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 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.

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.

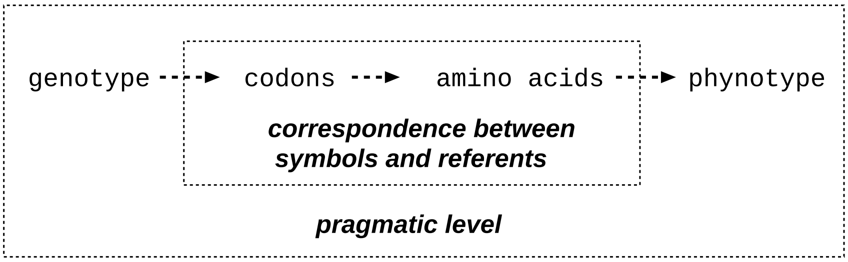

Figure 1.1: Relations between each codon and amino acid is correspondence between each symbol and its referent. It must be considered through relations between genotype and phenotype.

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.