博士学位論文

Evolution of Code and Communication in Dynamical Networks

（動的ネットワークにおける
コードとコミュニケーションの進化）

1996 年 3 月

東京大学大学院総合文化研究科

橋本　敬

(Takashi Hashimoto)

博士学位論文

Evolution of Code and Communication in Dynamical Networks

(動的ネットワークにおけるコードとコミュニケーションの進化)

1996年3月

東京大学大学院総合文化研究科

橋本 敬

(Takashi Hashimoto)

Contents

Acknowledgments

I record the warmest thanks to Associate Professor Takashi Ikegami, the supervisor at my doctoral course. All studies in this thesis was done in collaboration with him. Many of the ideas in these works were shaped in discussions with him. Talks with him about not only these studies but also other variable subjects gave me a great insight to complex systems.

I also would like to thank all the members and alumni of Ikegami's school, Eken S. Yoshikawa, Kouji Harada, Shin'ichirou Dora Nishimura, Kouji Kishi, Daisuke Komatsu, Masanao Matsushima, Dr. Hajime Anada and Dr. Noriyuki Matsuo. I am deeply grateful to Yukito Iba. He usually gives me many enlightening comments and valuable criticism. I thank all participants of KISSeminar, hold on every Thursday at my college, Professor Kunihiko Kaneko, Associate Professor Takashi Ikegami, Associate Professor Shin'ichi Sasa and their students. I give my thanks Associate Professor Yoshiki Nishimura. Prof. Ikegami and I often had discussions on language and cognition with him. They are very useful for the study reported in the second chapter. I also thank him for his critical reading of our papers.

I would like to record my thanks to the judging committee of this thesis, Professor Shun-ichi Amari, Associate Professor Takashi Ikegami, Professor Kunihiko Kaneko, Associate Professor Jun Makino and Associate Professor Kazunori Yamaguchi. Their comments and criticisms have made this thesis better.

The workshop "Complex Systems" has been sitting at Kyoto on every year. I began studying on complex systems by stimulated by discussions developed at the workshops about approaches to complex systems and their achievements. I give my thank organizing committee and discussants of the workshops for such interesting meetings. I organized a workshop entitled "Toward Construction and Description of Dynamical Systems" in December, 1994 with Tomoyuki Yamamoto and Akihiro Yamaguchi. Preliminary discussions with them gave me chances to know not only many superior researchers but also hardships to planning a workshop. The attendant and we could have fruitful discussions about dynamical view of nature. I express my gratitude

for the co-organizer, speakers, and all the attendants of the workshop. I indebted to Ayumu Yasutomi for interesting conversations about economical and social systems. I am also thankful Professor Keisuke Ito. He gave me a place to study these interesting topics at first. I also thank Associate Professor Yukio Pegio Gunji and members of Prof. Ito's school. I thank Professor Tonegawa and Associate Professor Kaburagi. They taught me statistical physics through my master course study.

Discussions about cultural anthropology, social science, philosophy at "Balinese Culture seminar" were stimulated me. I thank the members of the seminar, Professor Keiji Ueshima, Toru Momo, Mitsue Nakamura, Kumiko Yagihara and the others.

I offer hearty thanks my parents for giving me both mental and economic assistance. I am thankful to Kumiko Yagihara from the bottom of my heart for her encouragement and fruitful conversations.

Publications

- "Keisan Nouryoku no Shinka to Shudan no Keisei (Evolution of Computational Ability and Forming Groups) (in Japanese)"
  Takashi Hashimoto
  Suri Kagaku (1994) Vol. 32, No. 2, pp. 39-43
  (Chapter 2)

- "Keishiki Bunpou Sisutem no Nettowaku to sono Arugorizumikku na Shinka (Network of Symbolic Grammar Systems and Their Algorithmic Evolution) (in Japanese)"
  Takashi Hashimoto
  Bussei Kenkyu (1994) Vol. 61, No. 5, pp. 415-421
  (Chapter 2)

- "Communication Network of Symbolic Grammar Systems"
  Takashi Hashimoto and Takashi Ikegami
  in *Proceedings of the International Conference on Dynamical Systems and Chaos*, vol. 2, pp. 595–598
  Y. Aizawa et al. (eds.), World Scientific, Singapore, 1995
  (Chapter 2)

- "Evolution of Symbolic Grammar Systems"
  Takashi Hashimoto and Takashi Ikegami
  in *Advances in Artificial Life*, pp. 812–823
  F. Morán et al. (eds.), Springer, Berlin 1995
  (Chapter 2)

- "Emergence of Net-grammar in communicating Agents"

Takashi Hashimoto and Takashi Ikegami
BioSystems, 1996 (in press)
(Chapter 2)

- "Coevolution of Machines and Tapes"
  Takashi Ikegami and Takashi Hashimoto
  in *Advances in Artificial Life*, pp. 234–245
  F. Morán et al. (eds.), Springer, Berlin 1995
  (Chapter 3)

- "Active Mutation in Self-reproducing Networks of Machines and Tapes"
  Takashi Ikegami and Takashi Hashimoto
  Journal of ALife, (submitted)
  (Chapter 3)

These papers are available via WWW.
URL is `http://infidel.c.u-tokyo.ac.jp/~toshiwo/work.html`.

Chapter 1

Introduction

We study *emergence and dynamics of codes* by constructing abstract models in computers. A term *code* is defined as a set of rules which restrict the usages and interpretations of symbols in communication and expressions. A code of a system is organized either by a supervisor or by the system itself spontaneously. We are interested in the case of self-organization, a code emerging from interactions among agents. Two characteristic self-organized codes, linguistic code and genetic code, will be discussed thoroughly in the thesis.

Codes are often studied from the view points of both syntactic and semantic. The former studies connections among symbols and the latter does relations between each symbol and its referent. However, a code in a system must be organized for the system to work in practical situations. Thus, we must study organization or evolution of codes from pragmatic viewpoint, that is interpretations of and responses to information in practical situations. If we consider pragmatic case, the syntactic and semantic aspect of a code cannot be separately treated. Namely, we face with a problem of syntax and semantics mixture.

If many agents communicate, other problems may occur. One's interpretations of information may run counter to that of others. Code has to be fluctuating temporally by the actions of the agents. We see paradoxical nature of code, *syntax and semantics mixture* and *openness of code*.

Nowadays we often communicate through computer networks. Here we know that a sender and a receiver of messages share a common code to exchange the messages. The code used in computer networks is prescribed in a static way. It is a mere agreement at the syntactic level. We encode a message on symbols and decode symbols to get a message by following the prescribed code. This syntactic level of code was first studied by Shannon and Weaver [91]. They discussed its efficiency and capacity under noisy environment.

A lot of grammatical and lexical knowledge is input to computer systems in order to process natural language. The knowledge is taken as a code in natural language — the grammatical knowledge constitutes a code at the syntactic level and the lexical knowledge constitutes a code at the semantic level. However, processing natural language by computers turn out to be a very difficult issue. Usages and meanings of words in practical situations are not decided completely by given grammatical and lexical knowledge. Context dependency with synonymy and polysemy makes processing natural language be difficult. Mixture of syntax and semantics can be often seen. For example, Chomsky said that an active sentence can be transformed into a passive sentence without changing the meaning. But transformation between the active and passive sentences changes the meaning in practical situations. Interpretations and impressions become different for listeners. Moreover, we notice that creation of new words and phrases is more important characteristics of natural language. Such innovative nature of language cannot be brought about by mere grammatical and lexical knowledge.

As Wittgenstein said [106], rules in natural language can not be determined in advance. In practical communication, we just speak. A code does not exist from the beginning. Indeed, there is an arbitrariness to some degrees in usages of words or interpretations of information. The arbitrariness in a system undergoes restrictions by validities peculiar to the system. We can say anything, yet scarcely do we speak extraordinary ways in practical conversations. Hence we have smooth conversations. Thus we feel the existence of a tacit linguistic code. We can refer to a code only after conversations. It may change, however, after new conversations. For example, peculiar expressions or trend words are often created in a certain community, without mutual confirmation. Linguistic codes can only be described after events, and they dynamically change. What we must ask is how to understand and model the emergence and dynamics of the linguistic code.

Similarities between sentences in natural language and genetic sequences in genomes are often discussed (e.g. [81, 26, 95]). In the genetic systems, each codon is related with amino acid. Even if we know the relationships completely, we cannot understand how proteins act in real situations. Relations between genotype and phenotype constitute a genetic code in a pragmatic level (see Fig. 1.1). We cannot prescribe the code at this level as the linguistic code. Phenotype varies from individual to individual even with the same gene (incomplete penetrance). The phenotype can also vary through interactions with other genes and environmental factors (variable expressivity). Genetic codes have to be studied through the pragmatic levels.

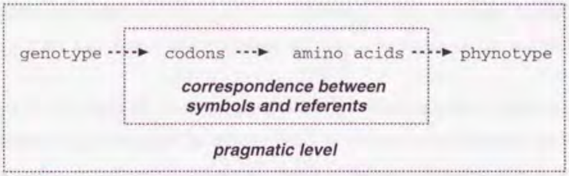


Figure 1.1: Relations between each codon and amino acid is correspondence between each symbol and its referent. It must be considered through relations between genotype and phenotype.

There is a general agreement that the genetic systems utilize the universal code over all creatures. Recently, different correspondences between codons and amino acids are found experimentally (e.g. [5, 62]). The fact conceives us that the genetic code can have dynamics as well as genetic code at the pragmatic level [89].

If systems with different code come to interact with, *mixture* of codes will be revealed. It causes interferences and conflicts between codes. And it may bring about dynamics of code.

For the linguistic examples, there are many different local and social dialects to coexist. In early stages of language acquisition process, children may have different codes from adults'. Children learn to modify their code, however, to incorporate into adults' in the course of development. Mixture of codes can be an innovative source for a new code. Examples of new codes can be seen in pidgin and creole languages. Pidgins are new languages which are invented for communication with each other, when more than two languages contact. Creoles are developed languages from pidgins by lexical and grammatical extensions to smoothly communicate with each other and to naturally express one's thought. Developments of child's language and pidgin/creole languages will be discussed in §2.1.2.

There might be mixture of codes in genetic systems. Suppose that a gene is interpreted as "produce a protein X". But, according to a different code, it is interpreted as "produce a protein Y". If the protein Y inhibits the protein X, these two codes generate contradiction. When a procaryotic cell and mitochondria fuse to form a eucaryotic cell, such mixture of codes can happen. One side of code will be driven away, or a new code will be invented.

Even within a single code, contradiction can be induced. Suppose that a protein X translates a gene to "protein Y which inhibits X". Such code induces a logical contradiction. By the logical contradiction the code can be improved. We refer to such nature of code as *openness*. It is related with a proposition "$A = \neg A$" or a sentence "This sentence is not valid." Such inconsistency in code can result in oscillating dynamics as Spencer Brown has first pointed out [11].

The mixture and openness of codes are inevitable characteristics of what is defined as a code.

We will propose a language game in communication network to study emergence and dynamics of linguistic code. A network consists of simple symbol processing agents with individual grammars. Agents speak and accept words in terms of their own grammars. We are interested in organizing codes in network through communication among agents. In other words, whether a specified usage of words can be selected in the network is our main concern. As we have discussed, it is impossible to determine rules of the game or individual grammars in advance. However, we can study the global dynamics of code in the communication network of individual grammars.

We assume in our language game that the more various words an agent can recognize or speak, the more the agent can score. Because recognizing words is related with the capability of information processing. How variety of words and computational ability of individual grammars change will also be discussed in detail.

We will see an emergence of groups with particular usages of words. It can be taken as an emergence of community code. We will also see evolution of computational ability. Two remarkable evolutionary processes will be observed. One is a module type evolution, the other one is a loop forming evolution. Mixture of codes, which originates in conflicts between two groups, disturbs smooth evolution to higher computational ability. Evolution of computational ability shows punctuated equilibria: emergence of higher computational ability agents follows by the community code.

As for the study of genetic code, we will propose a network model of machines and tapes. One primitive role of genetic code is replication. As von Neumann pointed out [102], self-reproduction requires separation of machine and its description tape due to self-referential problem. A system must observe itself to self-reproduce. If a system is unstable against its observation, self-reproduction is impossible. By introducing a stable description tape, von Neumann constructed a self-reproducing system in two dimensional cell automata. To study genetic code through a pragmatic level, we will discuss how the machines and tapes behave in the network for self-reproduction, and how the tapes are interpreted by the machines.

We introduce circular binary strings as description tapes, and machines as decoders and products of the tapes. Machines can only be reproduced by being read its description tape. We will see evolution from a minimal self-replicating network to a large complex network. To stably sustain the large network, a *core network* should appear. In the core network, each one of machines is produced by other machines. Not only each population of machine and tape but also network topology oscillate in the course of time. We will see the oscillation will be induced by openness of code.

Chapter 2

Evolution of Grammar Systems

2.1 Introduction

2.1.1 Language as an Evolutionary System in Ensemble Structures

Linguistic expressions look quite complex but look far from random. Therefore we feel that there exists a linguistic code. It is said that a grammar of a given language decides a structure of expressions. A grammar constitutes a part of the code. It is determined in part by the community which uses it. It is also commonly assumed that one has to have internal knowledge of one's language when one can derive and recognize appropriately structured expressions. Internal knowledge varies from person to person. We henceforth will refer to individual internal knowledge as an individual grammar. An individual grammar can undergo changes under physical and cultural environment and in its interaction with other grammars.

Language reflects historical factors such as evolutionary paths and growing processes, as shown in the next subsection. Therefore language can be treated as an evolving system. For these reasons, we have to discuss how a grammar evolves in an ensemble of individual grammars and how its complexity evolves.

It is often assumed that the complexity of language is mere reflection of the complexity in the world we live in, just as the complexity of living systems is said to be the reflection of their complex environments. But language is a highly autonomous system, with its own evolutionary dynamics.

MacLennan [72] has studied communication among simple rule agents. Agents exchange information about the local environments in which they live by emitting signals to each other. Fixed signals serve as names of objects which an agent encounters in its environment. Different names may correspond to the same object, resulting in the emergence of synonymous. Complex naming of objects reflects the complexity of the external environment. In Werner and Dyer's model [103], diversity of language is attributed to spatial inhomogeneity of the environment where agents live.

However, we believe that even without complex space or information, grammars can evolve and diversify by intrinsic mechanism. In general, evolving systems, such as evolutionary games [71, 50], Tierra world [84, 85], network of tapes and machines [52, 51] which is studied in the next chapter, constitute notable examples of evolutionary mechanisms. Each example has its own diversifying mechanisms such as host-parasite dynamics and self-referential paradox. If linguistic expressions are not mere labels of external objects, communication helps to develop grammar structures [1].

We use our language not just to describe entities and states of affairs in the external world but also to express our own thoughts. Language must be complex enough to express complex thoughts or concepts. On the other hand, language is used not only in order to communicate with others (exo-language), but also to construct one's "internal models" (endo-language). We regard grammar systems as representations of internal models. Exo-language also reconstructs internal models of both speakers and listeners. Language influences thought and vice versa; complex thoughts make language complex and complex language is conducive to the formulation of complex thoughts. To make false statements gives impetus to complex language: speaker must produce complicated expressions or encipher sentences to make it difficult for others to understand them. Listeners must recognize such complex expressions to correctly receive information. Communication therefore can be thought of as a source of rich grammar structures.

To study languages in evolutionary and network context, we consider a simple language game. Each player has his own grammar and communicates with each other by sending sentences. Player takes turns to speak and listen to the sentences. Each player's advantage is counted by what kind of sentences he speaks, how fast he recognizes sentences of the other and how one's sentences are recognized by other players. A player who get lower scores has to change their grammar and the grammars of players with higher scores are likely to be hereditary.

According to N. Chomsky [16], the computational ability of symbolic grammar is categorized into four classes:

type 0 phrase structure grammar

type 1 context sensitive grammar

type 2 context free grammar

type 3 regular grammar.

The higher a grammar is ranked in the hierarchy, the larger a set of words it can generate. To take a simple example, a word set,

$$\{0^n 1^n | n \geq 1\}, \tag{2.1}$$

can not be derived by a regular grammar (type 3) but can be derived by a context free grammar (type 2) and by any grammar higher in the hierarchy. Here a symbol xy is a concatenation

---

[1] Since our model has no external environments, naming of external objects lies outside the scope of this paper. But we will show that the diversification of words and grammars still occur.

of symbols x and y and x^n represents n times concatenation of symbol x. Hence a set (2.1) includes all strings beginning with any non-zero number of 0s followed by an equal number of 1s.

Evolution of grammar will be discussed as a process of escalation in the hierarchy. From a practical point of view, we have to take finite lengths of words and finite derivation steps into account. If we deal only with a finite set of words, e.g. $\{0^n1^n|N \geq n \geq 1\}$, Chomsky's hierarchical relationship does not always hold. In computation theory, there are no upper bounds to how much time can be put into deriving words and no ensemble structure is considered at all. We need to figure out what kind of grammar has a practical ability to derive and accept words in finite time steps. The computational ability of a symbolic grammar and hierarchy should be studied within an ensemble. Chomsky [17] said that regular grammars are not suitable for modeling the grammars of natural languages. We assume that the primitive structure of a grammar is a regular grammar and will see how it develop into higher grammars.

In the next subsection, we will review examples of evolution of complexity in linguistic systems.

2.1.2 Evolution of Linguistic Complexity

Evolution of language would be divided to two problems. One is the emergence of language, the other is evolution of linguistic complexity. The former is too complicated problem. To understand it we must investigate many subjects as well as linguistics. By the fact that the evolution of brain especially asymmetry of hemisphere is needed to deal with complex and abstract information, we must develop peleoneurology as well as conventional brain science [69, 34]. Since to utter vowels in a loud voice and articulated consonants need developments of vocal tract and facial muscle [69], we have to know the evolutionary path by paleobiological studies. It is said that evolution of stone tools is related with the evolution of language [70]. Teaching complex processes of making tools, e.g. Neanderthal's, seems impossible without language [34,4]. Language represents its own culture. Hence archaeological evidences should be examined. Communication is shown to be specific not only to human society but also to many other living things. There seems to be symbolic behaviors in some animals [64, 40].

On the other hand, development of lexical and grammatical complexity after establishment of language provides different but interesting subjects to study. Historical linguistics, child's acquisition of language and pidgin and creole languages are the examples [19].

Pidgin languages are the best examples of inventing new codes induced by mixture of codes. They are new languages when two or more languages fuse [99]. They are not mother tongues of speakers of pidgins. They have their own first language. For example, when superstratum language spoken by European overseers met substratum languages spoken by African slaves on Caribbean plantations, a relatively simpler new language was invented in order to communicate with people from other areas [108, 29].

Romaine reported some examples of simplification, reduction, restructuring and mixture of lexicon and grammar found in the pidginization processes [87]. Bickerton classified grammatical morphemes to two groups with respect to their robustness [10]. Ones are easily reconstructed morphemes if they are lost, e.g. tense, aspect, modality forms, question words, and irrealis complementizer. The others are hardly reconstructed morphemes if they are lost, e.g. gender agreement, number agreement and most prepositions. The former can give a primitive form of grammar. The evolution from grammar with the former type to more complex one including the latter type would have proceeded.

Creole languages are stabilized and extended ones from pidgins. Different from pidgins They are the mother tongues of speakers. Creolization usually proceeds through several generations. A good example is Tok pisin. Creole languages may approach to their superstratum languages after creolization, if the superstratum languages have large effects to the creoles — for example, when the superstratum language is used for education or broadcast. The approaching process from a creole to its superstratum language is referred to as decreolization. Languages between them are called post-creole continuum [99]. Pidginization, creolization and decreolization dynamics is illustrated schematically in Fig. 2.1.

Creolization can occur from any degrees of pidginization [87]. There are languages called "radical creoles". A radical creole is formed in one generation in a society where some pidgin languages are spoken. It is different from any languages previously spoken in the society. An example is Hawaii creole. Bickerton found that radical creoles have similar characters in common even though many of them are geographically separated. For example, they show simple morphologies, often no morphology at all. Three particles expressing anterior, irrealis, progressive used as auxiliary verbs are another example of the common character [9]. It is very suggestive, since we can imagine that language evolves in an invariant fashion. Bickerton argued that when children are exposed only insufficient model for acquisition, an innate syntactic structure arises. He called it "bioprogram" and asserted it was a candidate for the primitive

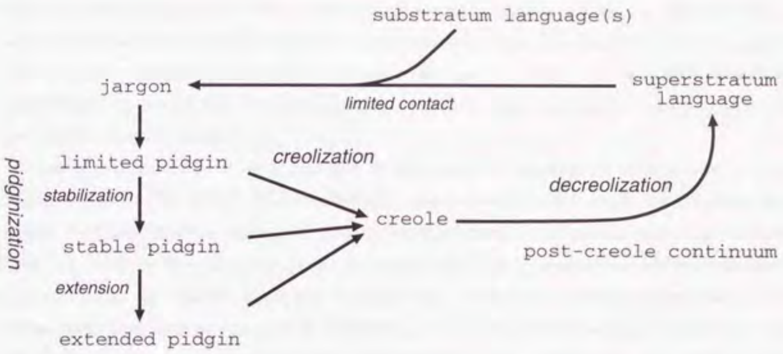


Figure 2.1: pidginization, creolization and decreolization processes

syntax at the emergence of language.

Creoles share some characters with children's language. For example, children often use verbs without inflection. Another example is the fact that they often do not exchange auxiliary verb and the subject word in an interrogative sentence – e.g. "Where I can put it?". Such sentence is normal in most creoles. Distinction between state and process is another example. In creoles, the progressive particle is by no means used with process verbs as "love" or "like". Children acquire this distinction very early. They scarcely say as "I am liking her".

Let us see development of children's language.

At first, children can utter only a word, which is called one word stage. They can recognize adult's syntactic structure, although they cannot speak syntactic structured sentences. It can be said that they have a passive grammar system [80]. At the two word stage, there is no unstressed morpheme as prepositions or auxiliary verbs [1]. Sentences in this stage called "telegraphic speech" [12]. Two main words are extracted from adult's sentences and are combined maintaining their word order in the adult's sentences.

Thereafter, children begin to use some morphemes. Brown reported learning process of 14 grammatical morphemes in English [12]. Present progressive, plural and locative prepositions are mastered in early stages of development. Contractible copula and contractible auxiliary are, on the other hand, acquired at last stages. Although the order is not held always in other language, acquisition order of several morphemes suggests a principle: simple rules are used at first and then complex rules are used in rough ways. To understand this acquisition order, several notions of complexity are proposed – semantic complexity (e.g. [13]), derivational complexity (e.g. [13,32,15]), formal complexity (e.g. [94]), and conceptual complexity (e.g. [18]) (see also Andersen's paper [1]).

Overregularizations have been reported in the course of acquisition of language by cross-linguistic studies [27,93,8,7]. Children typically acquire inflections through the following steps: at first, no inflection; then infrequent and appropriate use of both regular and irregular inflections, e.g. walk-walked, go-went; next, overregularized form – irregular words are inflected as regular words, e.g. "goed" for a past tense of "go"; at last, completely correct use. Pinker interpreted these facts as follows [79]. Children, at first, do not catch a rule of inflection. They only memorize words. Afterword they learn the rule for regular inflections and wrongly apply to irregular words. Thereafter, they acquire whole inflection rules. These facts show that children can generalize underlying rules from finite input and extend from simple to complex rules.

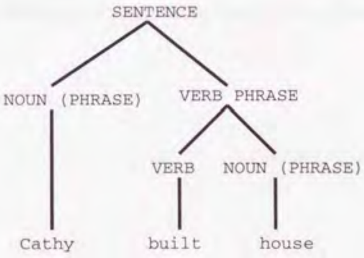


Figure 2.2: A hierarchical structure of a child's sentence

The next stage of development is the emergence of hierarchical structure of sentences. When children speak a sentence, they often say a part of the sentence at first, e.g. they speak a verb phrase, then speak the whole sentence. For example, they say "built house" immediately before speaking "Cathy built house." This fact suggests that their utterances are constructed as in Fig. 2.2. Beside, there is a fact that they never breathe in noun phrase. They seem to regard a noun phrase as a unit. It also suggests that their sentences are hierarchically structured.

In this subsection, we have looked that there are facts showing evolution of linguistic complexity through developments of pidgin and creole languages and children's language acquisition processes. They exhibit conspicuous evidences of dynamical codes. These dynamics are partly induced by mixture of codes – superstratum and substratum languages in the pidginization, and adults' and children's languages in the language acquisition processes.

2.1.3 Organization of this Chapter

Sections in this chapter are organized as follows. Theory of formal language and the Chomsky hierarchy are introduced in §2.2. A model of a language game and evolutionary dynamics are defined in §2.3. Detailed analysis of evolution of those grammar systems is presented in §2.4. Developing an ensemble sharing several words, we call it an ensemble with a common set of words (ECW), is found to be crucial for maintaining diversity of grammar structures. After establishment of an ensemble structure, each grammar system is likely to become complex again. An ECW is discussed in §2.5. A simulation of the model exhibiting a stepwise evolution over several eras will be reported in §2.6. Some other observations will be reported in §2.7. Several implications of ECW and evolution of grammar systems will be discussed in §2.8.

2.2 Basic of Formal Languages

2.2.1 Word and Language

A finite non-void set of symbols is called *alphabet* and is denoted by V. The finite strings of symbols are called *words* over V. The set of all words over V is denoted by V^*. The *length* of a word w is the number of symbols of w and is denoted by $|w|$. An arbitrary set of words is called *language* and is denoted by L.

Any language L is associated with several generative grammar systems G, which is characterized by a set of rules and symbols. By preparing relevant rules and symbols, at least one generative grammar can generate the whole words belonging to the given language L [46].

2.2.2 Generative Grammar

A *generative grammar* G is an ordered four-tuple (V_N, V_T, F, S). Symbols V_T and V_N are disjoint finite alphabet, called *terminal* and *nonterminal* symbols, respectively. A symbol $S \in V_N$ is an *initial symbol.* And a symbol F is finite set of ordered pair (α, β). Here, α and β is word over $(V_N \cup V_T)^*$ and α contains at least one symbol from V_N. The elements (α, β) in F are called *rewriting rules* and will be written in the form

$$\alpha \rightarrow \beta \,. \tag{2.2}$$

2.2.3 Derivation and Acceptance

Rewriting rules are used to derive new words from given ones. If the left-hand of a rule is equal to a part of a word, the part is replaced by the right-hand of that rule.

Given a grammar $G = (V_N, V_T, F, S)$ and two words $X, Y \in (V_N \cup V_T)^*$, we say that Y *is derivable from* X *in one step* and we denote it

$$X \Rightarrow Y, \tag{2.3}$$

iff there are words P_1 and P_2 in $(V_N \cup V_T)^*$ and a rewriting rule $\alpha \rightarrow \beta$ in F such that $X = P_1 \alpha P_2$ and $Y = P_1 \beta P_2$.

Given a grammar $G = (V_N, V_T, F, S)$ and two words $X, Y \in (V_N \cup V_T)^*$, we say that Y *is derivable from* X and we denote it

$$X \overset{*}{\Rightarrow} Y, \tag{2.4}$$

iff $X = Y$ or there is some word $X_0, X_1, X_2, \cdots, X_k (k \geq 0)$ in $(V_N \cup V_T)^*$ and $X_0 = X, X_k = Y$ and X_{i+1} is derivable from X_i in one step $(0 \leq i \leq k-1)$, i.e.

$$X = X_0 \Rightarrow X_1 \Rightarrow \cdots X_{k-1} \Rightarrow X_k = Y. \tag{2.5}$$

When no nonterminal symbols are left in the derived word, a derivation process terminates. If a word X is derived by a grammar G, we say that a word X is *acceptable* by G.

2.2.4 The Chomsky Hierarchy of Grammars

We can classify the generative grammars with respect to a set of rewriting rules. The classification below has been introduced by N. Chomsky [16].

A generative grammar $G = (V_N, V_T, F, S)$ is said to be of *type i* if it satisfies the corresponding restrictions in this list [86]:

$i = 0$ No restrictions, called *phrase structure grammar.*

$i = 1$ Every rewriting rule in F has form $Q_1AQ_2 \rightarrow Q_1PQ_2$, with Q_1, Q_2 and P in $(V_n \cup V_T)^*$, $A \in V_N$, and $P \neq \lambda$, except possibly for the rule $S \rightarrow \lambda$, which may occur in F, in which case S does not occur on the right-hand sides of the rules. Where λ is an empty word which contains no symbols. This grammar types are called *context sensitive grammar.*

$i = 2$ Every rule in F has a form $A \rightarrow P$, where $A \in V_N$ and $P \in (V_N \cup V_T)^*$. This type grammars are called *context free grammar.*

$i = 3$ Every rule in F has a form either $A \rightarrow PB$ or $A \rightarrow P$, where $A, B \in V_N$ and $P \in V_T^*$. This type of grammar is called *regular grammar.*

A language is said to be of *type i* if it is generated by a *type i* grammar. The classes of each type languages are related by the inclusions as follows:

$$\text{regular} \subset \text{context free} \subset \text{context sensitive} \subset \text{phrase structure}. \tag{2.6}$$

2.3 Modeling

2.3.1 Agent with Generative grammar

We define a communicative agent with generative grammar as follows:

$$G_i = (\{S, A, B\}, \{0, 1\}, F_i, S) \ . \tag{2.7}$$

All agents have the same sets of nonterminal and terminal symbols, and are identified by index i. A symbol F_i is a set of rewriting rules peculiar to each agent. The rewriting rules are written in the form $\alpha \to \beta$ as above mentioned. Here, the left-hand of any rule, denoted by α, is a symbol over V_N. The type of grammar which an agent can have is either a context free grammar or regular grammar. The right-hand of rule, denoted by β, is a finite string of symbols over $V_N \cup V_T$. The same symbol should not be included in α and *beta* to forbid a self loop. In this paper, we use 0s and 1s for alphabets so that words become bit strings.

2.3.2 Communication

Agents communicate with each other by trying speaking and recognizing words in terms of its own grammar.

All agents derive words using their own rewriting rules. To derive a word a leftmost symbol equal to left-hand side of a rewriting rule is rewritten by a right-hand of the rule. Derivation always starts from an initial symbol S. If there are more than two fitting rules in an agent's rule set, the agent selects one rule randomly. When no nonterminal symbols are left in the derived word, a derivation process terminates. And the derived word is spoken to all agents. An agent fails to speak a word when (i) the derivation does not terminate within 60 rewriting steps or (ii) there is no applicable rule in its rule set. The maximum length of a word is M here. The words longer than M are truncated after the M-th symbol and then are spoken to. The possible number of words (N_{all}) is limited to $2^{M+1} - 2$, and a full set of words speakable by an agent G_i is denoted by $L_{\text{sp}}(G_i)$.

For example, an agent which rules are $S \to A, A \to B$ and $A \to 01$ can derive only $\{01\}$. This agent always rewrites the initial symbol S to A via a rule $S \to A$. Then there are two rules to be applied, $A \to B$ and $A \to 01$. One rule is selected randomly from these two rules. If $A \to 01$ is selected the word become 01 which doesn't consist of any nonterminal symbols and are not rewritten any further. Then he speaks the word "01" to all agents. On the other hand, if the rule $A \to B$ is adopted, the derived word is B. Since there is no rule to rewrite, this agent fails.

Agents try to recognize words by applying their own rule in the opposite direction. If an agent can rewrite a given word back to the symbol S within 500 rewriting steps, we say that the agent can recognize the word. Detailed algorithm for recognition process are explained in Appendix. The language recognized by a agent G_i is denoted $L_{\mathrm{rec}}(G_i)$. Note that the inclusion relationship (i.e. $L_{\mathrm{sp}}(G_i) \supseteq L_{\mathrm{rec}}(G_i)$) holds, because of the truncation and limitation of rewriting steps.

In the above example, the agent can understand only a word "01" by writing back it to the symbol S as

$$01 \Rightarrow A \Rightarrow S \ . \tag{2.8}$$

It cannot recognize any other word since the agent does not have applicable rules as *non-terminal symbol → terminal symbols* except for $A \rightarrow 01$.

2.3.3 Language Game

We set a language game in a network consisting of P agents. Each agent takes turns to speak a word and it is given to all the agents. Then every agent including the speaker tries to recognize it. Each time step consists of R rounds. For each round, every agent has an opportunity to speak.

Each agent is ranked by three different scores:

1. **speaking** How long and how rare words an agent speaks.
2. **recognizing** How long words and how quickly an agent recognizes.
3. **being recognized** How long words which an agent speaks are recognized.

A word spoken by the l-th agent to the m-th agent at a round c is denoted by the symbol $w_{lm}(c)$. The scores for l-th agent at a round c is computed as follows:

For computing a score of speaking, it is given by,

$$p_l^{\mathrm{sp}}(c) = \begin{cases} \frac{|w_{lm}(c)|}{trend+1} \ , & \text{for speaking a word } w_{lm}(c) \\ -\frac{M}{2} \ , & \text{for failing to speak any word ,} \end{cases} \tag{2.9}$$

Where *trend* is defined as the frequency of the word spoken in the last 10 time steps. An agent gets a higher value of $p_l^{\mathrm{sp}}(c)$ when it speaks longer words and/or less frequent words.

For computing a score of recognizing, it is given by,

$$p_{kl}^{\mathrm{rec}}(c) = \begin{cases} \frac{|w_{kl}(c)|}{s} \ , & \text{for recognizing a word spoken by the } k\text{-th agent} \\ & \qquad \text{in } s \text{ rewriting steps} \\ -|w_{kl}(c)| \ , & \text{for not recognizing a word spoken by the } k\text{-th agent} \ . \end{cases} \tag{2.10}$$

A quick recognition of a long word provides a higher value of $p_{kl}^{\mathrm{rec}}(c)$.

For computing being recognized score, it is given by,

$$p_{lm}^{\mathrm{br}}(c) = \begin{cases} \frac{|w_{lm}(c)|}{P} \ , & \text{if the spoken word is recognized by the } m\text{-th agent} \\ -\frac{|w_{lm}(c)|}{P} \ , & \text{if the spoken word isn't recognized by the } m\text{-th agent} \ . \end{cases} \tag{2.11}$$

Agents recognizing each other makes a value of p_{kl}^{br} high.

The total score for the l-th agent in a time step is an average of a weighted sum of three scores over R rounds:

$$p_l^{\mathrm{tot}} = \frac{1}{R} \sum_{c=1}^{R} (r_{\mathrm{sp}} p_l^{\mathrm{sp}}(c) + r_{\mathrm{rec}} \sum_{k=1}^{P} p_{kl}^{\mathrm{rec}}(c) + r_{\mathrm{br}} \sum_{m=1}^{P} p_{lm}^{\mathrm{br}}(c)) \ , \tag{2.12}$$

where r_{sp}, r_{rec} and r_{br} are the respective weighting coefficients. For example, if r_{br} is given a positive value, those who can be recognized by more agents get scores more. But if the value is given negative, being recognized is no more favorable.

2.3.4 Evolutionary Dynamics

In each time step, new agents are produced. They inherit a rule set from their parents with a slight change. The change of rule set is defined by the following three processes:

adding mutation A new rule is added, which is transfered from parents by modifying randomly selected rule.

replacing mutation A randomly selected rule is replaced with a modified rule.

deleting mutation A randomly selected rule is deleted.

The modification ways are following;

1. Replacing a symbol of the left-hand of the rule with the other nonterminal symbol.
2. Replacing a symbol in the right-hand of the rule with the other nonterminal or terminal symbol.

3. Inserting a symbol in the right-hand side of the rule.

4. Deleting a symbol from the right-hand of the rule.

Adding mutation is applied to agents within the rate m_{add}, if their scores exceed the average score. Replacing and deleting mutations are applied to all the agents within the rate of m_{rep} and m_{del}, respectively. The same number of the least scored agents as new agents is removed from a network so that the total number of agents is kept constant.

2.4 Algorithmic Evolution

In this paper, a network consists of 10 agents ($P = 10$) and each agent tries to speak 10 times in each time step ($R = 10$). The score of language game is computed under the fixed parameters: $r_{\text{sp}} = 3.0, r_{\text{rec}} = 1.0$ and $r_{\text{br}} = -2.0$. Note that agents which can speak less acceptable words are benefited for a negative value of r_{br}. It is expected that a variety of the words spoken in a population will increase. All the mutation rates are set at equal value 0.04 ($m_{\text{add}} = m_{\text{rep}} = m_{\text{del}} = 0.04$). The maximum length of a word is limited to 6 ($M = 6$), therefore the number of possible words N_{all} is 126.

Initially, all agents assumed to have the simplest grammar, i.e. a single rule with one symbol in the both hand side. They are classified as type 3 grammars due to Chomsky's classification. At least, either a rule $S \rightarrow 0$ or a rule $S \rightarrow 1$ should be included in order to derive a word.

We find that evolution of grammar system is accelerated by the characteristic factors, one is a module-type evolution and the other one is a loop forming evolution. Computational ability of agents is measured by the ratio of recognizable words to the total number of possible words, i.e.

$$\text{computational ability} = \frac{N(L_{\text{rec}}(G_i))}{N_{\text{all}}} \ , \tag{2.13}$$

where $N(L_{\text{rec}}(G_i))$ is the number of words which can be recognized by the agent G_i. Fig. 2.3 represents the example of evolution of the computational ability from the initial network. The computational ability, as well as the number of the distinct words spoken in the network, we call a *variety* of words, evolves in the course time.

A tree that displays the derivation path of a given word is called a derivation tree of the word. We put all possible derivation tree of a grammar system in a directed, connected graph. A structure of the graph expresses the algorithm of the grammar. Algorithmic evolution can be seen in the topological changes of this graph.

2.4.1 Evolution during the Early Stage

It is shown in Fig. 2.3 that computational abilities of agents slowly evolve during initial 200 time steps. In Fig. 2.4 (a)$\sim$(c) the corresponding grammar systems are depicted in graph diagram. The initial agent has a weakest ability, having a direct derivation rule $S \rightarrow 0$ (Fig. 2.4(a)). The agent can increase the ability by the process of the adding mutation. Adding the rule $S \rightarrow 1$ to the initial graph generates a branch structure (Fig. 2.4(b)). Further, the multi branch structure

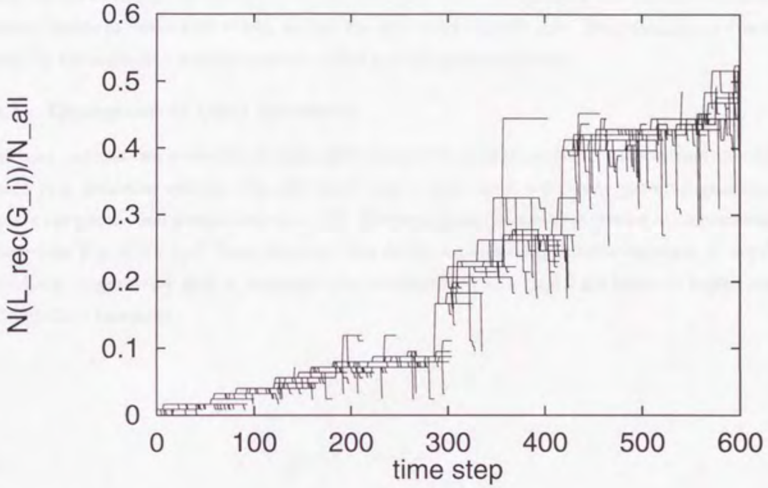


Figure 2.3: Time step v.s. $N(L_{\text{rec}}(G_i))/N_{\text{all}}$. Each line connects one agent to oneself or its offsprings. It branches off by the mutations. A line terminates when the corresponding agent is removed. These lines show upward trend. In initial 200 time step, computational ability gradually increases. After that, transitions to higher computational agent are frequently observed.

will evolve (Fig. 2.4(c)).

2.4.2 Module-Type Evolution

We find in Fig. 2.3 that an agent with the remarkably high ability (> 0.1) appears at time step 192. The change of grammar at this time step is sketched in Fig. 2.4(d). An acquired rule $A \rightarrow 00$ can double the size of acceptable words. Every intermediate word containing a symbol A can be rewritten by the rule $A \rightarrow 00$. In the sense that one common rule is used by many different words to make new words, we call the key rule a module rule. Evolutionary processes driven by the emergent module rule are called module-type evolutions.

2.4.3 Emergence of Loop Structure

Grammar systems can evolve by an alternative evolutionary process, that is, loop structures are formed in a grammar system. Fig. 2.3 shows that a new agent with more powerful grammar appears the population around time step 310. The new agent has a loop structure in its grammar system (see Fig. 2.4(e)). A loop structure can derive a potentially infinite numbers of words recursively. A grammar with a loop structure is categorized as a type 2 grammar or higher one in Chomsky's hierarchy.

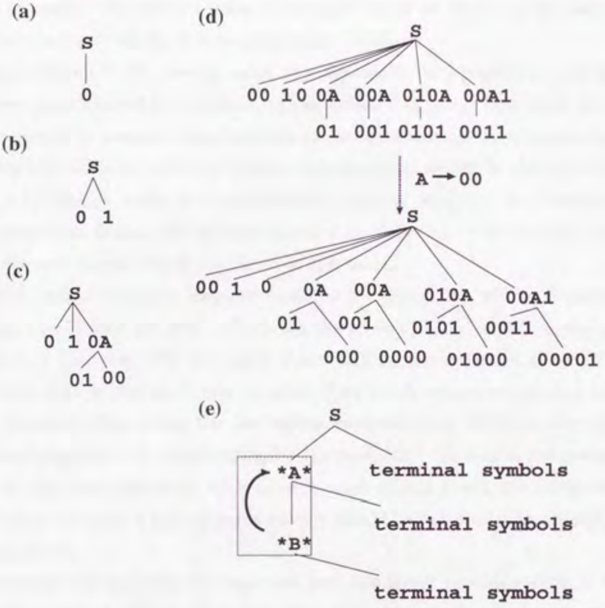


Figure 2.4: The examples of grammar structure are shown by graph diagrams: (a) a sequential structure, (b) a branch structure and (c) a multi branch structure. In the (d) an example of module-type evolution is shown. Acquiring a rule $A \to 00$, a grammar without bifurcation (upper tree) is evolved into one with bifurcated branches (lower tree). An example of grammar having a loop structure is schematized in (e). Asterisk stands for any symbols. With this grammar, new agent can rewrite words $*A*$ into $*B*$ and vise versa. Words derived from such grammar can not represented in a tree form.

2.5 Ensemble with a Common Set of Words

An upper structure, which is named an *ensemble with a common set of words (ECW)*, emerges in the population of agents. An ECW consists of agents which can speak and recognize a common set of words. The other agents which can't speak or recognize the common set of words are less benefited than those belonging to the ECW.

When there exists an ECW, even an agent of a high ability in a population will die out. For example, a new agent evolved by a module-type evolution dies out at time steps 192 and 310 in Fig. 2.3. Agents will be removed from the network nevertheless they have a power grammar.

At time step 403 an agent with the highest computational ability in the population is dies out (see Fig. 2.3). Agents taking too much rewriting steps to recognize very frequent words are likely to decrease their fitness. We indicate this fact by Table 2.1. The rewriting steps needed to recognize the well spoken words are shown in this table.

Agents which cannot recognize frequent words in the population will be removed in order. An agent G_{306} (agent with ID 306) , which has the second highest ability in the population, is removed first at time step 400. An agent G_{302} which cannot recognize a word "10" is next agent to be removed. At the next time an agent G_{307} which cannot recognize a word "0001" is removed. An agent G_{276} which has the highest computational ability in the population is removed at time step 403,as it cannot recognize the word "00". To stay in the ensemble, where a word "00" is the most commonly spoken, each agent should speak and recognize the word quickly. An ability to speak a kind of words quickly should be balanced with an ability to speak many but long words.

Numbers in bold font in Table. 2.1 represent first two larger rewriting steps to understand the words in the leftmost column. It is clear from this table that it takes much more time for agents G_{306} and G_{276} to recognize several words. To take more rewriting steps to recognize commonly spoken words of the majority is disadvantageous for the agents G_{306} and G_{276}. If a group containing agents G_{306} and G_{276} constituted the majority, the words as "001011" or "010101" would be the commonly spoken words. In such cases, agents G_{306} and G_{276} will take advantageous.

Fig. 2.5 shows the phylogeny of agents existing at time step 400. It shows that the group consists of the agents G_{276} and G_{306} and that of the other agents forms the different lines. They form two different ECWs. The agents in the major ECW have lower computational ability than those in the minor ECW consisting of G_{276} and G_{306}. Two ECWs conflict to survive in

Table 2.1: This table shows rewriting steps to recognize some words (the left most column) spoken at time step 400. Simulation parameters are $r_{\mathrm{sp}} = 3.0, r_{\mathrm{rec}} = 1.0andr_{\mathrm{br}} = -2.0$. In the second column, the trend of each word (frequency of words in the last 10 time steps) is written. Numerals in the first row are ID of each agent at time step 400 in order that the earlier a agent is removed the lefter it is located. If the agent can't recognize the word, no numerals is indicated. Bold numerals represent the first two longest steps to recognize the left most word.

word	trend	**306**	302	307	**276**	305	301	299	294	290	284
001001	30			121	**185**			121	**145**	133	121
011001	20			157	**423**			166	**214**	190	157
11100	16			52	**245**			46	**58**	52	52
11	14		**7**		**12**	**7**	**7**				
110	12		13		**32**	**14**	**14**				
00110	26		53	**57**		**62**	48	43	54	49	53
110010	11		**174**	147				121	**202**	160	147
00	69	**3**	**3**	**3**		**3**	**3**	**3**	**3**	**3**	**3**
10	57	**7**		5	**7**	5	5		5	5	5
011010	24	**431**	210	164	**431**	120	180	166	251	209	164
001010	31	**233**	134	124	**236**	106	106	116	161	141	124
1110	24	**125**	45	27	**122**	60	60	24	29	27	27
01110	9	**122**	91	69	**192**	55	76	66	83	75	69
00101	25	**91**	65	58	**91**	52	49	57	66	63	58
111	30	**53**	21	13	**52**	24	24	13	13	13	13
0001	78	**20**	19		**20**	21	16	18	18	18	17
001011	14	**401**			**404**						
010101	10	**190**			**193**						

the network. Those in the major ECW behave cooperatively as the result by speaking and recognizing a common set of words and get higher scores. At last all agent in the minor ECW is removed from the network. In this way the evolution toward the high computational ability is suppressed by establishing ECWs.

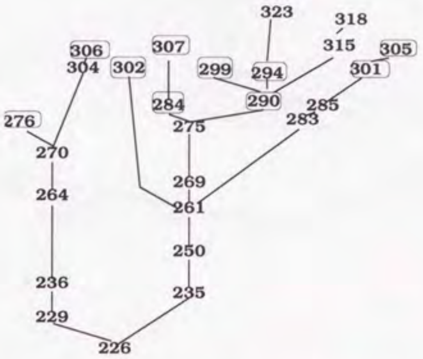


Figure 2.5: This picture represents the phylogeny of agents at time step 400 (in oval boxes) from common ancestor (G_{226}). A number represents ID of each agent. A line is drawn from parent agent (lower) to its offsprings (upper). Two genetic series are bifurcated from the common root (G_{226}), the agents G_{306} and G_{276} and that of the other agents. They are forming different ECW. The agent G_{306} and G_{276} are both contained in the left series.

Fig. 2.6 displays a situation of spreading out an ensemble in network. A high trend peak moves from low number of agents which recognize some words to high number of agents with time step going on. The broken arrow in this figure indicates this movement. It implies that new agents belonging ECWs come into existence and at last the ECWs dominate whole network.

After removing agents G_{306} and G_{276} from a network, an ECW spread to whole network as shown in Fig. 2.6. Then agents come to compete with each other within the same ECW. Proportional to the number of rewriting steps to recognize the commonly spoken words, the agents are removed from the network. In the ECW, a new agent with the high computational ability will then emerge through algorithmic evolution.

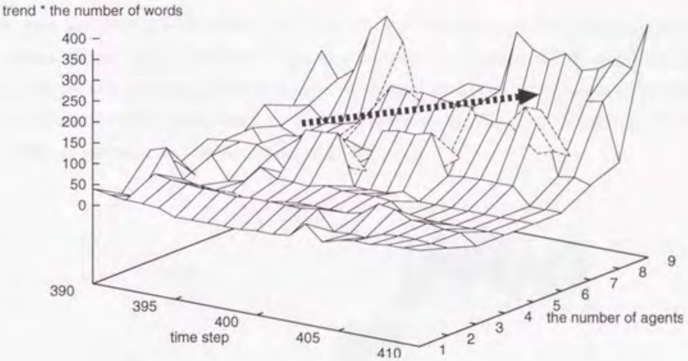


Figure 2.6: time step v.s. the number of agents which can recognize some words v.s. trend *times* the number of the words which each number of agents can recognize : We can see that an ECW spread out in network. This graph has a peak at time step 396 and the number of agents 5, where some words which 5 agents can recognize are spoken many times. After that this peak moves to higher number of agents in the course of time. That is to say, agents speaking the words increase. Increase of trend of particular words seems reasonable to say that agents in some ensembles increase. At last all of agents can speak and recognize such words. Namely, some ensembles have dominated whole network. The broken arrow indicates the movement of the peak.

2.6 Punctuated Equilibrium

We have seen that our system shows rapid algorithmic evolution in the grammar systems in certain stages. Since, algorithmic evolution is suppressed by forming ECW, rapid algorithmic evolution follows the quasi-equilibrium stages. Temporal evolution of amounts of handling information therefore shows punctuated equilibrium phenomena (Fig. 2.7). Handling information defined below is sensitive to the formation of ECW.

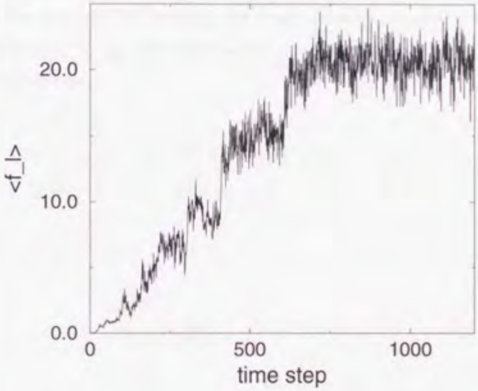


Figure 2.7: time step v.s. the average handling information (see the definition in the text): In the first 700 time steps, stepwise evolution can be observed. The stepwise changes reflect alternate evolution of ECW and the algorithmic evolution.

The handling information of the l-th agent is defined as the follows,

$$f_l = \frac{1}{RP^2} \sum_{c=1}^{R} \sum_{k=1}^{P} |w_{kl}^{\mathrm{rec}}(c-1)| \sum_{m=1}^{P} |w_{lm}^{\mathrm{rec}}(c)| \tag{2.14}$$

$$|w_{ij}^{\mathrm{rec}}(x)| = \begin{cases} 1 & \text{if } x = 0 \\ \text{the length of } w_{ij}(x) & \text{if } x > 0 \text{ and the word which spoken by the} \\ & \quad i\text{-th agent is recognized by the } j\text{-th agent} \\ 0 & \text{otherwise .} \end{cases} \tag{2.15}$$

Information contents of a word is simply given by the length of a word. The initial amount of handling information, i.e. $|w_{ij}(0)|$, is defined as 1.

If an agent gets high value of f_l, which suggests that the agent can recognize words spoken by others and its speaking words are recognized by other agents. When some ECWs conflict with the other ECWs, the averaged handling information in a population,

$$\langle f_l \rangle = \sum_{p=1}^{P} \frac{f_l}{P} \ , \tag{2.16}$$

does not increase. After some ECWs occupy the whole network, long and new words will be spoken and recognized again by agents. Punctuated equilibrium phenomena in the amount of $\langle f_l \rangle$ is explained by the scenario.

2.7 Results of Other Simulations

2.7.1 Minimal Almighty

We can make a *minimal almighty* agent. It is an agent which can speak and recognize all possible words with the least number of rules. For example, a minimal almighty agent has the rules such as:

$$S \rightarrow A, A \rightarrow SS, S \rightarrow 0, S \rightarrow 1 \ . \tag{2.17}$$

This grammar is categorized as a type 2 grammar. It recognizes and speaks all the words very quickly. However, it tends to speak a low variety of words because of random adoption from plural fitting rules. A minimal almighty agent cannot invade into the system composed of ECW because it can speak lower variety of words. In the case of $r_{\text{rec}} = 1.0, r_{\text{sp}} = r_{\text{br}} = 0.0$, an almighty agent can evolve, since low variety of speaking words has no disadvantage for its fitness. An example of evolution of computational ability when an almighty agent emerges is shown in Fig. 2.8. Mutants not only from the almighty agent but also ancestors of almighty agent have much lower computational ability than its parent as shown in this figure. The fragility of the almighty's grammar suggests that the rules in almighty agent strongly interact with each other.

2.7.2 Score of Being Recognized

We have three parameters in the definition of total score (see eq. 2.12). Here, we will see the effect of the parameter r_{br}. If it is given positive value, a tendency to recognize each other will be encouraged. But if it is given negative, the opposite tendency begins to spread. We display, how it depends on the parameter r_{br}. If r_{br} is larger than 4.0, both the variety of word spoken and the average handling information are suppressed. Since agents gets higher score by speaking and recognizing the same and short words. If r_{br} becomes less than -3.0, the average handling information will go down (Fig. 2.9 b)) since it reflects the degree of being recognized. Variety, on the other hand, is kept middle level in the range $r_{\text{br}} < -3.0$ (Fig. 2.9 a)). In such range, agents make higher score by speaking less frequent and less recognizable words. It maintains variety of words in the middle level even when r_{br} is less than -3.0. In earlier stages (time step < 1500), simulation with $r_{\text{br}} = -2.0$ has shown quick evolution. However it is saturated at $r_{\text{br}} = 0.0$ and below (time step ≥ 1500). We conclude that a slight negative value of r_{br} accelerates the evolution at the highest speed.

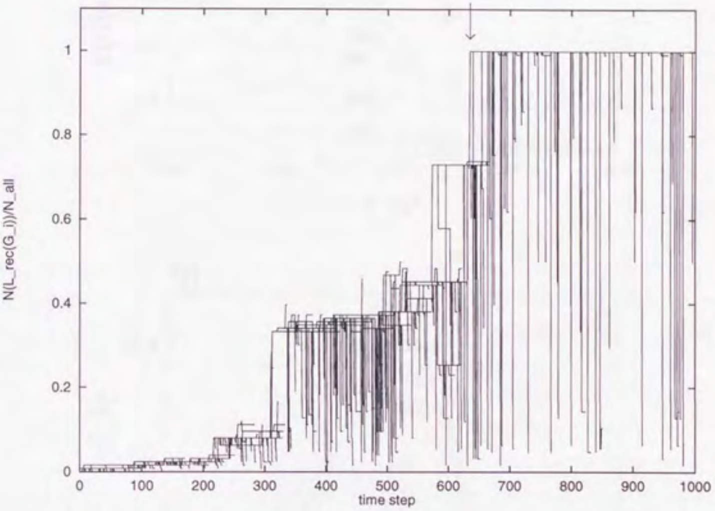


Figure 2.8: An example of emergence of almighty agent. Time step v.s. $N(L_{\mathrm{rec}}(G_i))/N_{\mathrm{all}}$. Each line connects one agent to oneself or its offsprings. It branches off by the mutations. A line terminates when the corresponding agent is removed. Coefficient for each score is $r_{\mathrm{rec}} = 1.0, r_{\mathrm{sp}} = r_{\mathrm{br}} = 0.0$. An almighty agent emerges at time step = 634 (the arrow indicates). Almost mutants from both almighty agent and almighty's ancestor have very low computational ability.

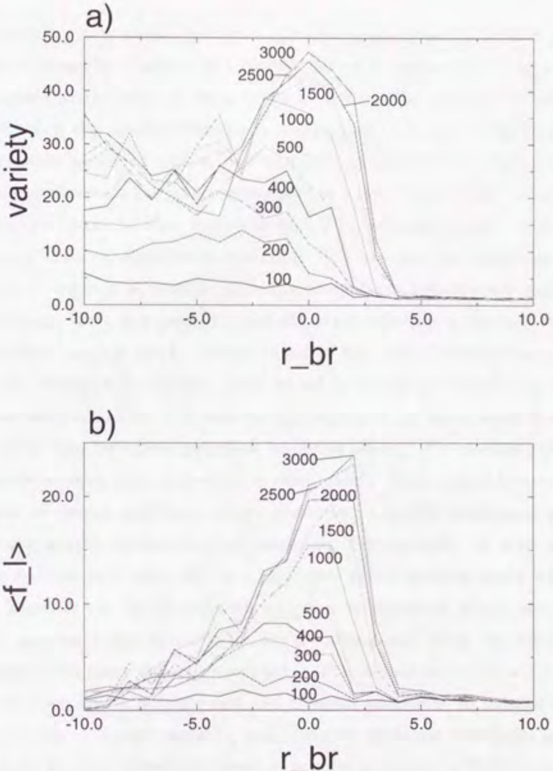


Figure 2.9: a)variety of words spoken and b)the average handling information vs r_{br}, coefficient of score of being recognized, at several time steps. The other parameter values are $r_{sp} = 3.0$ and $r_{\mathrm{rec}} = 1.0$. The number attached to each line represents time step. a) In lower range of the parameter ($r_{\mathrm{br}} \leq -3.0$) variety shows middle level, in midway of the parameter ($-3.0 \leq r_{\mathrm{br}} \leq 2.0$) it has high variety and in the higher range it shows low level. b) For each range $< f_l >$ shows low, high and quite low level, respectively.

2.8 Discussions

In the present study, we have shown two types of evolutionary dynamics. One is a module-type evolution. What it means for a rule to be a module is that it can be utilized by other rules in a grammar to generate nearly twice as many words as before. The number of recognizable word rapidly increases when this module emerges in a grammar. The other type is a loop forming evolution. A grammar equipped with a loop structure can derive recursively many words. A grammar with a loop structure cannot be represented in a tree shape. Thus the grammar system climbs up Chomsky's hierarchy from type 3 to type 2 by acquiring loops. Hence we call such a loop forming evolution an algorithmic evolution. We consider two significant evolutionary processes in natural language as possible correspondents of our evolutionary dynamics.

In natural language, there is a process called affixation, whereby a group of letters is added to a word to produce another word. Affixes added to the heads of words are called prefixes; un-, mis-, pre- are examples in English. Ones added to the ends of words are called suffixes; -ful, -less, -ish are such examples. The module-type evolution we have found in our simulation can be related with word formation processes involving affixes. The module rules are used to produce new words by attaching themselves to other words. They indeed behave as affixes.

The other kind of process produces nested sentences. Complex sentences, phrases within phrases and clauses within clauses are such examples. For example, "A man with a colorful umbrella who is walking over there will be a candidate of the political party which suffered a setback." This sentence can be decomposed into four independent simple sentences without any nesting: "A man has a colorful umbrella. He is walking over there. He will be a candidate of a political party. The party suffered a setback." This nested structure is a very important feature of natural language to produce rich and complex sentences. If natural language were only required to serves as a mere signaling tool, complex syntactic structures might not have evolved. In order to produce nested sentences, a complex grammar is likely to appear.

An ECW (ensemble with a common set of words) is characterized by a shared set of words. Individual grammars comprising an ECW are restricted to deriving words freely. A particular usage of words in an ECW emerges. Thus a code is organized. In order to speak and recognize sharing words quickly, it is advantageous to have their words as single rules (i.e. $S \rightarrow$ words), and to combine several rules of nonterminal symbols on their right hand sides (e.g. $A \rightarrow 0BS$) to speak/recognize other words. This can be regarded as double articulation in natural language. Because a sentence consists of words and a word consists of symbols, we can have infinite sentences with finite symbols. The double articulation also constitutes one of the most important features of natural language. Such structure is good for recognizing frequently spoken words quickly and speaking many longer words. When we only count a recognizing score in a language game, almighty agents appear and a variety of words is suppressed, since such structure as double articulation is not likely to develop.

In biological systems ranging from ants' society to human society, social laws and norms develop. Essential problem in a development of norms is how such ordered upper structures evolve from local interactions [60, 98]. It is suggested that homeochaos or the edge of chaos is formed to establish cooperation [58, 110]. In the iterated prisoner's dilemma game, the Tit For Tat strategy is formed in cooperative societies [2]. Once such a social structure appears, it in turn restricts local dynamics. A recursive loop between global structures and local structures continues endlessly. Taylor described such a recursive relationship as "LOCAL to GLOBAL back to LOCAL, inter-level feedback loops" [98]. This is stressed as evidence of "emergence", which is one of the central concerns in artificial life studies [98, 63].

We here propose "children's play" as an example of such emergent phenomena. Children sometimes change rules of a game while playing it. The rules dictate how the player is to behave and how new rules emerge from children's play successively. Such dynamics of children's play is like the dynamics of linguistic code. It is reported from sociolinguistics studies (e.g. [92]) that in general linguistic communities, social dialects are frequently observed. They are language variations which are used only in particular social classes or in different groups based on various social variables, such as social classes, education levels, occupations and age groups. Peculiar usages emerge from conversations in a small group. Once they are established, they constrain the variety of words. It is said that group words such as social dialects are likely to emerge when a group consists of about 8 members [92].

Within our language game, an ECW produces dialects, irrespective of individual grammars. We have used a term *net-grammar* to refer to the way an ECW produces dialects and restricts the potential computational ability of individual grammars in the proceeding paper [41]. We can say from our study that a code has restrictive force to individuals sharing the code. Linguistic codes can be a source of social codes.

Our language game favors the following agents: speak long and infrequent words; recognize long words quickly; speak words which are not recognized by other agents. These conditions force players to try to produce more expressions. As a result, diversity of words and complexity of grammar get enhanced. Such a tendency of players is valid through the whole situation, a tendency to speak/recognize a common set of words emerges in an ECW. Agents trying to speak the same words is tantamount to trying to communicate with each other. This is what we believe to be one of the main purposes of language. Conflicts between enhancing expressivity and establishing linguistic code result in punctuated equilibria in communicating information.

Agents in our game are in a dilemma. They try to speak words which others cannot recognize, but at the same time, to recognize what others say based on the same grammar. This dilemma somehow resembles imitation games proposed by Suzuki and Kaneko [96, 59]. It was a simple model for mutually mimicking birds. In their papers a bird with a complex song was stronger when it came to defending its territory, since complex songs were difficult for other birds to imitate. But at the same time, a bird imitated other birds' song by the same mechanism. A song is a time series generated by a logistic map. Evolution of a bird's song was achieved by mutating the parameter of logistic map. They observed evolution of a song towards the edge between the periodic window and chaos, that is, birds singing songs of marginal stability seemed stronger.

Crutchfield has shown that complexity of automata which has accepted time serieses of dynamical mappings has been the highest at the onset of chaos [23, 22]. In his model, input to automata was quantized time serieses of maps. By replacing this input by output of other automata, there may be a similarity to our model. Ensemble of Crutchfield's automata may give rise to complexity by language game.

Algorithmic evolution and restriction by an ECW causes punctuated equilibria even in language systems. It is brought about by two factors, modification of grammars and network structures. Punctuated equilibrium in genetic systems has been produced by genetic fusion operators, which combine genes with module genes [53]. (Note that the module rules which we define in the present study are different from the module species used by Ikegami and Kaneko [53]. The module species were frequently utilized as fusion partners by *many other* species. The module rules, in contrast, are used to speak or recognize many words within *a single* agent.) By crossover operators different schemata can evolve in parallel, being combined into a better fitness [45]. Consequently, intermittent patterns will appear if crossover is introduced as genetic operator. There are no module agent or parallel evolutions in the present study, as we do not incorporate such operators as fusion or crossover that combine one's grammar with others'. But rules in a grammar have epistasis, that is, they interact with each other to produce words, a slight modification of a rule may cause large changes in the ability to speak and recognize words and thus the algorithmic evolution can occur.

If agents given randomly generated words, the variety of words produced according to grammars exhibits no punctuated equilibrium, it almost linearly increases, because agents cannot form an ensemble structure.

Chapter 3

Coevolution of Tapes and Machines

3.1 Introduction

3.1.1 Evolution of Genetic Code

Biological code in molecular levels are called *the genetic code.* It carries genetic information which can be passed from a generation to a generation or from cell to cell for self-reproduction. The genetic code is composed of 64 codon triplets assigned to specific amino acids or stop commands (Table 3.1).

Table 3.1: Universal genetic code: Each of 61 codon triplet is assigned to a specific amino acid and the other 3 are stop codons that terminate the synthesis of a protein molecule. The great portion of biological systems have same code system with this universal code. But some exceptions are found. The abbreviations referring amino acids are the followings: Ala = Alanine, Arg = Arginine, Asn = Asparagine, Asp = Aspartic acid, Cys = Cysteine, Gln = Glutamine, Glu = Glutamic acid, Gly = Glycine, His = Histidine, Ile = Isoleucine, Leu = Leucine, Lys = Lysine, Met = Methionine, Phe = Phenylalanine, Pro = Proline, Ser = Serine, Thr = Threonine, Trp = Tryptophan, Tyr = Tyrosine, Val = Valine.

Codon	Amino acid	Codon	Amino acid	Codon	Amino acid	Codon	Amino acid
UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys
UUC	Phe	UCC	Ser	UAC	Tyr	UGC	Cys
UUA	Leu	UCA	Ser	UAA	stop	UGA	stop
UUG	Leu	UCG	Ser	UAG	stop	UGG	Trp
CUU	Leu	CCU	Pro	CAU	His	CGU	Arg
CUC	Leu	CCC	Pro	CAC	His	CGC	Arg
CUA	Leu	CCA	Pro	CAA	Gln	CGA	Arg
CUG	Leu	CCG	Pro	CAG	Gln	CGG	Arg
AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser
AUC	Ile	ACC	Thr	AAC	Asn	AGC	Ser
AUA	Ile	ACA	Thr	AAA	Lys	AGA	Arg
AUG	Met	ACG	Thr	AAG	Lys	AGG	Arg
GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly
GUC	Val	GCC	Ala	GAC	Asp	GGC	Gly
GUA	Val	GCA	Ala	GAA	Glu	GGA	Gly
GUG	Val	GCG	Ala	GAG	Glu	GGG	Gly

The evolution of the genetic code is one of the major subjects in the origins of life. Numerous attempts have been made to make clear how the code had evolved (for example [21, 107, 65, 73, 105, 6]). Here we are interested in how sequences of codons are interpreted to proteins and how the proteins act to serve self-reproduction.

The code in Table 3.1 was thought to be universal. But exceptions were found in human mitochondria [5] and in nuclear code [62]. These facts suggest that the genetic code is not frozen from the early stage of life. Evolution of biological systems in their forms and functions is accompanied by evolution of the genetic codes. Biases of the GC content in genome are related with changes of the genetic code. In eubacteria, the variation of the mean GC content is relevant tob variations in phylogeny [47,48]. It suggests that there exist directional mutation pressures. The directional mutation may be caused by internal factors such as errors in the reproduction of DNA [89]. (Osawa et al reviewed the evidences of evolution of genetic code and directional mutation pressure in [78].)

A code is to present ways to interpret symbols. The genetic code directs the way to synthesize proteins from DNAs. Because the genetic code is not established by somebody but autonomously organized, it is essential to discuss evolutionary paths of the code for self-reproduction as co-evolutionary processes of information sequences and their decoders.

3.1.2 Problems in Self-reproduction

Origin of life is often attributed to the emergence of self-reproductive properties. Therefore the genetic code must be organized to serve self-reproduction. John von Neumann first has proposed an automaton model for the self-reproduction problem [102]. In his abstract modeling, the fundamental problem in reproduction is that to copy something we first have to observe the object, however, the observation in some cases disturbs the object.

In addition to the problem of observation and copying, the self-referential problem known as Richard's problem occurs for self-copying. A self-reproducing automaton should interfere with itself for replication of itself. This generates a self-referential paradox [42]. To avoid the difficulty, von Neumann separated a machine from its description tape as well as proteins and RNAs/DNAs. In Neumann's model, he defined a tape as a pattern of stationary states. Without external disturbances, the replication scenario is perfect. But external noise causes error actions, replication can become unstable.

Another problem is caused in the case that many machines come to interact. A description tape can be read by distinct machines in different manners. Several different machines can be produced from a tape. A machine disturbing reproduction of other machines may be produce.

We propose the additional problem of self-replication, as is also stressed by Neumann himself.

It is evolvability, that is, how to evolve more complex or functional machines from simple self-replicating machines. Random update of automaton states brings mutation into Neumann's automaton model. If mutation occurs in a machine itself, a machine may change its function. But such mutation should not be copied onto the next generation. If mutation occurs in a tape, it will be copied onto the next generation.

How to encode machines on tapes and how to acquire stability and evolvability of them in ensemble of tapes and machines under noisy environment will be discussed here.

3.1.3 Difficulties for Evolution of Stable Autocatalytic Networks

Eigen and Schuster have tried to elucidate the origin of genetic code based on their hypercycle model [24]. But they didn't discuss evolution from simple autocatalytic systems to a higher order hypercycle. Here we study evolution of genetic code as evolution of stable autocatalytic networks of machines and their description tapes.

Several drawbacks of hypercycle model are known. In a hypercyclic loop all connectivity among pairs of replicator and enzyme must be exactly equal. All couplings in a hypercycle are restricted to homogeneous in this sense. Niesert, Harnash and Bresch has pointed out the following three processes breaking down an hypercycle by problems of stability [76]. (1) If a selfish replicator emerges by accident, the minimal replication loop takes over the whole loop. A selfish replicator makes an enzyme not to help the replication of the sequence at next position in the loop but to help the reproduction of the replicator itself. (2) There is a danger of short circuit catastrophe. A short cut loop can appear when a mutant catalyze a sequence farther than normal one. Loops with large and short loops sharing partial components cannot be sustained. The whole hypercycle always contracts to the shorter circuit. (3) A large hypercycle dilutes each component in the system. Therefore the large a hypercycle becomes, the more possible that concentrations of components go down to zero by fluctuation. A hypercycle collapses if a critical component is lost.

In order to overcome these drawbacks, Kauffman and Rasmussen have discussed several improvements. Kauffman has proposed possible countermeasures for above problems [61]. He stated that these problems could be evaded by introducing both activating and inhibiting coupling. Redundancy of connections will increase stability of a catalytic system. He also presented another plan. If an enzyme can catalyze many replicators at a time, the whole system can be more stable.

Rasmussen has conceived the emergence of autocatalytic network as self-organization in a random directed graph by introducing mutually catalyzing strand-pairs [83]. Although he envisioned the emergence of autocatalytic network, the following problem has been pointed out by Kauffman [61]: the larger the number of kinds of strand-pairs, the more difficult to find catalyzing pair in a graph, since each strand in Rasmussen's model only has a single pairing target for catalyzing.

In spite of their improvements, the essential weak point in Eigen and Schuster's hypercycle is inevitable. It is the homogeneous coupling. One of the purposes here is to study how large autocatalytic network with inhomogeneous couplings is stabilized.

In our study, machines and description tapes are introduced and they are influenced by random external noise. While a machine reads a tape, both probabilistic and deterministic change occurs on tape symbols. The probabilistic change is caused by external noise and is called a probabilistic mutation or a passive mutation. On the other hand, the deterministic change is caused by machine action and is named a deterministic mutation or an active mutation. We refer these two processes as mutations since a tape is rewritten into new tapes by machines in place of replication. As mentioned before, mutation induced by internal factors has been suggested in the study of real genetic code [89].

We see evolution of a perfect replicating network composed of a pair of a machine and a tape in low external noise regime. For high noise region, a complex autocatalytic network sustaining deterministic mutation evolves. Self-replication not as an individual but as a whole network now becomes important. In the other autocatalytic models [28, 24], reproduction only by machines or by tapes has been discussed. No machine can reproduce itself, however, without the coding tapes and vice versa. By considering both machines and tapes, we can discuss the evolution and selection of code in terms of evolution of an autocatalytic network.

3.1.4 Organization of This Chapter

Sections in this chapter are organized as follows. Our basic model is introduced in §3.2. Population dynamics of machines and description tapes are introduced with the notion of active and passive mutation. In §3.3, evolution of a network composed of machines and tapes is discussed. If external noise is low, a simple pair of self-copying machine and tape is formed, where the active mutation is suppressed. But in the realm of high external noise, a complex network can emerge by sustaining high active mutation. A mechanism of forming this complex but stable network is attributed to the formation of core structure. A network acquiring noise stability is defined as a "core network". The core network is studied in §3.4. In §3.5, classification of core networks will be presented. A core network consists of many autocatalytic loops. In §3.6, we study embedded autocatalytic loops in core networks. A network which both machines and tapes forming catalytic loops is found to stabilize large and complex networks. §3.8 is devoted to discussion of our results. Some possible connections with real biological systems will be discussed.

3.2 Modeling

Our system consists of two different objects, tapes and machines. They are simple analogues of the polynucleotide and polypeptide or ribosome, respectively. A tape has a bit string of a circular form. A machine which is an automaton like a Turing machine consists of 3 different parts, a head, a tail and a transition table. Each head and tail is expressed by a 4 bit string, whose pattern will be compared with binary patterns of tapes. A transition table consists of 4 mappings; $(\sigma^M, \sigma^T) \rightarrow (\sigma'^M, \sigma'^T)$, where σ^M and σ^T represents a current binary state of machine and tape. A tape and machine state will change to (σ'^M, σ'^T), respectively depending on a current state of machine and tape.

Introducing an ensemble of tapes and machines, we carry out the machine-tape reaction process as follows.

3.2.1 Interaction of Machines and Tapes

A machine $\mathbf{M}_i$ can react with a tape $\mathbf{T}_j$ iff the tape $\mathbf{T}_j$ has a head $\mathbf{h}_i$ and tail $\mathbf{t}_i$ of the machine $\mathbf{M}_i$ in a different site.

Machine $\mathbf{M}_i$ reads a tape $\mathbf{T}_j$ from the first site of $\mathbf{h}_i$ to the first site of $\mathbf{t}_i$. And a new tape written according to the transition tables of $\mathbf{M}_i$ is produced. And then a new machine is produced from the new tape. The site from $\mathbf{h}_i$ to $\mathbf{t}_i$ will be called the reading frame. A half population of machine starts to read a tape with the internal state 1 and the other half does with the state 0. As the result, it generates a new set of machine $\mathbf{M}_k$ and tape $\mathbf{T}_l$ per each interaction. This process s written as a reaction equation,

$$\mathbf{M}_i + \mathbf{T}_j \rightarrow \mathbf{M}_i + \mathbf{T}_j + \mathbf{M}_k + \mathbf{T}_l. \tag{3.1}$$

During the read/write process, both probabilistic and deterministic changes on bits of tape pattern are assumed to occur. The probabilistic change is caused by external noise and is called a **passive mutation**. On the other hand, the deterministic change is caused by machine action, and is named an **active mutation**. We call it mutation since it does not replicate a tape but actively rewrites it. A rewritten tape can be taken as a mis-copy of the original tape. The rate of passive mutation denoted by μ_P is measured by the bit flip rate per bit. The active mutation per a reaction is measured by the rewriting rate per a length of reading frame when a machine

i reads a tape j. Namely, it is given by,

$$\mu_{Aij} = \frac{w}{L_{ij}}, \tag{3.2}$$

where a symbol w denotes the number of rewritten bits and L_{ij} denotes a length where machine i read a tape j. Average of active mutation rate is given by,

$$\langle \mu_A \rangle = \frac{\sum_{ij} \mu_{Aij} m_i t_j}{\sum_{ij} m_i t_j}, \tag{3.3}$$

where, m_i, t_i, is the population of machine and tape, respectively.

3.2.2 Translation of Tapes

Not only bits of a reading frame, but every bit of tape is repeatedly picked up to construct a new machine from a first site of the reading frame. If a length of a tape is not enough, the same bit is used for coding several different part of a new machine. In the present model, we use a fixed length of 7-bit tapes with 16-bit machines. A first 8 bits are mapped onto head and tail parts in order. The next 8 bits are mapped onto a transition table. In order to cover 16 bits by 7 bits, several bits are multiply used. This complicated mapping from bits of tape onto machine function is assumed to reflect the nonlinear mapping from one-dimensional DNA to 3-dimensional protein folding structure.

3.2.3 Population Dynamics

We presume two abstract spaces of the size N for tapes and machines. A total m machines and t tapes distribute over the respective space. By iterating the following procedures, we simulate the machine/tape reacting system.

1. Compute concentration of machines and tapes by dividing the population number of each object, denoted by m_i, t_i, respectively, by the capacity size N as following equations,

$$f_i^M = \frac{m_i}{N}, \tag{3.4}$$

$$f_j^T = \frac{t_j}{N}, \tag{3.5}$$

   where, f_i^M and f_j^T is concentration of i-th machine and j-th tape, respectively.

2. Make $c^{M/T}N$ numbers of new machines and tapes from the reaction of machine i and tape j. The rate of reaction f_{ij} is given by,

$$f_{ij} = \frac{c^{M/T} f_i^M f_j^T}{\sum_{k,l} f_k^M f_l^T}. \tag{3.6}$$

   if the i-th machine can read the j-th tape. Otherwise it takes zero value. The reaction coefficients $c^{M/T}$ take a positive constant $c^M (= c^T)$.

3. Remove $d^M\%$ of old machines and $d^T\%$ of old tapes.

4. Put the new machines and tapes back in each space. Hence the population of machine i and tape j of the next generation becomes,

$$m'_i = (1 - d^M)m_i + \sum_{k+j \to i} f_{kj}N, \tag{3.7}$$

$$t'_j = (1 - d^T)t_j + \sum_{k+i \to j} f_{ki}N. \tag{3.8}$$

   It should be noted here that each machine has its unique corresponding description tape but the inverse is not true. Generally each tape encodes several machines depending on which site the translation starts in.

5. Taking an integer part of the above population, we obtain the actual population of the next generation. The machine or tape whose concentration is lower than N^{-1} is removed from the population. Not to go beyond the total population size N, the reaction coefficient should satisfy the relation $c^{M/T} \leq d^{M/T}$.

3.2.4 Effect of External Noise

Out of $c^{M/T}N$ new tapes, a number of tapes as well as machines, are erroneously generated by external noise. It is assumed that the rate of error depends on the reading frame. Namely, the rate of error replication by external noise is given by,

$$\varepsilon = 1 - (1 - \mu_P)^L, \tag{3.9}$$

where the symbol L is the length of reading frame. We use Monte Carlo methods to get the mutant objects. At most $\varepsilon c^{M/T} N$ mutant populations are obtained by randomly flipping the bit within a reading frame.

3.2.5 Source to Start Searching on Tapes

Each tape has a source where an attached machine starts to search for the head and tail pattern. Starting from the site, patterns are searched for in the clockwise direction of a circular tape. When a head pattern is found, a tail pattern starts to be searched in the clockwise direction. The site of source of a tape can be updated randomly when the tape is newly generated after an extinction. Note that every translational invariant tape has the same source.

3.3 Destabilization of a Minimal Self-replicating Loop

About 10 randomly selected machines with 2 or 3 tapes are prepared as the initial configuration. A machine without description tape is unstable and smoothly removed from the system. Hence an initial configuration which does not include any description tapes of the initial machines will die out if there is no mutation process. External noise may produce description tapes by mistake.

Even without external noise, a machine can acquire its description tape by the other machines as a normal product. To sustain the tape, we have to have the description tape of the machine which generates the tape of the first machine. In order to make it stable, a successive reproducing process should form a closed loop; each machine on the loop reproduces a machine for the next position. (Refer Fig. 3.8-a).) Detailed discussion will be given in §3.6.

We will see how the replicating networks evolve by changing the amount of external noise. Examples of temporal evolution of population of machines and tapes are shown in Fig. 3.1.

By introducing a lower amount of external noise in a system, we see a minimal autocatalytic loop evolve. In this example, a machine $\mathbf{M_{1002}}$ reads a tape $\mathbf{T_1}$ to replicate the same tape and its own machine. The number attached to tapes and machines are hexadecimal number converted from its binary representation. Many initial configurations reach this minimal autocatalytic system for a lower noise regime.

A system with the minimal autocatalytic loop is said to be metastable since it remains stable after turning off external noise. However the minimal loop is destabilized by increasing external noise. Translation under external noise generates many machines, most of which rewrites existing tapes. Increasing of parasite machines destabilizes the original self-replicating loop. In Fig. 3.1-a), a parasitic machine $\mathbf{M_{1222}}$ invades the network with its description tape $\mathbf{T_{41}}$.

A greater variety of machines and tapes induces unstable oscillation in Fig. 3.1-b). An original self-replicating pair becomes unstable if too many parasitic machines attach to it. Population of each machine and tape show unstable oscillation in time. An oscillation of large amplitude is caused by the original self-replicating loop. Other oscillations are caused by parasitic machines and tapes.

This temporal oscillation spontaneously crashes by exhausting the self-replicating loop. The system restarts by having the minimal self-replicating loop, otherwise it become extinction.

The minimal loop shows zero active mutation as being depicted in Fig. 3.1-a). Periodically rising of active mutation rates are caused by the parasite machine $\mathbf{M_{1222}}$ and the tape $\mathbf{T_{41}}$.

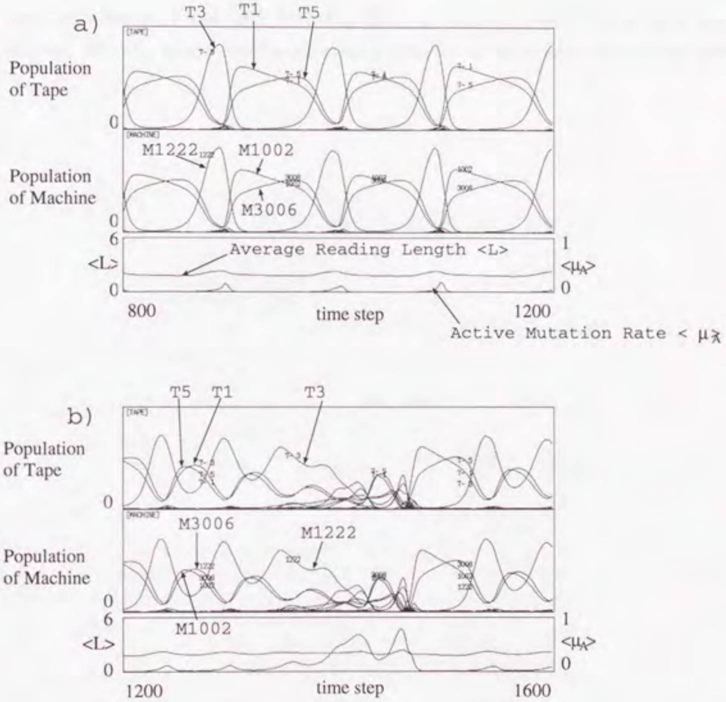


Figure 3.1: Temporal evolution of population of machines (the top row in both graphs) and tapes (the middle row). The bottom row displays temporal evolution of averaged active mutation rates and the averaged length of the reading frame. The parameters of population dynamics are $c^{M/T} = d^{M/T} = 0.6$ in the influence of, a) lower external noise ($\mu_P = 0.04$) and b) higher external noise ($\mu_P = 0.055$). Both start from the same initial states.

The active mutation rate intermittently bursts in time as shown in Fig. 3.1-b). It should be noted that a low rate of active mutations means that the network has more individual self-replication; each tape is self-replicated without mutation. But a high active mutation suggests that many machines read tapes and producing different machines with different tapes from the original ones. Namely, failure of self-replication is reflected in the amplitude of active errors.

3.4 Emergence of Core Network

In the region of high external noise ($\mu_p \geq 0.05$), a stable structure seems to be evolved. Unstable oscillation in population amplitude as we see in Fig. 3.1-b) is spontaneously stabilized around time step 600 in Fig. 3.2. At the same time, the active mutation rate is sustained at the high level.

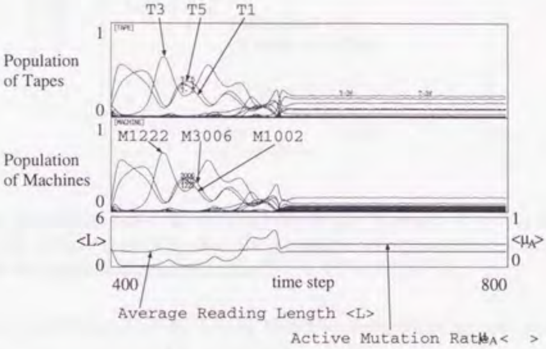


Figure 3.2: A spontaneous transition into a core network of a fixed point state. It displays temporal evolution of population of machines (the top row) and tapes (the middle row), and averaged active error rates and the averaged length of the reading frame (the bottom row). The rate of external noise is set at $\mu_P = 0.07$. The parameters of population dynamics are $c^{M/T} = d^{M/T} = 0.6$.

If we turn off external noise after time step 600, the numbers of machines and tapes remain stable. But if the noise is removed before time step 600, it will back to a minimal self-replicating loop. We call a network which acquired an implicit stable structure a core network.

Fig. 3.3 shows an example of temporal evolution of machines and tapes before and after a transition to a core network and after turning off external noise. After the transition the amplitude of total number of machines and tapes become less rugged than that of before the transition. Generally, machines disappear from the system when turning off the noise. Then a true core network is left in the system.

A minimal self-replicating loop (Fig. 3.1-a)) is also an example of core net. It cannot sustain

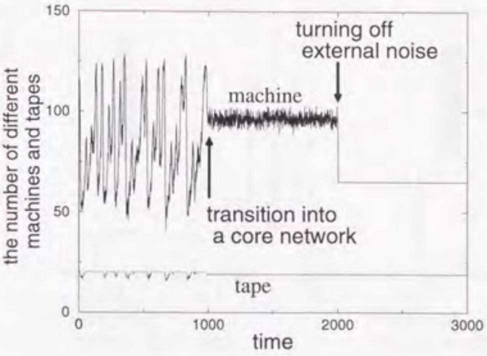


Figure 3.3: Temporal evolution of distinct numbers of machines and tapes. External noise($\mu_P =$ 0.08) is turned off at time step 2000 after the emergence of a core network at time step 1100. The parameters of population dynamics are $c^{M/T} = d^{M/T} = 0.6$.

an active mutation rate by definition. All the examples of complex core networks which appear at middle external noise range ($0.05 \leq \mu_P < 0.1$) have high active mutation rates. By increasing the amount of external noise more, an attainable core net again loses complexity, becoming a minimal self-replicating loop.

In Fig. 3.4, we depict the number of distinct machines and tapes and an active mutation rate of core nets as a function of external noise, which are attained by turning off external noise at time step 2000. This diagram depends on the initial configuration of machines and tapes. Some initial configurations never attain any core networks. However the general tendency is that complex core networks emerge in he mid range of ($0.05 \leq \mu_P < 0.1$) of external noise. Namely, there exist upper and lower bound on external noise to evolve rich core networks. Core networks with a fixed point state are more attainable than ones with oscillating states. A core network at external noise of 0.05 contains a rather smaller number of machines than the other core networks. It has been found that this network uses more 0-rich tapes, which contain a bit of state 0 more than that of 1, in autocatalytic loops. Whereas the other core networks in a fixed point state use more 1-rich tapes, which contain a bit of state 1 more.

As exemplified at a range of $\mu_P = 0.6$ in Fig. 3.4, a core network is not necessarily a fixed

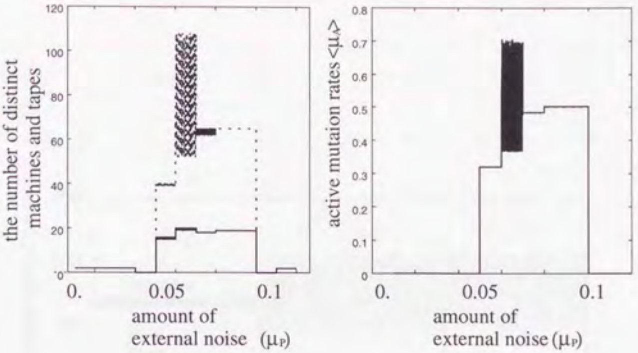


Figure 3.4: Temporal evolution of the distinct numbers of machines and tapes (left figure) and those of active mutation rates (right figure) during time step $[2500, 3000]$ are depicted as a function of external noise. The parameters of population dynamics are $c^{M/T} = d^{M/T} = 0.6$.

point state. It may start to oscillate after turning off the noise. We call it an oscillatory core network. Temporal evolution of the number of machines and tapes, and μ_A is described in Fig. 3.5. The topology of a network changes in time. Note that a network is called a fixed core net, if there is constant number of machines and tapes. Emergence of temporal oscillation of a network depends on the sustained network topology.

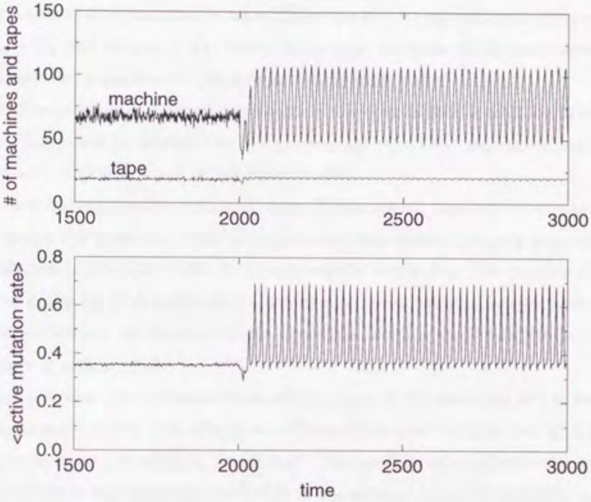


Figure 3.5: Temporal evolution of the distinct numbers of machines and tapes, and average of active mutation rate of oscillatory core. After turning off external noise ($\mu_P = 0.06$) at time step 2000, we observe oscillation of them. The parameters of population dynamics are $c^{M/T} = 0.6, d^{M/T} = 0.4$.

3.5 Taxonomy of Core Networks

As mentioned above, there are some attainable states of core networks depending on initial configurations of machines and tapes. They can be phenomenologically classified with respect to the dynamical states of active mutation rate.

Roughly, there are 3 distinguishable states – active mutation rate with zero, high constant and oscillating state. The zero active mutation rate state is a minimal self-replicating network which contains only a pair of a machine and a tape. There are two strong self-replicating machine-tape pairs, $\mathbf{M_{1002}} - \mathbf{T_1}$ and $\mathbf{M_{2004}} - \mathbf{T_1}$. If the latter type emerges, the system cannot develop a complex core network regardless of the amount of external noise.

There exist diverse kinds of network topologies in the core networks with high constant active mutation rate. They can be divided into two grades, one with lower number of machines ($\simeq 40$) and the other with higher number of machines ($\simeq 60$).

Many cores are in dynamically oscillatory state. These are fall into four subgroups: 1) Periodic oscillation of active mutation rate with varying its machine number ranging from 50 to 110. The period of oscillation is about $20 \sim 22$. 2) Quasi-periodic oscillation. The number of machines is same as that of group 1). 3) Amplitude of machine number is bounded between 80 and 110 with quasi-periodic oscillation. 4) Quasi periodic oscillation with a small amplitude. The average machine number is roughly 110.

These groups are also discriminated from return maps of the averaged active mutation rate. The typical figures are in Fig. 3.6. The width of amplitude are the same in Fig. 3.6-a) and -b). We can observe frequency locking in Fig. 3.6-a). The oscillatory core networks as depicted in Fig. 3.6-c) show higher and narrower oscillation in the average of active mutation rate than that of in Fig. 3.6-a) and -b). The oscillation of the number of machines is also high and narrow. the full number of tapes, on the other hand, are sustained in the networks of -c), 20 tapes exist all the time. Fig. 3.6-d) has also high and very narrow amplitudes of both the active mutation rate and the number of machines. The time series of the active mutation rate displays low frequency modulations and its power spectrum has many peaks.

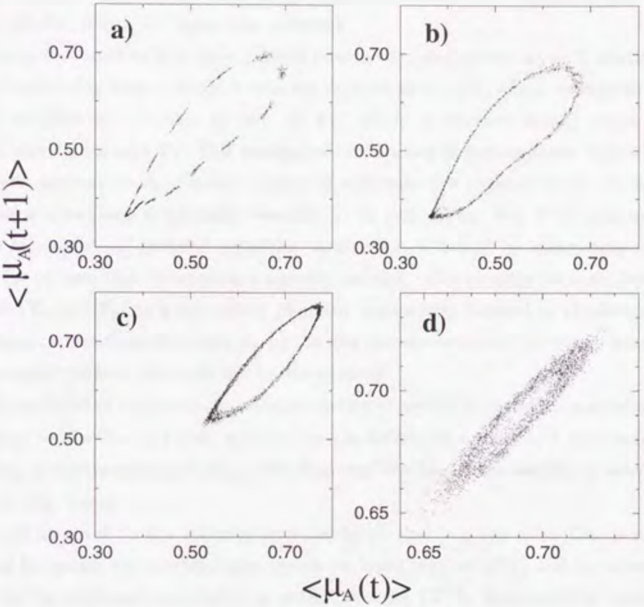


Figure 3.6: Return map of the averaged active mutation rate ($(\langle\mu_A(t)\rangle$ v.s. $\langle\mu_A(t+1)\rangle)$ for four distinguished oscillatory core networks. a) The widest amplitude, frequency locking. Period of main oscillation is around 21 or 22. b) the widest amplitude, c) middle amplitude, d) the narrowest amplitude, note that the range of this graph is different from the others. These are constructed from data taken after the cutting off the external noise. The rate of external noise is a) $\mu_P = 0.04$, b) $\mu_P = 0.04$, c) $\mu_P = 0.06$ and d) $\mu_P = 0.07$. The parameters of population dynamics are $c^{M/T} = 0.6$, $d^{M/T} = 0.4$ for all graphs.

3.6 Embedded Autocatalytic Loops

The embedded autocatalytic loops is an important criterion to distinguish network states. Since Eigen and Schuster's pioneering work [24], a notion of autocatalyticy has been known as a useful razor. We use this notion to dissect core networks.

Machines to be sustained in a system should possess their description tapes. Evolution from a simple self-replicating loop to complex ones are depicted in Fig. 3.7, where we express a system by actual machine-tape reaction graphs. In Fig. 3.7-a), a machine $\mathbf{M_{1002}}$ copies itself by reading its description tape $\mathbf{T_1}$. This minimal self-replicating loop exemplifies Eigen-Schuster's autocatalytic network type. Namely, a tape is self-replicated without error. In high noise regime, the minimal loop is gradually destabilized by side chains. Fig. 3.7-c) and -d) show a successive appearance of parasitic networks. A structure obtained by subtracting a network described in -c) from that in -d) gives a parasite network. This parasite network depends two input tapes ($\mathbf{T_3}$ and $\mathbf{T_5}$) on a network c). Similarly, a structure obtained by e) minus d) gives a hyper parasite, which depends a tape $\mathbf{T_b}$ on the first parasite network. Without a host network in -c), successive parasite networks will be extinguished.

A finally established core network is a combination of several loops which are autocatalytic. With respect to machine sets only, a closed loop is defined as a chain of k machines where a machine $\mathbf{M}_j$ generates a machine $\mathbf{M}_{j+1}$, the final machine $\mathbf{M}_k$ generating $\mathbf{M}_1$ as schematically displayed in Fig. 3.8-a).

A tape set attached to this machine loop can be divided into two sets. One is a tape set which need to sustain the machine loop, that is an input tape set $\{\mathbf{T}_j^{\mathrm{I}}\}$ and the other is a set produced by the machine loop, that is an output tape set $\{\mathbf{T}_j^{\mathrm{O}}\}$. Each machine reads at least one of the tapes in $\{\mathbf{T}_j^{\mathrm{I}}\}$ and generates one belonging to $\{\mathbf{T}_j^{\mathrm{O}}\}$.

A set $\{\mathbf{T}_j^{\mathrm{I}}\}$ is prerequisite for sustaining machine loops. If this set is automatically produced by the corresponding machine loop, it is called an autocatalytic loop. Namely, tape set in a network must be satisfy the following condition for the network to be independently autocatalytic,

$$\{\mathbf{T}_j^{\mathrm{I}}\} \subseteq \{\mathbf{T}_j^{\mathrm{O}}\} \, . \tag{3.10}$$

Otherwise a network has to use the products of other loops to compensate for necessary tapes. It is a necessary condition for any independent autocatalytic loops that a tape set needed to sustain the machine loop is self-reproduced by the machine loop.

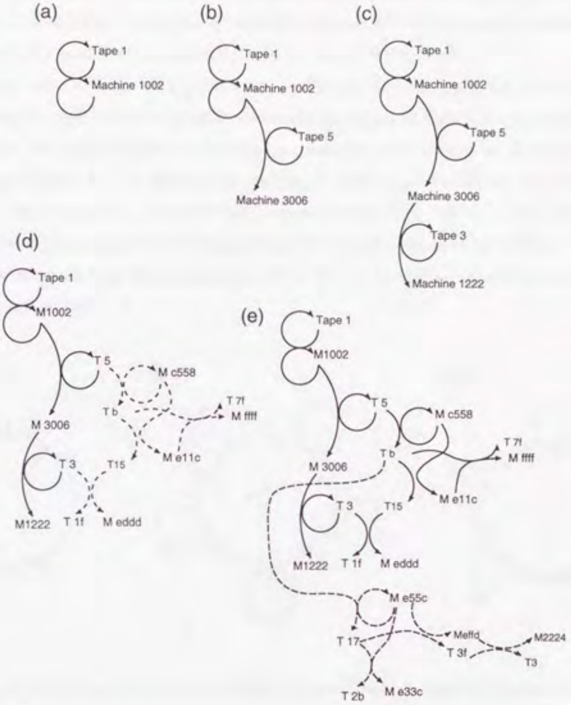


Figure 3.7: Evolution from a simple self-replicating loop to a complex network is described. a) An initially existing a minimal self-replicating loop, where a machine $\mathbf{M_{1002}}$ replicates itself by reading the tape $\mathbf{T_1}$. A net (a) will be successively exploited by $\mathbf{M_{3006}}$ with $\mathbf{T_5}$ (b)then by $\mathbf{M_{1222}}$ with $\mathbf{T_3}$ in (c). The net which is depicted on d) is a net c) with a parasitic network hanging on it. This is named a parasitic net, since as a network it depends on a net c) for the input tapes $\mathbf{T_5}$ and $\mathbf{T_3}$ to maintain its structure. Connections in the parasite network are drawn by broken arrow. In a network e), a hyper-parasite (drawn also by broken arrows) hanging on a parasite net in d) is depicted.

As special case, the autocatalytic condition is locally satisfied, hence each tape is self-replicated. A network of this type is called the Eigen-Schuster type (Fig. 3.8-b)). However, an autocatalytic condition is not locally satisfied in general. In the noisy environment, a network which globally satisfies the condition will be emerge (Fig. 3.8-c)).

In the general case of Fig. 3.8-c), not only machines but also tapes form a catalyzing loop structure. Hence we call the autocatalytic networks emerging in high noise regime a double loop network. In the low noise regime, a network can sustain its structure by Eigen-Schuster type, depicted in Fig. 3.8-b). For the high noise regime, in contrast, it is difficult to maintain prefect replication. It then switches to the double loop structure (Fig. 3.8-b)). Since Eigen-Schuster type only allows replication without error, it can be called a replicating system. On the other hand, a double network type locally changes tapes. It can be called an editing system, as tapes are used by being edited.

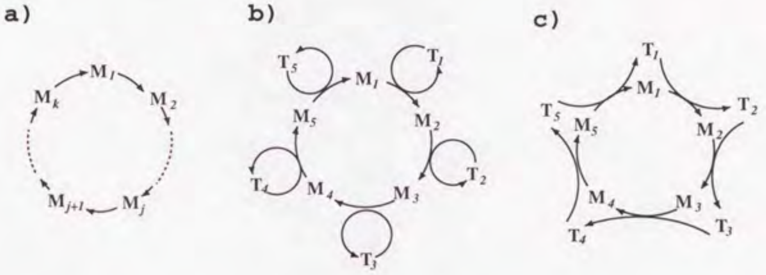


Figure 3.8: a) A loop of machines is schematically described. A machine produces a subsequent machine. b) and c) are illustrations of two different types of autocatalytic loops. They are Eigen-Schuster type b) and the double loop type c). In Eigen-Schuster type, tapes are replicated without mutations. In the double loop type, both each machine and tape are converted into subsequent one.

An example of core net in fixed point found in our simulation is depicted in Fig. 3.9. There are five double autocatalytic loops here. Each loop is extract in Fig. 3.10. Loops of Fig. 3.10-a) and b) belong to the Eigen-Schuster type and a loop of d) is the double loop type, which has no self-replicating tape at all. A loop of c) is also a double loop type but they have both replicating and editing reactions.

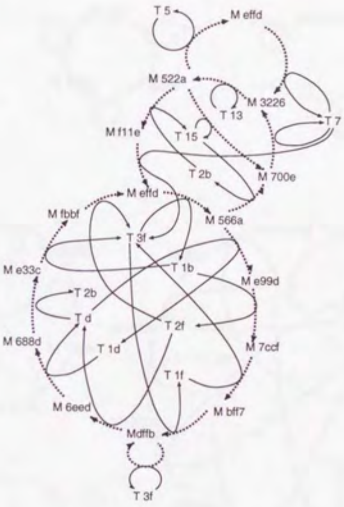


Figure 3.9: Embedded autocatalytic loops which are found in the core network after turning off external noise are depicted. We can see five independent autocatalytic loops here. Three are of the Eigen-Schuster type and the other two are of the double loop types.

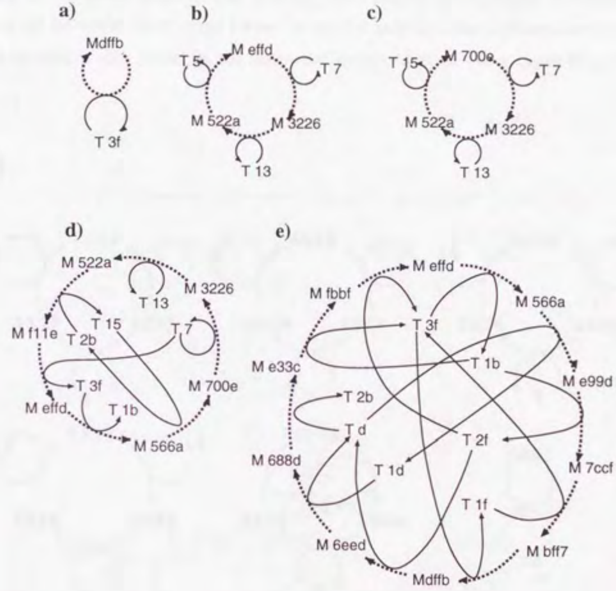


Figure 3.10: 5 loops in a fixed point core network in before figure are separately displayed. a), b) and c) are Eigen-Schuster type. There are both replicating and editing reaction in d) and e) consists of only editing connection.

Fig. 3.11-a) ~ g) is examples of loops consisting of an oscillating core network. The oscillation is found to be the first group classified in §3.5, active mutation rate periodically oscillates, the number of machines ranges from 50 to 110. The network topology changes in time. After networks depicted in Fig. 3.11-g), very long and many autocatalytic networks appear. The largest length in the loops is about 60 and the maximum number of loops attains 200. Thereafter, it goes back to the same loops in Fig. 3.11-a). The period of oscillation is around 22 steps. Though not all networks have same forms, it can be said to have a recursive structure. The recursive structure is also found in the quasi-oscillatory cases of other types 2) ~ 4) classified in §3.5.

a)

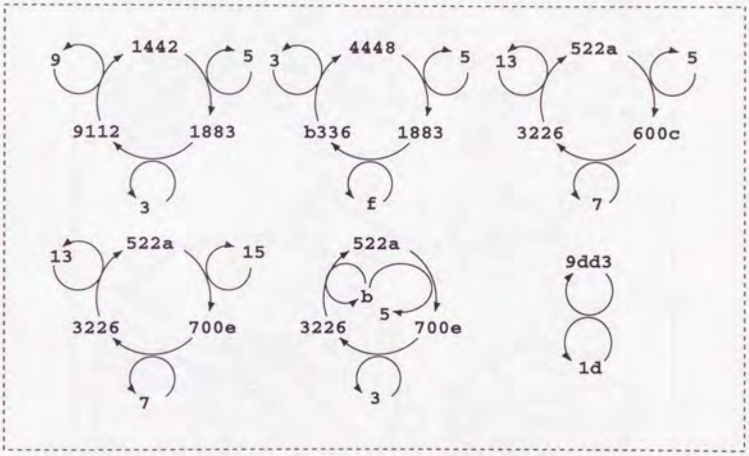


Figure 3.11: These figures are examples of embedded autocatalytic loops found in a oscillatory core network of first type. After g), very long and many loops appear. They come back to the same loops in a) after about 22 time steps.

We pick up in Fig. 3.11 loops of reaction paths which fulfill the autocatalytic condition defined by eq. (3.10). Note that networks depicted in Fig. 3.11 are not autocatalytic loops in a strict

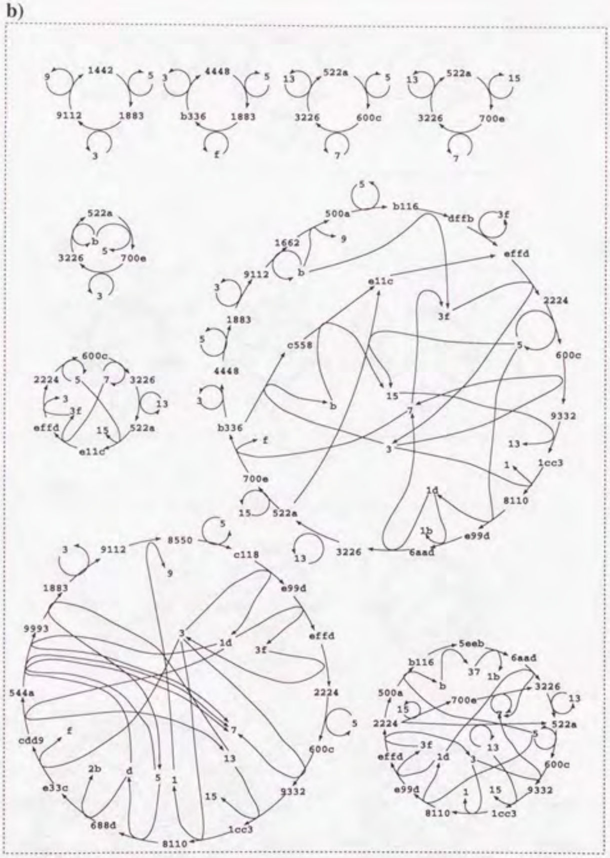


Figure 3.11: (continued)

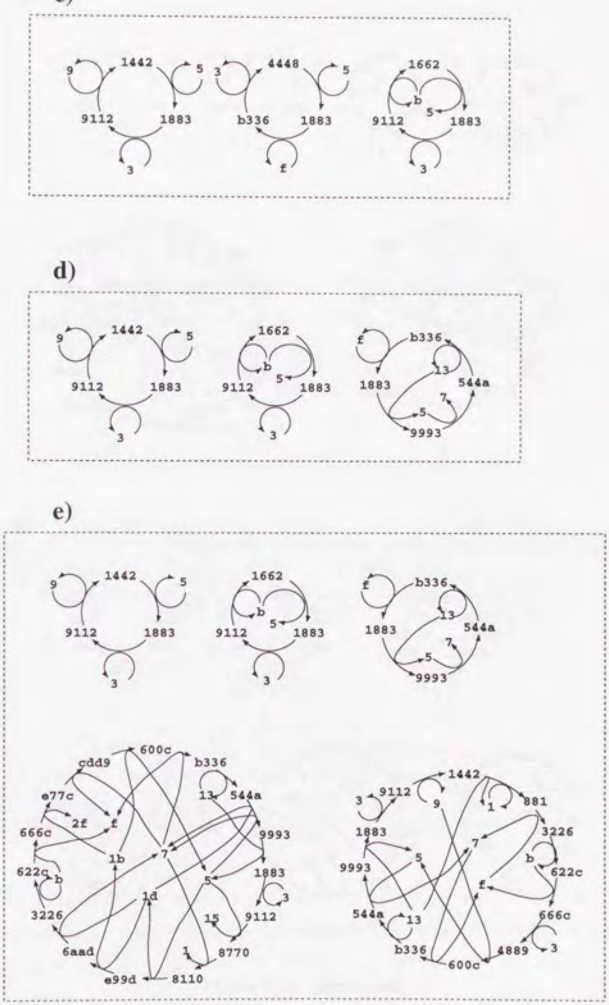


Figure 3.11: (continued)

f)

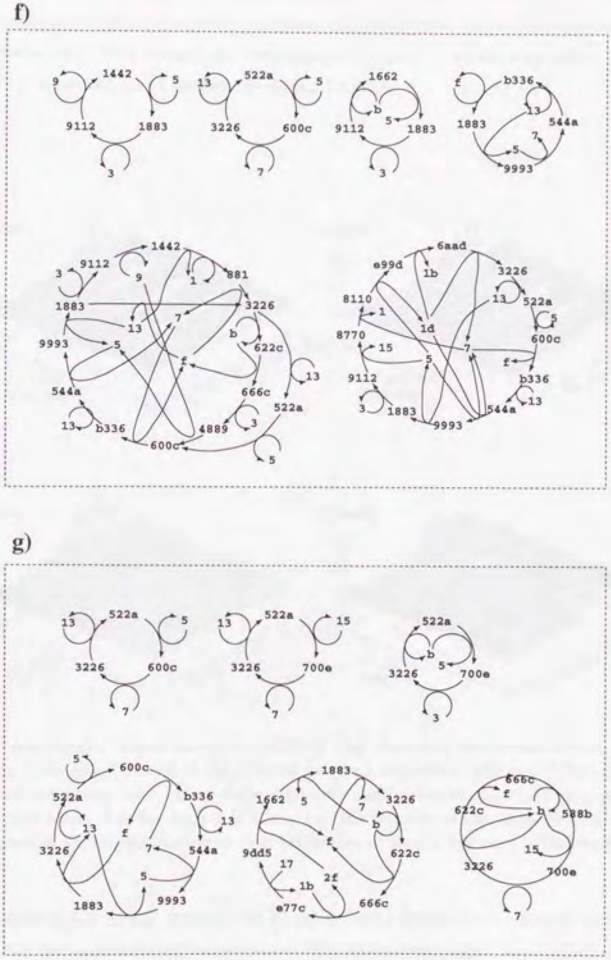


Figure 3.11: (continued)

sense, because they can not hold its structure stably.

Cores in oscillatory states change the number of autocatalytic loops. The number of Eigen-Schuster type autocatalytic loops shows less change than that of double loops type. In Fig. 3.12 we show temporal evolution of histogram of loop length.

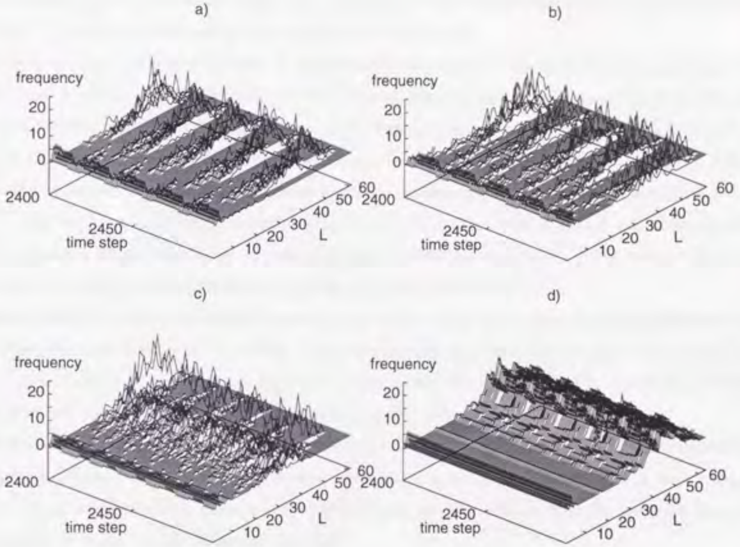


Figure 3.12: Histograms of lengths of autocatalytic loop change in the course of time as shown in these graphs. They are depicted in the interval between time step 2400 and 2500 after external noise is cut at time step 2000. They show relatively stable oscillations than immediately after cutting external noise. L is the length of a loop i.e. the number of machines in an autocatalytic loop. Four graphs are corresponding to four categories of oscillatory core networks described in Fig. 3.6.

The core nets shown in Fig. 3.12-a) and b) periodically switch their network structure from a network with few autocatalytic loops to one with many large loops. Fig. 3.12-c) shows many autocatalytic loops with length $L \simeq 30 \sim 40$ and periodic bursts to larger loops ($L = 45 \sim 61$).

Loop length around $L = 35$ grows when the burst occurs. These bursts are observed in a short period of time. Average of the loop length is around $L = 36$, which differs from the other networks, $L \simeq 42$. The core network depicted in Fig. 3.12-d) has two $L = 1$ loops, five $L = 3$ loops and a $L = 16$ loop for all time. The histogram of loop length shows small fluctuation. These facts suggest that the core network in Fig. 3.12-d) has a relatively stable structure similar to core network at fixed point state. Thus we observe that the return map of active mutation rate is torus with a small radius as described in Fig. 3.6-d).

Almost all core networks consists of autocatalytic loops with inhomogeneous couplings. Machines and tapes have the different input arrows from the output arrows apparently from Fig. 3.9. Besides, several autocatalytic loops shares part of their reaction paths. An example is Fig. 3.10-c) and d). They share the reaction paths $\mathbf{M_{700e}} \rightarrow \mathbf{M_{3226}} \rightarrow \mathbf{M_{522a}}$. Many cores are robust against perturbations that removes some tapes or machines from the network artificially. It is noted that we have introduced a machine to be able to create some different machines through interpretations of various tapes. A machine can catalyze several tapes. This redundancy contributes to complex and dynamical networks with sufficient stability.

Autocatalytic networks of double loop type are more stable than those of Eigen-Schuster type. For example, a loop with 16 machines of the double loop type is found in Fig. 3.12 for all time. In contrast, longer Eigen-Schuster type loops than three are not found. The stability of double loop network depends on the detailed topology of the network.

A tape can serve as templates of many different machines by being differently interpreted by different machines. But the core network evolves code, a set of interpretations. A core network selects some interpretations from diverse interpretations. We will describe the network from the viewpoint of "code" in the following section.

3.7 Diversity and Dynamics of Code

We assume an affinity between machines and tapes. The affinity is given by the condition that machines can read tapes if the head and tail patterns are found on tapes. Despite this affinity, there is a large arbitrariness by which machine a tape is read and as which machine the tape is interpreted. Selection of this interpretation ways constitutes a "code" of this system. A code of a network consists of interpretation rules as $(\mathbf{T}_i, \mathbf{M}_j) \to \mathbf{M}'_i$. For example, in the autocatalytic loop shown in Fig. 3.10-b), a machine $\mathbf{M_{effd}}$ reads a tape $\mathbf{T_7}$ and interprets it as a machine $\mathbf{M_{3226}}$. The interpretation rule for the process is $(\mathbf{T_7}, \mathbf{M_{effd}}) \to \mathbf{M_{3226}}$.

We depicted five autocatalytic loops in Fig. 3.10 consisting an example of fixed point core network. In Fig. 3.10-a), b) and c), there is a unique interpretation rule for each tape in the loops. And the loops b) and c) share their code, since interpretation ways for $\mathbf{T_7}$ and $\mathbf{T_{13}}$ is the same, $(\mathbf{T_7}, \mathbf{M_{effd}}) \to \mathbf{M_{3226}}$ and $(\mathbf{T_{13}}, \mathbf{M_{3226}}) \to \mathbf{M_{522a}}$. On the other hand, the other loops in Fig. 3.10 have more than two different ways to interpret a tape. A tape $\mathbf{T_7}$ in d), for example, is interpreted as a machine $\mathbf{M_{3226}}$ by $\mathbf{M_{700e}}$ and as $\mathbf{M_{effd}}$ by $\mathbf{M_{f11e}}$. The code in a core network composed of these five loops is shown in Table. 3.2. Both polysemy and synonymy are found. Polysemous tapes are $\mathbf{T_7}, \mathbf{T_{3f}}, \mathbf{T_d}, \mathbf{T_{2f}}$ and $\mathbf{T_{1b}}$. These tapes produce distinct machines through interpretations by distinct machines. The tape $\mathbf{T_7}$, for example, is interpreted as either $(\mathbf{T_7}, \mathbf{M_{effd}}) \to \mathbf{M_{3226}}$, $(\mathbf{T_7}, \mathbf{M_{700e}}) \to \mathbf{M_{3226}}$ or $(\mathbf{T_7}, \mathbf{M_{f11e}}) \to \mathbf{M_{effd}}$. Synonymous tapes produce one machine through interpretations by distinct machines. Tapes for $\mathbf{M_{effd}}$ is synonymous. It is produced from three different tapes, $\mathbf{T_7}, \mathbf{T_{2f}}$ and $\mathbf{T_5}$. We can also find that a tape is interpreted as one machine by distinct machines. For example, a tape $\mathbf{T_{15}}$ is interpreted as the same machine $\mathbf{M_{700e}}$ by the machines $\mathbf{M_{522a}}$ and $\mathbf{M_{566a}}$.

In minimal and fixed point core nets, the code cannot change even with large external noise. But for the oscillatory core network, the code can change temporally without noise. Table. 3.3 represents an example of temporal changes of coding system in an oscillatory core network. We depict interpretation ways only for tape $\mathbf{T_5}$ in autocatalytic loops of the oscillatory core net described in Fig. 3.11 in this table. There are 13 interpretation rules, 9 are distinct, in the period of time $t = 2408 \sim 2418$. A code at one time can not serve replication of the same core network. Constituents of a core net increase or decrease their concentrations by reactions according to the code at that time. But after a period, the same code reconstructed. For example, a machine $\mathbf{M_{4448}}$ exist at time steps $t = 2408 \sim 2412$. But by following the code at that time, the machine is not produced efficient amount. At last, it disappears from a core

Table 3.2: Example of code in a core network in fixed point state are shown in this table. This is a code in Fig. 3.9. A machine $\mathbf{M}$ reads a tape $\mathbf{T}$ and interprets it as a machine $\mathbf{M}'$ There are polysemous tapes, $\mathbf{T_7}, \mathbf{T_{3f}}, \mathbf{T_d}, \mathbf{T_{2f}}$ and $\mathbf{T_{1b}}$. These tapes are respectively interpreted as two different machines by two or three different machines. Synonymous tapes are also exist. $\mathbf{M_{effd}}$ is produced from three tapes, $\mathbf{T_7}, \mathbf{T_{2f}}$ and $\mathbf{T_5}$. A tape $\mathbf{T_{15}}$ is interpreted as the same machine $\mathbf{M_{700e}}$ by the machines $\mathbf{M_{522a}}$ and $\mathbf{M_{566a}}$.

T	M	M'	T	M	M'
7	effd	3226	3f	dffb	dffb
	700e			bff7	
	f11e	effd		effd	566a
15	522a	700e	d	688d	e33c
	566a			566a	e99d
2f	dffb	6eed	1b	e33c	fbbf
	fbbf	effd		e99d	7ccf
5	522a	effd	13	3226	522a
1d	6eed	688d	2b	522a	f11e

network at $t = 2413$. Its revival is at $t = 2418$. And the same interpretations with $t = 2408$ appear at $t = 2431$ again.

In general, diversity of code of core network in oscillatory state is higher than that of fixed point. It is also true for the length of autocatalytic loops. The diversity of code and length of an autocatalytic loop may have relevance to the stability of network. But it is not proved yet. Stability of autocatalytic loops can vary according to their topologies even if they have the same length.

Table 3.3: As an example of temporal changes of code, variable usages of tape $\mathbf{T_5}$ in the oscillatory core networks described in Fig. 3.11 are depicted in the period time step from 2408 to 2418. When a machine $\mathbf{M}$ reads a tape $\mathbf{T_5}$, it generate a machine $\mathbf{M'}$. In this example of core net, 13 interpretation mays of tape 5 appear. The tape is polysemously interpreted as 9 different machines. A machine can be produced by distinct machines. For example, a machine $\mathbf{M_{1883}}$ can be produced from machines $\mathbf{M_{1442}}, \mathbf{M_{4448}}$ or $\mathbf{M_{9993}}$.

time	2408 ∼ 2410			2411			2412			2413		
	T	M	M'	T	M	M'	T	M	M'	T	M	M'
1)	5	1442	1883	5	1442	1883	5	1442	1883	5	1442	1883
2)	5	4448	1883	5	4448	1883	5	4448	1883			
3)				5	9993	1883						
4)												
5)				5	544a	9993						
6)	5	522a	600c	5	522a	600c						
7)				5	2224	600c						
8)				5	8550	c118						
9)				5	522a	e11c						
10)				5	8110	e99d						
11)				5	500a	b116						
12)												
13)												

time	2414			2415∼2416			2417			2418		
	T	M	M'	T	M	M'	T	M	M'	T	M	M'
1)	5	1442	1883	5	1442	1883						
2)										5	4448	1883
3)												
4)										5	4448	9993
5)	5	544a	9993	5	544a	9993	5	544a	9993			
6)				5	522a	600c	5	522a	600c	5	522a	600c
7)												
8)										5	8550	c118
9)										5	522a	e11c
10)	5	8110	e99d	5	8110	e99d						
11)												
12)	5	9112	8770	5	9112	8770						
13)										5	c118	ebbd

3.8 Discussions

By introducing ensemble of machines and their description tapes, we have studied self-organization of a stable code for self-reproduction correlated with self-organization of an autocatalytic network. It is difficult to organize a stable code in general. An optimal coding of a machine may lead to the worse coding of the other. How to improve coding cannot be determined locally but should be determined in a network context. In a fixed core net, a stable code with polysemy and synonymy is organized. Alternation of coding driven by external noise has also been reported recently [44]. In their systems, a multi coding system, i.e. code with polysemous interpretations, is developed under a high noise environment. To make a stable code is to close logic consistently. It has succeeded in a core network in fixed point state. But for a oscillatory core network, its code dynamically changes. A code destabilizes itself and this results in oscillation of coding structure. We refer to such self-destabilizing nature of code as *openness of code.*

A minimal self-replicating loop, composed of one machine and one tape, emerges under influence of external noise. When external noise is elevated, a minimal self-replicating loop is exploited by parasitic networks. The populations of machines and tapes then begins to show unstable oscillation. In a realm of relatively high external noise, a system will evolve into a stable network against external noise spontaneously. When a network attains this state, it is stably sustained even after external noise is removed. We call the network acquiring such noise stability a "core network". The core network consists of many autocatalytic loops, being a fixed point or oscillatory state.

A transition from an unstable state to a core network state is similar to what has been seen in a host-parasite model [54, 58]. In the host parasite model, chaotic instability is shared by almost all species by sustaining a high mutation rate, leading to weak high-dimensional chaos, termed homeochaos. In the present model, roles of host and parasite emerge spontaneously in a network. Interaction between host and parasite loops causes dynamic instability as well. The core structure suppresses the instability and as the result a high active mutation rate is sustained in the present model.

We have introduced two different mutations – *passive and active mutation.* Taking the active mutation as a computation process, we propose a computational view of mutation and self-replication. A minimal self-replicating loop has no active mutation, that is no replication errors. A minimal self-reproduction loop can attain without any computation. If the computation process offers costs, the minimal loop may be the most economic self-replication system. Such autocatalytic network is stable only below some noise level. A large fixed core network, on the other hand, sustains a high active mutation rate in a high external noise regime. It absorbs external noise as active mutation of machine function. Namely, passive mutation caused by external noise is replaced as deterministic mutation by doing computation. The network carries and processes large amount of information in a core network.

Several autocatalytic loops are embedded in a core network. These loops are divided into two types. One has no active mutation. Tapes are replicated locally. It corresponds to the Eigen-Schuster's hypercycle. We call this type of network Eigen-Schuster (ES) type autocatalytic network. The other has a high active mutation rate. It consists of double loops of machines and tapes. Namely, each tape as well as machine produces a successive tape. Both entities are replicated as a whole, not individual. We call this type of network double loop type autocatalytic network. We also have mentioned that this type of network can be regarded as a editing system. Since tapes in the network are rewritten to templates of different machines, in contrast to tapes in ES type network, which are replicated without errors.

It is thought that DNA is replicated individually and RNA is a copy of DNA in real biological systems. However, an example [38] may be related to our double loop autocatalytic network. The authors of [38] has shown that DNA of macronucleus has been generated from DNA of micronuclear in Oxytricha nova. DNA of micronuclear is transcribed once into RNA. Then the exons are completely rearranged and reversely transcribed into the DNA of macronucleus. If editing and reversely transcribing RNA is more stable than replicating DNA itself, an active mutation as editing will be favored in early genetic systems.

It is said that biological system has evolved from RNA world to DNA world through RNP world [36, 35, 109]. RNA world consists of RNA molecules having catalytic functions as well as replicating function. Then it has evolved to the world including RNAs and proteins (RNP world). Even in such RNP world, self-reproduction is still unstable. Editing of RNA must repeatedly occur. By using more stable polynucleotide, DNA world has evolved. RNA, protein and DNA cooperatively work in the DNA world as in the present biological systems.

Our modeling can correspond to the second level of real evolution (RNP world). Differentiation of ES type and double loop type in autocatalytic networks can be taken as a transition from RNP to DNA world.

Autocatalytic loops in our model overcome the difficulties asserted by Niesert et al. [76]. They have discussed that an autocatalytic network become unstable either by selfish replicator, short circuit or zero concentrations (see §3.1.3). And the network is destabilized by inhomogeneous couplings. The inhomogeneous coupling is that a tape catalyzes more (less) tapes than being catalyzed and the similar asymmetry exists for machine reactions. In our core network, autocatalytic loops with different length, even minimal loop, can coexist. Oscillatory core nets does not break down by the zero concentrations of component, which is Niesert's third catastrophe. A similar topology is recovered after a period of time. The inhomogeneous couplings are also permitted. Polysemy in a tape, encoding several machines, are important to resolve there difficulties. As Kauffman has discussed [61], redundancy caused by non-specific replication makes an autocatalytic network robust.

The state where no subcomponents of the system are themselves self-replicating has been referred to as a collective reproducing state by Kauffman [61]. The emergence of such ordered state in his autocatalytic polymer model could also escape Niesert's catastrophes. But the inhomogeneous couplings were not able to be managed. There was not a gene like substance in his system. However, reproduction and evolution of unstable entity requires a stable description of itself.

Fontana's "algorithmic chemistry" [30,31] has shown evolution of replicating programs similar to our work. His chemical world consists of strings of variables and operators called functions. Reactions among functions are governed by λ-calculus. A system is dominated by single self-copying strings in a level-0 stage, which corresponds to our minimal self-reproduction loop. By forbidding self-copying function, a mutually catalytic set emerges (level-1 stage). This level-1 stage corresponds to the fixed point core network in our model. In spite of no explicit prohibition of self copying, our system spontaneously attains mutual catalyticy under noisy environment.

In Fontana's next level (level-2 stage), interconverting autocatalytic sets can be combined. This level corresponds to our advanced model [51], where tapes and machines are wrapped within a cell. In an ensemble of the cellular systems, evolution to complex self-reproduction is caused not by external noise buy by influx machines from other cells. Several translations are performed by external machines from other cells. This cooperative behavior brings about diversification of core networks.

Chapter 4

Conclusions and Developments

We define a *code* in a system as ways of using symbols and interpretations of information. We refer to successive emergence and dynamical changes of codes as *evolution of code*. Evolution of code in a communication system and a genetic system have been studied here.

4.1 Summary and Conclusion — Evolution of Grammar Systems

In Chapter 2 we have studied evolution of communicating agents in a network. To discuss evolution of linguistic code and grammar systems, we proposed a language game. Agents communicate with each other by deriving and accepting words in terms of their own generative grammar. They are ranked according to their communicative effectiveness. Agents which speak either long infrequent words, recognize long words quickly or speak words which are not recognized by the other agents overpower others. Mutation of rules are introduced to update agents' grammar. A code in this system is characterized by particular usage of words. Evolution of grammar systems was discussed as evolution of computational ability in the Chomsky hierarchy.

Computational ability of each agent's increases as well as variety of words spoken in the course of time. We have found that information in communication shows a stepwise evolution, where equilibrium states and sudden jumps occur alternately.

In the equilibrium states, agents communicate with common words. We call the community exchanging a common set of words an ECW. Different ECW is characterized by a different common words. When different ECWs conflict, an ECW composed of higher computational ability agents does not always outwit. An ECW with more number of agents wins. That is, ECWs suppress for individual grammar to evolve. Code of communication is discussed in terms of ECW. A particular usage of words determined within an ECW is a code.

On the other hand, breaking up ECWs with evolving grammar is caused by mainly two remarkable mechanisms. One is a module-type evolution. An agent can enlarge a set of recognizable words by acquiring a module rule. The role of modules may be corresponding to that of affixes in natural language. The other mechanism is a loop forming evolution. A grammar containing a loop structure can recursively derive words. If a grammar forms a loop, it can potentially derive infinite number of words. Such grammar can derive nested sentences. Long and rich expressions are produced by phrases within phrases or clauses within clauses structures. A tree to loop structural change presents an evolution from regular to context-free grammar.

The emergence of code and evolution of computational ability can also be seen by different parameter settings. But for the extreme case, the results are very different. Variety of words and information in communication remains very low, if ability for being recognized is a big advantage. In this case, only a code is established but it does not exhibit dynamics. In the opposite limit, agents speak many long words but without mutual understandings. In other words, any code is organized in the network.

Let us mention different points from related works. MacLennan [72] and Oliphant [77] think that codes in communication is owing to matching between each symbol and its referent among agents. But we think that the meaning and indication must not be thought to be identical (cf. [33, 82]). We assume that codes are based on usages of phrases. Therefore evolution of diversity of words and complexity of grammars are brought about without diverse external world different from MacLennan's model [1]. Werner and Dyer [103] has discussed that diversity of language attributes to physical barriers. Since our model consists of only 10 agents because we are interested in dynamics of formation of group word or trendy word in relatively small group, we cannot discuss diversity of code. However, we indicate that conflict between two codes prevents evolution of grammar systems and it results in punctuated equilibria evolution.

4.2 Summary and Conclusion — Coevolution of Tapes and Machines

In Chapter 3, coevolution of tapes and machines are studied. A machine reads a tape and then a new tape and a new machine are produced. We have simulated an ensemble of machines and tapes to discuss computational aspects of mutations and translations. By introducing external noise, we have investigated the role of randomness on machine's deterministic action. In this model, a tape can be interpreted as distinct machines depending on machines reading the tape. Changing tape symbols by machines' deterministic actions is named "active mutation". Probabilistic changing under external noise is called "passive mutation". A code of replicating network is, therefore, a whole set of interpretations of tapes by machines composing the network.

We see the evolution from a minimal self-replicating network to a larger one by increasing external noise. If external noise is small, the minimal network is a final state of evolution. A minimal network consists of a stable pair of machine and tape, which are reproduced locally. When external noise excesses a certain threshold, a spontaneous transition to a large stable net-

---

[1] Our further research suggests that the diversification of words and complication of grammars can be allowed without preference for long and rare words in the definition of score function.

work is observed. After the transition, the network retains a self-maintaining network involving stable core structure, named a "core network". Core networks replicate individual tapes and machines globally.

Dynamical states of the populations of tapes and machines in core networks can be fixed point or oscillating state. Large core networks have high active mutations. They seem to mimic the reaction paths which external noise have forced on.

Stability of core networks is attributed to the combinations of embedded autocatalytic loops. A fixed core consists of several autocatalytic loops. In oscillatory cores, the number and topology of embedded autocatalytic loops changes in the course of time. Autocatalytic loops can be divided into two type. One is named as Eigen-Schuster (ES) type, the other double loop type. Each tape in an autocatalytic loop of ES type is locally replicated. On the other hand, machines and tapes in that of the other type are globally replicated. The latter network has non-zero active mutation. Large autocatalytic networks with inhomogeneous couplings are stabilized by the latter one.

Interpretations of tapes in a network can not be determined locally. It must be organize consistently in the whole network. In a fixed core network, a set of interpretations are fixed. Thus a code for self-reproduction is organized. A network chooses one possible interpretations system to replicate the structure of network. Global restrictions for interpretations make a self-consistent logic in a network. If a logic does not made consistently, we call it *logical openness.* In an oscillatory core network, a code for self-reproduction is not stable. Oscillation of code and network topology can be related to logical openness. A set of interpretations are destabilized by itself. We refer to such self-destabilizing nature of code as *openness of code* .

At last we give a brief summary of new feature in the present study. Our model is development to ensemble context from Neumann's self-reproducing automaton [102]. And autocatalytic network is studied from evolutionary point of view different from Eigen and Schuster's work [24]. Indeed, we show evolution from minimal loop to large and complex autocatalytic network. It is revealed that introducing active mutation process stabilizes complex networks prevailing Niesert's catastrophes [76]. Moreover, the autocatalytic network in our study can have inhomogeneous couplings. Oscillation of network topology is a novel feature.

4.3 Further Developments

We are interested in oscillation induced by logical self-denial structure. A system with a code is threatened the maintenance of the code by itself. The underlying mechanisms of oscillation of network topology in our machines and tapes system are not clarified yet. We can say that oscillation of network topology can be an extension of logic oscillation, which is first discussed in Brownian algebra [11]. Such oscillating nature of logic can naturally occur in our model.

We exemplify other interesting subjects to discuss from the viewpoint of evolution of *code*.

- Social code

  Norms, common senses or ethics in a society govern behaviors and ways of thinking of members in the society. Most individuals have to obey them without intentions. They are spontaneously organized in a society, regulating behaviors of the constituents of societies. We refer such regulations in a society as social codes. Social codes cannot be determined in advance. Evolution of social code is also induced by internal dynamics and logical openness.

- Self-reference code

  A self-referential paradox of laws are worth noting. There are laws to legislate the way to enact laws. We find a self-referential paradox, when we make laws for laws. Hofstadter has introduced an interesting self-modifying game named NOMIC [43]. Players of the game make or modify rules of the game. Players can make some rules which forbid to make or modify other rules. This is an example of logical openness, since laws themselves destabilizes the laws.

- Code in the immune system

  The immune system distinguishes self and non-self in our body. Codes in the immune system are regulations to discriminate self image from non-self image. But what is self or non-self cannot be determined in advance. It is distinguished by whether the immune system responds to antigens or not. A network model of the immune system [55] characterizes the system as the network of antibodies stimulating each other. Both internal and external antigens' images are restructured by the network. The self/non-self image is not dynamically stable. An antibody which constitutes the self image may become non-self. The whole system may be destroyed by some internal/external antigen images. Selecting a network structure may be considered as constructing a code of self/non-self. Therefore a notion of dynamical code developed in our study must be compared with the immune network.

- Code for articulation

  We at first perceive our continuous external world, then articulate it. Assigning symbols comes after the articulation. Codes for articulation determine how articulate our continuous external world and how assign symbols. However, the dualism – the external world is continuous and the internal world is discrete – can be criticized (e.g. [49]). And we may not symbolically process information in our internal world. Lakoff insists that a conceptual system is metaphorically structured and emphasizes the importance of physical experiences [67, 66]. We think that the code for articulation can be determined partially through communication among agents and it can vary temporally and locally (cf. [90, 104]). Articulating the external world and structuring internal world is remained as future problems.

4.4 Remarks on Artificial Life Studies

Langton has proposed Artificial Life studies in his manifesting paper [68]. He summarized the essential features of computer-based Artificial Life models as follows:

- They consist of populations of simple programs or specifications.
- There is no single program that directs all of the other programs.
- Each program details the way in which a simple entity reacts to local situations in its environment, including encounters with other entities.
- There are *no* rules in the system that dictate global behavior.
- Any behavior at levels higher than the individual programs is therefore emergent.

Since both models we have reported here satisfy these features, our studies are kinds of Artificial Life Studies.

Langton and other many researchers on Artificial Life say that "emergence" is the key concept in the field [68, 63]. Here the concept means that the appearance of global or higher level structure which cannot be expected from local or lower level dynamics and interactions [3]. Our main results, ECWs and core networks, are such emergent global structures. We showed not only the appearance of global structures but also dynamical changes of the global structures, which is our main concern – *emergence and dynamics of code*. Taylar stressed the feedback from higher level to lower level as he described "LOCAL to GLOBAL back to LOCAL inter-level feedback loops" [98]. Dynamics, such as successive formation, collapse and oscillation, of global structures are induced from such inter-level interactions.

Computation is another key term in Artificial Life studies. Some classical works has been devoted to treat biological systems or language as computational systems. Neumann's theory of self-reproducing automaton has been a pioneering work of computational treatment of biological system [102]. Chaitin has tried to make an information theoretical definition of life in terms of algorithmic complexity theory for descriptions of biological systems [14]. Chomsky has opened the way to consider language as a computational process with derivational grammar theory [17].

Emmeche has divided concept of computation into four classes as [25]:

- The formal, or algorithmic, concept of computation, which has its theoretical footing in the notion of Universal Turing Machine.
- An informal, intuitive, or "mathematical" concept of computation that is not bounded by the known limitations of formal systems.
- A biological concept of computation. This seems to be a quasi-theoretical concept that can be understood in many ways: for example, as problem solving by learning and adaptability; as molecular processing of information in cells; or as computation by neural networks.
- A physical concept of computation, that might be nonrepresentationalistic. The entities that cooperate in computational enterprises are patterns that can transmit, store, and modify information, but these patterns seemingly do not have to "stand for" anything, as long as no functional constraints are imposed from a higher level.

Our studies are relevant to an informal or a biological concept of computation. We have modeled each agent in our models with simple information processing unit – with derivational grammar systems in the study of linguistic code, and binary strings and Turing-Machine-like automata in the study of genetic code – as the first feature of Artificial Life models listed above. By this modeling we try to catch computational aspects of dynamics of language, self-reproduction and catalytic networks. We has showed in the former study that gaining a loop structure, we named it algorithmic evolution, has enriched the diversity of recognizable words. We has indicated in the latter study that active computation as active mutation has permitted large and complex autocatalytic networks which have been able to storage larger information than hypercycle type autocatalytic networks. Unfortunately, the attempts have not been brought completion. More attention should be paid to interrelations among concepts of computation.

In addition to the concept of computation, the important roles of chaos for biological systems and cognitive systems has been pointed out [100, 58, 54, 57]. Our concern is not only relations between the formal and an informal, and the formal and a biological concept of computation in many agents system with dynamical nonlinear interaction [2] but also roles of chaos in the system. In the formal computation and formal logic system, it is known that there is undecidability such as halting problem of Turing Machine or Gödel's incompleteness theorem [37]. Such undecidability is induced by difference between internal and external viewpoint of the system. It is an interesting point to discuss how such undecidability is caused in modeled biological or cognitive system with constructing internal observers. [3] The importance of an internal observer to understand not only cognitive systems but also physical, chemical, biological system has been pointed out by Rösller [88], Conrad [20] and Tsuda [101, 56].

The aim of Artificial Life studies is said to give pictures of *life-as-it-could-be* [68]. Namely, we want to construct *lifelike* phenomena in artificial media such as computers, robots or biomaterials. Whether a phenomenon is lifelike depends on the definition of life. The definition has been discussed for a long time.

However, we cannot give a settlement to the problem. There must be gray-zone whatever criteria for life is stated. There will exist phenomena which we will thought of as alive but not satisfying these criteria, and phenomena which we will not thought of as alive but satisfying these criteria. Anyway, one of the most large contribution of Artificial Life studies is that they raise discussions on what it means to be alive. But, the discussions often fall into realism or externalism. We are likely to think a phenomenon as lifelike if it is displayed with creature-like representation. Life is neither substance nor concrete state, but is a process arises from cognitive

---

[2] Relation between dynamical system and formal computation system has been studied by Moore. He proved that Turing Machine could be embedded to two dimensional piecewise linear map [74, 75]. His work is remarkable result connecting dynamical systems and computation. It has been practically shown that dynamical systems can have different type undecidability from the sensitivity to initial value usually referred to as chaos.

[3] Undecidability in physical system has been also studied, for example by Svozil [97].

activity. We think that the process always makes unexpected behaviors – truly emergence.

Since such emergent property through cognitive action is essential for evolution, approaching with the viewpoint of internal observer to constructing complex systems comes to be imperative [56]. To model a system with internal observer, not only creatures but also cells or molecules [20,39] should be treated as internal observers. The system consisting of their activities has some kinds of arbitrariness or undecidability, but behavior of the system has to be assigned restrictions by validities peculiar to the system. Cohering between internal and external viewpoint results in dynamics of the system. Such dynamical mechanism will be correlated with the oscillation of code and network topology observed in the study of network of tapes and machines.

Chapter 5

Appendix

5.1 Algorithm for Recognition Process of a Grammar System

For a given grammar, i.e. a set of rewriting rules, there are some algorithm to decide whether a given word can be accepted by the grammar, if the grammar belongs to the class of context free [46, 86]. In this appendix we introduce one of such algorithms which used in our language game.

The basic problem in rewriting process is the undeterminisity about more than two applicable rules and positions to rewrite. In the derivation process, we introduced randomness to adopt a rule from plural applicable rules and the leftmost derivation. But in the recognition process we must determine without ambiguity whether a given word is involved in a language of a given grammar. To this end we introduce a marker to decide a position to apply a rule and test all possible sub-search-trees until the word is rewritten back to the initial symbol.

At first, we transform each rule $(\alpha \rightarrow \beta)$ of a given grammar G to a corresponding reverse rule with a marker which is denoted by #,

$$\beta\# \rightarrow \alpha\# \ . \tag{5.1}$$

Second, add two rules to move the marker at the end of rule list,

$$\#0 \rightarrow 0\# \ , \ \#1 \rightarrow 1\# \ . \tag{5.2}$$

A grammar with marker $G^{\#}$ is constructed by these transformation rules for the grammar G. Third, the marker is attached to the head of a given word w tested whether it is involved in the language of the grammar G,

$$\#w \ . \tag{5.3}$$

If the given word w is acceptable by the grammar G or not is attributed to whether the rules in the marked grammar $G^{\#}$ can rewrite the marked word $\#w$ to the target word,

$$S\# \ . \tag{5.4}$$

Note that there is no undeterminisity about the position to apply a rule in this grammar $G^{\#}$, since only one marker # can appear in a word.

Each agent have its rules in the list form. A rewriting rule which can rewrite the word are searched from the top of the list. If there is an applicable rule, the marked word is rewritten by the rule. No more applicable rules expect for the rules to move the marker can be found, the marker is moved to right by using one of the rules to move the marker. Once a word have been rewritten, then the rule search process is restart from the top of the list. The rewriting process finishes if the word is written as $S\#$ and we know that the given word w can be accepted by the given grammar G. If there is no applicable rule even if the word does not coincide with the target word $S\#$, we put the written word back one rewriting step, and the search process is restarted from the next rule of the last application. This recursive process is continued until there is no rule which is not tried. We know the given grammar G can not accept the given word w, if the written words have never accord with the target word in spite that all rules have tried. It is when the rewritten word become $w\#$.

We show some examples. For a grammar G with rule set,

$$S \to 0,\ S \to A1,\ A \to 10,\ A \to 1S,\ B \to A,\ S \to 10,$$

the corresponding grammar with marker $G^{\#}$ consists of the following rules:

$$0\# \to S\#,\ A1\# \to S\#,\ 10\# \to A\#,\ 1S\# \to A\#,$$
$$A\# \to B\#,\ 10\# \to S\#,\ \#0 \to 0\#,\ \#1 \to 1\#\ .$$

A word 101 is rewritten as the following process:

$$\#101 \Rightarrow 1\#01 \Rightarrow 10\#1 \Rightarrow 1S\#1 \Rightarrow A\#1 \Rightarrow A1\# \Rightarrow S\#\ .$$

Since it can be rewritten to the target word $S\#$, the word 101 is acceptable by the grammar G.

Next example is a rewriting process with putting back. A word 10 is rewritten as following processes:

$$\#10 \Rightarrow 1\#0 \Rightarrow 10\# \Rightarrow A\# \Rightarrow B\# \Rightarrow \times$$

Since the word $B\#$ cannot be rewritten any more, denoted by $\Rightarrow \times$, we put back the word one step to the word $A\#$. And then we restart the search process from the next of the last applied rule in the list. Namely, we begin to search the fitting rule from $10\# \to S\#$, since the word $A\#$ is rewritten by the rule $A\# \to B\#$.

$$A\# \Rightarrow \times$$

But there is not any rule in the remainders of the list. We put back again and the search process is restarted from the next rule of the rule which has been applied to rewrite the word $10\#$ to $A\#$, i.e. $1S\# \to A\#$, then the rewriting process is restarted as,

$$10\# \Rightarrow S\#.$$

Because we can attain to the target word, it is made sure that the grammar G accepts the word 10.

A failure process is displayed as a last example. The rewriting process of the word 00 proceeds as following:

$$\begin{array}{rll} \#00 \Rightarrow & 0\#0 \Rightarrow & S\#0 \Rightarrow S0\# \Rightarrow \times \\ & & S\#0 \Rightarrow \times \\ & 0\#0 \Rightarrow & 00\# \Rightarrow \times. \end{array}$$

Bibliography

[1] ANDERSEN, E. S. Complexity and language acquisition: influences on the development of morphological systems in children. In *The Evolution of Human Languages*, J. A. Hawkins and M. Gell-mann, Eds., vol. 11 of *Santa Fe Institute studies in the science of complexity.* Addison-Wesley, Redwood City, 1992, pp. 241–271.

[2] AXERLOD, R. *The Evolution of Cooperation.* Basic Books, New York, 1984.

[3] BAAS, N. A. Emergence, hierarchies, and hyperstructures. In *Artificial Life III* (Redwood City, 1994), C. G. Langton, Ed., Addison-Wesley, pp. 515–537.

[4] BARBER, E. J. W., AND PETERS, A. M. W. Ontogeny and phylogeny: What child language and archaeology have to say to each other. In *The Evolution of Human Languages*, J. A. Hawkins and M. Gell-mann, Eds., vol. 11 of *Santa Fe Institute studies in the science of complexity.* Addison-Wesley, Redwood City, 1992, pp. 305–352.

[5] BARRELL, B. G., BANKIER, A. T., AND DROUIN, J. A different genetic code in human mitochondria. *Nature 282* (1979), 189–194.

[6] BAUMANN, U., AND ORO, J. Three stages in the evolution of the genetic code. *BioSystems 29* (1993), 133–141.

[7] BERMAN, R. A. Language development and language knowledge: evidence from the acquisition of hebrew morphophonology. *J. Child Lang. 8* (1981), 609–626.

[8] BERMAN, R. A. Regularity vs. anomaly: the acquisition of hebrew inflectional morphology. *J. Child Lang. 8* (1981), 265–282.

[9] BICKERTON, D. *Roots of Language.* Karoma, Ann Arbor, 1981.

[10] BICKERTON, D. Creole languages and the bioprogram. In *Linguistic Theory: Extensions and Implications*, F. J. Newmeyer, Ed., vol. 2. Cambridge University Press, Cambridge, MA, 1988, pp. 268–284.

[11] BROWN, G. S. *Laws of Form.* George Allen and Unwin, London, 1969.

[12] BROWN, R. *A First Language.* Harvard University Press, Cambridge, MA, 1973.

[13] BROWN, R., AND HANLON, C. Derivational complexity and order of acquisition in child speech. In *Cognition and the Development of Language*, J. R. Hayes, Ed. John Wiley, New York, 1970.

[14] CHAITIN, G. J. Toward a mathematical definition of "life". In *The Maximum Entropy Formalism*, R. D. Levine and M. Tribus, Eds. MIT Press, 1979, pp. 477–498.

[15] CHOMSKY, C. *The Acquisition of Syntax in Children from 5 to 10.* MIT Press, Cambridge, MA, 1969.

[16] CHOMSKY, N. *Logical Structure of Linguistic Theory.* Plenum, New York, 1955.

[17] CHOMSKY, N. *Syntactic Structure.* Mouton, The Hague, 1957.

[18] CLARK, E. V. On the child's acquisition in two semantic fields. *J. Verbal Learning and Verbal Behavior 11* (1972), 750–758.

[19] COMRIE, B. Before complexity. In *The Evolution of Human Languages*, J. A. Hawkins and M. Gell-mann, Eds., vol. 11 of *Santa Fe Institute studies in the science of complexity.* Addison-Wesley, Redwood City, 1992, pp. 193–211.

[20] CONRAD, M. In *Real Brains, Artificial Minds*, J. L. Casti and A. Karlqvist, Eds. Elsevier Science Publishing, Amsterdam, 1987.

[21] CRICK, F. H. C. The origin of the genetic code. *J. Mol. Biol. 38* (1968), 367–379.

[22] CRUTCHFIELD, J. P. The calculi of emergence: computation, dynamics and induction. *Physica 75D* (1994), 11–54.

[23] CRUTCHFIELD, J. P., AND YOUNG, K. Computation at the onset of chaos. In *Complexity, Entropy and the Physics of Information* (Redwood City, 1990), W. H. Zurek, Ed., Addison-Wesley.

[24] EIGEN, M., AND SCHUSTER., P. *Hypercycle.* Springer-Verlag, Berlin, 1979.

[25] EMMECHE, C. Is life as a multiverse phenomenon? In *Artificial Life III* (Redwood City, 1994), C. G. Langton, Ed., Addison-Wesley, pp. 553–568.

[26] EMMECHE, C., AND HOFFMEYER, J. From language to nature: The semiotic metaphor in biology. *Semiotica 84* (1991), 1–42.

[27] ERVIN, S. Imitation and structural change in children's language. In *New Directions in the Study of Language*, E. Lenneberg, Ed. MIT Press, Cambridge, MA, 1964.

[28] FARMER, D. J., KAUFFMAN, S. S., PACKARD, N. H., AND PERELSON, A. S. *Ann. Rev. NY Acad.Sci. 504* (1987), 118.

[29] FOLEY, W. A. Language birth: the processes of pidginization and creolization. In *Language: The Socio-cultural Context*, F. J. Newmeyer, Ed., vol. 4. Cambridge University Press, Cambridge, 1988, pp. 162–183.

[30] FONTANA, W. Algorithmic chemistry. In *Artificial Life II* (Redwood City, 1991), C. G. Langton, C. Taylar, J. D. Farmer, and S. Rasmussen, Eds., Addison-Wesley, pp. 159–207.

[31] FONTANA, W., WAGNER, G., AND BUSS, L. W. Beyond digital naturalism. *Artificial Life 1* (1994), 211–227.

[32] FRASER, C., BELLUGI, U., AND BROWN, R. Control of grammar in imitation, comprehension and production. *J. Verbal Learning and Verbal Behavior 2* (1963), 121–135.

[33] FREGE, G. On sense and nominatum. In *Readings in Philosophical Analysis*, H. Feigl and W. Sellars, Eds. Appeton-Century-Crofts, New York, 1949, pp. 85–102.

[34] GAZZANIGA, M. S. *The Social Brain.* Basic Book, New York, 1985.

[35] GESTELAND, R. F., AND ATKINS, J. F., Eds. *The RNA world: the nature of modern RNA suggests a prebiotic RNA world* (1993), Cold Spring Harbor Laboratory.

[36] GILBERT, W. The RNA world. *Nature 319* (1986), 618.

[37] GÖDEL, K. Uber formal unentscheidbare satze der *principia mathematica* und verwandter system i. *Monatshefte fur Matematik und Physik 38* (1931), 173–198.

[38] Greslin, A. F., Prescott, D. M., Oka, Y., Loukin, S. H., and Chapell, J. C. Reordering of nine exons is necessary to form a functional action gene in oxytricha nova. *Proc. Natl. Acad. Sci. USA 86* (1989), 6264.

[39] Gunji, Y.-P. Global logic resulting from disequilibration process. *BioSystems 35* (1995), 33–62.

[40] Harper, D. G. C. Communication. In *Behavioural Ecology: An Evolutionay Approach*, J. R. Krebs and N. B. Davis, Eds., 3rd ed. Blackwell, Oxford, 1991.

[41] Hashimoto, T., and Ikegami, T. Emergence of net-grammar in communicating agents. *BioSystems* (in press) (1996).

[42] Hofstadter, D. R. *Gödel, Escher, Bach: An Eternal Golden Braid.* Basic Books, New York, 1980.

[43] Hofstadter, D. R. *Metamagical Themas – An Interlocked Collection of Literary, Scientific and Artistic Studies.* Basic Books, New York, 1985.

[44] Hogeweg, P., and Hesper, B. Evolutionary dynamics and the coding structure of sequences: multiple coding as a consequence of crossover and high mutation rates. *Computers Chem 16* (1992), 171.

[45] Holland, J. H. *Adaptation in Natural and Artificial Systems*, cambridge ed. MIT Press, 1992.

[46] Hopcroft, J. E., and Ullman, J. D. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

[47] Hori, H., and Osawa, S. Evolutionary change in 5s rna secondary structure and a phylogenic tree of 54 5s rna species. *Proc. Natl. Acad. Sci. USA 76* (1979), 381–385.

[48] Hori, H., and Osawa, S. Origin and evolution of organisms as deduced from 5s ribosomal rna sequences. *Mol. Biol. Evol. 4* (1987), 445–472.

[49] Iba, Y. Information integration and fundamental problems in artificial intelligence [in japanese]. *Jinkou-Chinou Gakkai-Shi* (1996).

[50] Ikegami, T. From genetic evolution to emergence of game strategies. *Physica 75D* (1994), 310–327.

[51] Ikegami, T., and Hashimoto, T. Replication and diversity in machine-tape coevolutionary systems. submitted to ALife V.

[52] Ikegami, T., and Hashimoto, T. Coevolution of machines and tapes. In *Advances in Artificial Life* (Berlin, 1995), F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, Eds., vol. 929 of *Lecture Notes in Artificial Intelligence*, Springer.

[53] Ikegami, T., and Kaneko, K. Genetic fusion. *Phys. Rev. Lett. 65*, 3352-3353 (1990).

[54] Ikegami, T., and Kaneko, K. Evolution of host-parasitoid network through homeochaotic dynamics. *Chaos 2* (1992), 397–407.

[55] Jerne, N. K. The immune system. *Sci. Amer. 229* (1973), 59.

[56] Kandko, K., and tsuda, I. Constructive complexity and artificial reality: an introduction. *Physica 75D* (1994), 1–10.

[57] Kaneko, K. Chaos as a source of complexity and diversity in evolution. *Artificial Life 1* (1994), 163–177.

[58] Kaneko, K., and Ikegami, T. Homeochaos: dynamics stability of a symbiotic network with population dynamics and evolving mutation rate. *Physica 56D* (1992), 406–429.

[59] Kaneko, K., and Suzuki, J. Evolution toward the edge of chaos in an imitation game. In *Artificial Life III* (Redwood City, 1994), C. Langton, Ed., Addison-Wesley.

[60] Kauffman, S. A. Requirements for evolvability in complex systems. In *Emergent Computation* (Massachusetts, 1987), The MIT Press.

[61] Kauffman, S. A. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, Oxford, 1993.

[62] Kawaguchi, Y., Honda, H., Taniguchi-Morimura, J., and Iwasaki, S. The codon cug is read as serine in an asporogenic yeast *candida cylindracea*. *Nature 341* (1989), 164–166.

[63] KAWATA, M., AND TOQUENAGA, Y. From artificial individuals to global patters. *Trends Ecol. Evol. 9* (1994), 417–421.

[64] KREBS, J. R., AND DAVIS, N. B. *An Introduction to Behavioural Ecology.* Blackwell, Oxford, 1981.

[65] KUHN, H., AND KUHN, C. Evolution of a genetic code simulated with a computer. *The Origin of Life 9* (1978), 135.

[66] LAKOFF, G. *Women, Fire, and Dangerous Things: What Categories Reveal abut the Mind.* The University of Chicago Press, Chicago, 1987.

[67] LAKOFF, G., AND JOHNSON, M. *Metaphors We Live By.* The University of Chicago Press, Chicago, 1980.

[68] LANGTON, C. G. Artificial life. In *Artificial Life* (Redwood City, 1988), C. G. Langton, Ed., Addison-Wesley, pp. 1–47.

[69] LENNEBERG, E. H. *Biological Foundations of Language.* Wiley, New York, 1967.

[70] LIEBERMAN, M. *On the Origins of Language.* Macmillan, New York, 1975.

[71] LINDGREN, K. Evolutionary phenomena in simple dynamics. In *Artificial Life II* (Redwood City, 1991), C. G. Langton, C. Taylar, J. D. Farmer, and S. Rasmussen, Eds., Addison-Wesley, pp. 295–312.

[72] MACLENNAN, B. Synthetic ethology: an approach to the study of communication. In *Artificial Life II* (Redwood City, 1991), C. G. Langton, C. Taylar, J. D. Farmer, and S. Rasmussen, Eds., Addison-Wesley, pp. 631–658.

[73] MAIZELS, N., AND WEINER, A. M. Peptide-specific ribosomes, genomic tags, and the origin of the genetic code. In *Cold Spring Harbor Symposia on Quantitative Biology*, vol. 52. Cold Spring Harbor Laboratory, Cold Spring Harbor, New York, 1987.

[74] MOORE, C. Unpredictability and undecidability in dynamical systems. *Phys. Rev. Let. 64* (1990), 2354–2357.

[75] MOORE, C. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity 4* (1991), 199–230.

[76] Niesert, U., Harnasch, D., and Bresch, C. Origin of life between scylla and charybdis. *J. Mol. Evol. 17* (1981), 348.

[77] Oliphant, M. The dilemma of saussurean communication. *BioSystems* (in press) (1996).

[78] Osawa, S., and Shimura, Y., Eds. *RNA no Sekai* [in Japanese] (Tokyo, 1990), Kodansha.

[79] Pinker, S. *Language Learnability and Language Development.* Harvard University Press, Cambridge, MA, 1984.

[80] Pinker, S. Language acquisition. In *Foundations of Cognitive Science*, M. I. Posner, Ed. MIT Press, Cambridge, MA, 1989.

[81] Pollack, R. *Sings of Life: The Language and Meanings of DNA.* Houghton Mifflin, New York, 1993.

[82] Quine, W. V. O. *From a Lofical Point of View – 9 Logiro-Philosophical Essays.* Harvard University Press, Cambridge, MA, 1980.

[83] Rasmussen, S. Toward a quantitative theory of the origin of life. In *Artificial Life* (Reading, Mass, 1989), C. G. Langton, Ed., Addison-Wesley, pp. 79–104.

[84] Ray, T. S. An approach to the synthesis of life. In *Artificial Life II* (Redwood City, 1991), C. G. Langton, C. Taylar, J. D. Farmer, and S. Rasmussen, Eds., Addison-Wesley, pp. 371–408.

[85] Ray, T. S. Evolution, complexity, entropy and and artificial reality. *Physica 75D* (1994), 239–263.

[86] Révész, G. *Introduction to Formal Languages.* Dover Publications, New York, 1991.

[87] Romaine, S. The evolution of linguistic complexity in pidgin and creole languages. In *The Evolution of Human Languages*, J. A. Hawkins and M. Gell-mann, Eds., vol. 11 of *Santa Fe Institute studies in the science of complexity.* Addison-Wesley, Redwood City, 1992, pp. 213–238.

[88] Rössler, O. E. Endophysics. In *Real Brains, Artificial Minds*, J. L. Casti and A. Karlqvist, Eds. Elsevier Science Publishing, Amsterdam, 1987.

[89] S. OSAWA, T. H. JUKES, K. W., AND MUTO, A. Recent evidence for evolution of the genetic code. *Microbiol. Rev. 56* (1992), 229–264.

[90] SAPIR, E. *Language: An Introduction to the Study of Speech.* Harcourt Brace, New York, 1921.

[91] SHANNON, C. E., AND WEAVER, W. *The Mathematical Theory of Communication.* University of Illinois Press, Urbana, 1949.

[92] SHIBATA, T. *Shakai Gengogaku no Kadai.* Sanseido, Tokyo, 1978.

[93] SLOBIN, D. I. Grammatical transformation in childhood and adulthood. *J. Verbal Learning and Verbal Behavior 5* (1966), 219–227.

[94] SLOBIN, D. I. Cognitive prerequisites for the development of grammar. In *Studies of Child Language Development*, Ferguson and Slobin, Eds. Hold, Rinehart and Winston, New York, 1973.

[95] SMITH, J. M., AND SZATHMÁRY, E. Language and life. In *What is Life? – The Next Fifty Years*, M. P. M. L. O'Neill, Ed. Cambridge University Press, Cambridge, MA, 1995, pp. 67–77.

[96] SUZUKI, J., AND KANEKO, K. Imitation games. *Physica 75D* (1994), 328–342.

[97] SVOZIL, K. Undecidability everywhere? In *Limits to Scientific Knowledge*, J. Casti and J. Traub, Eds. Wiley, New York, 1996.

[98] TAYLOR, C. E. "fleshing out" artificial life ii. In *Artificial Life II* (Redwood City, 1991), C. G. Langton, C. Taylar, J. D. Farmer, and S. Rasmussen, Eds., Addison-Wesley, pp. 25–38.

[99] TODD, L. *Pidgins and Creoles.* Routedge & Kegan Paul, 1974.

[100] TSUDA, I. *Chaos-teki-Nou-kan (Chaotic Scenario of Brain)* [in Japanese]. Saiensu-sha, Tokyo, 1991.

[101] TSUDA, I. Chaotic hemeneutics for understanding the brain. In *Endophysics.* Aerial Press, Santa Cruz, CA, 1993.

[102] von Neumann, J. *Theory of Self-reproduction.* the University of Illinois Press, Urbana, 1968.

[103] Werner, G. M., and Dyer, M. G. Evolution of communication in artificial organisms. In *Artificial Life II* (Redwood City, 1991), C. G. Langton, C. Taylar, J. D. Farmer, and S. Rasmussen, Eds., Addison-Wesley, pp. 659–687.

[104] Whorf, B. L. The relation of habitual thought and behavior to language. In *Language, Thought and Reality: Selected Writings of Benjamin Lee Whorf*, J. B. Carroll, Ed. MIT Press, Cambridge, MA, 1956.

[105] Wills, P. R. Self-organization of genetic coding. *J. Theor. Biol. 162* (1993), 267–287.

[106] Wittgenstein, L. *Philosophische Untersuchungen.* Basil Blackwell, 1953.

[107] Woese, C. R. *The Genetic Code: The Molecular Basis for Genetic Expression.* Harper & Row, New York, 1972.

[108] Wurm, S. A. Pidgins, creoles and lingue franche. In *Current Trends in Linguistics*, vol. 8. Mouton, The Hague, 1971, pp. 999–1021.

[109] Yanagawa, H. *Seimei ha RNA kara Hajimatta* [in Japanese]. Iwanami Shoten, Tokyo, 1994.

[110] Yoshikawa, E. S., and Ikegami, T. Paleto optimality at the edge of chaos. to be submitted, 1996.

論文の内容の要旨

論文題目　“Evolution of Code and Communication in Dynamical Networks”
(動的ネットワークにおけるコードとコミュニケーションの進化)

氏名　橋本 敬

1　イントロダクション

この論文では、記号の使用方やその解釈の体系を「コード」と呼ぶ。自発的に発生するコードは動的に変化しうるが、その運動には、統語レベルと意味レベルの混合 (syntax and semantics mixture) と全体として閉じたコードをつくる事の困難 (openness of code) が影響を与えるであろう。この論文では、言語におけるコードと遺伝コードのついてモデル化し、その創発、進化を論じる。

2　形式文法システムの進化

言語のコードを事前に書き下す事はできないが、事後的に言葉の使用について記述する事は可能である。しかし、この記述は常に変化しうる。言語コードはコミュニケーション自体による変化をまぬがれ得ない。

言語コードの進化を研究するために、会話を行う個体のネットワークを考える。各個体 (エージェント) は形式文法を持つシステムとして定義される。エージェントがそれぞれの文法に基づいて語を発し、全エージェントが発話された語の理解を試みる (会話ネットワーク)。このネットワークで、語の発話・受理に応じた得点が与えられるある種のゲームを行う。文法の変異過程により一部のエージェントが得点に応じて新しいエージェントに置き換えられるという、進化ダイナミクスを導入する。

あるエージェントの計算能力は、受理可能な語の集合の大きさで測られる。また、あるエージェントの処理する情報量は発した語の長さと理解した語の長さの積と定義し、ネットワーク内を流れる情報量は、その平均で測る。語の多様性は発話された語の種類とする。

得点は、より多くの語を発話・理解できると上がり、他のエージェントに理解されると少し下がるという設定にする。初期状態としてランダムに作った計算能力の低い文法(チョムスキー階層では正規文法に属する)を持つエージェントを10個体つくる。計算能力、ネットワーク内を流れる情報量、語の多様性は時間的に増大する傾向にあるが、あまり変化しない部分と急激な発展が交互に現れるという断続平衡的な様相を呈する。

平衡するところでは、共語集団(ensemble with common set of words, 以下ECW)が形成されている。ECWとは同じ語を発話・理解しあうエージェントの集団である。すなわち、ある特定の語の使用が選ばれており、コードが自律的に形成されたと見ることができる。異なる2つのECWが存在する時、高い能力のエージェントにより構成されるECWが淘汰される場合がある。より多く話される共語を発話・理解できないエージェントは、たとえ高い計算能力を持っていても淘汰される。ECWの共存(mixture of code)により高い計算能力への進化は抑制されるのである。

ほかのエージェントより多くの語を発話・理解するエージェントが有利な場合には、ECWがつぎつぎに現れるというコードの進化が起きる。理解しあう事が非常に有利である場合は、一つのコードができるが、コードの進化は起きず、語の多様性とネットワーク内の情報は低いレベルに留まる。逆の状況では、語の多様性は上がるが、情報の流れは低い。これはコードが成立せず、相互理解なしに多くの語が発せられている状態である。

計算能力の急激な発展は2種類の機構によりもたらされる。一つはモジュール型進化である。これは、一つの新しいルールをモジュール的に使う事で、多くの新しい文を発話・理解できるようになる発展過程である。自然言語との関連で考えるならば、接辞による語形成に対応するであろう。もう一方はループ構造の出現である。文法内にループを持つことで再帰的な書き換えが可能になり、原理的には無限個の語を発することができる。これは文法の構造的な変化であり、アルゴリズム的進化と呼ぶ。これは、複文を作る事により一つの文で長く豊富な記述ができることに対応する。ループを含む文法はチョムスキー階層において文脈自由のクラスに属し、初期の正規文法より一つ上のクラスである。すなわち、チョムスキー階層をのぼる形の進化が起きている。

ECWによる進化の抑制と、これらの文法の進化が交互に起きることで、断続平衡的な進化が見られるのである。

3 テープとマシンの共進化

遺伝情報系におけるコドンとアミノ酸の対応がわかっても、ある遺伝子型がどのように表現型に発現するかは完全にはわからない。このレベルでは、情報の解釈側の役割が重要である。また、遺伝コードはシステムが自己複製可能なように組織化されなければならないが、自己複製には自己言及のパラドックスがある。つまり、自己複製のために自己を観測せねばならず、観測に対して安定でないものの複製は困難となる。ノイマンはマシンとその記述により自己複製オートマトンを構成した。

自己複製を可能にする遺伝コードがどのように進化するかを見るために、ここでは、テープとそれを読むマシンのネットワークを考える。テープは円形のビット列で、マシンの記述となっている。マシンはテープを解釈し、新しいテープとマシンをつくる。マシンによってテープが書き換えられる事を「能動的ミューテーション」という。また外部ノイズにより確率的にテープ上の記号が変化することは「受動的ミューテーション」と呼ばれる。

外部ノイズの上昇により、最小自己複製ネットワークから大きな複製ネットワークへの進化が見られる。外部ノイズが低い場合は、能動的ミューテーションなしで自己複製する、一対のテープとマシンによる最小のネットワークが安定に存在する。受動的ミューテーションによりパラサイト的ネットワークができ大きなネットワークへと進化していく。外部ノイズがある程度大きくなると、大きく複雑なネットワークへ

の転移が起きる。この転移後は外部ノイズなしでもネットワークは安定 にその構造を維持する。 この自己維持ネットワークを「コアネット」と呼ぶ。これは個々のテープ・マシンの自己複製から、大きなネットワークが全体として複製する状態への進化である。このネットワークでは、 能動的ミューテーション率 (μ_A) が高い値に保たれる。μ_Aの値はマシンがテープを書き換える率である。これを計算を行っている程度と見ると、おおきなネットワークの維持のために計算が行われているという見方が可能である。 コアネットにおけるμ_Aは固定点だけではなく、時間的に振動するものもある。

コアネットには幾つかの自己触媒的なネットワークが埋め込まれている。自己触媒ネットワークには、テープが個々に複製されているものと、そうでないものという2つの種類がある。前者は、アイゲンらのハイパーサイクルと同じものであり、$\mu_A = 0$ である。 後者はテープとマシンの両方が相互につくりあうループをなしており、$\mu_A > 0$ である。 このタイプでは、非等方的な結合をもつ大きな自己触媒ネットワークが安定に存在できる。

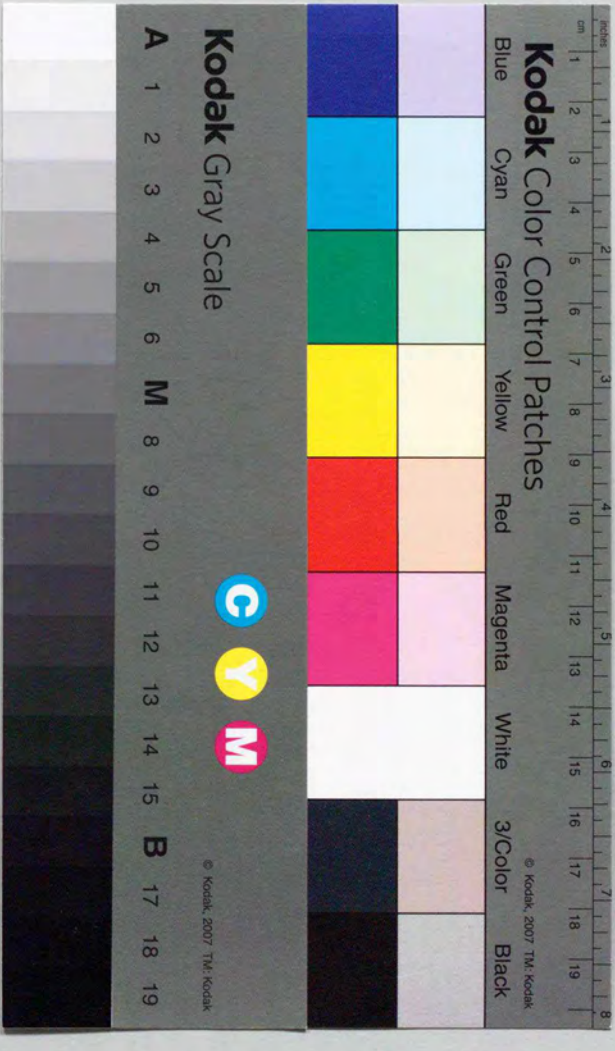
振動するコアネットでは、μ_Aやマシン、テープの数が振動するだけではなく、ネットワークのトポロジー自体が時間的に変動する。ある時は少数の小さな自己触媒ループでネットワークが構成されるが、その後大きなループが多数存在するようになり、また少数の小さなループに戻るという時間的な変動を準周期的に繰り返す。

マシンはテープを解釈し新たなマシンをつくる。この解釈の仕方の体系がこのシステムでの遺伝コードに対応するものである。固定点のコアネットではコードが組織化され安定に存在している。そのコードには多義性と同義性がある。振動コアの場合はコード自体が時間的に変動するのだが、これは「閉じたコード」をつくれていないと見れる (openness of code)。すなわちあるコードに基づいた解釈をする事自体がネットワークを不安定化させている。

4 結論

言語的コードの進化の研究において、ある語を共通に使う集団が現れた。そこでは、語の使い方が選ばれるというコードが共有されている。また、チョムスキー階層をのぼるかたちの進化が起きるが、二つのコードの対立で高い計算能力への進化が抑制される。高い計算能力への進化とコードの成立によるその抑制によって、ネットワーク内の情報量は断続平衡的な発展をする。

遺伝的コードの進化の研究においては、多くのテープとマシンが相互につくりあう事により、大きなネットワークが全体として自己複製する「コアネット」が進化した。そこでは、マシンによるテープの解釈という、遺伝コードに対応するものが組織化されている。また。ネットワークのトポロジーが時間的に変動する振動コアネットができるが、これは閉じたコードがつくれない事によって生み出される振動とみることができる。


Kodak Color Control Patches
Blue
Cyan
Green
Yellow
Red
Magenta
White
3/Color
Black
© Kodak, 2007 TM: Kodak
Kodak Gray Scale
A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19
C Y M
© Kodak, 2007 TM: Kodak