facts, an artificial chemistry system must feature a direct or indirect active transport method. Here, active transport refers to the operation of moving a symbol (or a symbol set) from one point in space to another. Out of the active transport, indirect transport refers to transport to the predefined destination (based on rules for transport of the symbols prepared in advance), whereas direct transport refers to transport to a specific destination (performed by a specific activator symbol). In general, in direct active transport the distance between the symbol (or a symbol set) transported and the destination is observed to decrease after transport.

Emergence Condition (11): Selective transport across walls of symbols must be possible.

As discussed earlier, membranes partition cell interiors into layered compartments to form organelles (including the nucleus, the Golgi apparatus, and mitochondria) and the liposomes used to transport molecules. Unlike the walls that (once formed) separate the interior from the exterior, the walls of these compartments are not completely sealed. Rather, these walls feature a mechanism involving numerous membrane proteins on the membrane surface that sense molecules inside or outside the compartments and selectively move molecules across the membrane in one direction or the other. A compartment lacking such a mechanism is essentially an inanimate object and cannot contribute to the functions of a living cell. Similarly, an artificial chemistry system must provide a mechanism allowing the selective transport of symbols across walls to allow the exchange of information in and out of compartments.

Emergence Condition (12): Multiple activator symbols must be able to operate simultaneously on a single operand symbol.

The genetic information programed without limits as required in Condition (2) is generally encoded as a one-dimensional symbol sequence on a long, thin symbol complex. With an increase in the amount of encoded information, this sequence grows to extraordinary length, which makes it takes time for an activator molecule to traces out the long sequence.

These operations, of course, are implemented in biological systems by informational macromolecules such as DNA and RNA or proteins such as polymerase. In living creatures, we need to pay attention to the fact that more than one protein acts on a single DNA molecule at any given time, which prevents degradation of replication or transfer rates as sequences grow longer. The same mechanism also enables mutual switching of genetic transcription (for example, a repressor protein suppresses the transcription of genes by attaching to a DNA molecule and preventing the polymerase from passing).

Such cooperative or competitive interactions between activator molecules are essential in generating intricate functions based on a single piece of genetic information. Likewise, an artificial chemistry system must allow multiple activator symbols to act simultaneously on a given operand.

Emergence Condition (13): Symbol complexes must be able to move across space, combine, and form new symbol complexes.

An active function in a biological system is not always achieved by a single activator molecule (protein); a multimeric complex is a molecule containing two or more polypeptides (chains of amino acids constituting proteins) in a single unit. In an extreme case, complexes like ribosomes (molecules involved in protein synthesis) contain several dozen amino acid chains to generate complex functions.

These molecular complexes form through a process called self-assembly. The constituent molecules move through the solvent, encounter each other, and combine (coalescing rather than reacting) with matching physical and chemical

properties. Large protein complexes—ribosomes, for example—form from self-assembly occurring at multiple, hierarchical stages.

What properties are required for a solution molecular system to generate such hierarchical self-assembly? Coalescence, the elementary process of self-assembly, can be described quantitatively in the same way as a chemical reaction. We can apply chemical reaction velocity theory (refer to Column 2, discussed further below) to estimate the rate of coalescence. Theory states that rates of chemical reaction in a solution remain constant regardless of molecule size. Even as molecules grow in size, they take part in reactions with no loss of activity, a property that ensures self-assembly will occur at hierarchical stages.

The discussion so far points to the condition that must be met when we design an artificial environment to allow functions to emerge hierarchically through self-assembly. In general, in an artificial environment, the designed symbol complex becomes heavier as it grows, and concomitantly, tends to lose mobility and efficiency—something we need to prevent. In other words, large symbol complexes must be able to move around and combine with each other with no loss of mobility or efficiency.

Column 1: Von Neumann's Self-reproducing Automata

John von Neumann's self-reproducing automata consist of three parts: Tape T (*) (the parentheses enclose the content of the tape) for storing genetic information, a Machine C for making copies of the tape, and a Machine S for building machines based on the information on the tape. Machine C is generally called the copier and Machine S is generally called the constructor. Machine S is also called the Universal Constructor, meaning it can construct any machine if the appropriate tape information is supplied. The functions of these parts are expressed by the equations below.

$$\forall X: \quad C + T(X) \xrightarrow{C} C + T(X) + T(X)$$

$$\forall X: \quad S + T(X) \xrightarrow{S} S + T(X) + X$$

Here, X is the information for encoding Machine X.

The self-replication cycle of von Neumann's self-reproducing automaton, "C + S + T(C + S)," is expressed by the equation below.

$$\begin{aligned} C + S + T(C + S) &\xrightarrow{C} C + S + T(C + S) + T(C + S) \\ &\xrightarrow{S} C + S + T(C + S) + C + S + T(C + S) \end{aligned}$$

Figure 3.4 in this column is a schematic diagram illustrating this mechanism. (a) Copier C makes a copy of Tape T and (b) constructor S constructs C and S, based on the tape information.

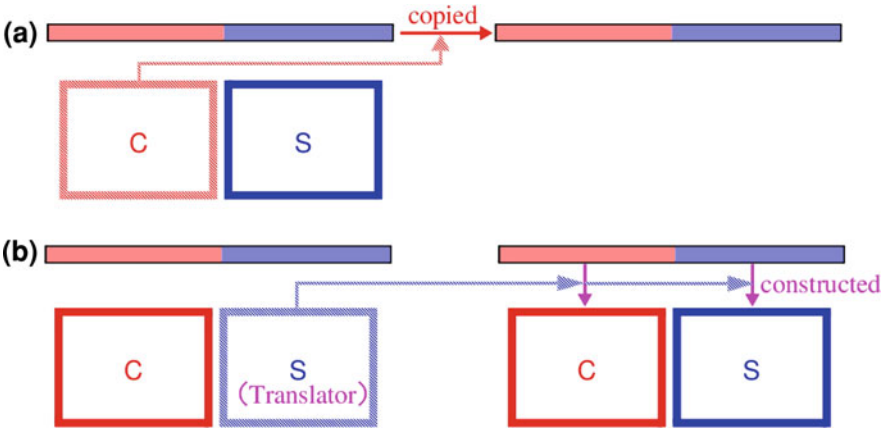

Fig. 3.4 Self-replication cycle of von Neumann automaton

When a mutation occurring during the course of evolution changes the content of the tape, the self-replication cycle is expressed by the equation below.

$$\begin{aligned} \mathrm{C}+\mathrm{S}+\mathrm{T}(\mathrm{C}+\mathrm{S}) &\overset{\text{mutation}}{\longrightarrow} \mathrm{C}+\mathrm{S}+\mathrm{T}(\mathrm{C}'+\mathrm{S}') \\ &\overset{\mathrm{C}}{\longrightarrow} \mathrm{C}+\mathrm{S}+\mathrm{T}(\mathrm{C}'+\mathrm{S}')+\mathrm{T}(\mathrm{C}'+\mathrm{S}') \\ &\overset{\mathrm{S}}{\longrightarrow} \mathrm{C}+\mathrm{S}+\mathrm{T}(\mathrm{C}'+\mathrm{S}')\mathrm{C}'+\mathrm{S}'+\mathrm{T}(\mathrm{C}'+\mathrm{S}') \end{aligned}$$

As the tape information changes to C′ and S′, the machines are also modified to C′ and S′. As indicated by this example, if mutations change the tape information in any manner during the self-replication of the automata, Machine S itself, which translates the genetic information (tape information) to the phenotype (machine), also changes. In von Neumann's self-reproducing automata, mutations can modify the translation relationship itself.

3.3 Topological Properties of Intermolecular Interactions

In the preceding section, we considered the properties of the solution as a biochemical reaction field from an evolutionary perspective and from the perspective of emergent functionality. In this section, we will discuss the properties of the solution reaction field from a slightly different perspective—that of its physical and chemical properties. In particular, we provide an overview of the topological conditions for molecular collisions—a prerequisite for reactions—and summarize their topological properties by introducing molecular networks that express the spatial relationships between molecules.

3.3.1 Intermolecular Forces and Chemical Reaction Velocity Theory

Generally speaking, the bonds between the atoms and molecules constituting living creatures fall into four categories, based on bonding energy. In order of increasing energy (each step being roughly a tenfold increase), we have van der Waals bonds, hydrogen bonds, ionic bonds, and covalent bonds (Fig. 3.5).

Table 3.1 shows the origins of the intermolecular forces—forces other than the covalent bond—in greater detail. Molecules can be ions (monopoles having static

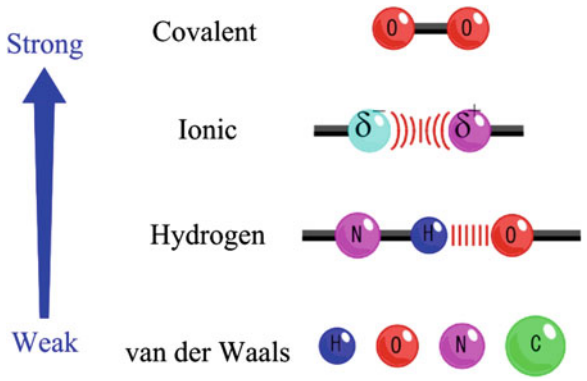

Fig. 3.5 Forces between the atoms and molecules that make up living creatures

Table 3.1 Classification of intermolecular forces of attraction

Ion ⇔ ion	⊕⇔⊖	$\phi(r) \propto r^{-1}$	
Ion ⇔ dipole	⊕⇔⊖⊕	$\phi(r) \propto r^{-2}$ fixed dipole $\phi(r) \propto r^{-4}$ rotable dipole	Ionic
Ion ⇔ induced dipole	⊕⇔⊖⊕	$\phi(r) \propto r^{-4}$	Hydrogen
Dipole ⇔ dipole orientation	⊖⊕⇔⊖⊕	$\phi(r) \propto r^{-6}$ (Keesom)	
Dipole ⇔ induced dipole induction	⊖⊕⇔⊖⊕	$\phi(r) \propto r^{-6}$ (Debye)	van der Waals
Induced dipole ⇔ induced dipole dispersion	⊖⊕⇔⊖⊕	$\phi(r) \propto r^{-6}$ (London)	

Type of intermolecular attractive force, schematic diagram, potential energy φ on intermolecular distance r

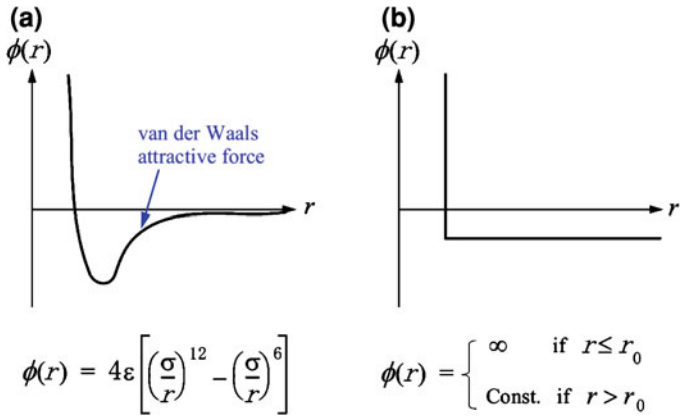

Fig. 3.6 Lennard-Jones potential and hard-sphere potential

charges); dipoles (which have uneven charge distributions in the static state); or induced dipoles (which lack electric charge in the static state but become dipoles when electric charge is induced by a nearby charge). The potential energies of the attractive forces are affected by distances in different ways, depending on the molecule type. In general, attractive forces between molecules without distinct electrostatic charges (for example, between induced dipoles) tend to fall off rapidly with increasing intermolecular distance. (The indices of r dependence of φ in Table 3.1 take large negative values.) Keesom, Debye, and London in this table refer to the individuals who formulated the inter-dipole interactions named orientation, induction, and dispersion, respectively.

Owing to overlapping electron clouds, molecules and atoms experience significant repulsive forces when they come into very close proximity, something not shown in this table. For this reason—for example, in the case of molecules without charge—they experience attractive forces (dispersion forces) when distant and repulsive forces when close together. We can express this as a Lennard-Jones potential (Fig. 3.6a). The Lennard-Jones potential is often used in molecular dynamics to help model interatomic forces. This section simplifies this potential into the hard-sphere potential (Fig. 3.6b) for use in subsequent discussions.

Chemical reaction, namely, the recombination of molecular bonds, starts with the molecular collision. In order to estimate the rate of chemical reactions, we have to calculate the frequency of molecular collisions incorporating the properties of environments (including the gas and liquid phases). This has long been one of the most important topics in chemical reaction velocity theory. The theory succeeded in formulating rate constant k as a function of various physical parameters (including molecule size and temperature) by assuming the molecules to be hard spheres and molecular distributions to be homogeneous. Column 2 describes this in greater detail; interested readers are referred there.

Column 2: Chemical Reaction Velocity Theory

The rate constant k [m^3/(mol s)] (the brackets enclose the unit), which indicates the rate of a chemical reaction, has a different parameter dependence in the gas phase and in the liquid phase. First, consider a basic chemical reaction in the gas phase, as shown in Fig. 3.7 of this column.

Here, I indicates the ingredient molecule. P indicates the product molecule. In general, the time dependence of the concentration $[P]$ [mol/m^3] of P: is expressed as

$$\frac{d[P]}{dt} = k[I]^2 \tag{3.1}$$

We introduce the following notations:

N:	Number of ingredient molecules [1]
V:	Volume of the gas phase considered [m^3]
m:	Mass of a single ingredient molecule [kg]
d:	Diameter of the ingredient molecule when regarded as a hard sphere [m]
$\langle v \rangle$:	Average speed of ingredient molecules [m/s]
$\langle v_r \rangle = \sqrt{2}\ \langle v \rangle$:	Average relative speed between ingredient molecules [m/s]
z:	Average collision frequency per ingredient molecule [1/s]
$Z = (1/2)(N/V)z$:	Number of collisions per second per unit volume [1/(m^3s)]
ε_0:	Activation energy of the reaction [J = m^2kg/s^2]
N_A:	Avogadro number [1/mol]
k_B:	Boltzmann constant [m^2kg/(s^2K)]
T:	Absolute temperature [K]
$\beta = 1/(k_B T)$:	Temperature factor [1/J = s^2/(m^2kg)]

Here, $\langle v_r \rangle$ is square root of two times $\langle v \rangle$; the ingredient molecules move near-randomly in the gas phase, and we can assume that the hopping vectors of the ingredient molecules form an average angle of 90°. The factor 1/2 in the expression for Z corrects for the double-counting of collisions (two ingredient molecules take part in a single collision). Based on this notation, we express the concentration of the ingredient molecules as:

$$[I] = \frac{N}{N_A V} \tag{3.2}$$

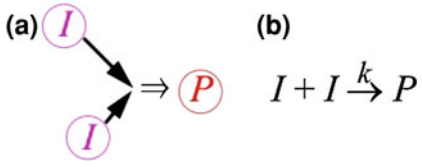

Fig. 3.7 Chemical reaction in gas phase

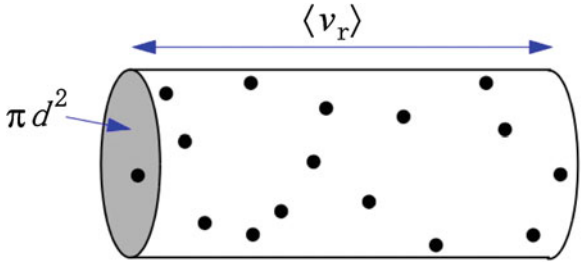

Fig. 3.8 Region through which a molecule in the gas phase moves each second

With the Maxwell–Boltzmann distribution derived from the kinetic theory of gases, we can derive

$$\langle v \rangle = \left(\frac{8k_{\mathrm{B}}T}{\pi m} \right)^{1/2} \tag{3.3}$$

Assume that each molecule is a hard sphere of diameter d. Then, each second, a molecule engages in z number of collisions with other ingredient molecules, with z being equal to the number of particles—the points representing the central position of other ingredient molecules—in the cylindrical region in Fig. 3.8 of this column. Accordingly,

$$z = \frac{N}{V} \cdot \pi d^2 \cdot \sqrt{2} \langle v \rangle$$

$$\therefore Z = \frac{1}{\sqrt{2}} \cdot \frac{N^2}{V^2} \cdot \pi d^2 \langle v \rangle \tag{3.4}$$

The final number of reactions occurring each second—d[P]/dt—is the number of collisions obtained multiplied by the Boltzmann factor which includes the activation energy as:

$$\frac{\mathrm{d}[P]}{\mathrm{d}t} = k[I]^2 = \frac{Z}{N_{\mathrm{A}}} \cdot \exp(-\beta \varepsilon_0) \tag{3.5}$$

Substituting Eqs. 3.2–3.4 for this equation, we obtain

$$\therefore k = \frac{N_{\mathrm{A}}}{\sqrt{2}} \cdot \pi d^2 \left(\frac{8k_{\mathrm{B}}T}{\pi m} \right)^{1/2} \cdot \exp\left(-\frac{\varepsilon_0}{k_{\mathrm{B}}T} \right)$$

Here, $\exp(-\varepsilon_0/(k_{\mathrm{B}}T))$ is called the Arrhenius factor; the other part is called the pre-exponential factor. Since m is proportional to d^3, the dependence of rate constant k on molecule size d is estimated to be $k \propto d^{1/2}$, indicating a gradual increase with molecular size.

Next, consider the reaction in the liquid phase illustrated in Fig. 3.9 in this column. The liquid phase differs from the gas phase in that the ingredient molecules I and the product molecules P are surrounded by solvent molecules S. An ingredient molecule hops by the collision with the solvent

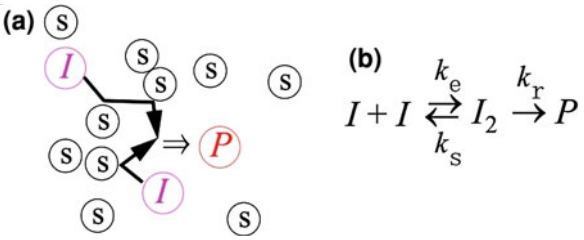

Fig. 3.9 Chemical reaction in liquid phase

molecules (Brownian motion), and as a result, encounters and collides with another ingredient molecule. In the liquid phase, these encounter and collision are defined as different events. Even if the two ingredient molecules encounter, if they cannot overcome the activation energy and do not generate P immediately, they collide with each other several times on the spot, and finally react or separate. This means the solute molecules are surrounded by the solvent in the liquid phase in a manner similar to conditions in a cage. Once the molecules belong to the same cage (i.e., once they encounter), they are not readily separated and tend to collide repeatedly. This is called the cage effect.

We can formulate such reactions in the liquid phase by assuming, as in Fig. 3.9b, that two ingredient molecules first form an encounter complex (activated complex) I_2 at the rate constant k_e, and I_2 reacts at rate k_r or separates at rate k_s. We consider the stationary state—the state in which intermediate product I_2 neither increases nor decreases—and derive

$$\frac{d[I_2]}{dt} = k_e[I]^2 - (k_s + k_r)[I_2] = 0$$
$$\therefore [I_2] = \frac{k_e}{k_s + k_r}[I]^2$$

This equation can be rewritten as:

$$\frac{d[P]}{dt} = k_r[I_2] = \frac{k_e k_r}{k_s + k_r}[I]^2 = k[I]^2$$

Here, k is the rate constant of the entire reaction defined as

$$k \equiv \frac{k_e k_r}{k_s + k_r} = k_e - k_s \frac{k_e}{k_s + k_r} = k_e - k_s \frac{[I_2]}{[I]^2} \tag{3.6}$$

With this equation, we can formulate the reaction velocity as:

$$\frac{d[P]}{dt} = k[I]^2 = k_e[I]^2 - k_s[I_2] = \frac{Z_e}{N_A} - \frac{Z_s}{N_A} \tag{3.7}$$

Here, Z_e [1/(m^{3}s)] and Z_s [1/(m^{3}s)] are the numbers of encounters and separations per second per unit volume, respectively; $k_e[I]^2 = Z_e/N_A$ and

$k_s[I_2] = Z_s/N_A$ are the numbers of encounters and separations per second per mole, respectively. In the stationary state, the number of product molecules, P, is equal to the number of encounters minus the number of separations.

Now, assume that an ingredient molecule moves with the Brownian motion and approach another (target) ingredient molecule. The relevant parameters are:

r: Distance to the target molecule [m]
$[I]_r$: Density of the ingredient molecules at distance r [mol/m^3]
D: Diffusion coefficient of the ingredient molecule in the solution [m^2/s]
η: Viscosity of the solution [kg/(ms)]
z_e: Average frequency of encounters of an ingredient molecule [1/s]
z_s: Average frequency of separations of an ingredient molecule [1/s]

Here, we can relate z_e and z_s to the numbers of encounters and separations, respectively, by

$$Z_e = (1/2)(N/V)z_e$$

and by

$$Z_s = (1/2)(N/V)z_s.$$

We can calculate D from η and other parameters using the Einstein–Stokes's relation

$$D = \frac{k_B T}{6\pi\eta(d/2)} \tag{3.8}$$

Though in a solution all ingredient molecules diffuse with D, here we assume that the target ingredient molecule is fixed in space and other molecules diffuse with magnitude $2D$, which gives an equivalent analysis. Around the target molecule, other ingredient molecules collide the target, disappear, and generate product P; hence in this area (Fig. 3.10a), we can assume that density $[I]_r$ of the ingredient molecules takes a minimum value at $r = d$ and increases as r increases until reaching bulk value $[I]$ at infinity ($r = \infty$) (shown by the upper curve in Fig. 3.10b of this column).

Here, using Fick's Law, the number j_r (1/s) of ingredient molecules that cross outwardly the surface of a sphere of radius r centered at the target molecule per unit time can be expressed as

$$j_r = -4\pi r^2 \cdot 2D \cdot N_A \cdot \frac{d[I]_r}{dr} \tag{3.9}$$

Since in the stationary state, j_r must have constant value j that is independent of distance r (the law of conservation of mass), if we integrate both

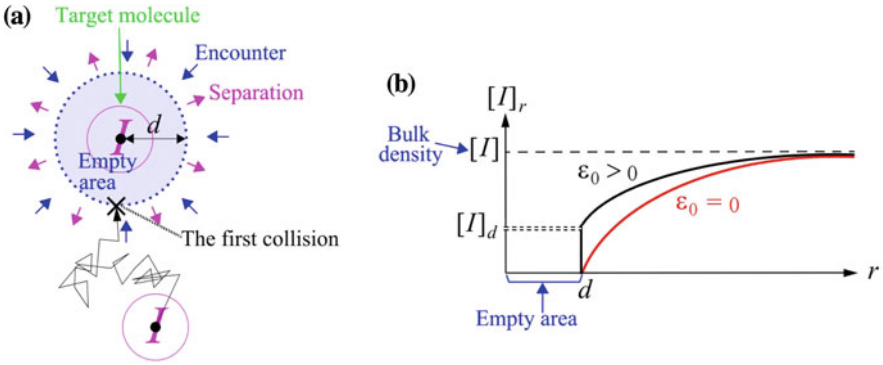

Fig. 3.10 Conditions around the target molecule in the liquid phase and distance dependence of molecular density

sides of the equation above with respect to r, j goes out of the integral and we obtain

$$\int_d^\infty \frac{j}{4\pi r^2}\mathrm{d}r = -\int_d^\infty 2D \cdot N_\mathrm{A} \cdot \frac{\mathrm{d}[I]_r}{\mathrm{d}r}\mathrm{d}r$$

$$\therefore j = -4\pi d \cdot 2D \cdot N_\mathrm{A} \cdot ([I] - [I]_d)$$

On the other hand, as shown in Fig. 3.10a, j is the frequency of separations minus the frequency of encounters for a target molecule: $j = z_\mathrm{s} - z_\mathrm{e}$. Thus, $Z_\mathrm{e} - Z_\mathrm{s}$ can be formulated as:

$$\begin{aligned} Z_\mathrm{e} - Z_\mathrm{s} &= \frac{1}{2}\frac{N}{V}(z_\mathrm{e} - z_\mathrm{s}) = \frac{1}{2}\frac{N}{V}(-j) \\ &= 4\pi d \cdot D \cdot N_\mathrm{A}^2[I]([I] - [I]_d) \\ &= 4\pi d \cdot D \cdot N_\mathrm{A}^2[I]^2 \exp(-\beta\varepsilon_0) \end{aligned} \tag{3.10}$$

The last equation is derived from the argument that positive activation energy ($\varepsilon_0 > 0$) induces separation. If we assume the reaction lacks activation energy ($\varepsilon_0 = 0$), all collisions instantly generate product P. (This state of affairs is said to be *diffusion-controlled*, since the rate of reaction is determined solely by diffusion.) When this happens, the ingredient molecules are no longer present on the surface of the sphere of radius d, which leads to $[I]_d = 0$; in other words, $[I] - [I]_d = [I]$ (the lower curve in Fig. 3.10b). If we assume activation energy is extremely large ($\varepsilon_0 = \infty$), on the other hand, collisions between the ingredient molecules never causes the generation of P. In this case, the number of separations equals the number of encounters ($Z_\mathrm{e} = Z_\mathrm{s}$), which gives $[I] - [I]_d = 0$. In intermediate cases ($0 < \varepsilon_0 < \infty$), we can assume that $[I] - [I]_d$ has the ε_0 dependence described by the Boltzmann factor.

From Eqs. 3.7, 3.8, and 3.10, we can formulate k as

$$\begin{aligned} k[I]^2 &= \frac{Z_e}{N_A} - \frac{Z_s}{N_A} = 4\pi d \cdot D \cdot N_A [I]^2 \exp(-\beta\varepsilon_0) \\ \therefore k &= \frac{4k_B T \cdot N_A}{3\eta_T} \cdot \exp\left(-\frac{\varepsilon_0}{k_B T}\right) \end{aligned} \tag{3.11}$$

Here, viscosity η is temperature-dependent and expressed as η_T. As with Eq. 3.5 for the gas phase, $\exp(-\varepsilon_0/(k_B T))$ on the right hand side of Eq. 3.11 is called the Arrhenius factor. The remaining part is called the pre-exponential factor. Equation 3.11 differs from Eq. 3.5 in that it does not depend on size d or mass m of the molecule. This is attributable, more or less, to the mutually canceling effects of the slower motion and larger cross section of the collision of larger molecules in the liquid phase.

The viscosity η_T is often formulated with Andrade's law as

$$\eta_T = A \exp\left(\frac{\varepsilon_0}{k_B T}\right) \tag{3.12}$$

This law applies to many liquids, but not, unfortunately, to water, the biological solvent.

Author's Refs. [87–89] for this column. Readers wishing to learn more about chemical reaction velocity theory are referred to these works.

3.3.2 Topological Conditions on Molecular Movement

The chemical reaction velocity theory is key to estimating rates of chemical reactions. However, the theory alone is inadequate for building virtual models to track each symbol (molecule) moving in information space within the artificial chemistry framework. Since chemical reaction velocity theory posits the diffusion of gases and liquids in three-dimensional Euclidean space and assumes homogeneous molecular distributions, it says nothing about what molecule, one particular molecule will most likely meet after colliding with another molecule.

A solute molecule that diffuses with Brownian motion is located in a three-dimensional Euclidean space. This means that a solute molecule is more likely to encounter a spatially close molecule and that molecules located far from each other are unlikely to meet soon. In preparing an artificial space and designing rules for the transport and collision of symbols, we need to consider such topological properties of molecular positions, which chemical reaction velocity theory does not specify explicitly.

In order to represent such constraint, here we set the three conditions as prerequisites for the information space and for the rules for transport of symbols.

These conditions set the ground rules for transport and compartmentalization of symbols in the artificial system, and should be satisfied for the system to comply with Emergence Conditions (5)–(11).

Topological Condition (1): The concept of distance applies between the symbols.

Topological Condition (2): The distance between the symbols can change over time.

Topological Condition (3): Collisions and reactions between symbols are more likely to occur if the distances separating them are smaller.

3.3.3 Intermolecular Distance and the Molecular Network

This section discusses the topological relationship between the molecules described in the Sect. 3.3.2 from a more quantitative perspective. First, we consider the vicinity matrix $V = (v_{ij})$ below. Element v_{ij} of the matrix $\mathbf{V}$ is a variable that can be expressed as $v_{ij} = 1/r_{ij}$, with distance r_{ij}, between the ith and jth molecules. Matrix $\mathbf{V}$ expresses the proximity relation (topological distance) between all molecules in the target region, which can be expressed, for example, as:

$$\mathbf{V} = \begin{bmatrix} \infty & 0.21 & 0.79 & \cdots & \\ 0.21 & \infty & 0.02 & \cdots & \vdots \\ 0.79 & 0.02 & \infty & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ & \cdots & \cdots & \cdots & \infty \end{bmatrix}$$

$\mathbf{V}$ is an $\mathbf{N} \times \mathbf{N}$ symmetric square matrix incorporating all topological information between the molecules, except information on angles ($\mathbf{N}$ is the total number of molecules). Since analyzing the matrix in this form is difficult, we simplify it based on our knowledge of intermolecular interactions. As discussed in Sect. 3.3.1, intermolecular forces, particularly those operating on molecules without electrostatic charge, rapidly fall to zero as the distance between the molecules increases. Thus, intermolecular interactions are negligible at distances above a certain value. As discussed with chemical reaction velocity theory in the liquid phase (refer to Column 2, discussed earlier), reactions in the liquid phase can be considered in two stages: encounters and reactions. In order to react, two molecules must approach each other and belong to the same cage (encounter). On that basis, we binarize the elements v_{ij} of the matrix $\mathbf{V}$ by substituting with a 0 when a value corresponding to an element is below a threshold value (i.e., when the intermolecular interaction is negligible) and by substituting with a 1 otherwise (i.e., when the molecules participate in an encounter). This generates encounter matrix $\mathbf{A} = (\delta_{ij})$, a new binary matrix. If the threshold value is 0.5, for example, the matrix $\mathbf{A}$ obtained from matrix $\mathbf{V}$ is expressed as:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & \cdots & \\ 0 & 1 & 0 & \cdots & \vdots \\ 1 & 0 & 1 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ & \cdots & \cdots & \cdots & 1 \end{bmatrix}$$

The graph (network) having this matrix as the adjacency matrix expresses the spatial relationship of the molecules as a graph topology, although only approximately. We call this graph an encounter network or molecular network.

3.3.4 Topological Properties of the Molecular Network

A molecular network is a graph, and we can apply graph theory to examine its characteristics. As the derivation in Sect. 3.3.3 clearly shows, a randomly generated binary matrix cannot be the adjacency matrix of a molecular network. A molecular network provides a simplified expression of the positional distribution of molecules in Euclidean space—the topology of which has special properties, reflecting the original spatial relationship of the molecules.

This section focuses on the topological properties of the molecular network and examines the static and dynamic properties of edge rewiring, which reflects the Brownian motion of the molecules. We simulate the movement of the molecules in the solution as a random walk of hard spheres in a two- or three-dimensional Euclidean space. We also calculate various characteristic quantities and edge creation or deletion probabilities in the encounter network of the hard spheres produced in the simulation [69].

The results obtained can be summarized as the topological properties of the molecular network:

Topological Property (1): Cluster coefficient C is approximately 10–20 times that for the random graph.

Topological Property (2): Average path length L is approximately 1.6–2 times that for the random graph.

Topological Property (3): Diameter D is small.

Topological Property (4): Tangledness T is low.

Topological Property (5): Edge joining probability P_{join} decreases linearly as the degree k of the end point increases.

Topological Property (6): Edge joining probability P_{join} decreases exponentially as shortest path length l between the end points increases.

Topological Property (7): Edge cutting probability P_{cut} is large when degree k of the end point or second shortest path length l_2 between the end points is large.

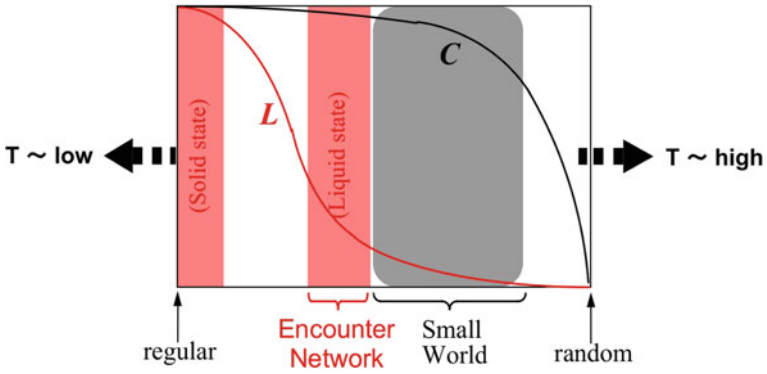

Fig. 3.11 Positioning of encounter network shown schematically in the C and L's plot with the Watts and Strogatz's scenario

Cluster coefficient C and average path length L are parameters that characterize small-world networks (see Column 3), investigated intensively in recent studies of complex networks. According to the C and L's values specified by Properties (1) and (2), we see that a molecular network is positioned somewhere between a regular network (corresponding to the solid phase) and a random network (corresponding to the gas phase). Figure 3.11 illustrates this state with plots for L and C values. As discussed in Column 3 further below, a small-world network is also a graph intermediate between the two network extremes. However, as the L value suggests, the encounter network is positioned somewhat closer to a regular network than a small-world network (Fig. 3.11).

Here, diameter D in Topological Property (3) is the maximum value of the shortest path length between all node pairs in the graph, which is large for a misshapen or distributed graph. Tangledness T in Topological Property (4) is measured by the number of edge crossings when the graph is projected onto a plane; it assumes a large value when the degree of tangling of the graph is large. Since a molecular network is a network defined by the positions of molecules in Euclidean space, it takes small values for D and T.

Column 3: Small-World Network

As described in a paper by Watts and Strogatz [1], a seminal publication in complex network studies, a *small-world network* refers to a group of graphs with special properties. In order to characterize a small-world graph, we need to define two parameters that express the graph's characteristics: cluster coefficient C and average path length L.

Cluster coefficient C is defined as follows:

$$C = \frac{1}{N}\sum_{i=1}^{N} C_i, \quad C_i = \frac{2E_i}{k_i(k_i - 1)}$$

Here, N is the number of nodes, k_i the degree of the ith node (the number of edges connected to the node), and E_i the number of edges connecting the node pairs adjacent to the ith node. C_i corresponds to the ratio of the edge pairs constituting triangles (clusters) among the $k_i(k_i - 1)$ edge pairs selected from the k_i edges.

Average path length L is defined as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i, \quad L_i = \frac{1}{N-1}\sum_{j \neq i} d_{ij}$$

Here, d_{ij}, the shortest path length connecting the ith and jth nodes, is assumed to have a finite value; in other words, the graph is connected.

A small-world network is defined, using these parameters, as a graph with large C and small L. A regular graph with highly regular topology has large C and large L; a random graph has small C and small L. A small-world network is positioned between these extremes. Alluding to a colloquial truism ("It's a small world!"), *small world* reflects the short paths connecting many node pairs; that is, small L.

In [90], Watts and Strogatz demonstrated the ubiquity of networks with small-world characteristics throughout the human sphere, including the Internet and social networks, and proposed a scenario for building such networks artificially (see Fig. 3.12 in this column). The simple scenario starts with a regular network with a one-dimensional structure. Then edges are randomly chosen and rewired, with which the graph gradually approaches a random network. This rewiring initially rapidly diminishes L, while C declines at a slower rate. A graph with small-world characteristics emerges as an intermediate state on the path to becoming a random graph.

Though a regular graph, the initial condition assumed in the Watts and Strogatz's scenario, is rarely observed in nature or in society, Davidsen et al. [91] subsequently proposed a scenario starting from a random graph and claimed that human social networks has small-world characteristics on account of the scenario. Similarly, the energy-based NAC rewiring rule

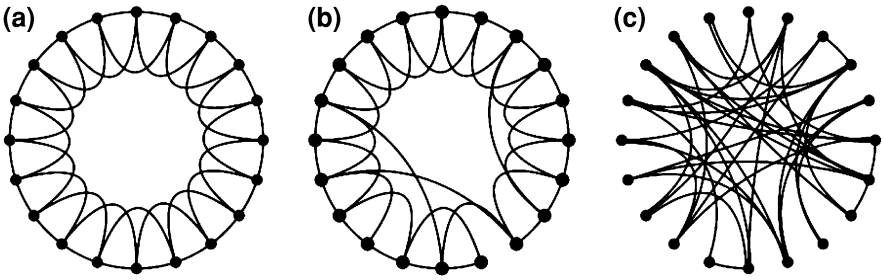

Fig. 3.12 Scenario by Watts and Strogatz (reproduced from [1]). **a** Starting from a one-dimensional regular network, the edges are randomly rewired. **b** A small-world network emerges. **c** A random network appears

described in Sect. 3.5 generates a molecular network (a kind of small-world network) starting from a random graph. These scenarios not starting from the initial regular graph create a small-world network, so to speak, in more realistic ways than by the Watts and Strogatz's scenario.

Chapter 5 of this book provides an overview of the application of small-world and other complex networks to the real world. An extensive body of literature exists on the topic, some in Japanese. We refer interested readers to these works [92–95].

Random-Walk Simulation of Hard Spheres

In order to model molecular diffusion in a solution (Brownian motion) by a random walk in Euclidean space, we consider a D-dimensional continuous space that satisfies the periodic boundary conditions as in Fig. 3.13. In this space, we introduce N D-dimensional hard spheres of radius R_1, and for each iteration, each hard sphere moves its center coordinates (hops) by a randomly chosen hopping vector. If the sphere hopping reduces the distance between the center points of the two hard spheres to below $2R_1$, the hopping is canceled. In order to simplify the problem and reduce computational costs, R_1 and hopping vector length Δ are assumed to be constant for all spheres and all hops. In addition, the direction of the hopping vector is not arbitrary, but randomly selected from $2D$ discrete values (a pair of positive and negative values for each dimension).

Collisions and contacts involving hard spheres are defined using contact radius R_2. When Euclidean distance d between the center points of a pair of hard spheres satisfies the condition $2R_1 \leq d \leq 2R_2$, the pair is regarded to be in contact (encounter). R_2 is taken to equal $2R_1$, reflecting the condition that the two spheres are regarded to be in contact if another hard sphere cannot enter between the two hard spheres ([8]; p. 890 of Japanese translation). While the contact graph (encounter network) thus created changes over time according to the random-walk model, we measure several graph characteristics.

First, we measure cluster coefficient C and average path length L of the encounter network, based on the usual definitions of these two variables. We also calculate edge joining and cutting probabilities P_{join} and P_{cut}, respectively, as follows.

Assume that Node A moves at a specific point in time and forms a new edge with Node C. Here, we measure directly l, the shortest path length between Nodes A and C, and degree k of Node C immediately before the edge forms, then add 1 to the frequency matrix $N_{\text{join}}(l,k)$. We also measure l and k for all nodes in the graph, including Node C, and add 1 to $N(l,k)$. At the end of the simulation, we calculate $P_{\text{join}}(l,k) = N_{\text{join}}(l,k)/N(l,k)$ to obtain the expectation value of P_{join}, the edge joining probability. Likewise, if Node A moves at a specific point in time, eliminating Edge AB, we measure directly l_2, the second shortest path length between Nodes A and B, and degree k of Node B, immediately before the edge disappears, then add 1 to frequency matrix $N_{\text{cut}}(l_2,k)$. We also check l_2 and k for all

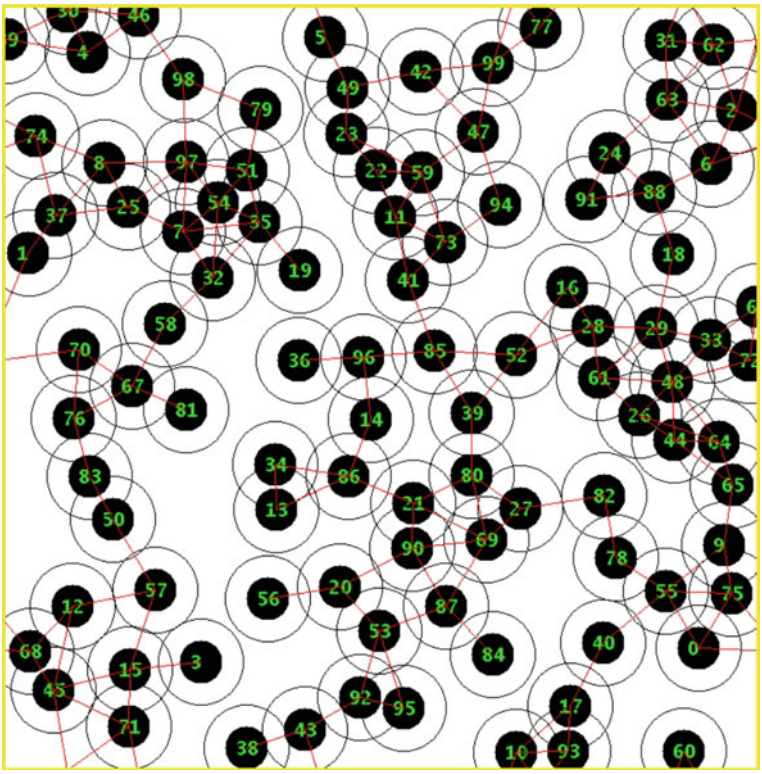

Fig. 3.13 Snapshot of hard-sphere random-walk simulation in two-dimensional space

nodes adjacent to Node A and add 1 to $N_2(l_2,k)$. At the end of the simulation, we calculate P_{cut} by using $P_{\text{cut}}(l_2,k) = N_{\text{cut}}(l_2,k)/N_2(l_2,k)$.

The parameter settings for this simulation are too detailed for the present discussion and are summarized in Column 4. We refer interested readers there.

Column 4: Parameter Setting for Hard Sphere Random-Walk Simulations

We calculate the typical values for the ratio R_1/Δ of the radius of the hard spheres and hopping vector length, based on the assumption that the hydrophilic head of the lipid molecule (solute) is surrounded by water (solvent) and is characterized by Brownian motion. We denote the number of collisions between the solute molecule and the solvent molecule per second as z, the radius of the solute molecule as r_1, and the average relative velocity of the solute molecule with respect to the solvent molecule as $\mathbf{V}_r$. Here, z equals the average number of solvent molecules in a cylinder with radius $R_1 + r_1$ and height $|\mathbf{V}_r|$ (see Column 2, previously discussed) and can be expressed as shown

in Fig. 3.14a. Here, ρ is the number of solvent molecules per unit volume. $\mathbf{V}_{\mathrm{r}}$ is the composite vector of the average velocity vector $\mathbf{V}$ of the solute molecules and the average velocity vector $\mathbf{v}$ of the solvent molecules. In this case, we can assume that the directions of the solvent molecules are completely random. Thus, $\mathbf{v}$ and $\mathbf{V}$ can be assumed to be orthogonal to each other. The resulting expression is as given in Fig. 3.14b. Here, we use the principle of equipartition of energy, $(1/2)M|\mathbf{V}|^2 = (1/2)m|\mathbf{v}|^2$, to estimate the velocity ratio as shown in Fig. 3.14c. $M = 255$ and $m = 18$ are the molecular weights of the lipid head ($C_8O_6PNH_{18}$) and water (H_2O), respectively. The equations shown in Fig. 3.14a, b give the R_1/Δ ratio shown in Fig. 3.14d. Based on this reference value, values for radius, contact radius, and hopping vector length of $R_1 = 4.0$, $R_2 = 8.0$, and $\Delta = 0.1$, respectively, are selected for this experiment.

We can evaluate average degree $\bar{k}$ of the encounter network if the hard spheres are randomly distributed in space as follows. Denote the volume of the D-dimensional sphere with radius r as $V_D(r) = \pi^{D/2} r^D/(D/2)!$. In view of

(a)
$$z = \pi(R_1 + r_1)^2 \cdot |\boldsymbol{V}_{\mathrm{r}}| \cdot \rho$$

(b)
$$|\boldsymbol{V}_{\mathrm{r}}| = \sqrt{|\boldsymbol{V}|^2 + |\boldsymbol{v}|^2} = \sqrt{1 + \frac{255}{18}} \cdot |\boldsymbol{V}| = 3.89 \cdot |\boldsymbol{V}|$$

(c)
$$\frac{|\boldsymbol{v}|}{|\boldsymbol{V}|} = \sqrt{\frac{M}{m}} = \sqrt{\frac{255}{18}}$$

(d)
$$\begin{aligned}
\frac{R_1}{\Delta} &= \frac{R_1}{|\boldsymbol{V}|/z} \\
&= \frac{R_1}{|\boldsymbol{V}|} \cdot \pi(R_1 + r_1)^2 \cdot |\boldsymbol{V}_{\mathrm{r}}| \cdot \rho \\
&= 3.89 \cdot R_1 \cdot \pi(R_1 + r_1)^2 \cdot \rho \\
&= 3.89 \times (3.63 \times 10^{-10}[m]) \times 3.14 \\
&\quad \times (3.63 \times 10^{-10}[m] + 1.5 \times 10^{-10}[m])^2 \times 3.34 \times 10^{28}[1/m^3] \\
&\simeq 39.0
\end{aligned}$$

(e)
$$\frac{N}{X^D - N \cdot V_D(R_1)}$$

(f)
$$\bar{k} = \frac{N}{X^D - N \cdot V_D(R_1)} \times (V_D(2R_2) - V_D(2R_1))$$

Fig. 3.14 Equations for parameter settings

the number density, the state in which N hard spheres with volume V_D (R_1) are distributed in a cube with sides of length X can be approximated by the state in which N points with zero volume are in the volume of $X^D - N \cdot V_D$ (R_1). Here, the number density is expressed as shown in Fig. 3.14e. $\bar{k}$ is this density multiplied by the volume of the D-dimensional spherical shell of inner radius greater than or equal to $2R_1$ and outer radius less than or equal to $2R_2$, which is obtained as shown in Fig. 3.14f. From this equation, for example, under the condition $D = 3$, $X = 93.0$, $N = 200$, $R_1 = 4.0$, and $R_2 = 8.0$, the average degree is estimated to be $\bar{k} = 4.0$.

Simulation Results

The experiment is performed with both $D = 2$ and $D = 3$. In both cases, the C and L values are confirmed to have the following characteristics. The C value increased to 10–20 times that for the random graph. The L value maintained a value approximately 1.6–2 times that for the random graph.

Figure 3.15 shows results for edge joining and cutting probabilities for the experiment with $D = 3$. (We confirmed that the results for $D = 2$ are similar.) Based on the results, where l is small, P_{join} (l, k) decreases exponentially as l increases, and decreases linearly as k increases. This result is intuitive based on what we know of molecular collisions. The collision probability is regarded to decrease as intermolecular distance increases due to the increase in l and with declining effective cross sections of collisions due to contact resulting from increased k. Here, according to Fig. 3.15a, in the region of $l = 2$ and $k \sim$ large, P_{join} increases slightly. This result is only observed under these particular conditions and is regarded to be attributable to noise. In the region with large l, P_{join} increases again, due to confined space. If the size of the space and the number of nodes in the graph are infinite, N (l, k) in the denominator increases infinitely as l increases. However, in a finite graph, N (l, k) decreases in a region with $l \sim$ large, and P_{join} is thus regarded to increase based on the actions of a small number of exceptional nodes.

Figure 3.15b shows that P_{cut} (l_2, k) is kept small in the region with $l_2 \sim$ small but rapidly increases when l_2 is above a certain value. The threshold value of l_2 decreases as k increases. For example, when $k = 8$, P_{cut} (l_2, k) is extremely large, even when $l_2 = 3$. This result can be understood by the "unnaturalness" of the graph topology with spatial restrictions considered. As the C and L values discussed above indicate, the encounter network has a type of regularity attributable to spatial restrictions. One conspicuous characteristic of a D-dimensional regular graph is that the nodes are distributed almost evenly in D-dimensional space and therefore the graph contains many paths of similar path length as the shortest path between a node pair. (In other words, the frequency is quite large near the shortest path in the histogram of the path length between node pairs.) An edge with a large second shortest path length ($l_2 \sim$ large) surrounded by many nodes ($k \sim$ large) contradicts this characteristic. The encounter network cuts such an edge with high probability, resulting in approaching a regular graph.

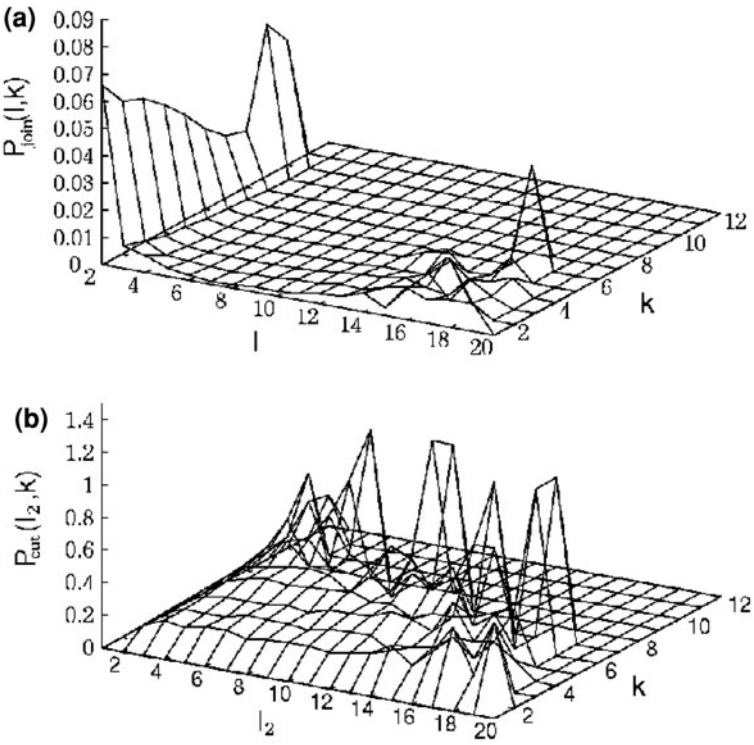

Fig. 3.15 Results of hard-sphere random-walk simulation. **a** Edge joining probability P_{join} between node pairs. (l is the shortest path length between pairs; k is the degree of one of the end points.) **b** Edge cutting probability P_{cut}. (l_2 is the second shortest path length between the end points of an edge; k is the degree of one of the end points.) $N = 200$ hard spheres are placed in Euclidean space with dimension $D = 3$ and length of sides $X = 93.0$. The radius of the hard spheres is $R_1 = 4.0$, the contact radius $R_2 = 8.0$, and the hopping vector length $\Delta = 0.1$. Under these conditions, the volumetric density of the hard spheres is 0.067, and the measured degree of the contact graph is $\bar{k} = 3.8$. These values are average values over 100 thousand iterations in a simulation that required about some 4 days to run on a desktop PC

3.4 Evaluating Artificial Chemistry Systems

Section 3.2 summarized the conditions necessary for the basic design of an artificial chemistry, based on the first stages in the evolution of life and functional emergence. Section 3.3 summarizes the topological conditions and topological properties of molecular networks, focusing on the positions and movement of molecules. Referring to these conditions, this section will verify and assess the basic designs of the artificial life and artificial chemistry systems proposed so far.

We select 26 representative systems to evaluate from those previously proposed and studied. Table 3.2 lists up the five elements of artificial chemistry systems comprising these systems. The systems are categorized by the following features:

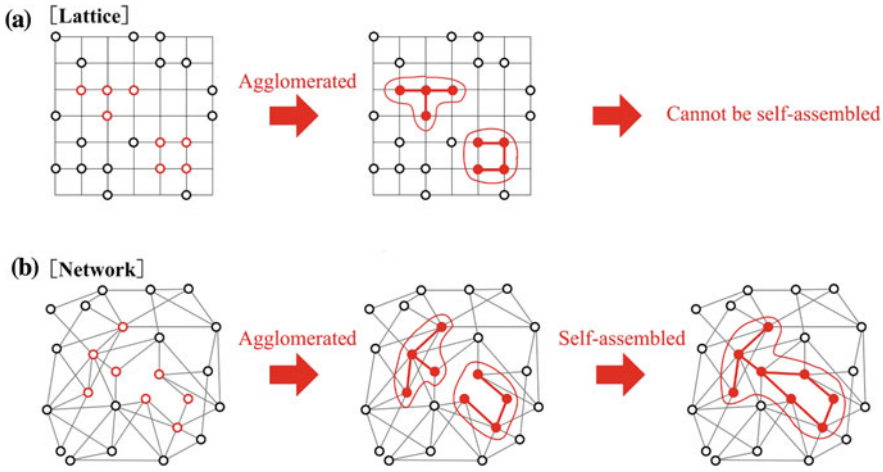

Fig. 3.16 Mobility of symbol complex

whether symbol types are fixed (determinative) or can be increased by combination (combinatorial); whether the space is rigid or elastic; and the dimensions of the space. (See Emergence Condition (6) in Sect. 3.2.2 for a detailed discussion of the rigidness of space.)

Table 3.3 gives the results of the evaluation. Reference [61] gives a detailed justification of the evaluation on which the results are based, although for part of the systems and for Conditions (1)–(10) only.

As Table 3.3 clearly shows, not one system among the 26 systems satisfies all 16 conditions. Emergence Conditions (12) and (13) have especially low pass rates. Condition (13) is met by few lattice spaces of discretized space. Figure 3.16 shows why.

In a lattice space like the one shown in Fig. 3.16a, when symbols assemble and form complexes occupying a large region in the lattice space, it becomes increasingly difficult to move the complexes within the space. Merely shifting a complex to the next lattice point entails checking the vacancy of many adjacent lattice points and associated computational costs. Rotational movement is out of the question. In contrast, these issues are less apparent in networks like the one shown in Fig. 3.16b. Among the systems evaluated, the planar graph of Speroni et al. and NAC fall into this category. Avida and Ameba are exceptions among lattice spaces. These systems always map a symbol complex to a single lattice point as a single program, so that the measure of mobility remains constant, regardless of the size of the complex.

The space and the transport rules for the symbols in the space form the most basic part of the framework of an artificial environment. The design of this framework significantly affects the quality of the model. All of the models examined here feature advantages and disadvantages, but none can generate self-organizing walls and flexibly express the transport/self-assembly of symbols.

Table 3.2 Five elements of artificial life and artificial chemistry systems

Example	(i) Information space	(ii) Elementary symbol	(iii) Reaction rule	(iv) Transport rule	(v) Higher-level manager
Tierra	1D core memory rigid	Machine instructions	Computational functions of machine instructions	Replication by copy command	Mutation, reaper (creature killer), time slicer, etc.
Modified Tierra	(Same as above)	(Same as above)	(Same as above)	(Same as above)	Same as above plus bonus conferrer (when the problem is solved)
SeMar	1D core memory elastic	Machine instructions and operands (classified as DNA/protein/membrane/nutrient)		Word transport rule by protein command	Mutation, reaper (creature killer), etc.
ID Proto-ell model	1D discrete elastic	Particles corresponding to solute and solvent molecules (classified as water/membrane/enzyme/resource and so forth)	Fixed metabolic reaction rules for synthesizing membrane particles from water and resource particles	Random walk of particles between lattice points, incorporating attractive and repulsive forces between nearby particles	
von Neumann's CA	2D discrete rigid (square lattice)	States of automata attributed to lattice points	Transition rules for automata	Transition rules for automata causing state propagation	
CA	2D/3D discrete rigid (square/cubic lattice)	(Same as above)	(Same as above)	(Same as above)	
Swarm chemistry	2D continuous rigid	Molecular agents with position, velocity, and characteristics vectors		Fixed rules for changing directions of agents	Human alchemist performing interactive selection and mixing of molecular species

(continued)

Table 3.2 (continued)

Example	(i) Information space	(ii) Elementary symbol	(iii) Reaction rule	(iv) Transport rule	(v) Higher-level manager
SCL-model	2D discrete rigid	Particles representing resources (substrate), catalysts, and membranes (link)	Inter-particle reaction rules for synthesizing/decomposing/joining membrane particles	Randomized transport rules conserving bonds between membrane particles	
Squirm3	2D continuous/discrete rigid	Particles ("atoms" or "enzymes") of fixed types and variable states	Externally prepared fixed state transition rules or state transition rules generated by interpreting enzyme states	Newtonian motion equation, incorporating attractive and repulsive forces between nearby particles	
Jonny Von	2D continuous rigid	T-shaped or cross-shaped particles (codons)	Transition rules for charges, states, and bonds of particles induced by collisions	Motion equation incorporating attractive and repulsive forces between nearby particles, friction (viscosity), and Brownian motion (noise)	
Dissipative particle dynamics (DPD)	2D/3D continuous rigid	Particles corresponding to solute and solvent molecules (particles having mass)		MD-like motion equation incorporating attractive and repulsive forces between nearby particles, friction (dissipative force), and random forces (noise)	
Langevin equation model	2D/3D continuous rigid	Particles corresponding to solute molecules (particles having mass, hydrophilic or hydrophobic characteristics, etc.)	Fixed metabolic reactions to synthesize membrane particles from resource particles	Langevin equation incorporating attractive and repulsive forces between nearby particles, friction (viscosity), and Brownian motion (noise)	
Cellular Potts model (CPM)	2D discrete rigid (square/hexagonal lattice)	Spin value of 0 or 1	Spin transition rules incorporating adhesion and spatial restrictions (energy minimization by the Monte Carlo method)		

(continued)

Table 3.2 (continued)

Example	(i) Information space	(ii) Elementary symbol	(iii) Reaction rule	(iv) Transport rule	(v) Higher-level manager
Avida	2D discrete elastic	Machine instructions	Computational functions of machine instructions	Replication by copy command, diffusion to adjacent lattice points associated with division of the creature, data transfers between adjacent lattice points	(Same as modified Tierra)
Ameba	(Same as above)	Patterns (codons) and machine instructions	(Same as above)	(Same as above)	(Same as Tierra)
SIVA	2D discrete elastic	Alphabets corresponding to small molecules and monomers	Functions encoded in virtual protein (short alphabetical strings)	Diffusion to adjacent lattice points associated with division of the creature	Modification of alphabets in the division of a creature (e.g., mutation and crossover operations)
2/3D Proto-cell model	2D/3D discrete elastic	(Same as 1D Proto-cell model, except that membrane particles have specific orientations)	(Same as 1D Proto-cell model)	(Same as 1D Proto-cell model)	
ARMS/ACS	Nested multisets	Characters (classified as passive or active)	Rewriting rules for passive characters defined in active characters	Transfer rules of characters between multisets	Division and removal rules for nesting, and mutation in character rewriting rules
AChemy	Tank	λ-expression	Rules for the λ calculus		Reaper for conserving the number of λ-expressions
Assembler automaton	Tank	Fixed-length binary numbers	Computational functions of machine instructions interpreting binary numbers		Reaper for conserving the number of λ-expressions

(continued)

Table 3.2 (continued)

Example	(i) Information space	(ii) Elementary symbol	(iii) Reaction rule	(iv) Transport rule	(v) Higher-level manager
Combinator model	Tank	Characters (basic combinators) or strings structured by "()" (combinators)	String rewriting rules defined in each basic combinator		Operation for conserving the number of basic combinators
SAC	Tank	Characters (classified as passive or active) or strings	Externally prepared fixed interpretation rules from active strings to rewriting rules		Operations for decomposing strings and supplying/ disposing of characters
Tominaga et al.	Tank	Characters (elements) or strings (objects)	Externally prepared fixed recombination rules for strings		
Tile automaton	2D continuous rigid	Tile clusters of various shapes	Rules for the creation and deletion of tile clusters caused by collisions of tiles	Newtonian motion equation and collision constraints	
Speroni et al.	2D triangular graph	(Same as combinator model)	(Same as combinator model)	Rules for the creation and deletion of contact edges resulting from reactions	
NAC	Graph assuming spatial restrictions	Characters (or strings) attributed to nodes (with hydro-properties, operation codes, or programs)	Interpretation and execution rules for operation codes/ programs for modifying strings and rewiring strong edges	Passive rewiring rules for weak edges (such as network energy minimization by Monte Carlo methods)	Operations for folding node chains, token generation, etc.

The references for the systems used as examples are given in the order of their appearance: Tierra [50, 33], modified Tierra [2], SeMar [58, 59], 1D Proto-cell model [39], von Neumann's CA [84], CA [26, 29, 30, 52, 53], Swarm chemistry [54], SCL-model [34, 35, 43, 75], Squirm3 [22–25], JohnnyVon [13, 55], DPD [12, 19, 21, 85], Langevin equation model [7, 37], CPM [17, 18, 46], Avida [1, 3, 4], Ameba [47–49], SIVA [42–45], 2D/3D Proto-cell models [31, 38, 40], ARMS/ACS [76–79], AlChemy [14, 15], Assembler automaton [10], Combinator model [56], SAC [41, 60], Tominaga et al. [81–83], Tile Automaton [86], Speroni et al. [57], NAC [62–70]

Table 3.3 Evaluation results for artificial life and artificial chemistry systems with respect to 16 conditions

			Emergence condition													Topological condition		
Symbol	Space	Example	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(1)	(2)	(3)
Determinative	1D rigid	Tierra	△	○	○	△	○	△	○	×	○	○	○	×	×	○	○	○
		Modified Tierra	△	○	○	△	○	△	○	○	○	○	○	×	×	○	○	○
	1D elastic	SeMar	△	○	○	△	○	○	×	○	○	○	×	○	×	○	○	○
		ID Proto-cell model	○	×	○	×	○	○	×	×	×	△	×	×	×	○	○	○
	2D/3D rigid	von Neumann's CA	×	○	○	○	×	△	○	○	○	○	×	×	×	○	○	○
		CA	×	×	○	△	○	△	○	○	○	○	○	×	×	○	○	○
		Swarm chemistry	○	×	×	×	×	×	×	○	×	○	×	×	×	○	○	○
		SCL-model	○	×	○	×	○	○	○	×	×	×	○	×	×	○	○	○
		Squirm3	○	○	○	△	○	○	○	○	○	×	○	○	×	○	○	○
		Jonny Von	○	○	×	×	×	×	×	○	○	×	×	×	○	○	○	○
		Dissipative particle dynamics	○	×	×	×	○	○	○	×	×	×	×	×	×	○	○	○
		Langevin equation model	○	×	○	×	○	○	○	×	×	×	×	×	×	○	○	○
		Cellular Potts model	×	×	×	×	○	○	○	×	×	×	×	×	×	○	○	○
	2D/3D elastic	Avida	△	○	○	△	○	○	○	○	○	○	○	×	○	○	○	○
		Ameba	△	○	○	△	○	○	○	×	○	○	○	×	○	○	○	○
		SIVA	○	○	○	△	×	×	×	○	○	×	×	×	×	○	○	○
		2/3D Proto-cell model	○	×	○	×	○	○	○	×	×	△	×	×	×	○	○	○
	Nested multisets	ARMS/ACS	×	○	○	×	○	○	○	○	○	△	△	×	×	×	×	×

(continued)

Table 3.3 (continued)

Symbol	Space	Example	Emergence condition													Topological condition		
			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(1)	(2)	(3)
Combinatorial	Tank	AlChemy	×	△	○	△	×	×	○	×	○	×	×	×	×	×	×	×
		Assembler automaton	△	×	○	△	×	×	○	×	○	×	×	×	×	×	×	×
		Combinator model	○	△	○	△	×	×	○	×	○	×	×	×	×	×	×	×
		SAC	○	○	○	○	×	×	○	×	○	×	×	×	×	×	×	×
		Tominaga et al.	○	○	○	△	×	×	×	×	×	×	×	×	×	×	×	×
	2D rigid	The automaton	×	×	×	×	○	×	×	○	○	×	×	○	○	○	○	○
	Planar graphs elastic	Speroni et al.	○	△	○	△	○	○	×	×	○	×	×	×	○	○	○	○
	Graphs elastic	NAC	○	○	○	○	○	○	△	△	○	○	△	○	○	○	○	○

In the table, *circles* indicate that conditions are met; *crosses* indicate conditions are not met. *Triangles* correspond to evaluation results as follows: satisfied with aid from higher-level manager (Condition (1)); possible, in principle, but not observed to date (Conditions (2), (4), (7), (8), (10), (11)); translation schemes exist, but are fixed (Condition (4); variable in the long term, but not in the short term (Condition (6)); only active transport using energy indirectly is possible (Condition (10))

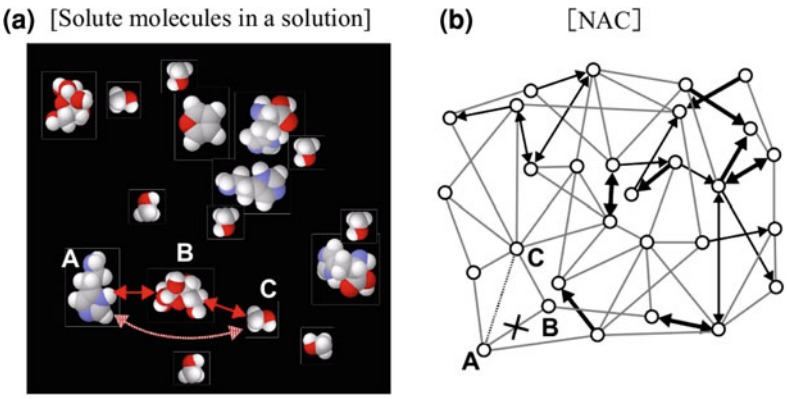

Fig. 3.17 Relationship diagram of solute molecules in solution and network artificial chemistry. **a** Solute molecules in a solution. The characteristics of the three-dimensional solution space restrict molecular motion. **b** Corresponding network artificial chemistry (NAC). The rewiring rules restrict node transport. After collisions between molecules A and B and between molecules B and C, the probability of a collision between molecules A and C is quite high

This poses a future challenge: to design systems meeting all of these conditions and featuring high emergent performance.

3.5 Network Artificial Chemistry

3.5.1 Basic Concept

Network artificial chemistry is the generic term for the artificial chemistry systems proposed and explored by Suzuki et al. [62–73] NAC systems express spatial relationships between symbols with purely topological graphs, making spatial descriptions more flexible. This section starts by describing some basic concepts of NAC.

The solute molecules, which play the central role in bio-chemical reactions, undergo repeated collisions when moving through the solvent (water), and their movements are restricted by the physical properties of the three-dimensional solution space and the size and shape of each molecule. NAC replaces these constraints with rewiring rules for the edges of a network. Figure 3.17 compares and draws parallels between this solution system and NAC.

The networks used in NAC are graphs consisting of nodes and edges. The nodes represent molecules or atomic clusters; the edges represent contacts or bonds. Since biological molecular interactions generally fall into one of four bonds—van der Waals, hydrogen, ionic, or covalent bonds—NAC has four edge types: wa, hy, io, and cv. The four edges are defined by different bond strengths. Weak edges are rewired one after another by a passive rule prepared in advance. Strong edges are formed by the active functions of the nodes and clusters.

Section 3.5.2 introduces typical examples of early NAC studies, while Sect. 3.5.3 introduces research on passive edge rewiring rules, which establish the basis for NAC dynamics. Section 3.5.4 describes the implementation of active programs into nodes to enhance the self-organization capabilities of the network structure.

3.5.2 Control Flow Cluster as Active Machine

Here, we introduce a model of the NAC in which one-dimensional node chains are converted to form clusters by "folding," functioning as active machines in a manner similar to proteins [68]. This mathematical folding and the folding of bio-protein molecules have the following common features.

Completeness of Expression

Both biological and mathematical foldings can generate clusters of various shapes based on differences in their one-dimensional sequences. The diverse range of protein functions and the diversity in the life forms indirectly demonstrate sufficient variations in the shapes of bio-proteins. When clusters are constructed according to the algorithm presented in this section, a node cluster (partial network) of any topology may also form, in principle, given the appropriate sequences.

Complementary Matching

Both are based on complementary matching of shapes or characters. Protein folding depends on the shapes, sizes, and physicochemical properties of the amino-acid residues making up the protein. (See Column 5 further below.) If a protein is folded with better matching between the residue sequences, the resulting protein is energetically stable. The folding of the node chain from Fig. 3.22b–c illustrated in this section can be regarded as a simplified mathematical counterpart of this process.

Column 5: Specificity and Complementarity of Chemical Reactions

In almost all of the reactions in molecular biology, from reactions of enzymes to protein self-assembly reactions, molecules specify their partners by complementary shape-matching (Fig. 3.18 in this column). Based on molecular shape, which vary as widely as human faces, molecules "choose" their partners in the reactions. This characteristic (called "specificity")

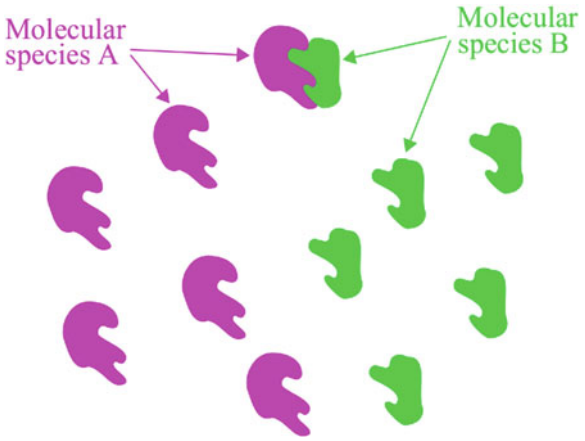

Fig. 3.18 Specificity based on complementary matching of shapes. The illustration schematically shows the shapes of molecular species A and B in a two-dimensional plane. In reactions between molecular species A and B, molecules of species A and molecules of species B have parts whose surface shapes complement each other. In this case, a molecule of species A will match with (react with) molecules of species B, but never with molecules of species A, its own species

propels one specific reaction in a timely manner in the presence of many other molecules within a cell.

From the viewpoint of intermolecular relation, complementary matching and equality matching (i.e., matching of like to like) have totally different outcome. If we observe only the matching relationship in specific areas on the surface of the molecules, interaction relationships characterized by only equality matching are transitive (i.e., if a $\Leftrightarrow$ b and b $\Leftrightarrow$ c, then a $\Leftrightarrow$ c). As a result, the interacting molecules form clusters that tend to separate from the remaining molecules. With complementary matching, on the other hand, reactions do not occur among molecules of the same species; all reaction interactions involve different species. This creates the potential for more complex topologies in the interaction network.

Figure 3.19 in this column illustrates this situation with the most basic example of interactions among three molecules. In the case of equality matching (Fig. 3.19a), the molecules separate into two clusters. In the case of complementary matching (Fig. 3.19b), the graph is more complex—it is connected by links but not completely bonded.

In fact, the complementarity of the intermolecular interaction implicitly generates complementary characteristics at various levels of interactions in the biological sphere, including interactions between hosts and parasites, fertilization interactions within the same species, and interactions between different species, such as predation and symbiosis. To model such complementarity, some artificial life systems (e.g., [96]) implement complementary

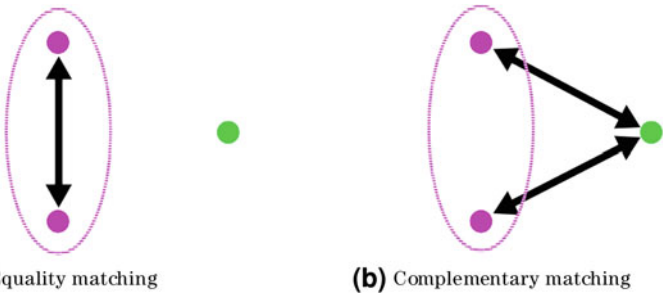

Fig. 3.19 Graphs of interactions among three molecules based on matching. **a** Equality matching; **b** complementary matching. The two nodes enclosed in the ellipse represent molecules of species A. The other node represents a molecule of species B. Two molecules connected with a *black two-headed arrow* interact (react) with each other; two molecules not connected with an *arrow* do not interact

matching between character strings or bit sequences to select the target of an operation, thereby generating diverse interactions between artificial creatures.

The clusters created by this mathematical folding ultimately are bound with strong edges, and function as data-flow machines (DFMs) by operating in parallel while exchanging tokens along the edges. (See Column 6 further below.) In the following experiment, one-dimensional node chains undergo mathematical folding to form two types of clusters: splitase and replicase. We present how active processing by these machines, such as modification of nodes and rewiring of strong edges, separates hydrophilic and hydrophobic regions in the network and replicates a molecular chain. The folding proposed here, a type of conversion from a genotype (node chain) to a phenotype (DFM), presents a method for introducing a large functional machine into NAC.

Column 6: Data-Flow Machine

A DFM or a data-flow computer (DFC) [97–99] is a computational architecture intensively studied by researchers around the world from the 1970s to 1980s as a parallel processing computer system potentially capable of circumventing the limitations of von Neumann computers. A DFC expresses a program as a graph consisting of operation nodes and data edges linking the nodes (Fig. 3.20b in this column). Data are carried by tokens along the edges; each node fires on receiving a token and transmits the results of an operation as another token to the output edges. A DFC always performs the operation (node firing) when the data (token) arrives, creating a highly parallel and distributed architecture.

Data-flow computer architecture and software were first explored at universities and research institutions including MIT (USA), the University

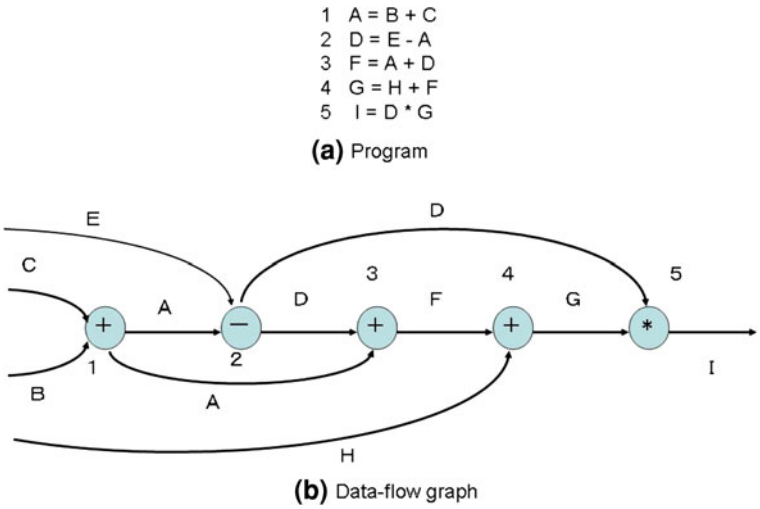

Fig. 3.20 Examples of **a** program and **b** data-flow graph used in data-flow computing (reproduced from [97])

of Utah (USA), the University of Manchester (UK), Toulouse University (France), NTT (Japan), and the Electrotechnical Laboratory (Japan). While several prototypes were built, the DFC architecture imposed unexpectedly severe constraints on software and prevented the sharing of existing software assets. Rapid advances in hardware technologies for von Neumann computers eclipsed the potential advantages of dedicated parallel machines, and DFC-based computers never reached the stage of commercial production. Interest in developing such machines dwindled in the 1990s.

Although DFCs failed to supplant von Neumann computers commercially, current superscalar processors inherit technologies from these research efforts, including out-of-order execution and register renaming [100].

Basic Model

The NAC graph used in this section consists of two types of nodes and three types of edges. Using an analogy drawn from living creatures, the nodes are classified as hydrophilic or hydrophobic nodes and the edges as Covalent (cv, directional, covalent bond), Hydrogen (hy, directional, hydrogen bond), or van der Waals (wa, non-directional, van der Waals bond) edges, in order of decreasing strength.

The wa edges of NAC are locally rewired by repeating the process sequences given in Steps (1)–(4) below.

(1) Reference Node A is randomly selected from the network.
(2) Node B with maximum degree is selected from adjacent nodes of A, connected by wa edges.
(3) Node C with minimum degree is selected from nodes connected by two or three wa edges from A.
(4) The hydrophilic and hydrophobic properties of Nodes A and C are investigated. If they satisfy specific conditions for bonding, Edge A–B is cut and Edge A–C created.

Here, the bonding conditions for Step (4) are set so that bonding is formed preferentially in the following order: hydrophilic–hydrophilic; hydrophobic–hydrophobic; and hydrophilic–hydrophobic [40]. Under his rewiring rule the graph tends to evolve toward having stronger small-world properties (see Column 3 discussed earlier) and more uniform degree distribution.

Mathematical Folding of Node Chains

The present NAC model expresses genetic information in terms of one-dimensional node chains connected by cv edges (Fig. 3.21). Each node has a character or character string composed of functional characters (a, b, …) or template characters (0, 2, …). The genetic information in the figure is the string obtained by joining the characters in sequence (e.g., a1122e0…). Each of the functional characters has a predefined function, but they do not operate in the form of the node chain.

Such node chains are folded after processing as shown in Fig. 3.22. First, the "agglomeration" process divides the adjacent node sequences directly before a functional character and converts each of the divided node chain sections into a

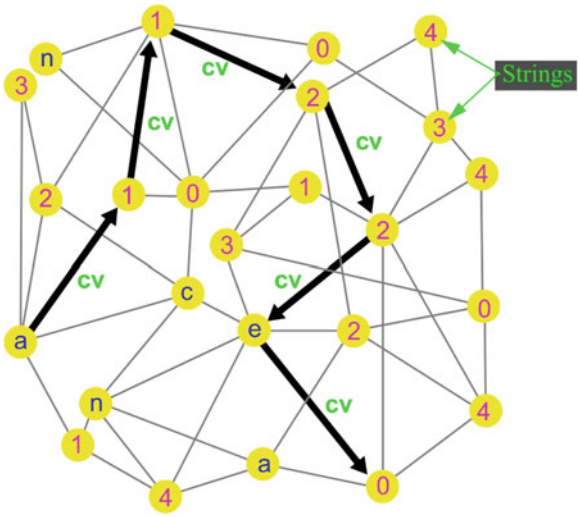

Fig. 3.21 Example of NAC node chain

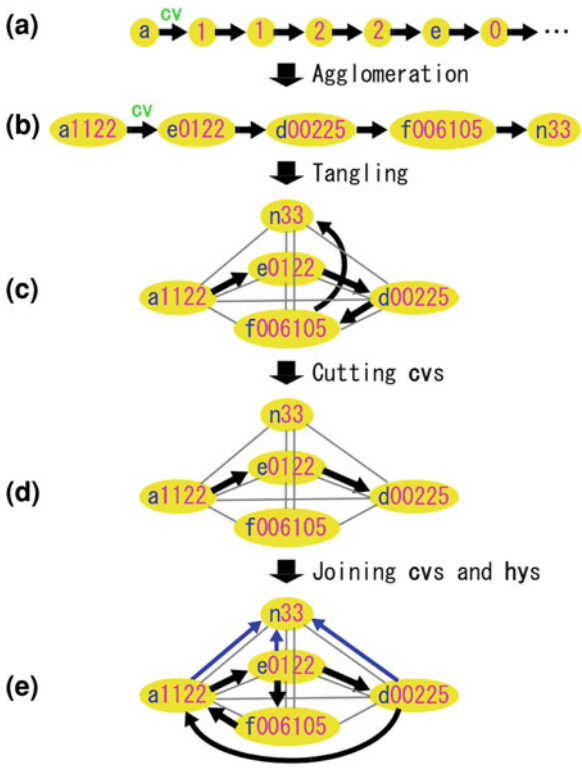

Fig. 3.22 Node chain folding process. **a** Initial node chain; **b** node chain after agglomeration; **c** cluster after tangling; **d** cluster after cutting cvs; **e** cluster after folding

single node with character strings such as a1122 or e0122 (Fig. 3.22b). Then, the wa edges link each of the node pairs in the node chain in (b), tangling the node chain ("tangling"; Fig. 3.22c). The template characters in each node are checked, and if "5" is included, the cv edge directly after is cut (cutting cvs; Fig. 3.22d).

Once the node chain is prepared for folding in this manner, the "folding" process ("folding," Fig. 3.22e) creates the cv and hy edges, based on complementary matching between the template character strings. Here, the template characters, 0–3, are matched complementarily based on the relationships 0 ⇔ 1 and 2 ⇔ 3. Nodes with matched templates are newly linked to cv or hy edges. The directions of the edges go from a template starting with 0 to the template starting with 1 and from the template starting with 2 to the template starting with 3. In the example of Fig. 3.22, new cv edges are formed from 00 to 11 and 01 to 10, and new hy edges are created from 22 to 33.

Operation of Data-Flow Clusters

When the node chain is folded into a node cluster, it operates as a DFM functioning in parallel while transmitting tokens along the cv edges. In general, when a

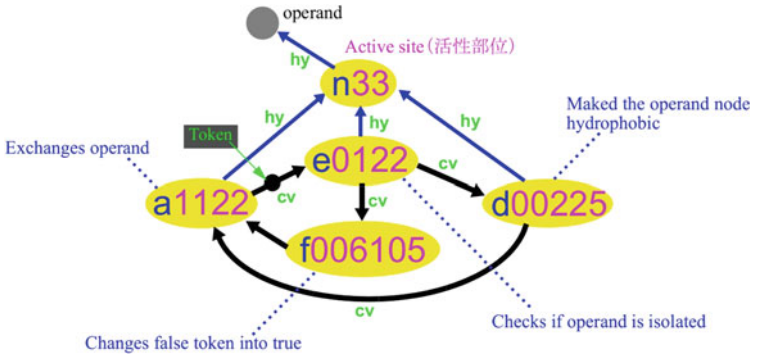

Fig. 3.23 Example of data-flow machine, "splitase." The cluster is made from the folding of a node chain consisting of 26 characters: "a1122e0122d00225f006105n33"

node receives a token, it fires if the value is "true," executing a routine assigned to the functional character (a, d, …). The value of the token is set according to the result and passed on to the next node.

Figure 3.23 shows an example of a DFM, called "*splitase*," generated by folding in the previous section. This machine consists of an active site (n33) and three operation nodes (a1122, e0122, d00225, and f006105). The operation nodes share a single active site via the hy edges and execute various processes to external nodes through this site. The processing starts with Node a1122 firing and the active site rewiring the hy edge to another operand node. Then, node e0122 fires. A check is performed to determine whether the new operand node is isolated with respect to cv. If so, the "true" token is transmitted from Node e0122 along the cv edges. If not, a "false" token is transmitted. When Node d00225 receives the true token, it operates on the operand node, changes its property to hydrophobic, and then returns the token to Node a1122. The function of Node f006105 is to fire when it receives a false token. If this node fires, the token value is converted to "true" and transmitted to Node a1122. This data-flow cluster has the capability of changing the operand nodes adjacent to Active site n33 to hydrophobic, one after another. This process rapidly produces one oily molecule after another.

Figure 3.24 shows a more complex example of designing the "*replicase*" data-flow cluster. The cluster consists of three active site nodes and 16 operation nodes. The basic replication processing of the node chain is based on processing proposed by Hutton [22]. The cv loop composed of Nodes 11–17 on the right half of this machine plays the main role. With the nodes in this part operating one after another by transmitting tokens, the active site randomly creates hy edges to nearby nodes, compares the character strings with the chain node, creates cv edges if they are the same, and moves one step along the chain. When the replication of the node chain is completed, the active site moves away from the node chain (cuts the hy edge) and searches for another node chain.

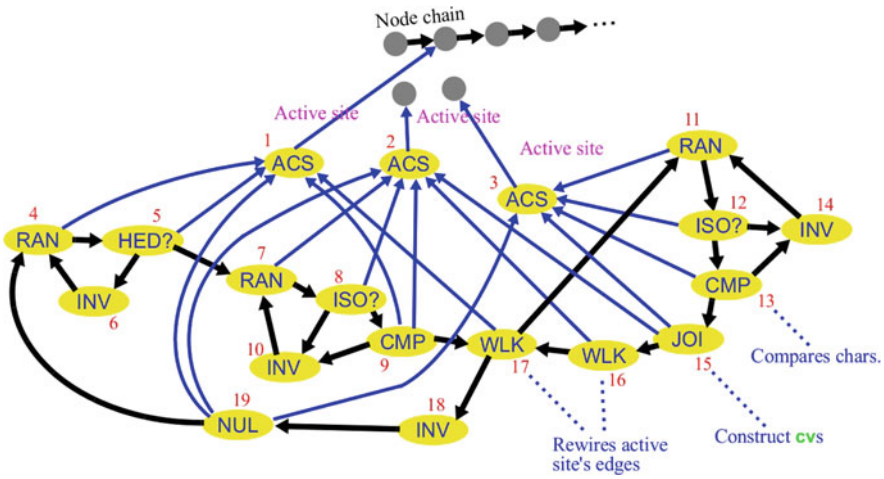

Fig. 3.24 Design example of data-flow machine, "replicase." The cluster is made from the folding of a node chain consisting of 191 characters: "a1101122311000k00011223f001005a111 00611001232e01000232g22300101232f001106101115a11110222e00000222g22201001223f111 16000015h101102326222i232i1101022300001f00010j22262326223001115n3325n3235n333" Mnemonics consisting of three to four characters in the node help identify the functions of the functional characters

Experimental Results

Network separation. We perform a network separation experiment using the *splitase* shown in Fig. 3.23. A node chain consisting of 26 nodes is introduced into an initial random network. After agglomeration, tangling, and cv cutting, the following three processes are performed: (1) rewiring of passive wa edges; (2) folding by wa edge connection; and (3) active processing by token transmission. Repeating these three processes changes the network topology and causes its structure to self-organize. Figure 3.25 shows a typical result of this experiment. As the hydrophobic nodes are generated by the *splitase*, the graph is separated into hydrophilic and hydrophobic regions and generates a structure with the hydrophilic cluster enclosed in the hydrophobic network.

Replication of Node Chain

A *replicase* node chain consisting of 191 nodes and an unfolded operand node chain consisting of ten nodes are introduced into an initial random network and processed in a manner similar to the above. Figure 3.26 shows a typical result. In this example, the *replicase* cluster creates replicated node chains with the same character strings as the operand node chain in the network.

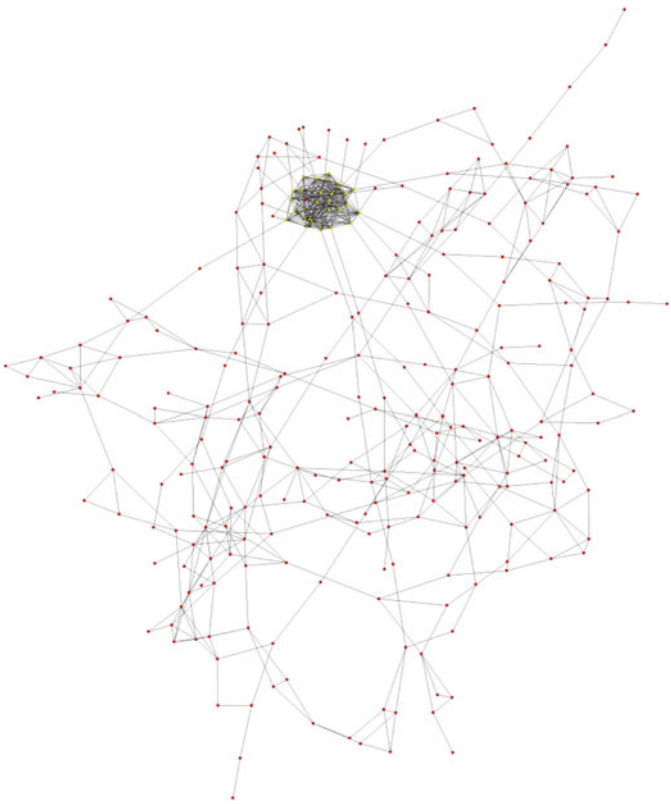

Fig. 3.25 Example of separation of network into hydrophilic and hydrophobic regions. The small cluster in the upper section is the hydrophilic cluster, and the rest is hydrophobic. A splitase chain is placed in a network with number of nodes $N = 300$ and average degree $W_p = 5$. After initial processing, the process is repeated $t = 260$ times. The wa edge joining probability is set as follows: $P_{ww} = 1.0$ between the hydrophilic–hydrophilic pair, $P_{ww} = 0.3$ between the hydrophobic–hydrophobic pair, and $P_{ww} = 0.0$ between the hydrophilic–hydrophobic pair. Since the present model has no mechanism to halt the splitase, if we continue the repetitive processing, the hydrophilic region other than the cluster itself disappears after 40 iterations. For all the figures shown hereafter in this chapter commercially available software aiSee [5] is used for the visualization (planar drawing) of networks

3.5.3 Passive Rewiring Rule Based on Energy Minimization

Network artificial chemistry generally provides four edge types: wa (van der Waals), hy (hydrogen), io (ionic), and cv (covalent). Of these, the passive rewiring rule of the weakest wa edges forms the basis of the dynamics. As shown in Fig. 3.6a, the intermolecular distance at which van der Waals attraction exerts its influence approximately equals the distance between molecules in the encounter state. (This distance is defined in this chapter as no greater than twice the hard-sphere diameter.) This means the passive rewiring rule for wa edges in NAC must be designed so that

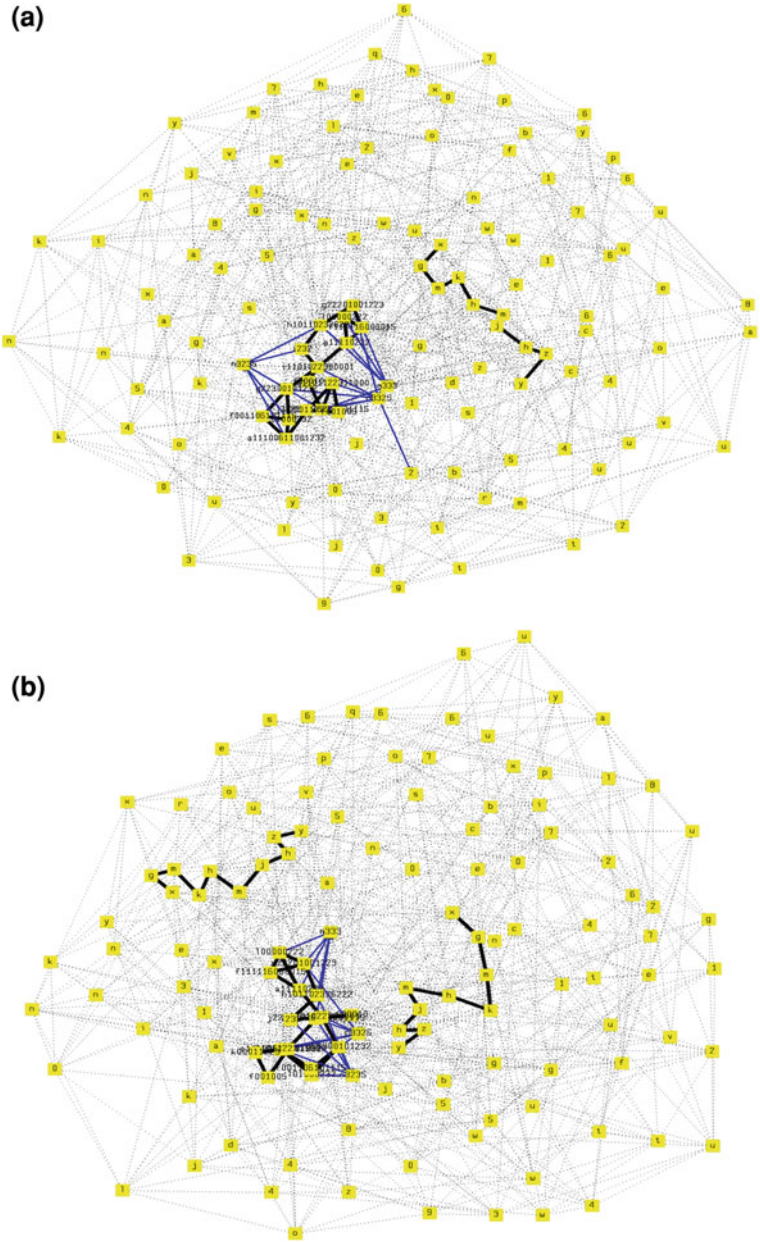

Fig. 3.26 Example of node chain replication by replicase. **a** Snapshot after 50 iterations; **b** snapshot after 710 iterations. The number of nodes is $N = 300$, the average degree is $W_p = 5$, and the wa edge joining probability is $P_{ww} = 1.0$

the topological properties of the resulting network are at least approximately equal to that of the encounter network of molecules discussed in Sect. 3.3.4.

Here, we describe an attempt to design the rewiring rule for weak edges [69]. The goal is to implement the feature of the design conditions previously mentioned. In order to formulate the rewiring rule, we first introduce energy defined for the entire graph. We then formulate an algorithm to minimize the energy stochastically by the Metropolis method and allow edges to be created or deleted as determined by the algorithm. Under this rule, we perform a NAC rewiring simulation, measure the topological properties using the graph obtained as described in Sect. 3.3.4, and try to determine whether they agree with the properties of the molecular network. The results are summarized below.

Topological Properties (1) and (2): The C and L values are intermediate between those for a regular graph and a small-world graph.

Topological Property (3): The resulting graph projected onto a plane forms a unity without distortions.

Topological Property (4): The resulting graph projected onto a plane has few edge crossings and small tangledness.

Topological Properties (5)–(7): The parameter dependence is qualitatively consistent with that of the hard-sphere simulation.

NAC Rewiring Simulation

Since the specific formulation of the rewiring rule used in this experiment is highly mathematical, it is summarized later in Column 7 and not discussed in depth in the main text. Based on preliminary experiments, we set the parameters defined in the formulation as: mean free path: $\lambda = 2$; upper limit for the shortest path length: $L_{\max} = 15$; indices in the equation for spatial restriction energy: $\mu = 0.01$, $\sigma = 4$, $\nu = 0.02$, $\gamma = 0.1$, and $\alpha = 2$; average degree: $\bar{k} = 4$; the threshold for energy change in the modified Metropolis method: $dE_{\text{th}} = 0.0015$; the acceptance probability for $-dE_{\text{th}} \leq \Delta E < 0$: $\kappa = 0.2$; and the temperature factor $\beta = 1{,}500$. Since this section focuses on spatial restriction energy, we use only wa edges and set the bonding energy at $\varepsilon^{(\text{wa})} = 0$.

Experimental Results

Starting from an initial random graph with number of nodes $N = 200$ and average degree $\bar{k} = 4$, we rewire the NAC graph. Figure 3.27 plots the cluster coefficient (C) and the average path length (L) as functions of time. As the figure shows, after approximately 100 thousand iterations, the C value of the NAC graph is roughly 19.4 times the initial value ($C \approx 0.018$) for the random graph. The L value is roughly 1.5 times the initial value ($L \approx 3.95$) for the random graph. The values indicate the resulting graph has properties approaching those of the encounter network of hard spheres.

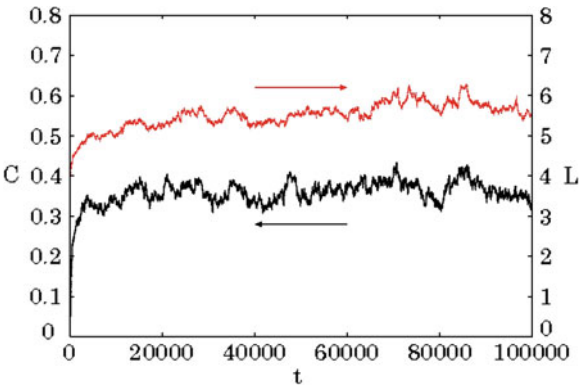

Fig. 3.27 Cluster coefficient (C) and average path length (L) in NAC rewiring simulation. The time propagation is measured for 100 thousand iterations of graph rewiring

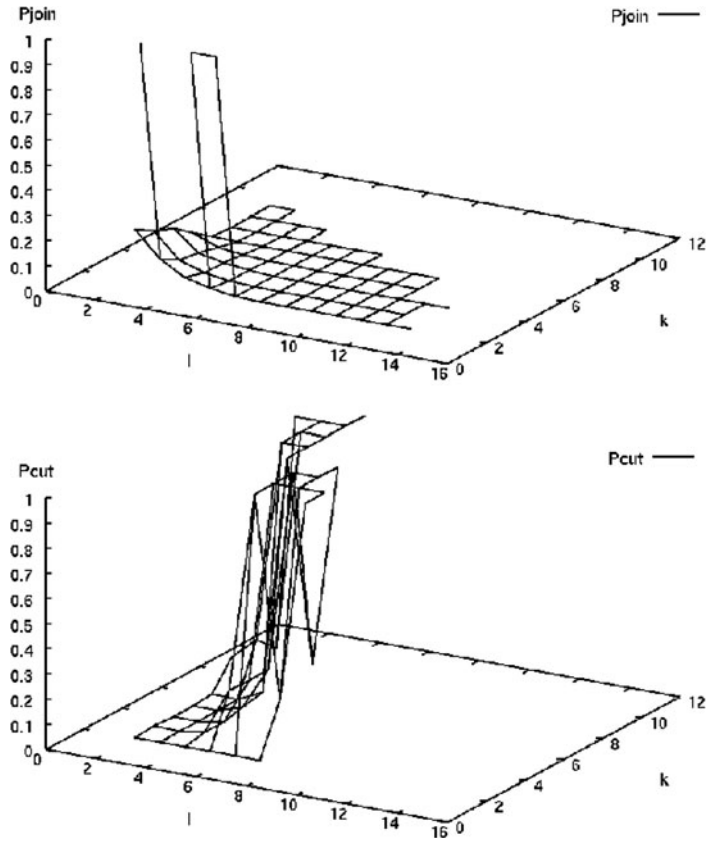

Fig. 3.28 Edge joining and cutting probabilities calculated based on NAC rewiring simulation result. We apply Eq. 3.19 from Column 7 (discussed earlier) to convert the average value for ΔE calculated for 200 thousand iterations of rewiring to P. As in Fig. 3.15, P_{join} is plotted as a function of the shortest path length (l) between pairs and the degree (k) of one of the end points; P_{cut} is plotted as a function of second shortest path length (l_2) between the end points of an edge and degree (k) of one of the end points

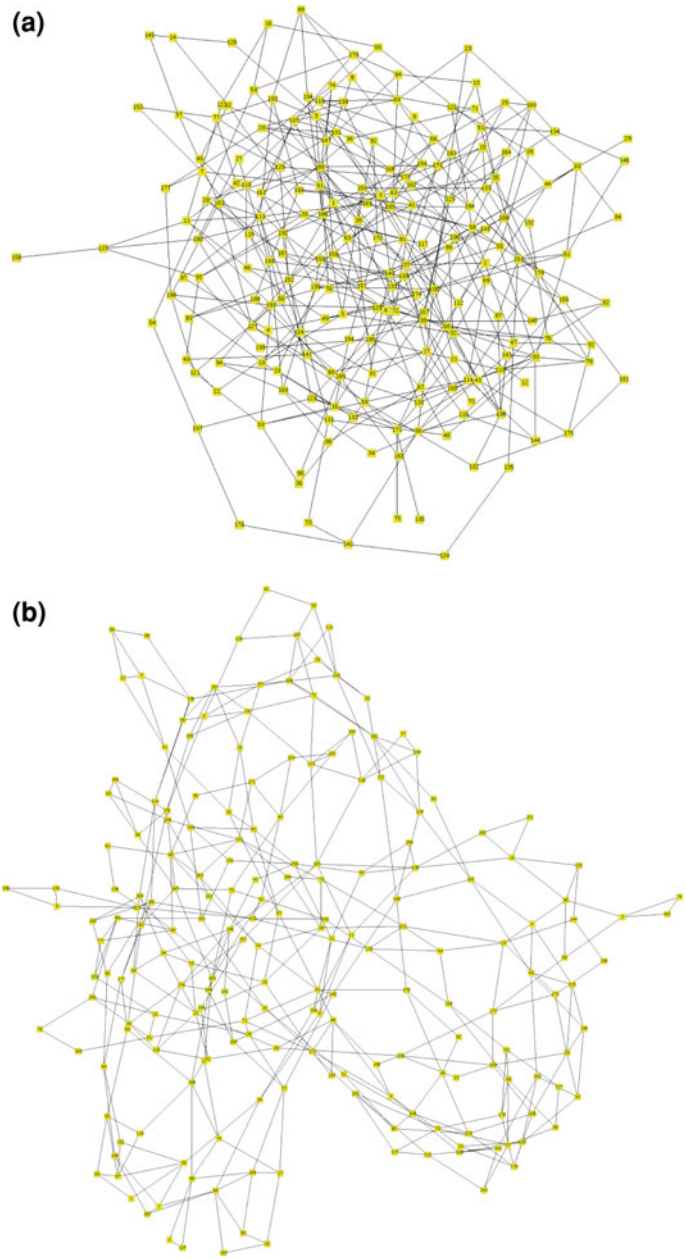

Fig. 3.29 Two-dimensional rendering of graphs created in NAC rewiring simulation. **a** Initial random graph; **b** NAC graph after 80 thousand iterations of rewiring visualized with aiSee [5]

Figure 3.28 shows the expectation values of the probabilities P_{join} and P_{cut} measured in the simulation. They agree qualitatively with the results for the random-walk simulation (Fig. 3.15).

Figure 3.29 shows the initial NAC graph and the NAC graph after 80 thousand iterations projected onto a plane. The graph after rewiring (Fig. 3.29b) is bunched and contains fewer edge crossings, generating a topology easier to project onto a plane.

Column 7: Formulation of Energy-Based NAC Edge Rewiring

Network artificial chemistry expresses molecular collisions as the generation of edges. Determining what molecule a given molecule will collide with next is equivalent to the problem of determining between what node pairs an edge should be generated. Here, we assume that the intermolecular distance in a NAC is measured by the shortest path length between the node pair and that the node pair for which the new edge generation is attempted is selected in proportion to the free path function below.

$$P_{\text{fp}}(l) = \exp(-l/\lambda) \tag{3.13}$$

This equation is based on Topological Property (6) of the molecular network in Sect. 3.3.4. Here, the parameter λ is called the NAC mean free path and corresponds to the mean free path of the molecules in the statistical physics (average distance traveled by a molecule without colliding with another molecule). As λ in the physics is independent of temperature (namely, the speed of the molecules) and depends only on the size and density of the molecules, so λ in a NAC takes a constant value independent of temperature. Here we also set the restriction $l \leq L_{\max}$, to avoid occasional situations where the system searches the graph for far-distant node pairs, resulting in the driving up of the computational cost.

We define the energy of the NAC graph as the sum of the topology-related spatial restriction energy E_{s} and edge bonding energy E_{b}, as:

$$E = E_{\text{s}} + E_{\text{b}} \tag{3.14}$$

E_{s}, the sum of the term (μ-term) for suppressing fluctuations in the degree of each node and the term (ν-term) for providing regularity, is expressed as:

$$E_{\text{s}} = \frac{\mu}{N}\sum_{n}\left|k_n - \bar{k}_n\right|^{\sigma} + \frac{\nu}{N}\sum_{\langle mn\rangle}\left\{\left(\frac{k_m k_n}{\bar{k}^2}\right)^{\gamma}(l_2)_{mn}\right\}^{\alpha} \tag{3.15}$$

In the expression, Σ_n is the summation over all nodes, each denoted by n, while $\Sigma_{\langle mn\rangle}$ expresses the summation over all adjacent node pairs, each denoted by mn. Here, k_n is the degree of Node n, $\bar{k}_n$ the expectation value of the degree of Node n (which differs for each node), $\bar{k}$the average degree for all nodes, and $(l_2)_{mn}$ the second shortest path length for Node Pair mn. When

the constant σ is positive, the μ-term has the function of making the node degree closer to the target value, $\bar{k}_n$, (in the case of hard spheres of uniform size distributed in space, a single hard sphere cannot randomly come into contact with an unlimited number of hard spheres), and the v-term has the function of cutting the edges between nodes with large degrees on both ends and a large second shortest path length with high probability.

E_b, the edge bonding energy, is expressed as:

$$E_\mathrm{b} = -\frac{1}{N}\sum_{\langle mn\rangle} e_{mn} \tag{3.16}$$

Here, e_{mn}, the bonding energy for Edge mn (the energy required to cut the edge), is defined by the following, according to the edge type: $(0 \le \varepsilon^{(\mathrm{wa})} \le \varepsilon^{(\mathrm{io})} \le \varepsilon^{(\mathrm{cv})})$.

$$e_{mn} = \begin{cases} \varepsilon^{(\mathrm{cv})} & \text{when Edge } mn \text{ is a cv edge} \\ \varepsilon^{(\mathrm{io})} & \text{when Edge } mn \text{ is an io edge} \\ \varepsilon^{(\mathrm{wa})} & \text{when Edge } mn \text{ is a wa edge} \end{cases} \tag{3.17}$$

The rule for joining or cutting the edges using the formulated E is determined by the modified Metropolis method to stochastically minimize E. More strictly speaking, edge rewiring is done by executing the two following steps at each iteration.

Step 1: Edge joining

Select a random node in the graph. Select randomly another node with a probability proportional to $P_\mathrm{fp}(l)$ where l is the path length between the two nodes. Based on network energy change ΔE before and after the new edge is generated between the two nodes, calculate acceptance probability $P(\Delta E)$, and then join the edge at this probability.

Step 2: Edge cutting

Select a random edge in the graph. Calculate network energy change ΔE before and after the edge is cut. Based on the energy change, calculate acceptance probability $P(\Delta E)$, and cut the edges at this probability.

The energy change $\Delta E = \Delta E_\mathrm{s} + \Delta E_\mathrm{b}$ associated with edge joining or cutting is obtained from Eqs. 3.15–3.17 and Eq. 3.18 below. (Here, $\varepsilon^{(\mathrm{cv\text{-}act})}$ is the activation energy of a cv edge.)

$$\Delta E_\mathrm{b} = \begin{cases} \frac{1}{N}\varepsilon^{(\mathrm{cv-act})} & \text{when joining a cv edge} \\ \frac{1}{N}\left(\varepsilon^{(\mathrm{cv-act})} + e_{mn}\right) & \text{when cutting a cv edge} \\ -\frac{1}{N}e_{mn} & \text{when joining an edge other than a cv edge} \\ \frac{1}{N}e_{mn} & \text{when cutting an edge other than a cv edge} \end{cases} \tag{3.18}$$

Accurately evaluating the change in the v-term in Eq. 3.15 requires calculations over all edges in the graph. Here, we consider only the partial

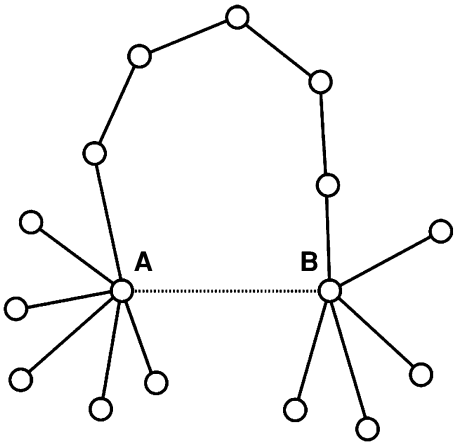

Fig. 3.30 Nearby section graph for approximate calculation of change in the spatial restriction energy term. The graph consists of edges directly connected to Nodes A and B, the end points of the edge about to be joined or cut, and edges constituting the shortest path (when joining) or the second shortest path (when cutting) between Nodes A and B

graphs shown in Fig. 3.30 in this column to evaluate the summation $\Sigma_{\langle mn \rangle}$. The results of a preliminary NAC rewiring experiment confirm that the energy difference ΔE, calculated as such, provides a satisfactory approximation of the true value. To reduce the computational cost, we also restrict the value of l_2 to be $l_2 \leq L_{\max}$.

Based on the above formulation, we calculate the final $P(\Delta E)$ from the following equation, which slightly modifies the acceptance probability of the Metropolis method.

$$P(\Delta E) = \begin{cases} \kappa \exp(-\beta \Delta E) & \text{when } \Delta E \geq 0 \\ \kappa & \text{when } -dE_{th} \leq \Delta E < 0 \\ 1 & \text{when } \Delta E < -dE_{th} \end{cases} \tag{3.19}$$

We would expect that setting probability $\kappa < 1$ for $\Delta E \geq -dE_{\text{th}}$ at a value different from the probability for $\Delta E < -dE_{\text{th}}$ would approximately reproduce the rapid rise in probability, as observed in Fig. 3.15b of the main text.

3.5.4 Organization of Network Structure by Active Node Program

The rewiring rule based on the network energy discussed in Sect. 3.5.3 produces qualitatively the same properties in the NAC graph as the contact graph of hard spheres. Nevertheless, when we view the network generated in this way as a virtual space replacing the three-dimensional solvent space, the results remain unsatisfactory. One reason for this is that this network fails to model liquid crystallization.

For example, water as a material at room temperature forms a network in which water molecules are loosely bonded by hydrogen bonds. At temperatures below 0°C, this network becomes regular, forming a crystal structure (ice). In an extreme sense, this phenomenon is generated by the physical properties of the water

molecules and three-dimensional spatial restrictions. However, it is difficult to create such a regular structure in a graph without spatial information such as angles and positions, based only on the energy minimization principle. The cluster coefficient and average path length of the graph obtained in the numerical simulation of Sect. 3.3 are also limited by the values of the small-world network (see Column 3, discussed earlier) or slightly closer to those of a regular network. Simulations with various energy functions that use a lower temperature value (higher value of the temperature factor β of the Metropolis method) cannot prompt the graph to converge into one with a large average path length (L)—in short, one with high regularity—while maintaining the connectivity of the overall graph [66, 69].

One of the reasons for this difficulty may be the specialty of the water molecules, ignored in the past NAC solvent node. Unlike organic solvents such as benzene, water is an abnormal liquid that establishes a pseudo-regular lattice over the space that it occupies based on hydrogen bonds. This nature is also the main cause of other behavior, including hydrophobic interactions and lipid bilayer formation. (Hydrophobic interactions are generally described as a phenomenon in which non-polar molecules cannot join the regular network of the water molecules, but are instead repelled and thereby segregated.) Intermolecular interactions based on hydrophilic and hydrophobic properties constitute one of the basic dynamics of bio-molecules. Thus, ultimately, water is capable of nurturing life because water molecules are not inert, but active molecules with unique functions. Up to this point—as in Sect. 3.5.2, for example—we have assumed that the active functions in a NAC result from the assembly of many nodes—such as a data-flow cluster (corresponding to a protein)—and that the solvent nodes (corresponding to water) are static and lack specific functions. However, the preceding discussion indicates that a graph, a pure mathematical model, has certain limitations in producing the regularity (crystal structures) created by spatial restrictions. In order to circumvent these difficulties, we need to implement more active functions in the nodes and allow them to interact.

Based on this concept, here we describe a new simulation model of the NAC which attributes programs (active functions) to NAC nodes and executes them in parallel [70]. In the solvent nodes, assumed to be inert in the NAC studies up to this point, we implement a program designed to model the complex functions of water. The program rewires the surrounding edges based only on local information, generating pseudo-regular structures in the network such as sheets and strings.

This example allows a network, a purely mathematical model, to self-organize an emergent structure similar to that from physical restrictions. The result suggests that the simplest and most efficient way to induce self-organizing structures in a graph is to attribute functions of sufficient complexity to the nodes and allow them to interact vigorously. Out of many possible ways to describe such functions, this simulation takes a method to implement functions as programs, allowing an artificial local program to generate global structures in a network, in a manner similar to amorphous computing (see Column 8).

Column 8: Amorphous Computing

The word *amorphous* means something lacking a defined shape or integrity. As used here, the term *amorphous* has a slightly different meaning: We can better understand the term *amorphous computing* as the formation of patterns from a bottom-up approach. This technique has been studied since the late 1990s, mainly by researchers at MIT, including Knight and Weiss [101].

These researchers pursued the following line of thought. When microfabrication technologies reach the point at which line widths in the integrated circuits formed on a silicon wafer reach the nanoscale order, design needs may outstrip current technologies, which assume that humans design patterns and impose them, top-down, on the wafer. At this point, perhaps the solution would be bottom-up patterning. Perhaps patterns might form from functional particles scattered on the silicon wafer—a wondrous result, if true.

Although the concept of such particles was merely dreamed up by computer scientists, they proceeded by positing that the problem, the fabrication of functional particles, had been solved. Based on this assumption, they proceeded to address an interesting informatics problem: What particle program should they prepare to form the patterns desired?

Here, let us consider a well-known example: PL-Gakuen's human letter displays at the summer Koshien baseball games. Each member of the cheerleading battalion holds several color cards, all numbered. When the captain of the cheerleading team calls out a number, each team member holds up the card corresponding to that number. This is an example of a top-down process; the resulting patterns are based on instructions issued by the captain. Let us take a slightly different approach and form the patterns with a bottom-up approach. Now, each member holds a program that specifies when and how the cards should be held up, and the team captain issues no instructions. Each member must determine the card to be held up next based only on local information—for example, the cards that the member or member's neighbors are currently holding up. What program do we need to prepare for each member to create the desired patterns? This, in fact, is a profound problem in informatics which information scientists today still have no generic prescription to solve with.

The informatics problem addressed in amorphous computing by Knight et al. entails the design of a program under such conditions. Afterward, the researchers involved in this study shifted the direction of research to an approach that applied engineering methods to real living cells (such as yeast) to equip cells with specific functions [102]. Living creatures are the only known systems that take an amorphous approach—behaving according to genomes, not instructors—and function in an orderly fashion as a whole. Perhaps life can inspire a successful approach to amorphous programing. This is one of the hottest research areas in the bioengineering. The fabrication of functional particles is also currently a topic of intense interest in nanotechnology [103, 104].

Model

Here, we consider the NAC solvent nodes as active nodes and design the node program. The design is based on the complex functions of water molecules. We run the program to investigate how structures form in the network. As a target structure, we consider the two-dimensional square lattice.

We implement the experimental program in Java, where each node or edge is expressed as an instance of the node class or edge class in Java. These instances are linked to each other with instance variables. The node program is implemented as a method of the node class. The node instance has a pointer array hy [ ] of size 4 and can indicate up to four hy edges connected to other nodes.

Figure 3.31 shows the algorithm of the node program used to implement rewiring. The method ad_rand() selects an adjacent node via an hy edge; the

Fig. 3.31 The node program algorithm

```
public void conduct_nd_prog()
{
  // choosing pointer's displacement
  int dp = [either -1 or 1];

  // choosing this node's initial
  //   adjacent node randomly
  Node nda = ad_rand();

  // choosing the next node
  Node ndb = nda.ad_next(this,dp);
  if(ndb==null) return;

  // choosing the second next node
  Node ndc = ndb.ad(nda,dp);

  // choosing this node's another
  //   adjacent node
  Node nde = ad_next(nda,-dp);
  if(ndc==null OR ndc==this){
    if(nde==null) return;
    if(nde==ndb) return;

    remake_edge(ndb,nde);
    return;
  }

  // choosing the third next node
  Node ndd = ndc.ad_next(ndb,dp);
  if(ndd==this) return;

  remake_edge(this,ndc);
}
```

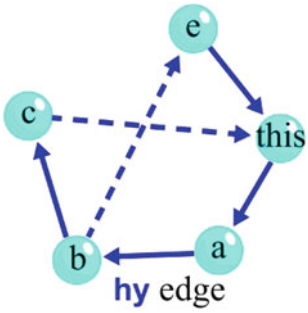

Fig. 3.32 Schematic diagram of the algorithm from Fig. 3.31. By creating the edge indicated by the *broken line*, the algorithm can create a tetragonal loop

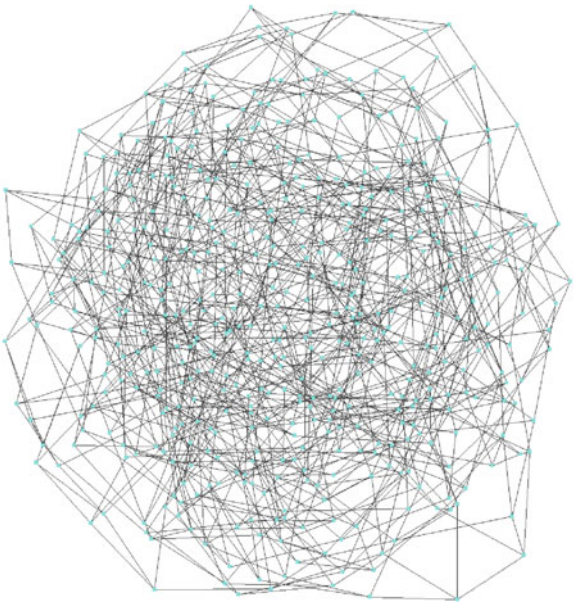

Fig. 3.33 Two-dimensional rendering of initial random graph based on the number of nodes $N = 512$ and uniform degree $K = 4$ (the cluster coefficient is $C \approx 0.0020$; the average path length is $L \approx 5.0$)

method ad_next(nd1, dp) returns an adjacent node via hy[p + dp] if nd1 is adjacent via hy[p]; the method remake_edge(nd0, nd1) generates a new hy edge between the two nodes nd0 and nd1 with a pointer; the variable dp $= \pm 1$ specifies the direction of the path search. Two adjacent hy edges in a loop are registered to locations that differ by dp with hy[ ] of the node between them. The Java operator "." represents a class member—for example, nda.xx specifies member xx belonging to nda.

Figure 3.32 is a schematic diagram illustrating the functions of the algorithm. This algorithm first extracts the path, nde–this–nda–ndb–ndc–ndd, with reference to the present node (this); it then creates the edge, ndb–nde or ndc–this, forming a loop of length 4.*Experimental results*. We start from a random graph with number of nodes $N = 512$ and uniform degree $K = 4$ and run conduct_nd_prog() K times in each iteration in each node. Figures 3.33, 3.34, and 3.35 show typical results.

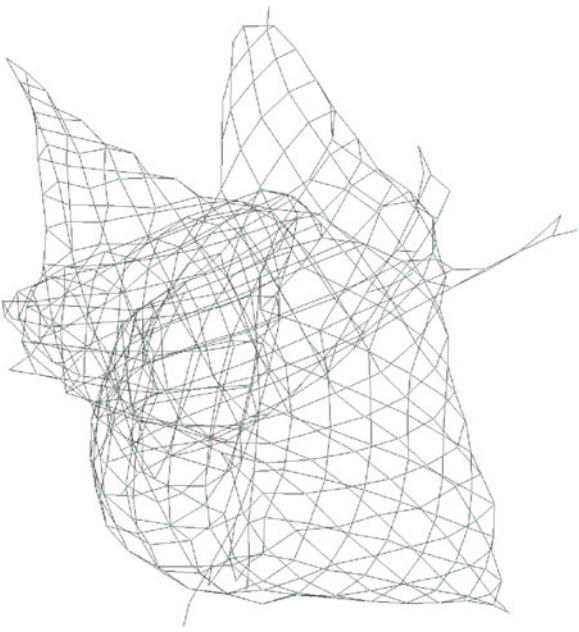

Fig. 3.34 NAC graph after 100 iterations (the average degree is $K \approx 3.81$, the cluster coefficient is $C \approx 0.0$, and the average path length is $L \approx 13.5$. Approximately 87% of the nodes have degree $K = 4$)

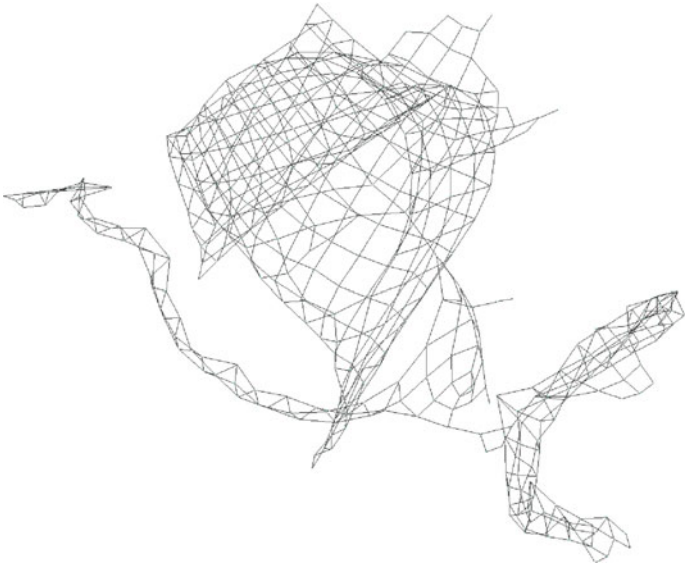

Fig. 3.35 NAC graph after 1,000 iterations (The average degree is $K \approx 3.84$, the cluster coefficient is $C \approx 0.0$, and the average path length is $L \approx 18.0$. Approximately 90% of the nodes have degree $K = 4$)

As shown in these figures, the rewiring by conduct_nd_prog() gradually creates regularity within the graph. After approximately 100 iterations, the graph assumes the structure of a folded two-dimensional square lattice sheet (Fig. 3.34). Beyond this stage, the sheet is gradually unfolded while maintaining regularity. After approximately 1,000 iterations, the graph assumes a structure with sheets and strings combined (Fig. 3.35). The algorithm conduct_nd_prog() specifies only the local links of the graph, not the overall structure. It does not distinguish between two-dimensional sheets and one-dimensional strings as the overall structure. (Several experimental results suggest that the current algorithm tends to converge the graph to a string rather than a sheet.)

3.6 Modified Network Artificial Chemistry

3.6.1 Concept

In the last part of the NAC studies described in the previous section, we implement a functional program at each node and demonstrate that the graph structure can be organized emulating the physical world. The program (Figures 3.31 and 3.32) used is implemented at each node instance as a class method. Interactions between nodes are implemented with reference to the nearby nodes ndc and ndd at a distance of 3–4. However, this program is not perfect with respect to the localization of processing. As described in Sects. 3.3 and 3.5.3, the intermolecular interactions in a solution work between encountering molecules—in other words, between a pair of nodes connected by an edge in NAC. A program that remotely operates the third- or fourth-neighbor nodes would violate this restriction.

In this section, we describe a study starting from a token [72] introduced into the network to resolve this problem. The term *token* refers to an information unit that transfers data and starts node operations in a DFC (see Column 6 discussed earlier). In this section, we add other functions to this token—which we will refer to as an *agent* hereafter—using it as an information carrier that transfers not just a "value," but a "program." As in the previous section, molecular functions are expressed in programs. However, these programs are carried by agents; they are not resident in each node. A program executes (firing or reaction) when it reaches a node, causing the rewiring of edges or the creation or deletion of agents.

If we modify the NAC in this way, how should we assign the correspondence between the nodes and agents and elements of the biological system? Through Sect. 3.5, we have assumed that a NAC node represents a molecule or group of atoms and that edges express the bonding or contact between the molecules. If the main body of the operation is not the nodes but the agents, if the nodes provide only a place for the agents to execute programs, perhaps agents—not nodes—would better represent molecules. Perhaps nodes are better interpreted as sites

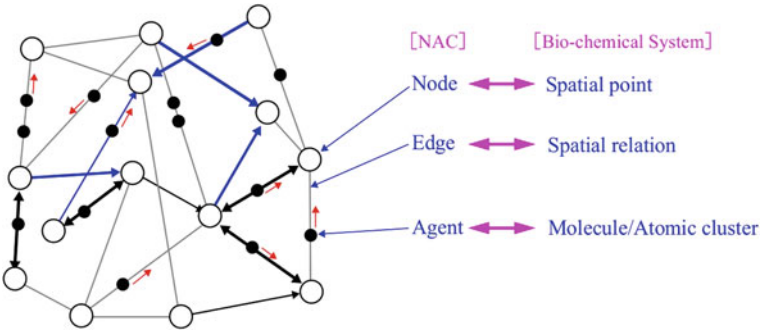

Fig. 3.36 Relationship diagram of modified network artificial chemistry (MNAC) and bio-chemical reaction system. (Contact and bonding relationship between molecular groups belonging to a spatial point.) A molecule or an atomic cluster is expressed as an agent moving in the network. A node represents a minute spatial region within which the molecular agent is located at a given point in time, and the edges collectively represent the contact or bonding relationship between the molecules or atomic clusters belonging to the spatial region

where molecular agents act. We will call an artificial chemistry model based on such a correspondence a Modified NAC (MNAC) to distinguish it from a conventional NAC. Figure 3.36 summarizes the correspondence between a graph in MNAC and a bio-chemical reaction system.

Moreover, if we regard the nodes as CPUs, edges as communication lines, and agents as programs, MNAC becomes a computational model that could be called "program-flow computing," in which many CPUs connected by communication channels execute numerous diverse programs in parallel. We discuss the significance and future research agenda of this model in Sect. 3.7.

In Sect. 3.6.2, as a typical example of MNAC studies, we describe the simulation of formation and splitting of a hydrophilic cluster using molecular agents.

3.6.2 Formation and Splitting of Hydrophilic Cluster by Molecular Agents

Here, we describe an example experiment in which the network structure is organized within the MNAC framework by the functions of the molecular agents [71, 73]. The experiment uses three types of designed agent programs (one each for rewiring the wa edges, rewiring the hy edges, and splitting the cluster). The programs are written as a sequence consisting of 45 machine instructions defined in advance. In the experiment, the programs fire and execute when the agents with the programs arrive at the nodes, resulting in the emergence of a hydrophilic cluster structure containing a pseudo-lattice structure in the graph, with a "centrosome" finally splitting the structure.

If we apply the scheme of artificial chemistry, the five elements stated in Fig. 3.2, to MNAC, the space is expressed by the network, the symbol is

expressed by the agent with a program, the rules for reaction of symbols are expressed by the computational functions of the program, and the rules for transport of symbols are expressed by the rule for restricting the movement of the agents. While no higher-level manager is in the picture at this point, we are likely to need one if we seek to develop MNAC into an evolving system. Based on this correspondence, MNAC is evaluated with reference to the emergence conditions (Sect. 3.2.2), topological conditions (Sect. 3.3.2), and topological properties (Sect. 3.3.4) discussed earlier. The results are summarized below. Here, we omit Topological Properties (5)–(7), since the edges in this MNAC do not represent the encounter and bonding relationships between molecules. Rather, they collectively represent the encounter and bonding relationships between agent groups belonging to spatial points and cannot be compared directly to the encounter network.

Emergence Condition (1): (Not met) A symbol can be independently created and removed by an agent.
Emergence Condition (2): (Met) There are no restrictions on the amount of genetic information held by the agent program.
Emergence Condition (3): (Met) Machine instructions that modify other agent programs can be prepared.
Emergence Condition (4): (Not met) Both genotypes and phenotypes are written as agent programs and do not require translation.
Emergence Condition (5): (Met) Splitting of clusters means walls.
Emergence Condition (6): (Met) The cluster size is variable.
Emergence Condition (7): (Partially met) The change in the order of clusters (mingling) does not occur naturally, but may be prompted by external disturbances.
Emergence Condition (8): (Met) The appropriate design of transport control for agents produces signal agents.
Emergence Condition (9): (Met) Randomness in agent movement may randomize the system.
Emergence Condition (10): (Met) The agent movement is transport-controlled by agent type and by edge type and direction.
Emergence Condition (11): (Partially met) Appropriate design of the transport control enables movement across walls.
Emergence Condition (12): (Not met) The symbol sequence is implemented as a program and cannot describe many-to-one interactions.
Emergence Condition (13): (Not met) Agents cannot combine with each other.
Topological Condition (1): (Met) A topological distance is defined in the graph between agents containing symbols.
Topological Condition (2): (Met) The topological distance changes with agent transport.
Topological Condition (3): (Met) A reaction is possible only between agents belonging to the same node.

Topological Properties (1)–(4): (Met) The hy network in the hydrophilic cluster produces a structure close to a pseudo-two-dimensional square lattice.

As shown here, the MNAC expressing molecules as agents rather than nodes gives molecules (complexes) significant mobility, whereas it fails to satisfy Emergence Conditions (12) and (13). Improving the MNAC to satisfy these conditions remains a problem for the future.

Graph Elements

The experimental program is coded in Java with nodes, edges, and agents implemented as classes. The nodes are classified into hydrophilic and hydrophobic types. The edges are classified into covalent (cv, directional), hydrogen (hy, directional), and van der Waals (wa, non-directional) types. The agents are classified into Type 0 (for splitting clusters, centrosomes), Type 1 (for rewiring the hy edges), and Type 2 (for rewiring the wa edges).

Figure 3.37 schematically illustrates the functions of these class instances with major variables and methods. Each instance has basic variables such as label and type and additional variables for the link. Changing these variables by executing the methods changes the link relationship in the graph.

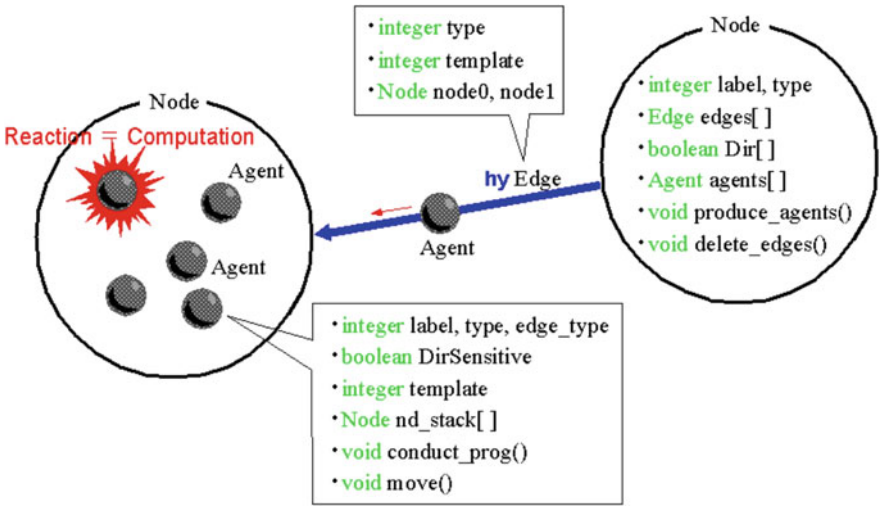

Fig. 3.37 Schematic diagram of nodes, edges, and agents in MNAC and their major instance variables and methods. The node variable, edges[ ], is a pointer to the edges connected to the node. The node variable, agents[ ], is a pointer to the agents belonging to the node. The edge variables, node0 and node1, are pointers to nodes on the end points of the edge. The agent stack, nd_stck[ ], is a pointer to nodes previously visited by the agent. The node methods, produce_agents() and delete_edges(), and the agent methods, conduct_prog() and move(), are methods of execution (see the main text)

Execution Operations

The simulation progresses by repeating the following four step operations:

Step 1: Node.produce_agents ()

(Agent creation by node):

This operation creates or removes the agents at each node to reach the target number specified in advance. The target number is separately determined according to node and agent types.

Step 2: Agent.conduct_prog ()

(Program execution by agent):

This operation executes the agent program, rewiring the edges.

Step 3: Node.delete_edges ()

(Edge cutting by node):

This operation cuts the edges so that the edge degree in each node does not exceed the upper limit. The upper limit of the degree is determined by node and edge types.

Step 4: Agent.move ()

(Agent movement):

This operation moves all agents from the current to the next node. The edge through which an agent passes is determined by agent variables: edge_type, DirSensitive, and template.

Among the four operations above, the three other than Agent.conduct_prog () are common to all nodes and agents. However, the algorithm executed by Agent.conduct_prog () depends on the agent type. Roughly speaking, Type 1 or 2 agents function locally to form hy-squares and wa-triangles, while Type 0 agents recognize differences between the labels of the other Type 0 agents in the same node, removing agents with different labels and cutting the edges through which they pass. For further discussion of this algorithm, see [71, 73].

The target number and upper limit degree used in Steps 1 and 3 specify the node type (spatial point). For example, the degree of the hy edge and the number of Type 1 agents for a hydrophilic node are 4, independent of the direction or template, and set to zero for a hydrophobic node. This specifies the hydrophobic nature of the node (the absence of hy edges). Table 3.4 summarizes the criteria used when the agent selects the edges in Step 4.

Table 3.4 Edge transport control of agents in Agent.move()

Edge type	Agent type		
	cv	hy	wa
Type 0	×	○	×
Type 1	×	○[a]	×
Type 2	○	○	○

The *circles* indicate that the agent can move through an edge of the type; the *crosses* indicate that the agent cannot move through an edge of the type

[a] Edge selection considering both direction and template

Experimental Results

Basic Setting

The experiment starts from an initial regular random graph (random graph with uniform degree) containing $N = 2{,}000$ nodes (1,400 nodes are hydrophilic, 600 nodes are hydrophobic). The degrees of the hy and wa edges are set to $K = 0$ and $K = 4$, respectively, in all nodes. Each of the 600 hydrophobic nodes is linked to a hydrophilic node with a cv edge to form 600 amphipathic dipoles in a manner analogous to biological lipid molecules. Since this experiment does not prepare operations for cutting and creating cv edges, these dipoles are completely maintained throughout the experiment.

After starting the execution operation, we prepare two Type 0 agents with different labels at the point of 100 iterations and substitute them into two randomly selected hydrophilic nodes. We observe the change in the graph topology as the operation executes.

Formation and Splitting of Hydrophilic Clusters with Pseudo-regular Structures

Figure 3.38 shows the typical results obtained. The Types 1 and 2 agents created and their creation and rewiring of hy and wa edges change the topology of the initial random graph shown in Fig. 3.38a. After 100 iterations, the graph assumes a shell structure in which hydrophilic nodes are densely connected by the hy network and accumulated at the center and hydrophobic nodes are pushed out to peripheral areas (Fig. 3.38b). Owing to the functions of the Type 1 agent, the hy network here has a structure close to a pseudo-two-dimensional square lattice with large average path length (L), forming the basis for the split in the next stage.

Although not shown here, each graph contains (a) 0, (b) 7,867, (c) 22,222, or (d) 28,449 molecular agents.

Figure 3.38b and the subsequent figures show the growth and percolation of the injected Type 2 agents. This agent behaves much like a centrosome in a living cell and clusters (splits) the hydrophilic nodes by their label values. This split occurs between iterations 150 and 250. Figure 3.38c shows the intermediate graph. The state with split clusters continues through to the final iteration, 500, until the simulation is stopped (Fig. 3.38d).

This experiment is performed ten times with different random number seeds. Figure 3.39 compares the graph for each case after 300 iterations projected onto a plane. As the figures show, the sizes of the clusters formed differ in each case, but all of the experiments produce the same qualitative results.

Although we do not show the resulting graph here, according to a separate supplementary experiment performed to investigate the function of the Type 1 agent, if only Type 1 agents operate in a graph consisting of $N = 512$ hydrophilic nodes, the average path length of the hy network formed can be as large as

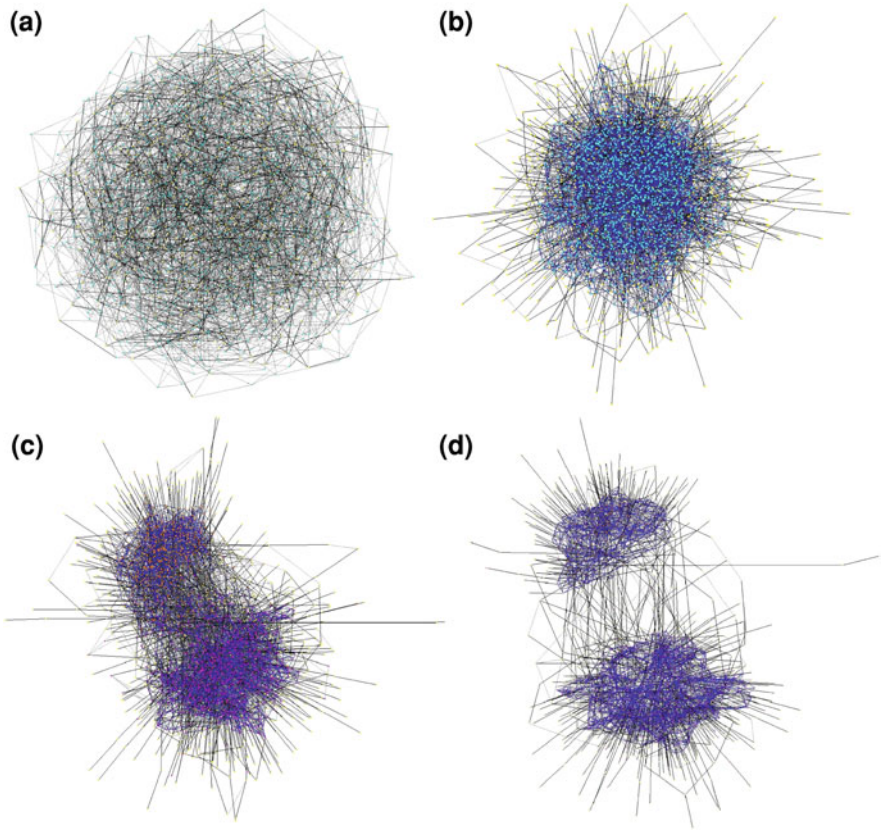

Fig. 3.38 Typical results from the experiment. **a** Two-dimensional rendering of initial regular random MNAC graph; **b** two-dimensional rendering of MNAC graph after 100 iterations; **c** two-dimensional rendering of MNAC graph after 180 iterations; **d** two-dimensional rendering of MNAC graph after 500 iterations

$L \approx 6.7$. This value exceeds $L = 6.01$, the value for a three-dimensional cubic lattice with the same number of nodes.

3.7 Future Prospects

This chapter discussed studies of artificial chemistry performed over the past 17 years, focusing mainly on two aspects: investigations and evaluations of design methodologies and progress in research on NAC. As described in Sect. 3.6.2, the network artificial chemistry (NAC and MNAC) systems obtained via this research remain incomplete—many problems remain to be solved before we can use this NAC to construct an artificial evolutionary system.

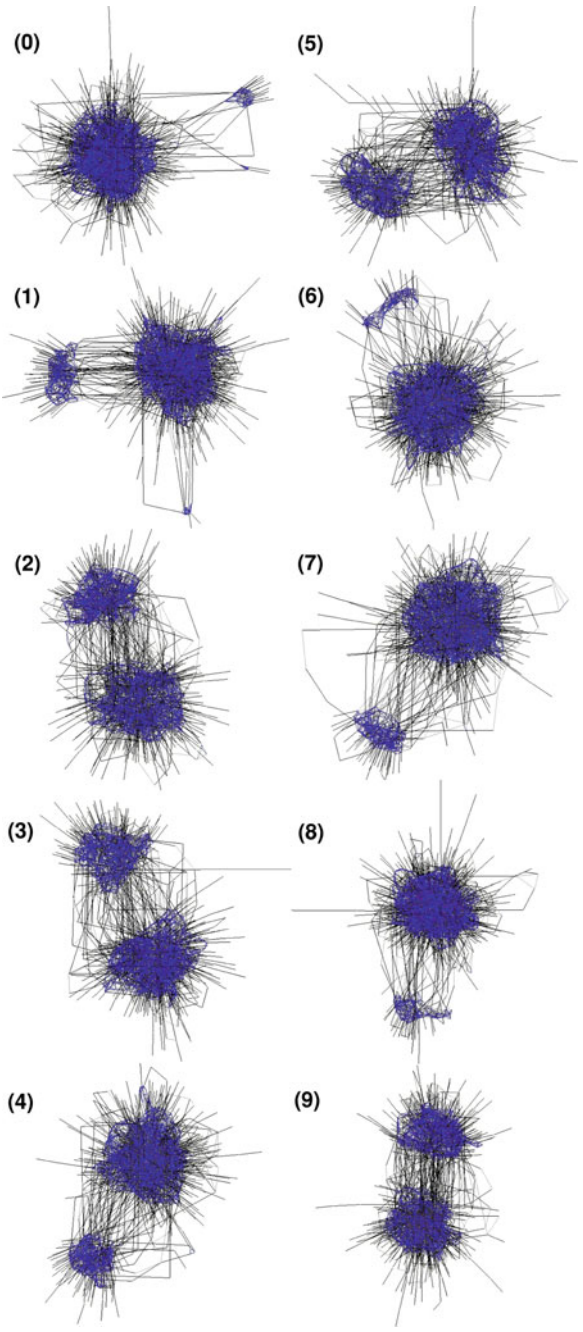

Fig. 3.39 Two-dimensional rendering of MNAC graphs after 300 iterations for each of ten experiments conducted with different random number sequences

In lieu of a conclusion, the last section of this chapter focuses on the new program-flow computing computational model, which has emerged as a byproduct of MNAC research, and describes future research problems and potential engineering applications.

In short, program-flow computing is a network computational model in which CPUs (nodes) connected by communication lines (edges) execute in parallel many programs that pass through them. This model differs from an ordinary parallel programing model in that the programs themselves are carried by the agents. This means the functions of the nodes or the operations performed in the nodes can change dynamically during execution. In this manner, for example, we can prepare various different types of agent programs, transport them in the network, and execute them on each node asynchronously, imparting complex functions to the entire network. We can design new engineering models for several systems based on program-flow computing. Some examples are given below.

3.7.1 Application to the Graph Coloring Problem

If the functions of the agents are selected appropriately, program-flow computing can be used to solve an optimization problem embedded in a network in which the agents operate. Figure 3.40 shows one example. Here, to solve the node coloring problem of a graph, a simple rule is prepared as the agent program: "If the color of the node to which the agent currently belongs is the same as the color of the node to which the agent belonged one step previously, the agent changes the color of the present node to another color." The figure shows how the movement and parallel execution of the agents change the node colors of the graph over time. This preliminary experiment shows that the initial randomly colored graph (Fig. 3.40a) gradually eliminates competition between the nodes through the execution of the agent program (Fig. 3.40b). In the end, the graph has different colors assigned to the ends of all edges (Fig. 3.40c). This example results in an optimal global solution for "graph coloring" based on the local selection by a single type of simple program. We believe we can apply a similar method to combinatorial optimization problems related to many other graphs.

3.7.2 Application to Neural Network Modeling

Another application and development of program-flow computing involves applications to neural network information processing. Neural networks, which began as mathematical models, are information processing models based on the signal processing that takes place in animal brains. Information processing within and between living neurons in the brain is basically performed by "molecules." Likewise, implementing molecular agents that perform processing while moving

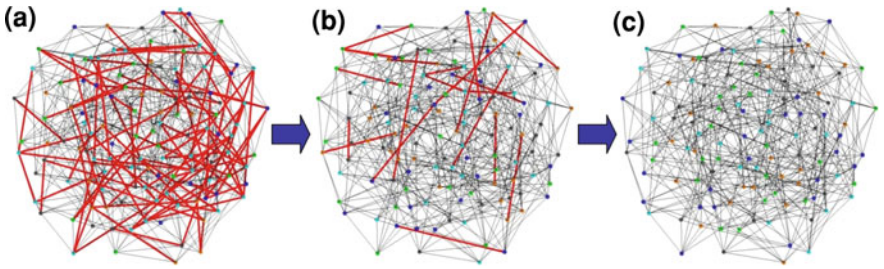

Fig. 3.40 Example of applying program-flow computing to node coloring problem of regular random graph. The problem is coloring the nodes of a regular random graph with the node number $N = 128$ and the uniform degree $k = 8$ using five predefined colors. **a** Is the initial state, **b** is the intermediate state, and **c** is the final state. In the figure, the *edges* indicated by heavy lines have nodes of the same color at both end points. The graph in the final state contains 1,024 molecular agents

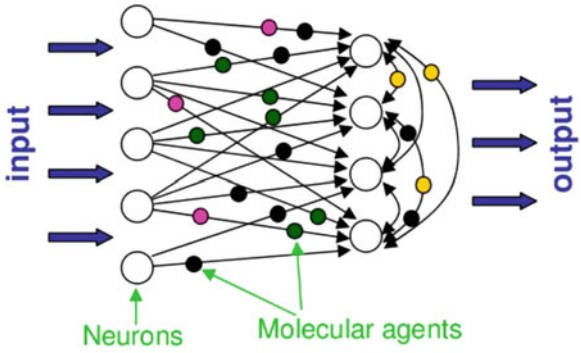

Fig. 3.41 Applying program-flow computing to neural network information processing. Various molecular agents are prepared, including those for the Hebbian rule, lateral inhibition, axon generation, and cell differentiation. The firing (reaction) and movement of these agents emulate signal propagation and learning in the neural network

between the nodes (neurons) via the edges (axons) in an artificial neural network may help generate an integrated account of processing related to signal propagation and learning (Fig. 3.41).

Our ongoing research efforts include building these models and evaluating their validity. One future goal is to propose new models for non-von Neumann information processing and information communication.

The Academic Frontier, Intelligent Information Science Project

The research described in this chapter was supported in part through the Academic Frontier, Intelligent Information Science (AFIIS) project entitled “The Clarification of Intelligence in Humans and Living Systems and its Applications”

(Representative: Mitsunori Miki, Professor at Doshisha University) at Doshisha University when the author was affiliated with both the ATR Network Informatics Laboratories (former Director, Dr. Katsunori Shimohara, now Professor at Doshisha University) and the NICT.

References

1. C. Adami, C.T. Brown, Evolutionary learning in the 2D artificial life system "Avida", in *Artificial Life IV: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systemsed*, ed. by R. Brooks, P. Maes (MIT Press, Cambridge, 1994), pp. 377–381
2. C. Adami, Learning and complexity in genetic auto-adaptive systems. Physica D **80**, 154–170 (1995)
3. C. Adami, *Introduction to Artificial Life* (Springer-Verlag, Santa Clara, CA, 1998)
4. C. Adami, C. Ofria, T.C. Collier, Evolution of biological complexity. Proc. Natl. Acad. Sci. U S A **97**, 4463–4468 (2000)
5. aiSee: Commercial software for visualizing graphs with various algorithms such as rubberband. http://www.aisee.com/
6. U. Alon, *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman & Hall (Crc Mathematical and Computational Biology Series, 2006)
7. Y. Asada, R. Suzuki, T. Arita, A simple artificial chemistry model based on the relationships between particles. In: Proceedings of the 36th SICE Symposium on Intelligent Systems, The Society of Instrument and Control Engineers (2009), pp. 227–232
8. G.M. Barrow, *Physical Chemistry* (McGraw-Hill Education, New York, 1988), Chapters 15–17. Japanese translation: Barrow, translated into Japanese by H. Daimon, K. Domen, Butsuri Kagaku Dai 6 Han (Ge) (Physical Chemistry, 6th edn.). (Tokyo Kagaku Dojin, Tokyo, 1999)
9. M. Bedau, P. Husbands, T. Hutton, S. Kumar, H. Suzuki (eds.) Workshop and Tutorial Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX). (2004)
10. P. Dittrich, W. Banzhaf, Self-evolution in a constructive binary string system. Artificial Life **4**, 203–220 (1998)
11. P. Dittrich, J. Ziegler, W. Banzhaf, Artificial chemistries-a review. Artificial Life **7**, 225–275 (2001)
12. P. Espanol, P.B. Warren, Statistical-mechanics of dissipative particle dynamics. Europhysics Letters **30**(4), 191–196 (1995)
13. R. Ewaschuk, P.D. Turney, Self-replication and self-assembly for manufacturing. Artificial Life **12**(3), 411–433 (2006)
14. W. Fontana, Algorithmic chemistry, in *Artificial Life II: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*, ed. by C.G. Langton et al. Santa Fe Institute Studies in the Sciences of Complexity, Vol. 10 (Addison-Wesley, 1992), pp. 159–209
15. W. Fontana, L.W. Buss, 'The arrival of the fittest': toward a theory of biological organization. Bull. Math. Biol. **56**, 1–64 (1994)
16. T. Gánti, Organisation of chemical reactions into dividing and metabolizing units: the chemotons. BioSystems **7**, 189–195 (1975)
17. J.A. Glazier, F. Graner, Simulation of the differential adhesion driven rearrangement of biological cells. Phys. Rev. E **47**, 2128–2154 (1993)
18. F. Graner, J.A. Glazier, Simulation of biological cell sorting using a two-dimensional extended Potts model. Phys. Rev. Lett. **69**, 2013–2016 (1992)

19. R.D. Groot, P.B. Warren, Dissipative particle dynamics: bridging the gap between atomistic and mesoscopic simulation. Journal of Chemical Physics **107**, 4423–4435 (1997)
20. N. Hirokawa, Kinesin and Dynein Superfamily Proteins and the Mechanism of Organelle Transport. Science **279**, 519–526 (1998)
21. P.J. Hoogerbrugge, J.M.V.A. Koelman, Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. Europhysics Letters **19**(3), 155–160 (1992)
22. T.J. Hutton, Evolvable self-replicating molecules in an artificial chemistry. Artificial Life **8**, 341–356 (2002)
23. T.J. Hutton, Information-replicating molecules with programmable enzymes, in *Proceedings of the Sixth International Conference on Humans and Computers (HC-2003)*, 2003, pp. 170–175
24. T.J. Hutton, Evolvable self-reproducing cells in a two-dimensional artificial chemistry. Artificial Life **13**(1), 11–30 (2007)
25. T.J. Hutton, The organic builder: a public experiment in artificial chemistries and self-replication. Artificial Life **15**(1), 21–28 (2009)
26. K. Imai, T. Hori, K. Morita, Self-reproduction in three-dimensional reversible cellular space. Artificial Life **8**(2), 155–174 (2002)
27. N. Ishii, K. Nakahigashi, T. Baba, M. Robert, T. Soga, A. Kanai, T. Hirasawa, M. Naba, K. Hirai, A. Hoque, P.Y. Ho, Y. Kakazu, K. Sugawara, S. Igarashi, S. Harada, T. Masuda, N. Sugiyama, T. Togashi, M. Hasegawa, Y. Takai, K. Yugi, K. Arakawa, N. Iwata, Y. Toya, Y. Nakayama, T. Nishioka, K. Shimizu, H. Mori, M. Tomita, Multiple high-throughput analyses monitor the response of E. coli to perturbations. Science **316**(5824), 593–597 (2007)
28. H. Kitano, Systems biology: a brief overview. Science **295**(5560), 1662–1664 (2002)
29. C.G. Langton, Self-reproduction in cellular automata. Physica D **10**, 135–144 (1984)
30. J.D. Lohn, J.A. Reggia, Automatic discovery of self-replicating structures in cellular automata. IEEE Transactions on Evolutionary Computation **1**, 165–178 (1997)
31. D. Madina, N. Ono, T. Ikegami, Cellular evolution in a 3D lattice artificial chemistry, in *Advances in Artificial Life (7th European Conference on Artificial Life Proceedings)*, ed. by W. Banzhaf, T. Christaller, P. Dittrich, J.T. Kim, J. Ziegler (Springer-Verlag, Berlin, 2003), pp. 59–68
32. T. Maeshiro, M. Kimura, The role of robustness and changeability on the origin and evolution of genetic codes. Proc. Nat. Acad. Sci. USA **95**, 5088–5093 (1998)
33. J. Maynard-Smith, E. Szathmáry, *The Major Transitions in Evolution*. Springer-Verlag, Berlin (1995). Japanese translation: J. Maynard-Smith, E. Szathmáry, translated into Japanese by K. Nagano, Shinka Suru Kaisou (Springer-Verlag Tokyo, 1997)
34. B. McMullin, F.R. Varela, Rediscovering computational autopoieses, in *Proceedings of the 4th European Conference on Artificial Life*, ed. by P. Husband, I. Harvey (MIT Press, Cambridge, MA, 1997), pp. 38–47
35. B. McMullin, D. Groß, D, Towards the Implementation of evolving autopoietic artificial agents, in *Advances in Artificial Life (6th European Conference on Artificial Life Proceedings)*, ed. by J. Kelemen, P. Sosik (Springer-Verlag, Berlin, 2001), pp. 440–443
36. D. Noble, *The Music of Life: Biology Beyond Genes* (Oxford University Press, 2008). Japanese translation: D. Noble, translated into Japanese by Y. Kurachi, *Seimei-no Ongaku – Genomu wo Koete System Biology Heno Shoutai* (Shin-you sha, 2009)
37. H. Noguchi, M. Takasu, Self-assembly of amphiphiles into vesicles: a Brownian dynamics simulation. Phys. Rev. E **64** (2001) 041913
38. N. Ono, T. Ikegami, Model of self-replicating cell capable of self-maintenance, in *Advances in Artificial Life (5th European Conference on Artificial Life Proceedings)*, ed. by D. Floreano (Springer-Verlag, Berlin, 1999), pp. 399–406
39. N. Ono, T. Ikegami, Self-maintenance and self-reproduction in an abstract cell model. J. theor. Biol. **206**, 243–253 (2000)
40. N. Ono, T. Ikegami, Artificial chemistry: computational studies on the emergence of self-reproducing units, in *Advances in Artificial Life (6th European Conference on Artificial Life Proceedings)*, ed. by J. Kelemen, P. Sosik (Springer-Verlag, Berlin, 2001), pp. 186–195

41. N. Ono, H. Suzuki, String-based artificial chemistry that allows maintenance of different types of self-replicators. The Journal of Three Dimensional Images **16**(4), 148–153 (2002)
42. T. Oohashi, H. Sayama, O. Ueno, T. Maekawa, Programmed self-decomposition model and artificial life. *Prodeedings of the 1995 International Workshop on Biologically Inspired Evolutionary Systems*. Sony CSL, Tokyo (1995), pp. 85–92
43. T. Oohashi, T. Maekawa, O. Ueno, E. Nishina, N. Kawai, Requirements for immortal ALife to exterminate mortal ALife in one finite, heterogeneous ecosystem, in *Advances in Artificial Life (5th European Conference on Artificial Life Proceedings)*, ed. by D. Floreano et al. (Springer-Verlag, Berlin, 1999), pp. 49–53
44. T. Oohashi, T. Maekawa, O. Ueno, N. Kawai, E. Nishina, K. Shimohara, Artificial life based on the programmed self-decomposition model: SIVA. Journal of Artificial Life and Robotics **5**, 77–87 (2001)
45. T. Oohashi, O. Ueno, T. Maekawa, N. Kawai, E. Nishina, M. Honda, An effective hierarchical model for the biomolecular covalent bond: an approach integrating artificial chemistry and an actual terrestrial life system. Artificial Life **15**(1), 29–58 (2009)
46. N.B. Ouchi, J.A. Glazier, J.P. Rieu, A. Upadhyaya, Y. Sawada, Improving the realism of the cellular Potts model in simulations of biological cells. Physica A **329**(3–4), 451–458 (2003)
47. A.N. Pargellis, The spontaneous generation of digital ''Life''. Physica D **91**, 86–96 (1996a)
48. A.N. Pargellis, The evolution of self-replicating computer organisms. Physica D **98**, 111–127 (1996b)
49. A.N. Pargellis, Digital life behavior in the Amoeba world. Artificial Life **7**, 63–75 (2001)
50. T.S. Ray, An approach to the synthesis of life, in *Artificial Life II: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*, ed. by C.G. Langton et al. Santa Fe Institute Studies in the Sciences of Complexity, Vol. 10 (Addison-Wesley, 1992), pp. 371–408
51. T.S. Ray, J. Hart, Evolution of differentiated multi-threaded digital organisms, in *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*, ed. by C. Adami, R.K. Belew, H. Kitano, C.E. Taylor (MIT Press, Cambridge, MA, 1998), pp. 295–304
52. H. Sayama, A new structurally dissolvable self-reproducing loop evolving in a simple cellular automata space. Artificial Life **5**(4), 343–365 (2000)
53. H. Sayama, Self-replicating worms that increase structural complexity through gene transmission, in *Artificial Life VII: Proceedings of the Seventh International Conference on Artificial Life*, ed. by M.A. Bedau et al. (MIT Press, Cambridge, 2000), pp. 21–30
54. H. Sayama, Swarm chemistry. Artificial Life **15**(1), 105–114 (2009)
55. A. Smith, P. Turney, R. Ewaschuk, Self-replicating machines in continuous space with virtual physics. Artificial Life **9**(1), 21–40 (2003)
56. P. Speroni di Fenizio, W. Banzhaf, Stability of metabolic and balanced organisations, in *Advances in Artificial Life (6th European Conference on Artificial Life Proceedings)*, ed. by J. Kelemen, P. Sosik (Springer-Verlag, Berlin, 2001), pp. 196–205
57. P. Speroni di Fenizio, P. Dittrich, W. Banzhaf, Spontaneous formation of proto-cells in an universal artificial chemistry on a planar graph, in *Advances in Artificial Life (6th European Conference on Artificial Life Proceedings)*, ed. by J. Kelemen, P. Sosik (Springer-Verlag, Berlin, 2001), pp. 206–215
58. H. Suzuki, An approach to biological computation: unicellular core-memory creatures evolved using genetic algorithms. Artificial Life **5**(4), 367–386 (1999)
59. H. Suzuki, Evolution of self-reproducing programs in a core propelled by parallel protein execution. Artificial Life **6**(2), 103–108 (2000a)
60. H. Suzuki, N. Ono, Universal replication in a string-based artificial chemistry system. The Journal of Three Dimensional Images **16**(4), 154–159 (2002)
61. H. Suzuki, N. Ono, K. Yuta, Several necessary conditions for the evolution of complex forms of life in an artificial environment. Artificial Life **9**(2), 537–558 (2003)
62. H. Suzuki, Artificial chemistry on small-world networks, in *Proceedings of the 18th Annual Conference of JSAI*, 2H4-03 (2004)

63. H. Suzuki, Spacial representation for artificial chemistry based on small-world networks, in *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (Artificial Life IX)* (2004), ed. by J. Pollack, M. Bedau, P. Husbands, T. Ikegami, R.A. Watson, pp. 507–513
64. H. Suzuki, Network artificial chemistry—molecular interaction represented by a graph, in *Workshop and Tutorial Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX)* (2004). ed. by M. Bedau, P. Husbands, T. Hutton, S. Kumar, H. Suzuki), pp. 63–70
65. H. Suzuki, Computational folding of molecular chains in network artificial chemistry, in *Proceedings of the 32th SICE Symposium on Intelligent Systems*. The Society of Instrument and Control Engineers (2005), pp. 383–386
66. H. Suzuki, N. Ono, *Statistical mechanical rewiring in network artificial chemistry. In: The 8th European Conference on Artificial Life (ECAL) Workshop Proceedings CD-ROM* (Canterbury, UK, 2005)
67. H. Suzuki, N. Ono, Network rewiring rules representing molecular diffusion, in *Proceedings of the 11th Emergent System Symposium (ESS) "Emergence Summer School 2005"*, Toyama, Japan. The Society of Instrument and Control Engineers (2005), pp. 127–130
68. H. Suzuki, Mathematical folding of node chains in a molecular network. BioSystems **87**, 125–135 (2007)
69. H. Suzuki, An approach toward emulating molecular interaction with a graph. Australian Journal of Chemistry **59**, 869–873 (2006)
70. H. Suzuki, A node program that creates regular structure in a graph, in *International Conference on Morphological Computation, Conference Proceedings*, March 26–28, 2007, ECLT, Venice Italy
71. H. Suzuki, A network cell with molecular tokens that divides from centrosome signals, in *Proceedings of the Seventh International Workshop on Information Processing in Cells and Tissues (IPCAT)*, ed. by N. Crook, T. Scheper (Oxford Brookes University, 2007), pp. 293–304
72. H. Suzuki, Structural organization in network artificial chemistry by node programs and token flow, in *SICE Annual Conference 2007 Proceedings*. The Society of Instrument and Control Engineers (SICE), Japan (2007) 1C10, pp. 884–889
73. H. Suzuki, A network cell with molecular agents that divides from centrosome signals. BioSystems **94**, 118–125 (2008)
74. H. Suzuki, P. Dittrich (eds.), Special Issue on Artificial Chemistry. Artif. Life **15**(1) (2009)
75. K. Suzuki, T. Ikegami, Shapes and self-movement in protocell systems. Artificial Life **15**(1), 59–70 (2009)
76. Y. Suzuki, S. Tsumoto, H. Tanaka, H, Analysis of cycles in symbolic chemical system based on abstract rewriting system on multisets, in *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, ed. by C. Langton, K. Shimohara (MIT Press, Cambridge, MA, 1997), pp. 521–528
77. Y. Suzuki, H. Tanaka, Order parameter for a symbolic chemical system, in *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*, ed. by C. Adami, R.K. Belew, H. Kitano, C.E. Taylor (MIT Press, Cambridge, MA, 1998), pp. 130–139
78. Y. Suzuki, H. Tanaka, Chemical evolution among artificial proto-cells, in *Artificial Life VII: Proceedings of the Seventh International Conference on Artificial Life*, ed. by M.A. Bedau et al. (MIT Press, Cambridge, 2000), pp. 54–63
79. Y. Suzuki, Y. Fujiwara, Y., J. Takabayashi, H. Tanaka, Artificial life applications of a class of P systems: abstract rewriting systems on multisets. *Lecture Notes in Computer Science*, Vol. 2235 (Multiset Processing) Springer, Berlin/Heidelberg (2001), pp. 299–346
80. E. Szathmáry, J. Maynard-Smith, From replicators to reproducers: the first major transitions leading to life. J. theor. Biol. **187**, 555–571 (1997)
81. K. Tominaga, Modelling DNA computation by an artificial chemistry based on pattern matching and recombination, in *Workshop and Tutorial Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX)*, ed. by M. Bedau, P. Husbands, T. Hutton, S. Kumar, H. Suzuki (2004), pp. 56–62

82. K. Tominaga, T. Watanabe, K. Kobayashi, M. Nakamura, K. Kishi, M. Kazuno, Modeling molecular computing systems by an artificial chemistry - its expressive power and application. Artificial Life **13**(3), 223–247 (2007)
83. K. Tominaga, Y. Suzuki, K. Kobayashi, T. Watanabe, K. Koizumi, K. Kishi, Modeling biochemical pathways using an artificial chemistry. Artificial Life **15**(1), 115–129 (2009)
84. J. von Neumann, *Theory of self-reproducing automata* (University of Illinois Press, Urbana. Edited and completed by A.W. Burks, 1966)
85. S. Yamamoto, Y. Maruyama, S. Hyodo, The dissipative particle dynamics study of spontaneous vesicle formation of amphiphilic molecules. Journal of Chemical Physics **116**(13), 5842–5849 (2002)
86. T. Yamamoto, K. Kaneko, Tile automation: a model for an architecture of a living system. Artificial Life **5**, 37–76 (1999)
87. Barrow G. M. Physical chemistry, McGraw-Hill Education, Chapters 15–17, 1988. Japanese translation: Barrow, translated into Japanese by Hiroshi Daimon and Kazunari Domen, Butsuri Kagaku Dai 6 Han (Ge) (Physical Chemistry 6th Edition), Tokyo Kagaku Dojin, 1999
88. Masahiro Kotani, Kiyohiko Someda, and Seiichiro Kouda, edited by Tamotsu Kondo, Daigakuin Kogi Butsuri Kagaku (Graduate School Course Physical Chemistry), Tokyo Kagaku Dojin, 1997
89. Vemulapalli G.K. Physical chemistry, Prentice-Hall Inc., Chapters 23–33, 1993. Japanese translation: Vemulapalli, translated into Japanese and supervised by Ueno et al., Butsuri Kagaku III Kagaku Hanno Sokudoron to Tokei Netsurikigaku (Physical Chemistry III Chemical Reaction Velocity Theory and Statistical Thermodynamics), Maruzen Co. Ltd., 2000
90. R. Albert, A.L. Bárabasi, Statistical mechanics of complex networks. Reviews of Modern Physics **74**(1), 47–98 (2002)
91. J. Davidsen, H. Ebel, S. Bornholdt, Emergence of a small world from local interactions - Modeling acquaintance networks. Physical Review Letters **88**(12), 128701 (2002)
92. Naoki Masuda, Norio Konno, Fukuzatsu Nettowaku no Kagaku (Science of Complex Networks), Sangyo Tosho, 2005
93. M.E.J Newman, The structure and function of complex networks. SIAM Review **45**, 167–256 (2003)
94. Wataru Soma, Katsunori Shimohara, Sumoru Warudo Nettowaku no Yakuwari (Role of Small-world Networks), Documents for Category II Meeting of Institute of Electronics, Information and Communication Engineers, NGN2001-12, 13–20, 2001
95. D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks. Nature **393**(6684), 440–442 (1998)
96. T.S. Ray, An approach to the synthesis of life, C.G. Langton , et al. (eds.). Artificial Life II: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (Santa Fe Institute Studies in the Sciences of Complexity, **10**), 371-408, Addison-Wesley, 1992
97. Yoichi Muraoka, Heiretsu Shori (Sofutowea Koza) (Parallel Processing (Software Course)), Shokodo Co., Ltd., Section 5.7, 1986
98. Sharp J.A. Data flow computing (Ellis Horwood Series in Computers and Their Applications), Ellis Horwood Ltd., 1985
99. Masahiro Sowa, Deta-furo Mashin to Gengo (Sofutowea Koza) (Data flow Machine and Languages (Software Course)), Shokodo Co., Ltd., 1986
100. Ryota Shioya, Hidetsugu Irie, Masahiro Goshima, Shuichi Sakai, Evaluation of Area-Oriented Register Cache, Proceedings of Information Processing Society of Japan, 2008-ARC-178, pp. 13–18, 2008
101. Abelson H., Allen D., Coore D., Hanson C., Homsy G., Knight T.F., Nagpal R., Rauch E., Sussman G.J., Weiss R. Amorphous computing, Communications of the ACM, vol. 43(5), pp. 74–82, 2000

102. The Bio FAB Group, Baker D., Church G., Collins J., Endy D., Jacobson J., Keasling J., Modrich P., Smolke C., Weiss R. Engineering life: building a fab for biology, Scientific American, vol. 294(6), pp. 44–51, 2006. Japanese translation: Bio FAB Group, Baker D., Church G., Collins J., Endy D., Jacobson J., Keasling J., Modrich P., Smolke C., Weiss R., Gosei Seibutsugaku wo Kasokusuru Baio Fabu, Nikkei Science, vol. 36 (9), pp. 32–41, 2006
103. Katsuhiko Ariga, Toyoki Kunitake, Iwanami Koza Gendai Kagaku eno Nyumon <16>Chobunsikagaku eno Tenkai (Iwanami Lecture Series, Introduction to Modern Chemistry<16> Development of Supermolecular Chemistry), Iwanami Shoten, 2000
104. Seiji Shinkai, Kazuki Sada, Masayuki Takeuchi, Norifumi Fujita, Bunshi Kikai: Seitai wo Tegakari to shite (Molecular Machine: Using Living Bodies as Clues), Edited by Naoki Sugimoto, Kagaku Furontia 13 Nano Baio Enjiniaringu - Seimei to Busshitsu no Yugo wo Mezasite (Chemistry Frontier 13 Nanobioengineering—Toward a Fusion of Life and Materials), Kagaku-Dojin Publishing Company, Inc., 50–60, 2004