Emergence of Net-grammar in Communicating Agents

Takashi Hashimoto[*]
and
Takashi Ikegami[†]
Institute of Physics, College of Arts and Sciences,
The University of Tokyo,
3-8-1, Komaba, Meguro-ku, Tokyo 153, Japan

Abstract

Evolution of symbolic language and grammar is studied in a network model. Language is expressed by words, i.e. strings of symbols, which are generated by agents with their own symbolic grammar system. Agents communicate with each other by deriving and accepting words in terms of their own grammar. They are ranked according to their communicative effectiveness: an agent which can derive less frequent and less acceptable words and accept words in less computational time will have higher scores. They can evolve by mutational processes, which change rewriting rules in their symbolic grammars. Complexity and diversity of words increase in the course of time. The emergence of modules and loop structure enhances the evolution. On the other hand, ensemble structure lead to a net-grammar, restricting individual grammars and their evolution.

Key words: Net-grammar; Algorithmic evolution; Module-type evolution; Evolution of language; Symbolic grammar systems

1 Introduction

Linguistic expressions look quite complex but are far from random. It is commonly assumed that one has to have internal knowledge of one's language when one can derive and recognize appropriately structured expressions. G. Lakoff and M. Johnson (1980) made several important remarks on how we understand our experiences. We form less clearly delineated concepts in terms of more clearly delineated ones through metaphorical processes. Our conceptual system is metaphorically structured and defined. Metaphors have a basis in our physical and cultural experience, and the basis may vary from culture to culture.

The grammar of a given language is also determined in part by the community which uses it. 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, 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 a 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 (1991) 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 the objects an agent encounters in its environment. Different names may correspond to the same object, resulting in the emergence of synonyms. Complex naming of objects reflects the complexity

[*]e-mail: `toshiwo@sacral.c.u-tokyo.ac.jp`
[†]e-mail: `ikeg@sacral.c.u-tokyo.ac.jp`

of the external environment. In Werner and Dyer's model (1991), the 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 mechanisms. In general, evolving systems, such as evolutionary games (Lindgren, 1991; Ikegami, 1994), Tierra world (Ray, 1991, 1994), network of Turing machines (Ikegami and Hashimoto, 1995) 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 (external language), but also to construct one's "internal models" (internal language). We regard grammar systems as representations of internal models. External 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. Conversation therefore can be thought of as a source of rich grammar structures.

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

According to N. Chomsky (1955), the computational ability of symbolic grammar is categorized into four different classes (Révész, 1991):

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 set of words it can generate. To take a simple example, a word set,

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

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

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 deriving steps into account. If we deal only with a finite set of words, e.g. $\{0^n 1^n | N \geq n \geq 1\}$, this 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. That is, the computational ability of a symbolic grammar and hierarchy should be studied within an ensemble. Chomsky (1957) 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.

The present paper is organized as follows. Formal language and the Chomsky hierarchy are introduced in §2.1. A concrete model of a language game is defined in §2.2 and §2.3. Detailed analysis of evolution

[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 those grammar systems is presented in §3.1. 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 achieving an ensemble structure, each grammar system is likely to become complex again. An ECW is discussed in §3.2. A simulation of the model exhibiting a stepwise evolution over several eras will be reported in §3.4. Some implications of ECW will be discussed in §5 and main conclusions are given in §6.

2 Modeling

2.1 Basic of Formal Languages

2.1.1 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 (Hopcroft and Ullman, 1979).

2.1.2 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 $\alpha \rightarrow \beta$.

2.1.3 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 the same 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 $X \Rightarrow Y$, 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 $X \overset{*}{\Rightarrow} Y$, 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.$X = X_0 \Rightarrow X_1 \Rightarrow \cdots X_{k-1} \Rightarrow X_k = Y$.

When no more 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.

2.1.4 The Chomsky Hierarchy of Languages

We can classify the generative grammars with respect to a set of rewriting rules. The classification below has been introduced by N. Chomsky (1955).

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 (Révész, 1991):

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

$i = 1$ Every rewriting rule in F has form $Q_1 A Q_2 \rightarrow Q_1 P Q_2$, 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 letters. This grammar types are called *context sensitive grammar.*

$i = 2$ Every rule in F has form $A \to P$, 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 form either $A \to PB$ or $A \to P$, 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:

$$\text{regular} \subset \text{context free} \subset \text{context sensitive} \subset \text{phrase structure}. \tag{2}$$

2.2 Agent with Generative grammar

2.2.1 Agent

We define a communicative agent with generative grammar as follows:

$$G_i = (\{S, A, B\}, \{0, 1\}, F_i, S)\ . \tag{3}$$

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 is not included in α. It forbids a self loop in a rule. In this paper, we use 0s and 1s for alphabets so that words become bit strings.

2.2.2 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 finish 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 $L_{\mathrm{sp}}(G_i)$.

For example, an agent which rules are $S \to A, A \to B$ and $A \to 01$ can derive only $\{01\}$. This agent has only one rule $S \to A$ to rewrite symbol S, then the word become A. At the next stage, there are two rules, $A \to B$ 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 to speak words.

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. 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

$$01 \Rightarrow A \Rightarrow S\ . \tag{4}$$

It cannot recognize any other word since the agent does not have applicable rules as nonterminal symbol $\to$ terminal symbols except for $A \to 01$.

2.3 Language Game and Evolutionary Dynamics

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.

2.3.1 Score

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,

$$p_l^{\mathrm{sp}}(c) = \begin{cases} \frac{|w_{lm}(c)|}{trend+1} \ , & \text{for speaking a word } w_{lm}(c) \\ -\frac{M}{2} \ , & \text{for failing to speak any word } , \end{cases} \tag{5}$$

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 he speaks longer words and/or less frequent words.

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

$$p_{kl}^{\mathrm{rec}}(c) = \begin{cases} \frac{|w_{kl}(c)|}{s} \ , & \text{for recognizing a word spoken by } k\text{-th agent} \\ & \qquad \text{in } s \text{ rewriting steps} \\ -|w_{kl}(c)| \ , & \text{for not recognizing a word spoken by } k\text{-th agent } . \end{cases} \tag{6}$$

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

For computing being recognized score, it is given by,

$$p_{lm}^{\mathrm{br}}(c) = \begin{cases} \frac{|w_{lm}(c)|}{P} \ , & \text{if the spoken word is recognized by } m\text{-th agent} \\ -\frac{|w_{lm}(c)|}{P} \ , & \text{if the spoken word isn't recognized by } m\text{-th agent } . \end{cases} \tag{7}$$

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

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

$$p_l^{\mathrm{tot}} = \frac{1}{R}\sum_{c=1}^{R}(r_{\mathrm{sp}}p_l^{\mathrm{sp}}(c) + r_{\mathrm{rec}}\sum_{k=1}^{P}p_{kl}^{\mathrm{rec}}(c) + r_{\mathrm{br}}\sum_{m=1}^{P}p_{lm}^{\mathrm{br}}(c)) \ , \tag{8}$$

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.

2.3.2 Mutations.

In each time step, new agents are produced. They inherit a rule set from their ancestor's with a bit of 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 rules 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 least scored agents as new agents is removed from a network so that the total number of agents is kept constant.

3 Results of Simulation

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.

3.1 Algorithmic Evolution

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.

$$\text{computational ability} = \frac{N(L_{\text{rec}}(G_i))}{N_{\text{all}}} \ , \tag{9}$$

where $N(L_{\text{rec}}(G_i))$ is the number of words which can be recognized by the agent G_i. Fig. 1 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.

3.1.1 Evolution during the Early Stage

It is shown in Fig. 1 that computational ability of agents slowly evolves during initial 200 time steps. In Fig. 2 (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(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(b)). Further, the multi branch structure evolves (Fig. 2(c)).

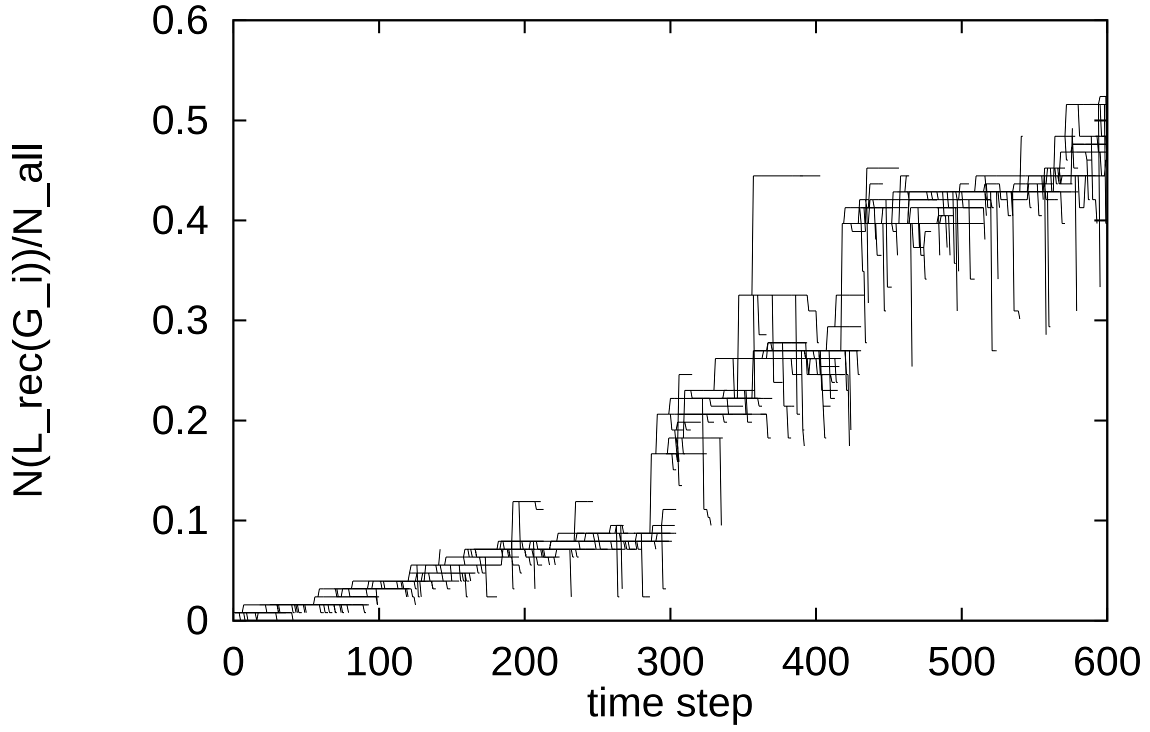


Figure 1: Time step v.s. $N(L_{\rm rec}(G_i))/N_{\rm all}$. Each line connects one agent to oneself or its offsprings. It branches off by the mutations. A line terminates when the corresponding agent is removed. These lines show upward trend. In initial 200 time step, computational ability gradually increases. After that, transitions to higher computational agent are frequently observed.

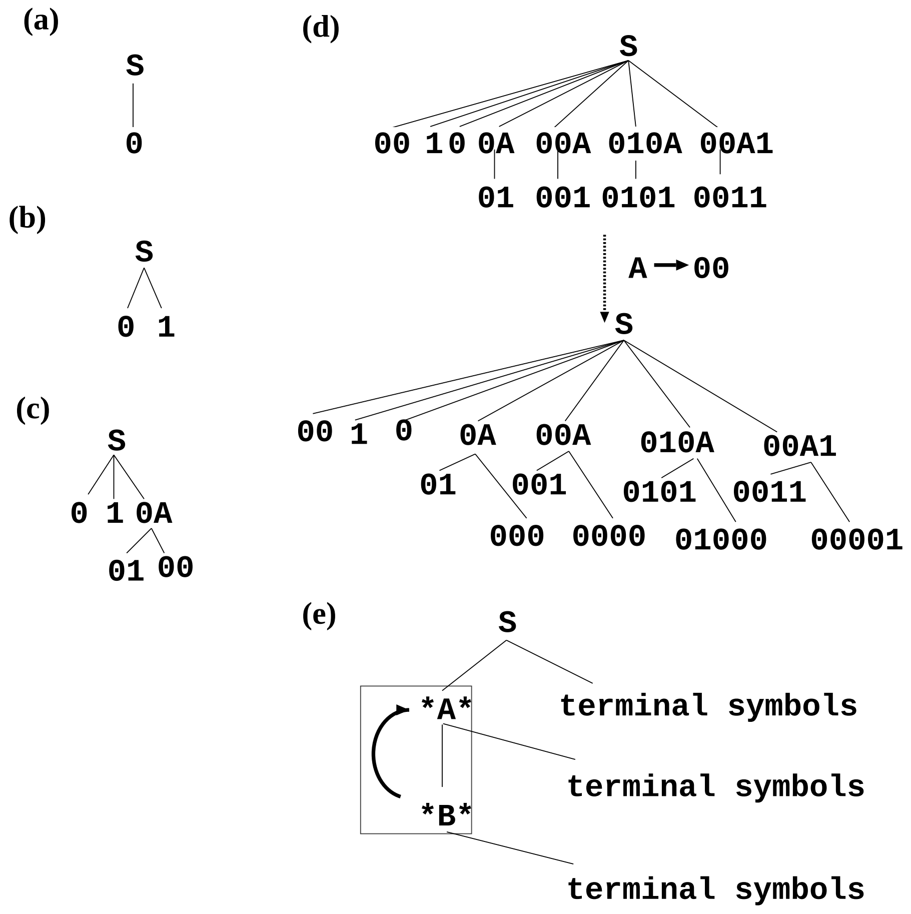


Figure 2: The examples of grammar structure are shown by graph diagrams: (a) a sequential structure, (b) a branch structure and (c) a multi branch structure. In the (d) an example of module-type evolution is shown. Acquiring a rule $A \rightarrow 00$, a grammar without bifurcation (upper tree) is evolved 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.

3.1.2 Module-Type Evolution

We find in Fig. 1 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(d). An acquired rule $A \rightarrow 00$ can double the size of a set of words for the agent to accept. 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 a module rule are called module-type evolutions.

3.1.3 Emergence of Loop Structure

Grammar systems can evolve by an alternative evolutionary process, that is, loop structures are formed in a grammar system. Fig. 1 shows 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(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.

3.2 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 shared set of words. The other agents which can't speak or recognize the common set of words are less benefited than those in 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 step 192 in Fig. 1 and around time step 310 in Fig. 1, 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. 1). Agents taking too much rewriting steps to recognize very frequent words are likely to decrease the fitness. We indicate this fact by Table 1. The rewriting steps to recognize the 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. 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 long words.

Numbers in bold font in Table. 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. 3 shows the phylogeny of agents 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. Two ECWs conflict to survive in 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.

After removing agents G_{306} and G_{276} from a network, 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 emerge through algorithmic evolution.

Table 1: This table shows rewriting steps to recognize some words (the left most column) spoken at time step 400. Simulation parameters are $r_{\mathrm{sp}} = 3.0, r_{\mathrm{rec}} = 1.0 and r_{\mathrm{br}} = -2.0$. 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 put. Bold numerals represent the first two longest steps among agents.

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**						

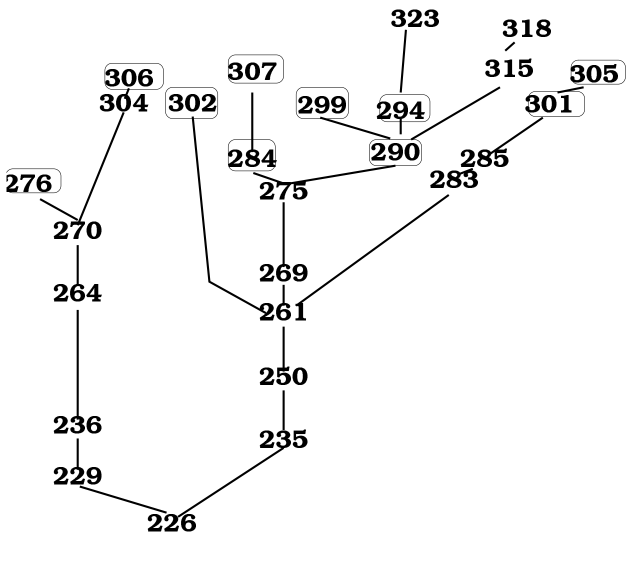


Figure 3: This picture represents the phylogeny of agents at time step 400 (in oval boxes) from common ancestor (G_{226}). A number represents ID of each agent. A line is drawn from parent agent (lower) to its offsprings (upper). Two genetic series are bifurcated from the common root (G_{226}), the agents 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.

3.3 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:

$$S \rightarrow A, A \rightarrow SS, S \rightarrow 0, S \rightarrow 1 \ . \tag{10}$$

This grammar is categorized as a type 2 grammar. It recognizes all the words very quickly and speaks all the words. However, it shows 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. Hence in the case of $r_{\mathrm{rec}} = 1.0, r_{\mathrm{sp}} = r_{\mathrm{br}} = 0.0$, an agent which grammar contains the rules same as rules in (10) has evolved with these parameters, for low variety of speaking words has no effect on its fitness.

3.4 Punctuated Equilibrium

We have seen that our system shows rapid algorithmic evolution of the grammar systems in certain stages. On the other hand, algorithmic evolution is suppressed by forming ECW. Rapid algorithmic evolution follows quasi-equilibrium stages. Temporal evolution of amounts of handling information therefore shows punctuated equilibrium phenomena (Fig. 4). Handling information defined below is sensitive to the formation of ECW.

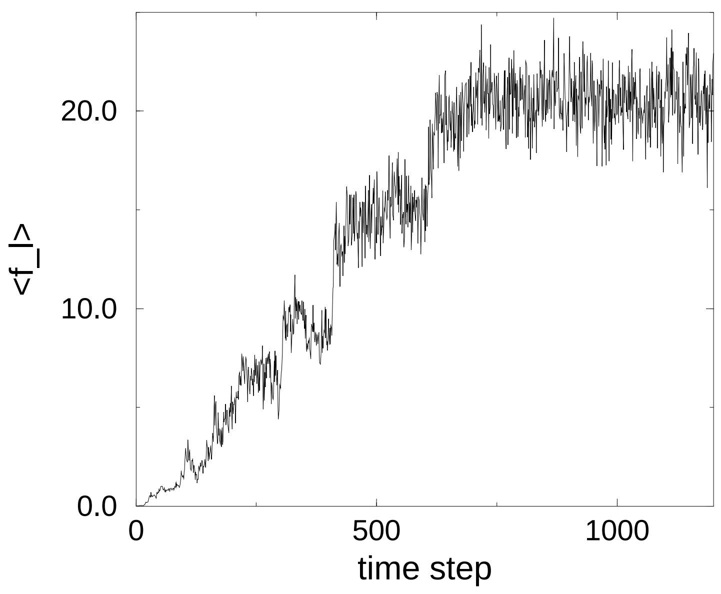


Figure 4: time step v.s. the average handling information (see the definition in the text): In the first 700 time steps, evolution can be observed. The stepwise changes reflect alternate evolution of ECW and the algorithmic evolution.

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

$$f_l = \frac{1}{RP^2} \sum_{c=1}^{R} \sum_{k=1}^{P} |w_{kl}^{\mathrm{rec}}(c-1)| \sum_{m=1}^{P} |w_{lm}^{\mathrm{rec}}(c)| \tag{11}$$

$$|w_{ij}^{\mathrm{rec}}(x)| = \begin{cases} 1 & \text{if } x = 0 \\ \text{the length of } w_{ij}(x) & \text{if } x > 0 \text{ and the word which spoken by the} \\ & \quad i\text{-th agent is recognized by the } j\text{-th agent} \\ 0 & \text{otherwise .} \end{cases} \tag{12}$$

Information contents of a word is simply given by the length of the 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, that 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, $\langle f_l \rangle = \sum_{p=0}^{P} f_l/P$, 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.

4 Score of Being Recognized

We have three parameters in the definition of total score (8). Here, we will see the effect of the parameter r_{br}. If it is given positive value, 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 2.0, the average handling information will go down (Fig. 5 b)) since it reflects the degree of being recognized. Variety, on the other hand, is kept high in two range: $(2.0 \geq r_{\mathrm{br}} \geq -2.0)$ and middle $(r_{\mathrm{br}} < -3.0)$ level (Fig. 5 a)). In such range agents make higher score by speaking less frequent and less recognizable words. It maintains variety high or middle level even when r_{br} is less than -3.0. In earlier stages (time step < 1500), simulation with $r_{\mathrm{br}} = -2.0$ has shown quick evolution. However it is saturated at $r_{\mathrm{br}} = 0.0$ and below (time step ≥ 1500). We conclude that a slight negative value of r_{br} accelerates the evolution at the highest speed.

5 Some Implications of ECW

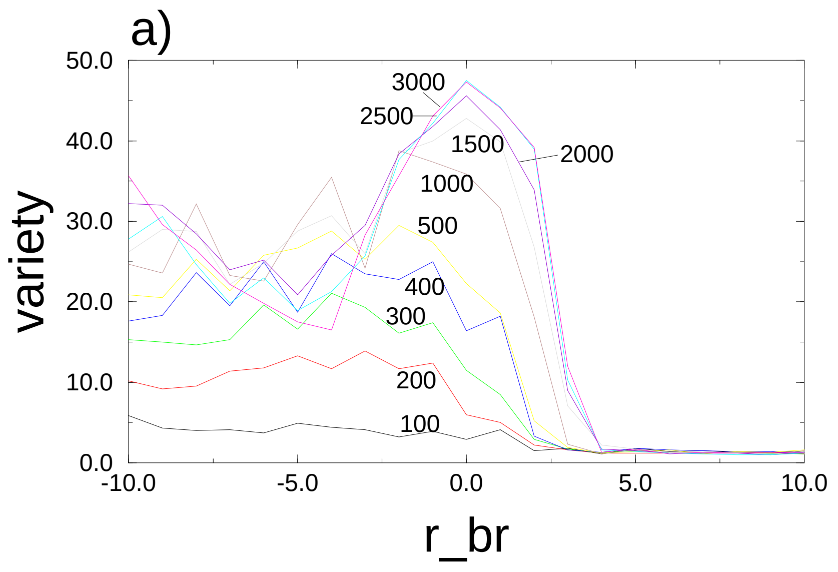
5.1 Connection with Natural Language

We consider two significant evolutionary processes in natural language.

In 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. That is, they 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 which has to deal with situations in finite time steps. If natural language were only required to serves as a mere signaling tool, complex syntactic structures might not have evolved. In order for nested sentences to be produced a complex grammar is likely to appear.

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. 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. Double articulation 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.



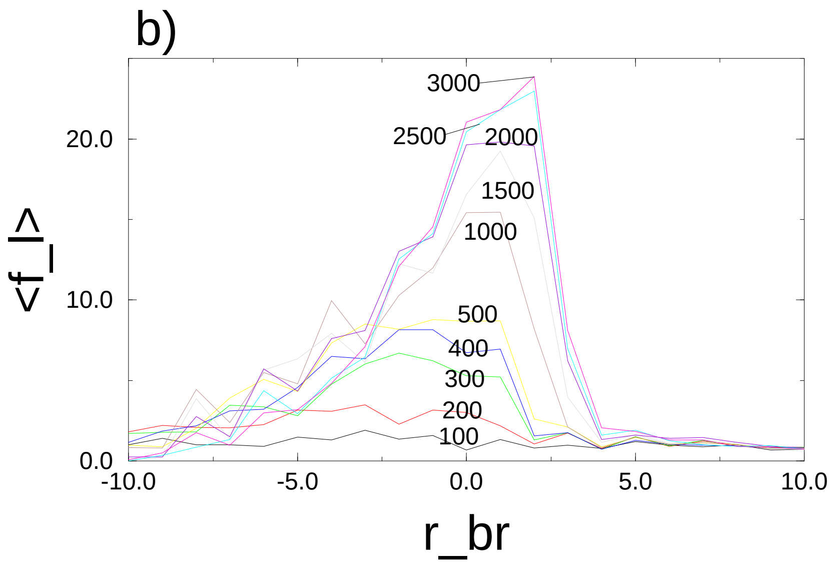


Figure 5: a)variety and b)the average handling information vs r_{br}, coefficient of score of being recognized, at several time steps. The other parameter values are $r_{sp} = 3.0$ and $r_{\mathrm{rec}} = 1.0$. The numbers near each line represent time step. a) In lower range of the parameter ($r_{\mathrm{br}} \leq -3.0$) variety shows middle level, in midway of the parameter ($-3.0 \leq r_{\mathrm{br}} \leq 2.0$) it has high amount and in the higher range it has quite low level. b) For each range $< f_l >$ shows low, high and quite low level, respectively.

5.2 Net-grammar – an Emergent Upper Structure

In biological systems ranging from ants' society to human society, social laws and norms develop. Essential problem is how such ordered upper structures evolve from local interactions (Kauffman, 1987; Taylar 1991). It is suggested that homeochaos or the edge of chaos is formed to establish cooperation (Kaneko and Ikegami 1992; Yoshikawa and Ikegami, 1995). In the iterated prisoner's dilemma game, The Tit For Tat strategy is formed in cooperative societies (Maynard Smith, 1982). Once such a social structure appears, it in turn restricts local dynamics. A recursive loop between global structures and local structures continues endlessly [2] . This is stressed as evidence of "emergence", which is one of the central concerns in artificial life studies (Taylar, 1991; Kawata and Toquenaga, 1994).

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 new rules emerge from children's play. Children's play often looks like a language game; trendy words continuously come and go. In general linguistic communities, social dialects are frequently observed (Shibata, 1978). Social dialects are language variations used 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 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 (Shibata, 1978).

Within our language game, an ECW produces dialects, irrespective of individual grammars. We will use the term *net-grammar* to refer to the way an ECW produces dialects and restricts the potential computational ability of individual grammars.

Our language game favors agents that: (i) speak long and infrequent words, (ii) recognize long words quickly and (iii) speak words which are not recognized by other agents. Therefore players try to encipher their 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, which we call a net-grammar. 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 encryption and establishing language result in punctuated equilibria.

5.3 Communication at the Edge of Chaos

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 of the kind proposed by Suzuki and Kaneko (Suzuki and Kaneko, 1994; Kaneko and Suzuki, 1994). It is a simple model for mutually mimicking birds. In their papers a bird with a complex song is stronger when it comes to defending its territory, since complex songs are difficult for other birds to imitate. But at the same time, a bird imitates other birds' song by the same mechanism. A song is a time series generated by a logistic map. Evolution of a bird's song is achieved by mutating the parameter of logistic map. They observe evolution of a song towards the edge between the periodic window and chaos, that is, birds singing songs of marginal stability seem stronger.

Crutchfield shows that complexity of automata which accept time serieses of dynamical mappings is the highest at the onset of chaos (Crutchfield and Karl, 1990; Crutchfield, 1994). In his model, input to automata is 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.

Their works treat communication and language as encryption processes. However, processes of constituting smooth communication should be treated as something runs counter to encryption. We establish language because we want to communicate. Encryption comes afterwards. On the other hand, the best mutual cooperation is attained at the edge of chaos (Yoshikawa and Ikegami, 1995). We do not know whether the edge of chaos still work ins mixed situation as our language game.

[2] Taylor described such a recursive relationship as "LOCAL to GLOBAL back to LOCAL, inter-level feedback loops" (Taylar, 1991).

5.4 Punctuated Equilibrium

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 is produced by genetic fusion operators, which combine genes with module genes (Ikegami and Kaneko, 1990). (Note that the module rules which we define in the present paper are different from the module species used by Ikegami and Kaneko (1990). The module species are frequently utilized as fusion partners by *many other* species but the module rules 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 (Holland 1992). Consequently, intermittent patterns will appear if crossover is introduced as genetic operator. There are no module agent or parallel evolutions, 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 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.

6 Conclusion

We have studied the evolution of symbolic grammars, by introducing a network model of communicating agents. Each agent has its own grammar system, being expressed as a set of rewriting rules. Via a rule set, each agent speaks words to the other agents and tries to recognize words spoken by the other agents. When mutational dynamics of rules is introduced, agents' grammars change in the course of time. Generally, an agent can speak and recognize more words if it has more rules. Hence grammars with more rules are more hereditable. However the number of recognizable words is not a simple function of the number of rules.

In the present paper, 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 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. That is, 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 have found that no other process than a loop formation brings about an algorithmic evolution.

It is generally believed that a grammar system higher in the hierarchy will perform better. Hence it may be argued that agents belonging to the highest rank in the hierarchy will come out sooner or later. This argument will be valid if grammars without ensemble structures can evolve. But this is not necessarily so when agents form communication network. Synergetic behavior of agents generates a macrostructure named ECW (ensemble with a common set words). Within this ensemble, the common words which characterize the ECW should be spoken and recognized in order for agent to stay in the ECW. This becomes a restrictive condition for individual grammars. Any agent to survive in the ensemble has to evolve their grammars within this restriction.

An ECW disturbs the smooth evolution to the higher ability grammar systems. Therefore an almighty agent, i.e. a grammar system which can speak and recognize all possible words, is difficult to evolve. The emergent restriction on each grammar can be called a net-grammar. If we define a meta-grammar that places restrictions on a grammar, this ensemble structure can be a source of such a meta-grammar.

We assumed that there are grammars and language games beforehand. But grammars may not actually exist in our brain. Nor do we know why we play a language game. Although we started with such an assumption, we have succeeded in showing the interesting structure and dynamics conducive to understanding an actual language system. Next, we will study the emergence of language games.

7 Acknowledgments

The authors would like to thank Y. Nishimura for critical reading of the manuscript. They thank Eken S. Yoshikawa for stimulating discussions. One of the authors (T.H.) wishes to express his gratitude to T. Yamamoto and N. Matsuo for helpful comments.

8 References

Chomsky, Noam, 1955, Logical Structure of Linguistic Theory (Plenum).
Chomsky, Noam, 1957, Syntactic Structure (Mouton).
Crutchfield, James. P., 1994, The calculi of emergence: computation, dynamics and induction. Physica D 75, 11 – 54.
Crutchfield, James. P. and Young, Karl., 1990, Computation at the Onset of Chaos. in: Complexity, Entropy and the Physics of Information, Zurek, Wojciech H. (eds.) (Addison-Wesley, Redwood City) pp. 223 – 269.
Hopcroft, John E. and Ullman, Jeffrey D., 1979, Introduction to Automata Theory, Languages and Computation, (Addison-Wesley, Redwood City).
Holland, J. H., 1976, Adaptation in Natural and Artificial Systems (University of Michigan Press, Ann Arbor).
Ikegami, Takashi, 1994, From genetic evolution to emergence of game strategies. Physica D, 75, 310 – 327.
Ikegami, Takashi and Hashimoto, Takashi, 1995 Coevolution of Machines and Tapes. in: The Proceedings of European Conference on Artificial Life 1995.
Ikegami, Takashi and Kaneko, Kunihiko, 1990, Genetic Fusion. Phys. Rev. Lett. 65, 3352–3353.
Kaneko, Kunihiko and Ikegami, Takashi, 1992, Homeochaos: dynamic stability of a symbiotic network with population dynamics and evolving mutation rate. Physica D, 56, 406 – 429.
Kaneko, Kunihiko and Suzuki, Junji, 1994, Evolution toward the edge of chaos in an imitation game. in: Artificial Life III, C. Langton et. al (eds.), (Addison-Wesley, Redwood City) pp. 43–54.
Kauffman, S. A., 1987, Requirements for evolvability in complex systems: orderly dynamics and frozen components. in: Emergent Computation, Stephanie Forrest (eds.) (The MIT Press, Massachusetts) pp. 135-152.
Kawata, Masakado and Toquenaga, Yukihiko, 1994, From artificial individuals to global patterns. Trends in Ecology and Evolution 9, 417-421.
Lakoff, George and Johnson, Mark, 1980, Metaphors We Live By, (The University of Chicago Press, Chicago).
Lindgren, Kristian, 1991, Evolutionary Phenomena in Simple Dynamics. in: Artificial Life II, C. G. Langton, C. Taylar, J. D. Farmer and S. Rasmussen (eds.), (Addison-Wesley, Redwood City) pp. 295-312.
MacLennan, Bruce, 1991, Synthetic Ethology: An Approach to the Study of Communication. in: Artificial Life II, C. G. Langton, C. Taylar, J. D. Farmer and S. Rasmussen (eds.), (Addison-Wesley, Redwood City) pp. 631–658.
Maynard Smith, John 1982, Evolution and Theory of Games (Cambridge University press, Cambridge).
Ray, Thomas S., 1991, An Approach to the Synthesis of Life. in: Artificial Life II, C. G. Langton, C. Taylar, J. D. Farmer and S. Rasmussen (eds.), (Addison-Wesley, Redwood City) pp. 371–408.
Ray, Thomas S., 1994, Evolution, complexity, entropy and and artificial reality. Physica D, 75, 239–263.
Révész, György E., 1991, Introduction to Formal Languages (Dover Publications, New York).
Shibata, Takeshi, 1978, Shakai Gengogaku no Kadai (Sanseido, Tokyo).
Suzuki, Junji and Kaneko, Kunihiko, 1994, Imitation Games. Physica D, 75, 328 – 342.
Taylor, Charles E., 1991, "Fleshing Out" Artificial Life II. in: Artificial Life II, C. G. Langton, C. Taylar, J. D. Farmer and S. Rasmussen (eds.) (Addison-Wesley, Redwood City) pp. 25-38.
Werner, Gregory M. and Dyer, Michael G., 1991, Evolution of Communication in Artificial Organisms. in: Artificial Life II, C. G. Langton, C. Taylar, J. D. Farmer and S. Rasmussen (eds.) (Addison-Wesley, Redwood City) pp. 659-687.
Yoshikawa, Eken S. and Ikegami, Takashi, 1995, Pareto optimality at the edge of chaos. (preprint)