Title	意味の曖昧さを考慮した音楽・言語構造の繰り返し学習
Author(s)	須藤，洸基
Citation	
Issue Date	2019-03
Type	Thesis or Dissertation
Text version	ETD
URL	http://hdl.handle.net/10119/15787
Rights	
Description	Supervisor:東条　敏，先端科学技術研究科，博士

Doctoral Dissertation

Iterated Learning of the Music/Language Structure with Ambiguous Meaning

Hiroki Sudo

Supervisor: Satoshi Tojo

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
[Information Science]

March, 2019

Abstract

Among the research area of linguistic evolution, cultural evolution is especially important for us to form languages. Even if the human is said to have obtained the language ability biologically, we cannot use any language by birth. There are several study about cultural evolution to research how language is constructed through generations. Among which, we focus on the generation of computer-tractable language. In this thesis, we investigate the self-organization of meaning through ambiguous communication because we cannot understand the speaker's intention every time completely. We can guess the intention of utterance together with body actions and a situation when they utter. On the other hand, what the listener has guessed may not be the exactly same one as the original intention. The so-called meaning only exists in our mind because we do not know what the meaning is. Our approach is to construct meanings, in our definition, from syntax learning.

We have investigated the effect of the symmetry bias on the language evolution, which decides the meaning in an ambiguous environment. For this task, we constructed a Meaning Selection Iterated Learning Model (MSILM) based on the Simon Kirby's Iterated Learning Model (ILM), and simulated with three strategies: Perfect Matching Symmetry Bias (PMSB), Imperfect Matching Symmetry Bias (IMSB), and Random strategy. As a result of applying IMSB, the language of the agent evolved into more compositional one, and the agent acquired a more expressive and similar language to the parent's one than that with the Random strategy agent. On the other hand, as a result of applying PMSB, the language of the agent did not evolve well, the agent acquired a less expressive and different language to the parent's one than that with Random strategy agent. Our experimental results showed that the effect of IMSB accelerated linguistic evolution to obtain more compositional one, whereas PMSB disturbs linguistic evolution.

As a target of syntax-based meaning composition, we consider music scores. We assume that the rules of music progression are in a subclass of context-free language, and we let computers find them autonomously. We employ the ILM, and ask if the computer can find a music knowledge that is common to us, and also if the computers can compose music independently of our music knowledge. In this research, we have shown an example set of rules found in the 25 études of Burgmüller regarding a symbol as a set of notes on one beat. Although many of categories in the tree seem redundant and futile, some of them reflect probable progressions, which well match with our human intuition. This experiment has several virtues compared with other grammar-based formalism for music. One is that we do not need to provide a dictionary beforehand. The other is that we can exclude the human-biased intuition, which had hindered the definition of creativity.

Key Words: **Linguistic Evolution; Machine Learning; Symmetry Bias; Agent-based model; Grammar acquisition; Iterated Learning Model; Music Analysis; Self-organization**

Acknowledgments

I have had the opportunity to study under one of the best environment at JAIST. I would like to appreciate the supervisors, Satoshi Tojo (principal supervisor), Minh Le Nguyen (second supervisor), and Takashi Hashimoto (supervisor of minor research). Great instructions they give enabled me to argue this dissertation.

I would like to express my sincere gratitude for careful review and kind advice of this dissertation by referees, Kazuo Okanoya (Professor at Graduate School of Arts and Sciences, Tokyo University), Kiyoaki Shirai (Associate Professor at Entertainment Technology Area, JAIST), and Shogo Okada (Associate Professor at Intelligent Robotics Area, JAIST).

Besides supervisors and referees, I would like to thank researchers who are related to Tojo laboratory, sincerely. Hitoshi Omori (Junior-Professor of Nonclassical Logic at Department of Philosophy I, Ruhr-Universität Bochum) who was an assistant professor at Tojo laboratory has taught me the attitude what a good researcher should be. Katsuhiko Sano (Associate Professor at Department of Philosophy, Graduate School of Letters, Hokkaido University) who was also an assistant professor at Tojo laboratory lectured the basic of mathematical logic which became a new viewpoint of my research. Matoba Ryuichi (Associate Professor at Department of Electronics and Computer Engineering, National Institute of Technology, Toyama College) and Shingo Hagiwara (Associate Professor at Department of International Business, National Institute of Technology, Toyama College) are the longest advisors about language evolution since I was a student of National Institute of Technology, Toyama Colledge. Makoto Nakamura (Associate professor at Department of Engineering, Niigata Institute of Technology) gives me constructive comments based on his broad knowledge including legal informatics, language evolution, and grammar acquisition.

Finally, I would like to offer my special thanks to my wife, Tomomi Sudo. I received generous support from her. Without her encouragement, this dissertation would not have been possible.

I devote my sincere thanks to all of them and my colleagues.

Contents

List of Figures

Chapter 1

Introduction

1.1 Cultural Evolution

The cultural evolution in linguistics is an important factor to form human language. After the human acquired an ability to handle our language, they could develop the more complicated language from the simple primitive one where a meaning consists of one word. Therefore, we should know not only the biological reason but also the cultural reason to clarify a mechanism of language. Our language has two characteristic features; compositionality and recursion. Compositionality means that a sentence is decomposed into multiple words, and the whole meaning is composed from the meaning of each word efficiently. Recursion is that a structure of sentence can be embedded into the inside of others, to create a more complex sentence. These features are basic functions on our language, and thus we should focus on the reason why these features can be acquired by human through generations.

Kirby (2002) proposed ILM (Iterated Learning Model) which is the model of language evolution[30]. Among the language transition, there are two types of transmissions. One is the vertical transmission which is repeating the one from a parent to a child; one-to-one transmission through generations. The other is horizontal transmission which is the communication in one generation; that is bidirectional transmissions in the community. Kirby (2002) conducted the simulation of only vertical transmission on the computer. His work showed that a language with highly compositional and recursive structure would be acquired by Language Acquisition Device (LAD), without defining any language model. The limitation of his work is that the child can see the parent's intention completely.

Our language transmission is explained by I-language and E-language[6]. The former is knowledge of language in our brain which contains lexicon, syntax, meanings, structures of meaning, and the grammar to represent a meaning. The latter is the utterance to outside which is generated by I-language. The speaker agent transmits the intention using E-language, and the listener agent cannot acquire their intention indirectly.

Considering the way of transmission of intention is the issues to work in the language evolution. We assume that transmission of the intention may be considered on our communication as follows.

- Select the meaning of the speaker's intention from a set of alternatives which are obtained by human (*e.g.*, cognitive biases). If we hear "this one" in front of Figure 1.1, we may guess alternatives which are a pen and a bottle of water. We will select a pen or a bottle of water as the meaning of "this one".

Figure 1.1: A Red Pen and a Bottle of Water

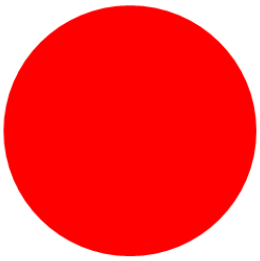


Figure 1.2: What's the difference? "This is red"

- Organize the meaning from the sensory information (*e.g.*, RGB value, wave voice signal, and objectively data) and the samples of sentence. If we hear "This is red" and "This is blue" in front of Figure 1.2 and Figure 1.3 respectively, we may understand what "red" and "blue" would represent. In this case, we cannot define our meaning space because of the possibility of infinite meaning. Therefore, we need the methodology to organize the meaning in the language evolution.

We need the methodology of language evolution to understand natural language. In general, NLP (Natural Language Processing) is developed as practical technology for the usage of computing power, and could not explain the process of the calculation and what the result means. Language evolution fills this gap of linguistic knowledge.

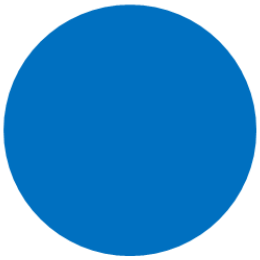


Figure 1.3: What's the difference? "This is blue"

1.2 Selection by the Cognitive Bias

In section 1.1, we denoted two ways of transmission of intention. In this section, we discuss the transmission of intention from the alternatives of the meaning. In general, the number of meanings recognized from the situation is infinite, because the meaning is mapped by the sensory data of the focused object which relates to the other object, and the verbal relation between meanings of the objects. We described multiple recognition for a situation in Figure 1.1 in Section 1.1. In addition, we cannot know that each person realizes an object as the same meaning respectively. Therefore, we assume that alternatives of the meaning of the utterance are limited by the human cognitive sense and nonverbal language of the speaker. The issue that the listener selects the true meaning on alternatives of meanings can be handled as the multi-armed bandit problem[28]. Cognitive biases are known to be effective to solve the problem. It may cause the illogical decision, but can accelerate the learning.

In the area of language evolution, we should investigate the variation of the accuracy of the selection by cognitive bias over time. We also consider if the learner's acquired knowledge is the same as the teacher's one. Cognitive bias may create a misunderstanding, so we should focus on how the gap influences the occurrence of language evolution.

1.3 Self-organized Meanings

In the case of no explicit meaning transmission in the communication, the learner acquires the meaning by self-organization from experiences of the utterance and the situation. The learner on the vertical transmission of language evolution learns the language from the state with no knowledge. Therefore, the learner needs the self-organization of the meaning to express the arbitrary situation.

E-language is the utterances recognizable by everyone. On the other hand, I-language is the internal knowledge of language which resides in each agent. Thus, there may be a gap between the intentions; one is observed as E-language by a listener and the other is of the speaker that based on I-language.

Natural language has ambiguities in understanding[43]. In NLP, many researchers have

tried to resolve the ambiguity of meaning space in understanding using Deep Learning, self-organizing map, and other machine learning techniques[27, 23, 43]. These research are done on unsupervised machine learning for ambiguities, without referring to the meanings of word and syntactical category. It is impossible that the given input data cover all of the language knowledge, thus, we need to compensate semantic features from the given data which are able to parse sentences with unknown words and structure.

In language evolution, self-organization should be investigated because the function of language changes dynamically over time. Simple meanings are represented by a few number of utterances in the early stage of language evolution, however, when we are forced to express complex meanings for communications in the grownup society, we must compose corresponding complicated utterances. In other words, we cannot simulate language evolution using well-defined semantics and syntax.

NLP requires much calculation amount and enormous input data. Therefore, the calculation of NLP would take long time to fix the model parameters. We do not take the calculation like *Deep Learning* and complex probabilistic model in own brain. In our hypothesis, the processing like self-organized map happens in our brain, hence we consider about dynamic acquisition of meanings in cultural evolution.

1.4 Relationship between Language and Music

Hauser (2016) argued the universality of human linguistic ability[21]. He called the linguistic generative faculty as "Universal Generative Faculty" (UGF). In his opinion, we can observe the same hierarchical structure with language on mathematics, morality and music, *e.g.*, noun phrase, noun clause, and noun in language, as well as chord progression, chord with the key, and chord in music.

The study of machine learning developed by NLP has been applied to music in various cases[13, 39, 38, 48]. And also, the structure generation through cultural evolution which Kirby (2002) proposed has been applied to laboratory experiments of music; the structure of rhythm[42] and the structure of playing music[49] was produced by iterated learning.

The paper of Kirby (2002) denoted that the hierarchical structure in language is caused by cultural evolution, but, every hierarchical structure cannot be what we handle as UGF; the hierarchical structure which has no recursion is not the whole expressive power of UGF. The feature of our language requires recursion in general[7, 10], and the structure which has no recursion is not the structure of UGF even if the structure is hierarchical. Therefore, the model which generates the hierarchical structure is not enough condition for human-like model. If we assume the mechanism of UGF, we should challenge to verify the model on the various areas which are expressed by UGF.

1.5 Formalization of Language and Music

In computer science, we handle language and music by formalization to generate intellectual products automatically and to analyze the property. In particular, the generative system which is modeling by formal grammar has the advantage to observe the generation process explicitly as compared to stochastic and statistic model and Deep Learning. To investigate language transmission in the complex system, we consider that the simple generation process as feedback is important.

We can refer to the animal languages as an observable parallel case of our progenitor's languages[20]. There exists a research of birdsong as a kind of animal language[31, 2, 46]. There are three steps in the recognition of birdsong; symbolization on continuous voice, classification of a symbol, and generation of syntactic rules. In linguistics, the first and second issues correspond to phonology, and the third issue corresponds to theoretical and cognitive linguistics. The methodology for the birdsong is the same as natural language processing. The birdsong has simple grammar which is executable by computer, that is finite deterministic automaton. Therefore, the initial human language must have been simple grammar as the birdsong, then the language is getting complex through cultural evolution.

One of the formalizations of music is GTTM (A Generative Theory of Tonal Music) which is proposed by Jackendoff and Lerdahl (1982) to formalize western tonal music formally based on music theory[11]. However, we need the human judgment to select the applicable rule to analyze the music. To resolve the problem regarding the applicable rule, there are researches using *Deep Learning*[16, 17]. It is difficult to implement the abstracted operations (*e.g.*, When the contradictory situation happens, the calculation process goes back to the lower layer process). Therefore, we cannot analyze the music completely based on GTTM.

On the other hand, there are researches using formal grammar which Chomsky proposed[48, 12]. These researches describe the music theory as a set of rules on the formal grammar (*e.g.*, Context-Free Grammar, Head-Driven Phrase Structure Grammar, and so on). In the analysis step, the existing music score is parsed by the set of rules predefined, and we will evaluate the parsed structure. When we describe music theory in formal grammar, the set of rules may become counter-intuitive to us.

1.6 Our Proposals

In this research, we propose new models of cultural evolution and show their simulation results. We constructed the iterated learning models that the one employs alternatives on meaning transition in Chapter 3 and the other develops the self-organized meaning without meaning transmission in Chapter 4, and simulate the language transition in the various conditions. And also, we challenged to formalize language and music.

Ambiguous environment like that we indicated in Section 1.1 is a kind of important features in language evolution. When we suppose that the meaning transition is realized by the joint attention (Section 1.2), we need the simulation model of language transmission based on the joint attention frame for study of language evolution. Meaning Selection Iterated Learning Model (MSILM) is ILM using transmissions of the language on ambiguous environment based on the joint attention. The second is ILM using self-organizing meanings corresponding to syntax. We evaluate the possibility of language evolution in an ambiguous environment and usefulness of the self-organizing meanings on language evolution using our model on computer simulation.

In Chapter 3, we experiment the iterated learning simulation on the meaning selection situation. In this circumstance, we employ the cognitive bias as a way of meaning selection. We focused on symmetry bias, and simulated two implementations of symmetry bias; the one is PMSB (Perfect Matching Symmetry Bias) which works for learned utterances, the other is IMSB (Imperfect Symmetry bias) which is based on similarity of meanings and

utterances. PMSB has an integrated meaning and assigns the same meaning to the same strings, and does not works for unknown utterances. IMSB has totality for all utterances because works for all utterances. IMSB is able to infer for all utterances, however, the selection considering similarity includes logical fault (*e.g.* When we do not have experience in language, it is wrong to map an utterance "eats pie" to a meaning "eats pine" which is similar to it). We investigate the effect of biases on the meaning selection situation in language evolution. In the experiment of PMSB, language evolution is not happened, and an agent's language is prone to the inherited language from a previous generation. We appraised the effect of language evolution by IMSB that causes compositionality of an agent's language in the joint attention environment.

If we suppose the meaning transition by self-organization like that we argued in Section 1.3, not to select a meaning from universal meaning space, we need to investigate the validity of the method of self-organization using the data grounded on the real world. In this research, we used music to verify our proposed method as the target of the real world. We assess soundness of our self-organization by music, not language, in terms of UGF as indicated Section 1.4.

In the Chapter 4, we investigate whether the agent acquires the meaning structure to express music theory through generations on the circumstance that the intention of utterance is not transmitted to the learner. The model in Chapter 4 employ a created music by agent's knowledge, different inputs with parent's one, and our learning method based on Kirby (2002)[30] as an utterance, bottleneck, and learning in iterated learning, respectively. On the situation which has no transmission of the meaning in language evolution, we cannot assess the correctness of the meaning structure directly. Therefore, we evaluate whether the structure of the meaning can be the expressible structure of music theory.

We experimented the iterated learning for 25 études of Burgmüller, and generated parse trees based on an agent's set of rules. We can appraise relationship between musical notes and phrases on a parse tree. As a result of assessment, we found the progress from dissonant chord to harmonic chord. The progress represents a part of music theory. However, we found much meaningless rules in a agent's knowledge. The rules is necessary for creations of more abstract rules, and we have possibility of more beneficial result if give the model audio signal information.

1.7 Thesis Outline

The rest of this thesis is organized as follows.

Chapter 2 introduces background needed in the rest of the thesis. Chapter 2 does not contain our contributions. Section 2.1 studies about CFG which is basic grammar in our simulations. Section 2.2 indicated basic ILM which is proposed by Kirby (2002). We described the concept of limitation of meanings on ambiguous situation in Section 2.3.

Chapter 3 shows language evolution on joint attention environment as ambiguous situation. We propose the methodologies to select a meaning from the alternatives, based on symmetry bias. This chapter denotes one of our work as linguistic transition on ambiguous environment. Section 3.1 showed the proposals in Chapter 3. In Section 3.2, we proposed new criterion to evaluate dissimilarity between two language knowledge. Section 3.3 also indicated the way of acceleration of the simulation using the clipping

agent's utterances. In Chapter 3, we argued occurrence of language evolution by the symmetry biases we indicated in Section 3.4, and characteristics of the symmetry biases. We proposed iterated learning model employed joint attention in Section 3.5, and inference of a meaning from an utterance based on symmetry bias in Section 3.6. Section 3.7 shows the experimental condition using clipped utterance (Section 3.3) and evaluation criteria including language distance (Section 3.2). Conclusion in Chapter 3 is indicated in Section 3.8.

Chapter 4 shows application of iterated learning to music scores. We propose the learning which acquires semantics and syntax at the same time, and investigated the effect of the learning for music scores. this chapter denotes one of our work as usefulness of self-organizing meaning to recognize music structure. In Section 4.1, we indicated the grammar of music as background in Chapter 4. Section 4.2 introduced categorial level which is semantics to work self-organization, and learning method for our proposed grammar. On our model, we clearly can recognize the meaning of utterance created by an agent (Section 4.3), and Section 4.4 is explanation how to generate an utterance from agent's knowledge. Furthermore, we can realize how an agent understands symbol sequence by the parse tree of the sequence. Therefore, our proposed model is a generative descriminative model of music. In our model, agent may have multiple understanding for a music score. An assessed parse tree of each score is randomly generated (Section 4.6). Our experimental results shows that a part of an agent's knowledge represents music theory (Section 4.7). Section 4.8 indicated that argumentation of validity of self-organization and usefulness as music model.

We concluded the achievement in language evolution on ambiguous environment, and indicated a further direction of language transmission on ambiguous situation.

Chapter 2

Preliminaries

2.1 Context Free Grammar

CFG (Context-Free Grammar) is one of 4 formal grammars which Chomsky (1959) proposed as Type 2 Grammars[5, 4]. Chomsky (1959) defined 4 layer of grammars which is PSG with restrictions. PSG has the same expressive power with Turing Machine. In particular, CFG has the mathematical feature which is the similar to language, is used to be the base model of NLP and programming language. The definition of CFG as follows.

Context Free Grammar G_0

$$\begin{aligned} G_0 &= (N, T, P, S) \\ N &: \text{Non-terminal symbols} \\ T &: \text{Terminal symbols} \\ P &: \text{A set of rules} \\ & \quad \alpha \to \beta \left(\alpha \in N, \beta \in (N \cup T)^+\right) \\ S &: \text{Start non-terminal symbol } S \in N \end{aligned}$$

2.2 Iterated Learning

Infants learn language from their parents, without observing the internal language of them, that is the innate but hidden grammar formalism. As a result, the infants may acquire a different language from their mothers. In Figure 2.1, the baby guess possible chunking from the few sample sentences given by its mother.

Simon Kirby described compositionality and recursion as fundamental features of grammar[30]. In Kirby's ILM, an infant's linguistic knowledge is evaluated by expressivity of linguistic knowledge and a number of grammar rules. Compositionality is the property whereby "the meaning of an expression is a monotonic function of its parts and the way they are put together[3].". High compositional grammar provides higher expressivity and a smaller number of grammar rules. Kirby's ILM is the model whereby a pair of a parent agent and an infant agent resides in a generation, and the infant agent becomes the parent agent of the next generation (Figure 2.2).

The parent agent and the infant agent are given the same meaning from the predefined

Figure 2.1: Chunking by Infants

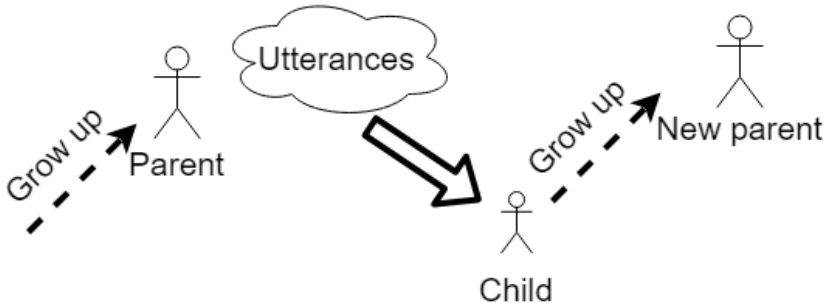


Figure 2.2: Illustration of Kirby's ILM

meanings. The parent agent generates an utterance which represents a given meaning. This process is iterated for a predefined number of times. In ILM, a meaning represented by a form of predicate-argument structure (PAS), and, the size of predefined meanings is 100 distinct meanings constructed by 5 predicates and 5 objects (Figure 2.3). The infant agent learns grammar rules from pairs of the given meaning and a parent's utterance. Meanwhile, the infant agent generalizes their language knowledge. In ILM, agent's knowledge represented by Labeled Context Free Grammar (LCFG) as follows.

Labeled Context Free Grammar G

$$
\begin{array}{rcl}
G & = & (N, T, M, V, P, S) \\
N & : & \text{Non-terminal symbols} \\
T & : & \text{Terminal symbols} \\
M & : & \text{Predefined meanings (PAS)} \\
V & : & \text{Variables for } M \\
P & : & \text{A set of rules} \\
 & & \alpha \rightarrow \beta \left(\alpha \in N \times (M \cup V), \beta \in ((N \times V) \cup T)^{+}\right) \\
S & : & \text{Start non-terminal symbol } S \in N \\
 & & \alpha \text{ must have all variables in } \beta.
\end{array}
$$

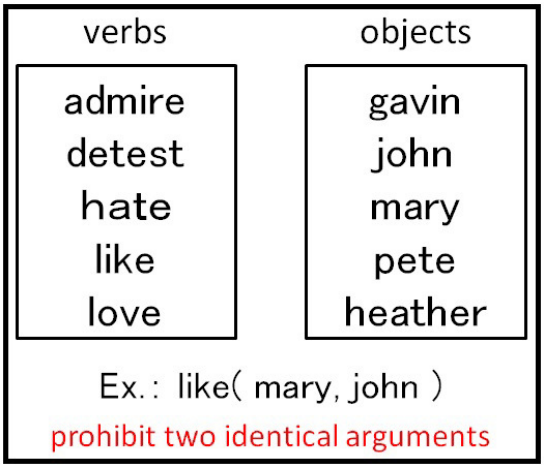


Figure 2.3: A Form of Meaning in ILM

The three operations employed in the ILM are as follows[1].

- Chunk

$$
\begin{aligned}
&S/like(mary, john) \rightarrow marylikesjohn \\
&S/love(mary, john) \rightarrow marylovesjohn \\
&\qquad\qquad\qquad \downarrow chunk \\
&S/X_1(mary, john) \rightarrow mary \;\; N_0/X_1 \;\; esjohn \\
&N_0/like \rightarrow lik \\
&N_0/love \rightarrow lov
\end{aligned}
$$

- Merge

$$
\begin{aligned}
&S/hate(X_2, X_3) \rightarrow N_1/X_2 \;\; hates \;\; N_2/X_3 \\
&N_1/gavin \rightarrow gavin \\
&N_1/pete \rightarrow pete \\
&N_2/gavin \rightarrow gavin \\
&\qquad\qquad\qquad \downarrow merge \\
&S/hate(X_2, X_3) \rightarrow N_2/X_2 \;\; hates \;\; N_2/X_3 \\
&N_2/gavin \rightarrow gavin \\
&N_2/pete \rightarrow pete
\end{aligned}
$$

- Replace

$$
\begin{aligned}
&S/admire(john, pete) \rightarrow johnadmirespete \\
&N_3/admire \rightarrow admire \\
&\qquad\qquad\qquad \downarrow replace \\
&S/X_1(john, pete) \rightarrow john \;\; N_3/X_1 \;\; pete \\
&N_3/admire \rightarrow admire
\end{aligned}
$$

[1]Chunk and merge is which Kirby defined, but, replace is which Nakatsuka(2006) defined as independent operation from the Kirby's research[47].

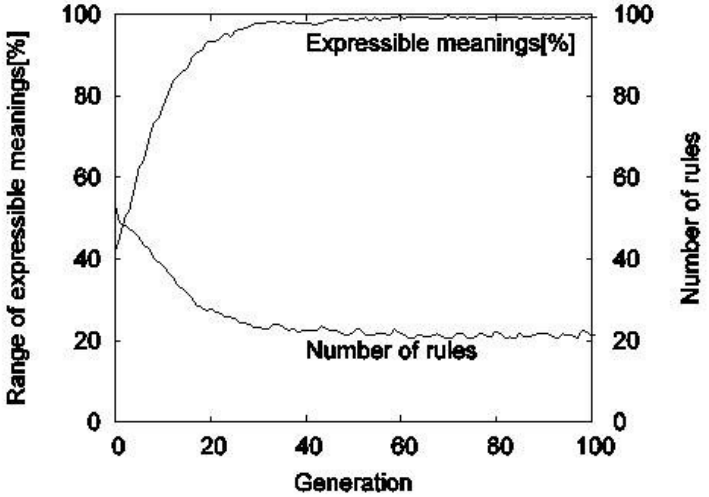


Figure 2.4: Kirby's ILM Expressivity and Number of Grammar Rules

Since the number of utterances is limited to 50 in the Kirby's experiment, the infant agent cannot learn all the predefined meanings directly (the size of predefined meanings is 100), thus, the infant agent has to generalize their linguistic knowledge by learning to the predefined meanings. When the parent agent cannot utter because of a lack of the grammar rules, she invents a new rule. This process is called the invention. Even if the invention does not work to complement the parent agent's grammar rules, she utters a randomly composed sentence.

We implemented Kirby's ILM, and evaluated knowledge of the agent for each generation. In addition, we experimented 100 times each of whose random-seed was changed at each time. Figure 2.4 shows the simulation result of the mean of expressivity and number of grammar rules in an exploratory experiment of Kirby's ILM. The vertical line represents the mean of expressivity, number of grammar rules, respectively, and the horizontal line represents generations. From Figure 2.4, expressivity started at 41.00% and finished at 99.32%, and number of grammar rules started at 41.92 and finished at 23.98. Therefore, the final agent of Kirby's ILM can generate utterances which represents most meanings. The agent's expressivity is increasing through generations, nevertheless, the agent's number of grammar rules is decreasing, because compositionality is emerged in the agent's grammar.

2.3 Joint Attention

Joint attention frame is an idea to share the intention of utterance realistically[22]. In joint attention frame, the transmission of the meaning is realized to understand the speaker's intention by eye-gazing, pointing and non-verbal indication. However, if we assumed that the meaning is a combination of the sensory data, a number of the meaning of an object is infinite because of continuity of sensory data. We can avoid the problem of a number of recognized meaning by assuming the faculty which is to decide the unique meaning to the situation. At the circumstance, there is "gavagai" problem[41]; that is ambiguities of the selected meaning from the same situation on joint attention frame (Figure 2.5).

Figure 2.5: “gavagai” problem

Chapter 3

Iterated Learning on Joint Attention Environment

3.1 Selection on candidates of the meaning

When we are put in unacquainted situations, we often employ illogical inferences, and some of them are called cognitive biases. At a period in the formation of language, cognitive biases work well for lexical acquisition [36, 32]. Cognitive biases restrict a flood of concepts for disambiguation, and enable easy mapping from a word to its meaning.

For example, once an infant maps a label to an object, it does not map the same label to other objects by the mutual exclusivity bias. Hence, unknown labels preferentially are mapped to new objects, and this helps the infant's lexical acquisition. In particular, the symmetry bias is known to be efficient to acquire lexica and sentence structures [35, 34, 26, 25]. Human recognition is affected by the symmetry bias.

The symmetry bias is a tendency to recall a meaning of an utterance that a hearer has already known when they receives a new utterance, and allows human to connect an utterance with its meaning each other. The symmetry bias is unique to human, *i.e.*, even great apes such as chimpanzees, orangutans, bonobos and so on, do not have it [18, 44]. They also do not have as complicated language as humans. Furthermore, it is considered that the symmetry bias and human's language-acquisition are strongly related, and this relationship may affect language evolution.

Also, there is similar researches concerning Saussurean sign [24, 37] toward researches of the symmetry bias. In the idea of Saussurean sign, it is important for human's communication, that is a bidirectional mapping between a phonological form and some representation of a concept. On the other hand, a bidirectional mapping between an utterance and a meaning is important for human's communication. When the utterance which they already know is identical to the former one, the symmetry bias works as an inference of what the utterance implies. In the research of Sassurean sign, the bidirectional mapping is important, however, the symmetry bias do not make mapping, and is just a bias for recognition of a meaning. It allows that a meaning can represent several other different utterances. The target of the research of Sassurean sign focuses on the lexical acquisition, however, the target in this research focuses on the grammar acquisition.

It is beneficial to investigate the effects of the symmetry bias on language evolution for unravelling a mechanism of language acquisition. However, in general, the computer simulation for the language evolution is difficult since we cannot reconstruct our research

environment exactly as the same as what is going on in our mind, and thus, we need to set up our simulation environment according only to our ungrounded intuition. If we could observe human mind in the more precise way and could clarify how our mental states shift, we might able to make this reconstruction process easier, but such an observation is unrealistic. Fortunately, we can simulate language evolution in a simpler model through computer simulation. Using this, we can control the environment of experiment, and we can estimate a human's internal state changes through that of an agent. Hence, computer simulation is effective to test various hypotheses of language-acquisition or language evolution, and to design experiments with reproducible conditions, such as learning times, population, and various difficulty levels of lexical acquisition and so on.

So far, we have implemented a computer simulation model based on Kirby's Iterated Learning Model (Kirby's ILM) [30] to investigate the effect of the symmetry bias. In Kirby's ILM, a parent agent generates an utterance based on a meaning from a predefined meaning set, and an infant agent receives a pair of the meaning and the utterance for 50 times which are the half size of the predefined meaning set, in a generation. After getting pairs of an utterance and its meaning, the infant agent generalizes its knowledge through learning, and the infant agent becomes a parent agent of next generation. Matoba removed some meanings from the pairs which the infant agent received [34].

In the study, the infant agent receives pairs with and without a meaning of its utterance. When the infant agent receives an utterance without its meaning, it infers a meaning of the utterance using the symmetry bias. Consequently, the infant agent can infer the meaning of its utterance using its acquired knowledge. When the infant agent is not given the meaning, the infant agent with the symmetry bias makes an inference based on the acquired knowledge, whereas the infant agent that has no symmetry bias allocates a meaning to an utterance randomly. As a result, the agent which has the symmetry bias acquires a more expressible, and compact language more efficiently.

Next, we need to evaluate the similarity between two languages, because we must evaluate how the cognitive bias helps to acquire the same language as the parent's one. To solve the task, we formulated the language distance between two linguistic knowledge. In addition, we revised the simulation model in the more realistic way, and the revised model that we call it MSILM (Meaning Selection Iterated Learning Model), is incorporating a joint attention frame as a new feature into Kirby's ILM. The difference between Kirby's ILM and MSILM is a way of transmission of meanings from a parent agent to an infant agent. Joint attention frame is for sharing the common state of environment between two individuals in the real world. Though the parent transmits a meaning to an infant correctly in Kirby's ILM, it is difficult to transmit a meaning to another individual where language of an each individual do not know language of another individual in real world. In MSILM, the joint attention frame introduces a state where multiple meanings are given for an utterance.

In this study, we verify the effect of the symmetry bias on language evolution, and we make a comparison of two kinds of implementation of the symmetry bias. Our aim is to observe the effect of language evolution by the strategy for meaning selection. The first implementation is named PMSB (Perfect Matching Symmetry Bias) which behaves the same as the existing research[34]. The other is named IMSB (Imperfect Matching Symmetry Bias) of which the constraint is weaker than the PMSB. The constraint of the IMSB depends on the similarity of utterances and meanings. The reason for including similarities is that a human does not repeat the same action perfectly every time, so

every human has the aptitude to accept a variation of each other's behavior, *i.e.*, we supposed the infant agent would recognize an utterance as a clue to decide a meaning. To infer a meaning of an utterance, the infant agent employs three strategies which are i) Random, ii) PMSB, and iii) IMSB. In this paper, we employ the constructive approach, and investigate the effect of these strategies on linguistic evolution.

3.2 Evaluation of Similarity between two Languages

We designed language distance which evaluates the dissimilarity between two kinds of linguistic knowledge. The smaller the value of language distance is, the more similar the linguistic knowledge is. We employed Levenshtein distance and Hamming distance on language distance. Levenshtein distance evaluates similarity between two strings of terminal symbols in terms of translating operations through inserting, deleting and replacing, *e.g.*, Levenshtein distance between two utterances which are "abc" and "abde" becomes 2 (replacing 'c' to 'd' and inserting 'e' at end). Hamming distance evaluates similarity between two strings of terminal symbols which are the same length, unlike Levenshtein distance. Translating operation on Hamming distance is only replacing, *e.g.*, Hamming distance between two meanings which are "like(mary, john)" and "like(mary, pete)" becomes 1 (just replacing 'john' to 'pete'). The following is the procedure to calculate language distance between the infant agent and the parent agent.

Step 1 Make pairs of a meaning and an utterance as much as possible using each linguistic knowledge.

Step 2 Pick up one of the infant's pairs.

Step 3 Select one of the parent's pairs which has the most similar meaning to the one of the picked up pair at step 2.

Step 4 Calculate Levnshtein distance between the utterance of the picked-up pair at step 2 and the utterance of selected pair at step 3.

Step 5 Repeat step 2 to step 4 in each infant's pairs, and calculate the mean of Levenshtein distance at step 4. This value becomes language distance of two kinds of linguistic knowledge.

Figure 3.1 shows the image of language distance between a parent and an infant. In the Figure 3.1, each black dot in the ellipse represents a pair of a meaning and an utterance, and an arrow which connects two black dot means a combination to calculate Levenshtein distance between two utterances. There also exists a black dot without an arrow, *viz.*, there is a pair that is not associated with calculating language distance in a parent's pairs. The situation issues that, language distance is designed for a parent's comprehension of an infant's language, because our aim of language distance is to evaluate the change of language from a parent to an infant. Figure 3.2 shows the transmission of utterances. A child have the part of the parent's utterances. In ILM, parent cannot give all utterances to express the whole meanings. Therefore, the child fills the lack of knowledge from given utterances. The language distance evaluates the difference of knowledge that the child fills.

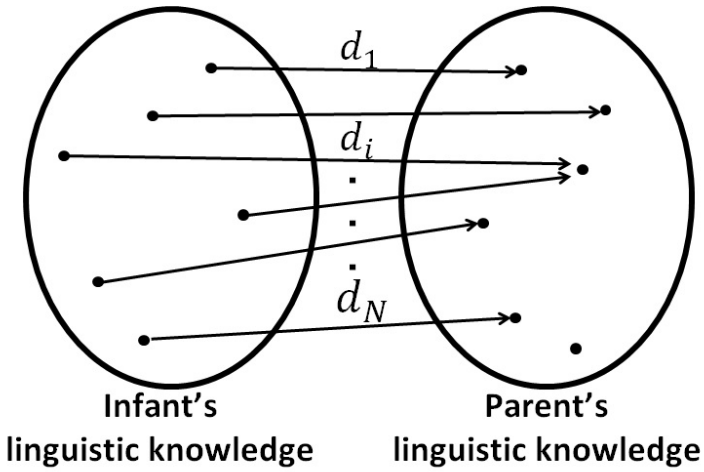


Figure 3.1: Processing of Language Distance

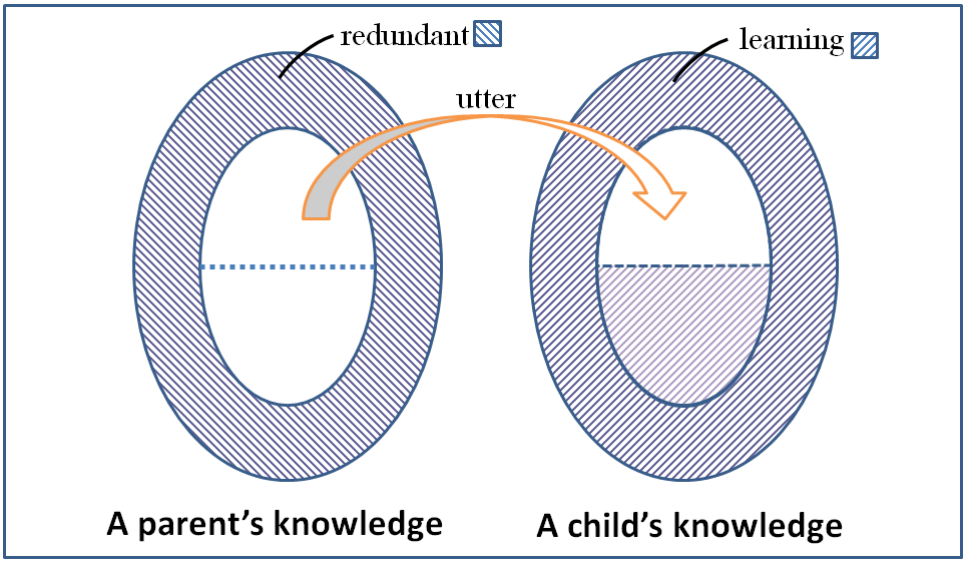


Figure 3.2: Extension of Knowledge in a Generation

An infant's language is learned from only a parent's utterances. If an infant acquires a parent's language perfectly, language distance would be 0, and, if an infant understands a parent's grammar of language perfectly (the infant parses given utterances perfectly), the language distance also would be 0. The following Equation 3.2 denotes the calculation of the language distance between knowledge K^a and knowledge K^b where u_i^a and $u_{u_i^a}^b$ are generated by K^a and K^b respectively. But also, $u_{u_i^a}^b$ should be generated from a meaning which is most similar to the meaning of u_i^a.

$$D_{K^a K^b} = \frac{1}{n} \sum_{i=0}^{n} \frac{d\left(u_i^a, u_{u_i^a}^b\right)}{|u_i^a| + \left|u_{u_i^a}^b\right|} \tag{3.1}$$

We experimented on language distance with the same experimental setting as Kirby's ILM(section 2.2). Figure 3.3 shows the result of applying language distance to Kirby's ILM. The vertical line represents the mean of language distance, and horizontal line represents generations. In Kirby's ILM, the language distance started at 0.406 and finished at 0.098. Since an infant agent receives a parent's utterance and its meaning, an infant understands a parent's language almost certainly through some pairs. It is expected that the higher expressivity is, the smaller the language distance is. In fact, from Figure 2.4 and Figure 3.3, the result of language distance shows the expected behavior, furthermore we can observe that the generation of the converged expressivity and that of the language distance is almost the same.

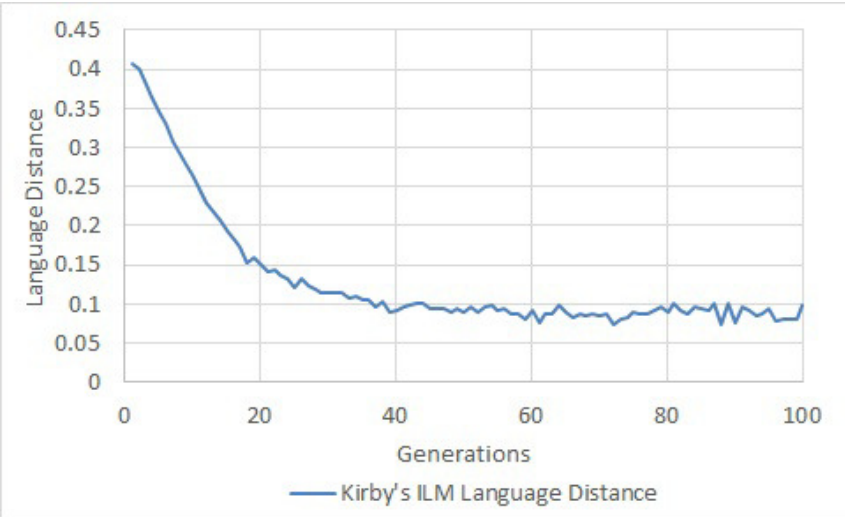


Figure 3.3: Kirby's ILM Language Distance

3.3 Acceleration of Iterated Learning Simulation by Clipped Utterances

In expanded ILM (*e.g.*, adding of population), we know that the mean of length of an utterance increases endlessly. This problem make it difficult to calculate the simulation

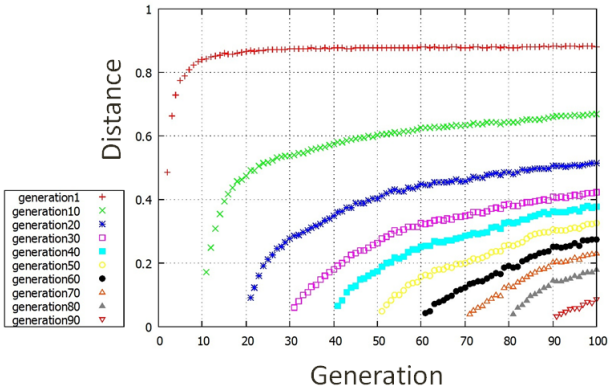


Figure 3.4: Language Distance: ancestor to offspring

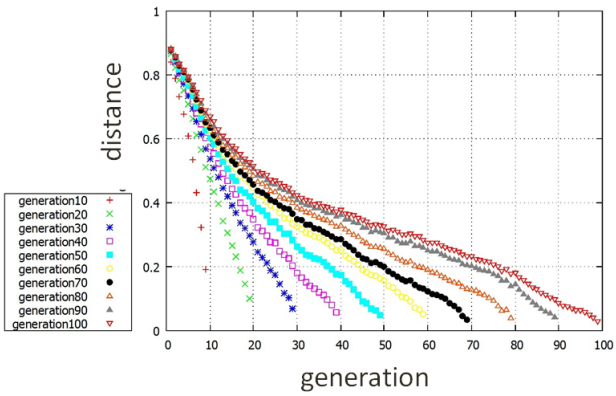


Figure 3.5: Language Distance: offspring to ancestor

and the evaluation of language distance. We employ clipped utterances to ILM, and verify an influence of it by evaluations which are expressivity, number of rules, and language distance.

Figure 3.6 shows an example of clipping. At first, the child collects rules to make an utterance, then erases the string in a rule until the erased string match any string in the child's knowledge or the length of the erased string becomes 1. In this case of Figure 3.6, "cba" and "ed" become "cb" and "ed" respectively because the erased string is 1. And moreover, "abef" becomes "abe" because the erased string become "ab".

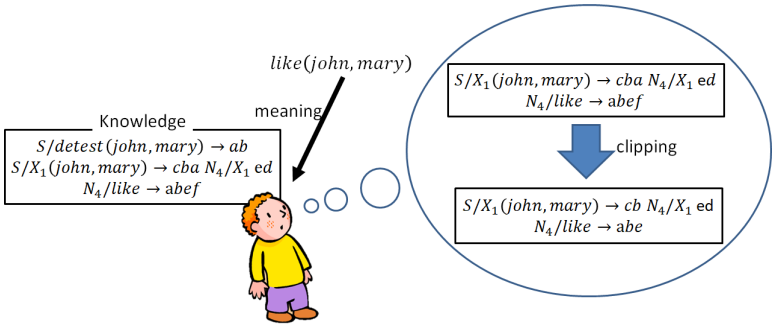


Figure 3.6: Illustration of Clipping

We experimented the simulation using the clipped utterances, and evaluated agent's knowledge by expressivity (Figure 3.7), number of rules (Figure 3.8), language distance (Figure 3.9) and a mean of utterance length (Figure 3.10, 3.11). We obtained no difference between agent with clipping and agent without clipped about expressivity, number of rules and language distance. And, we confirm decreasing of utterance length by clipping utterances.

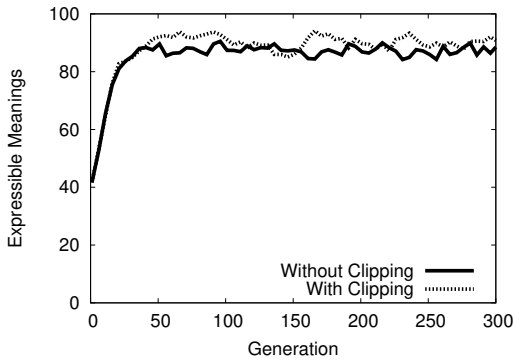


Figure 3.7: The Result of Clipping: Expressivity

3.4 Symmetry Bias

The symmetry bias makes a symmetric relation between an object and a label. Figure 3.12 shows a situation where the symmetry bias works. If infants are taught that a red sphere

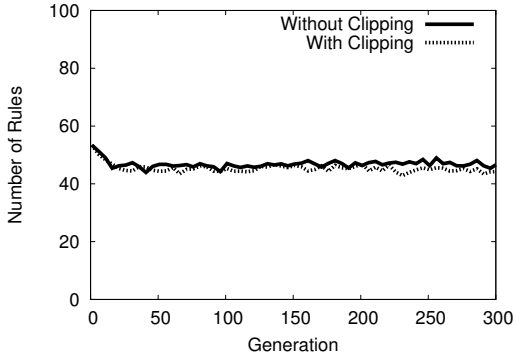


Figure 3.8: The Result of Clipping: Number of Rules

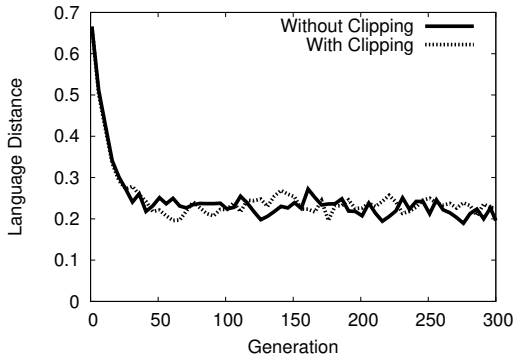


Figure 3.9: The Result of Clipping: Language Distance

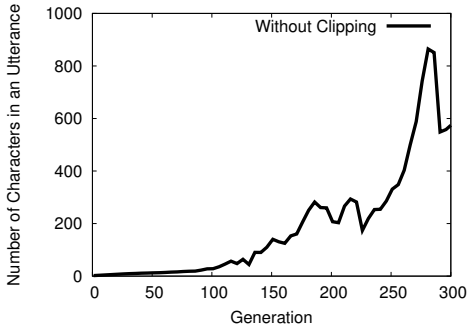


Figure 3.10: The Result of Clipping: Average of Utterance Length(without Clipping)

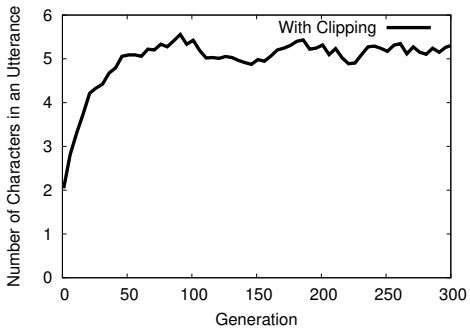


Figure 3.11: The Result of Clipping: Average of Utterance Length(with Clipping)

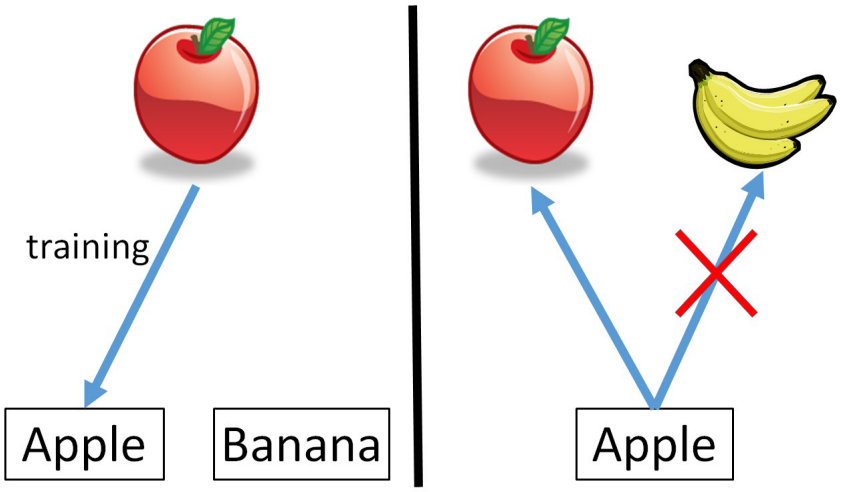


Figure 3.12: Illustration of the Symmetry bias

object has a lexical label "Apple", then they apply the label "Apple" to the red sphere object, *i.e.*, although infants learned that the object implies the label, infants recognized that the label implies the object. This action is considered as human specific, and is an illogical inference. Illogicality can be shown by the following incorrect sequence: By the symmetry bias, "It is rainy, because a person is opening an umbrella." is derived from "There is a person opening an umbrella because it is rainy", even though the opening of an umbrella does not imply rain. This inference is clearly incorrect. In language acquisition by the symmetry bias, the infant might make a mistake because of illogical inference. Thus, it is important to research the effect of the symmetry bias in language acquisition, and also language evolution.

In this paper, the symmetry bias acts in mapping a meaning and an utterance. Concretely, when an infant received an utterance from a parent, the infant appropriates the same meaning to a meaning of the utterance which the infant knows. In the other words, the infant could reverse the relationship of an utterance and a meaning by the symmetry bias.

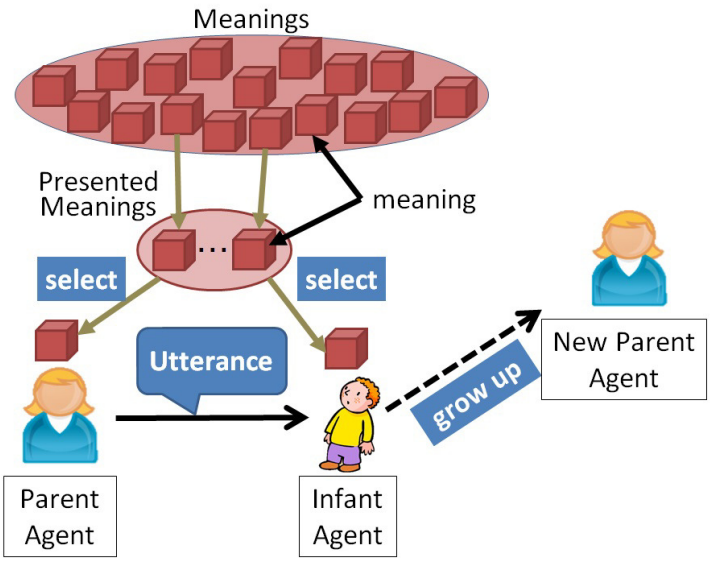


Figure 3.13: Illustration of our model wich is MSILM

3.5 Meaning Selection Iterated Learning Model

This section explains MSILM and assessment criteria of agent's linguistic knowledge. In this paper, we experimented simulations using MSILM based on ILM proposed by Kirby (2002), and evaluated the agent's linguistic knowledge by assessment criteria.

We have constructed MSILM which employed the notion of joint attention frame in Kirby's ILM (Figure 3.13). In MSILM, both the parent agent and the infant agent are presented with multiple meanings. The parent agent selects a meaning from the presented multiple meanings, and utters it to the infant agent. The infant agent infers a meaning of the utterance from the presented multiple meanings, and learns a pair of an inferred meaning and the utterance. Thus, the infant agent is not always able to infer the same meaning to the parent's selected meaning. So far, an agent's linguistic knowledge is evaluated by the expressivity of linguistic knowledge and the number of grammar rules. However, these criteria represent just about compositionality, *i.e.*, the similarity between the parent agent's linguistic knowledge and the infant agent's one is not represented.

In MSILM, it is important whether the infant agent can acquire the same language to the parent agent's one. Therefore, to evaluate the similarity between two languages, we employ language distance which is based on similarity between two utterances which are strings of terminal symbol[35].

3.6 Inference Strategy

In this study, we proposed two inference strategies which are PMSB and IMSB, based on the symmetry bias. Also, we employed Random strategy as a comparative indicator. These strategies are the ways to select a meaning of an utterance from presented multiple meanings. The descriptions of the strategies are as follows.

Random The infant agent selects a meaning from presented meanings randomly.

PMSB The infant agent compares the parent agent's utterance to all utterances which they can generate, and selects the same one. Next, they compare a meaning of the selected utterance to the presented meanings, and select the same meaning from them. If the infant agent cannot find the same utterance or meaning, the infant selects a meaning randomly.

IMSB The infant agent compares the parent agent's utterance to all utterances which it can generate, and selects the most similar one in terms of Levenshtein distance. Next, they compare a meaning of the selected utterance to presented meanings, and selects the most similar meaning from the presented meanings in terms of Hamming distance.

In the inference of IMSB, an agent selects the same utterance and meaning, however, in the inference of PMSB, an agent selects the most similar utterance and meaning. In terms of IMSB, when the infant agent can generate the same utterance to the parent's one, the infant agent infers the same meaning to the parent's selected one. This behavior in IMSB agent is equal to the one in PMSB agent, *viz.*, the condition which IMSB infers correctly includes one that PMSB infers correctly. Furthermore, IMSB also works even if PMSB does not work.

3.7 Experiment and Result

In the experiments, we employed 100 meanings which are constructed by five predicates and five object words. The number of presented meanings is two (if the number is set as over three, the experiment result will be unstable because of difficulty of the meaning selection task). The presented meanings are chosen randomly at each time. We examined the efficacy of each inference strategies when the infant agent infers the meaning of an utterance in two presented meanings. In MSILM, we measured expressivity, number of grammar rules, language distance, and ratio of correct meanings which is accuracy rate for the infant to select a parent's meaning in two presented meanings. We experimented 100 times, each of whose random-seed was changed deferentially. The value of each criterion is calculated as the mean of 100 values. Figure 3.14, 3.15, 3.16 and 3.17 show the comparison of the result of expressivity, number of grammar rules, language distance, and ratio of correct meanings, respectively with each strategy.

From Figure 3.14, we can observe that the IMSB agent's expressivity records the highest value in the three strategies. The PMSB agent's expressivity is the lowest even though the infant agent applies the symmetry bias as PMSB. In the case of applying PMSB, an infant agent acquires the same meaning if they know the parent's utterance, however, in the case of applying Random, an infant agent has possibility to acquire new information (an utterance represents a new meaning) even though they know the parent's utterance. Thus, the PMSB agent's expressivity records the lowest. In the case of applying IMSB, an infant agent acquires the same meaning if they know the parent's utterance, and an infant agent acquires a pair which has a more similar meaning and utterance to that they already knows if they does not know the parent's utterance. That the IMSB agent selects a similar meaning, makes the relationship between a meaning and an utterance; the more similar the utterances are, the more similar the meanings are. Therefore, the IMSB agent makes his/her language to be generalized easily over generations. In terms of IMSB,

there is a high probability that generalization of language happens. Notwithstanding, in all trials, an agent can not necessarily generate an utterance for all meanings. That expressivity is around 70 % in Figure 3.14 represents this.

From Figure 3.15, we observed that the number of grammar rules is almost impartial with each strategy.

In the symmetry bias (the both of PMSB and IMSB), the infant agent cannot appropriate some meanings for an utterance. Therefore, number of grammar rules as results of the symmetry bias is less than one as Random strategy, because the same rules with other one is not counted.

From Figure 3.16, we can observe that the IMSB agent's language distance is the smallest, *i.e.*, the grammar of the parent and infant agent with IMSB is the most similar in the three strategies. The PMSB agent's language distance is almost the same as the Random agent's one.

Language distance represents the change from language of the parent to that of the infant, and, it is considered that the languages in Random and that in PMSB strategy differ much. The reason of this is confirmed by unimproved expressivity over generations. After an infant inherited a parent's grammar, an infant's language becomes more compositional than parent's language. Therefore, in Random and PMSB, an infant does not inherited a parent's language, and, this means that language changes significantly.

Also, language distance of Random and PMSB are almost the same because each strategy of them has its own advantage. Advantage of PMSB is to appropriate a meaning more correctly than Random, by the symmetry bias. On the other hand, an advantage of Random is to acquire more expressive language than PMSB. In the case of IMSB, an infant selects the most similar utterance to what they already knows for an received utterance, and, selects a most similar meaning toward a meaning of the most similar utterance, from the presented meanings. Therefore, by IMSB, the agent's language acquires the feature that the more similar meanings to each other represent the more similar utterances to each other. This indicates that an infant inherited the information of similarity of utterances and meanings from a parent. So, an infant inherited a parent's language, and, the language distance becomes small.

From Figure 3.17, we can observe that the IMSB agent's ratio of correct meanings is the highest, and the Random agent's one is the lowest. Through generations, the tendency which indicates increasing of ratio of correct meaning is not observed, however, the IMSB agent's the ratio of correct meanings increases.

In the case of IMSB, an agent's language changes to the language which the more similar meanings to each other represents the more similar utterances to each other. In IMSB, we observed the relationship between the similarity of utterances and that of meanings, over generations. Therefore, IMSB inference becomes more rational over generations. In PMSB, when an infant selects a meaning correctly at the first time, it selects a meaning correctly after the second time because PMSB gives the perfect symmetric relation between a meaning and an utterance. On the other hand, when an infant does not select a meaning correctly at the first time, it looks like that this inference has a bad influence on the ratio of correct meanings, but in fact, it is not a disadvantage. In the case of that there is a no meaning which an infant selected at the first time in presented meanings, it selects a meaning randomly. If the meaning which an infant selected at the first time is not the same meaning as parent's one, the chances that the presented meanings include the wrong meaning at the first time are low, when

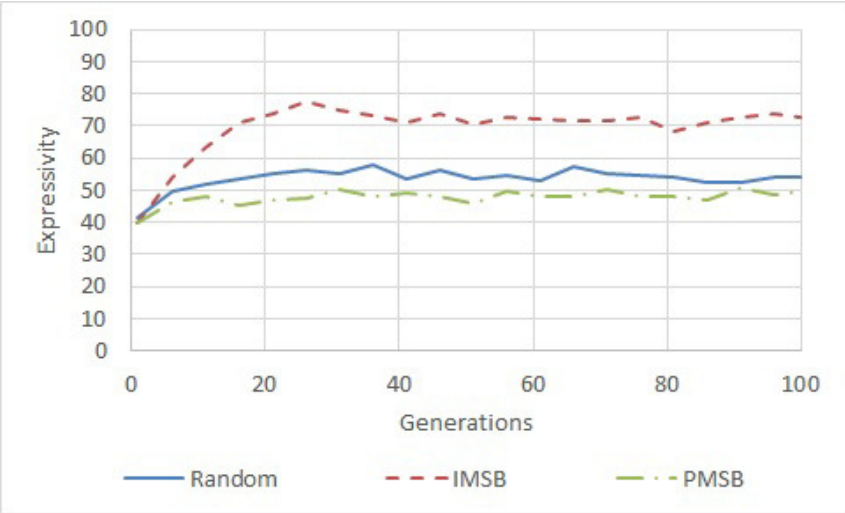


Figure 3.14: MSILM Expressivity

an infant received the same utterance after the second time. An infant received the same utterance after the second time when the parent generates based on the same meaning after the second time. The probability that the presented meanings include the meaning and the wrong meaning, is $\frac{1}{{}_{100}C_2} = \frac{1}{50 \times 99} = 0.00020202$. Consequently, PMSB is the more effective approach as the way to select the meaning than Random in condition of MSILM. Furthermore, IMSB which is the way to change the agent's language, is the more effective approach than PMSB which is the way to select the parent's meaning.

IMSB infant agent selects the parent's selected meaning with more precision through generations, and according to the increasing of expressivity, the language distance decreases. In the case of PMSB, we can observe an improvement of selecting the parent's meaning, however, the PMSB agent's expressivity is lower than the Random agent's one, and the PMSB agent's language distance is tantamount to the Random agent's one.

3.8 Effect of the Symmetry Bias

The aim of this study is to investigate the effect of the symmetry bias on language evolution. We constructed MSILM which incorporated the notion of joint attention frame in Kirby's ILM. First, we experimented simulation in Kirby's ILM, and evaluated the language of each generation by expressivity, number of grammar rules, and language distance.

These evaluation we used is not related to real condition of language evolution, but, we presume these evaluation a clue about feature of language evolution. We assume the results of Kirby's ILM the most efficient result of MSILM because Kirby's ILM is equivalent to a situation that an infant selects the parent's meaning every time in MSILM. In the results of Kirby's ILM, the mean value of expressivity, number of grammar rules and language distance are 99.32%, 23.98 and 0.098, respectively. After then, we compared simulation result in three inference strategies which are Random, IMSB and PMSB. IMSB and PMSB are inference strategies based on the symmetry bias, meanwhile, Random is the strategy of random selection.

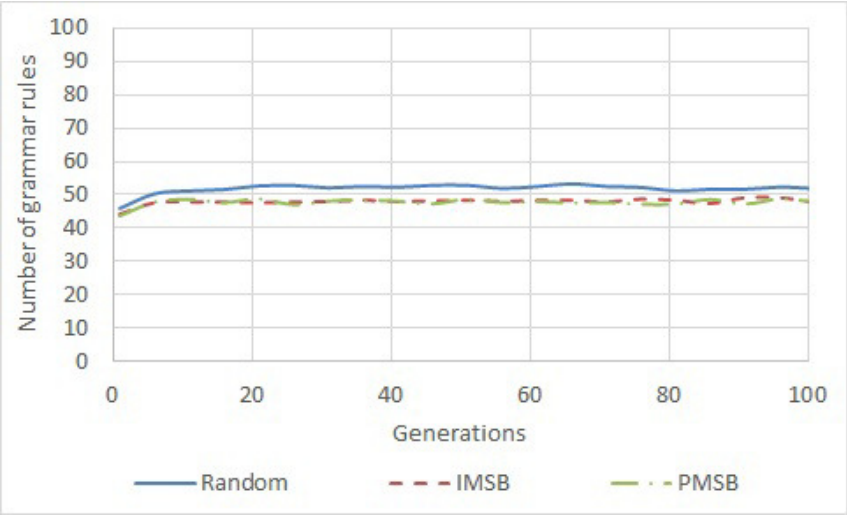


Figure 3.15: MSILM Number of Grammar Rules

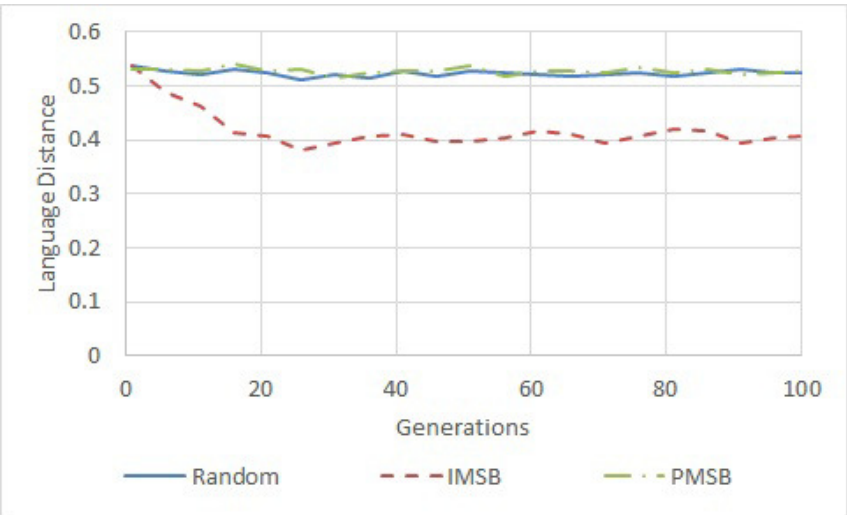


Figure 3.16: MSILM Language Distance

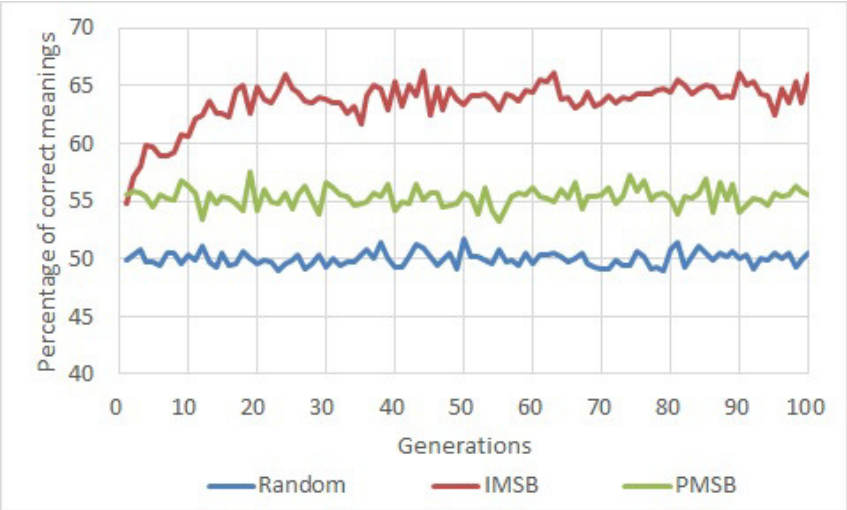


Figure 3.17: MSILM Percentage of Correct Meanings

In the results of the experiments, the effectiveness of IMSB for language evolution in real world is indicated. In this study, we incorporated the problem of joint attention frame with the computational simulation model, and evaluated the way which we assumed it more human-like in MSILM. As the results of the simulations, IMSB shows that an agent's language changes to more learnable language in the past generation. And, the results of PMSB and Random show that an agent's language does not occur language evolution. In the condition with the joint attention frame, *viz.*, is difficult to promote language evolution, an agent's language evolves into learnable and compositional language on an IMSB experiment. Therefore, IMSB accelerates language evolution whereas PMSB disturbs language evolution in spite of some improvement of selecting the parent's meaning.

Depending on the implementation of the symmetry bias, its effect derives different results on language evolution. IMSB showed a positive effect, and PMSB showed a negative effect. On machine learning, IMSB is expected to solve a problem like joint attention frame, *e.g.*, application to human-robot interaction system.

Chapter 4

Extension of Iterated Learning to Music Analysis

4.1 Finding the Grammar of Music

It is generally accepted that the origin of language and that of music are one and the same [45, 1]; we employ throat to utter or to sing, and ears to hear, and furthermore we are said to use the same parts of our brain to parse them. We say birds sing, but this is only a metaphor; in most of the species among birds only male birds utter languages to woo female birds to bear descendants

Thus far, many linguists and musicologists, as well as computer scientists, have tried to find a grammar structure in music [50, 48, 40, 13, 14]. Some of them are successful enough to analyze certain genre of music, *e.g.*, we can find the rules of chord progression in context-free grammar for early classicist music. Among which, the approach from the GTTM[11, 15, 29, 33] seems worth noting, employing the viewpoint of Chomsky hierarchy of tree structure. However, as many musicologists would agree, the syntactic rule for music is quite loose compared with languages and is easy to change rules. Furthermore, the history of music has devoted to deviate from the old tradition, to tolerate freer framework. Therefore, it seems less fruitful for us to fix a certain class of formal language, which plausibly lies between CFG and CSG (Context-Sensitive Grammar) in Chomsky hierarchy, in music.

In traditional linguistics, the main stream has been set to distinguish the theory of syntax and that of semantics, aside from phonetics and pragmatics. We regard that the discussion on what is the meaning of music is beyond the current scope; instead, we consider that the tree structure owes a part of what we call meaning. Also, we contend that the syntax is the universal generative faculty of human beings, that is the source of expressive power common in language, mathematics, morality, and music.

Cope has discussed in [8, 9] the creativity by computers, in which he has claimed that human is biased to degrade the computer music only because they are composed by the computer. Again, we would avoid the discussion on what is the creativity, but we simply try to make the computer compose music in a quite naïve sense, disregarding its quality.

In this paper, we assume that the progression rules of music are in a subclass of context-free language, and we let computers find them autonomously. We employ the Iterated Learning Model (ILM) by Simon Kirby[30]. The research questions of us are two-fold; the first is if the computer can find a music knowledge that is common to us without listening

to music. The second is if the computers can compose music independently of the music knowledge, that is familiar to us.

4.2 Music Analysis Iterated Learning Model

A model which we propose has two important features. One is importing categorial level into the model. This is to express categorial equivalence between different meanings like hyponymy and hypernymy. The other is that there is no restriction on the number of arguments and variety of meanings. Number of arguments and variety of meanings are dynamically increased by learning of input data. In this model, a meaning is defined by agent's self. These features implement an unlimited hierarchical compositional meaning which has recursion. It is different from ILM which is using predefined meanings (PAS).

Categorial Level

We employ a set of attributes as an interpretation of a meaning "${m_i}^{\mathcal{I}}$" as follows.

$$\text{Ex. } {m_i}^{\mathcal{I}} = \{\langle measure \rangle, \langle I/F \rangle, \langle F \rangle\}$$

This interpretation means that m_i is a measure in a piece of music which has a chord "F" and a chord with represented by tonal "I/F" ("m_i is a $\langle measure \rangle$" $\wedge$ "m_i is a $\langle F \rangle$" $\wedge$ "m_i is a $\langle I/F \rangle$").

We define three operations $(\cap, \cup, -)$ for interpretation of a meaning.

$$\begin{aligned} {m_i}^{\mathcal{I}} \cap {m_j}^{\mathcal{I}} &= \left\{x | x \in {m_i}^{\mathcal{I}} \wedge x \in {m_j}^{\mathcal{I}}\right\} \\ {m_i}^{\mathcal{I}} \cup {m_j}^{\mathcal{I}} &= \left\{x | x \in {m_i}^{\mathcal{I}} \vee x \in {m_j}^{\mathcal{I}}\right\} \\ {m_i}^{\mathcal{I}} - {m_j}^{\mathcal{I}} &= \left\{x | x \in {m_i}^{\mathcal{I}} \wedge x \notin {m_j}^{\mathcal{I}}\right\} \end{aligned}$$

Operation $\cap$, $\cup$ are the same as set theory. Operation '$-$' is a calculation of difference, and explained by $A \backslash B$ in set theory ($A = {m_i}^{\mathcal{I}}, B = {m_j}^{\mathcal{I}}$). The interpretation of a new meaning is defined by these operation in learning.

Extension of Expression of Meanings

In ILM, a predefined meaning formed into PAS enables to map between a meaning and a symbol string in knowledge. We define syntax for a meaning to express unlimited meaning as follows.

"$p(\epsilon)$" and "$p(args, \epsilon)$" is represented by "p" and "$p(args)$" as an abbreviation respectively.

$$\begin{aligned} \phi &::= p(args) \\ args &::= \phi \mid \phi, args \mid \epsilon \\ & \quad\quad \text{(Comma“,” is a part of syntax.)} \end{aligned}$$

Ex. $m_1(m_2(m_3, m_4(m_5), m_6), m_7(m_8))$
p is an atomic meaning.

$$\begin{aligned} \phi' &::= p(args') \\ args' &::= x' \mid x', args' \end{aligned}$$

x is a variable.

We call ϕ a meaning and ϕ represents a combination of grammar rules and categorial meaning. ϕ' represents a variable expression in a rule. Each variable matches ϕ.

A size of meaning sequence $|\phi|$ is defined as follows.

$$\begin{aligned} \text{Ex.} \quad & |\phi| = |p(args)| = |p(\phi, \cdots)| = \langle \text{number of } \phi \rangle \\ & |m_1(m_2(m_3, m_4(m_5), m_6), m_7(m_8))| = 2 \\ & |m_2(m_3, m_4(m_5), m_6)| = 3 \end{aligned}$$

A size of ϕ is defined by learning, and possible to be any number. It implies that the structure of a meaning is un-predefined in our model.

A predefined meaning in ILM is represented by an extended expression in our model (4.1). In the following equation, m_i denotes unclear meanings of sentence and structure of PAS (The PAS needs three arguments.).

$$like(mary, john) \subseteq m_i(like, mary, john) \tag{4.1}$$

Application of Extensions to LCFG

We employed a categorial level and unlimited recursion in meaning structure. The grammar specification is as follows.

Extended Labeled Context Free Grammar G'

$$
\begin{array}{rcl}
G' & = & (N, T, M_a, V_a, P, S) \\
N & : & \text{Non-terminal symbols} \\
T & : & \text{Terminal symbols} \\
M_a & : & \text{Atomic meanings} \\
V_a & : & \text{Atomic variables} \\
M & : & \text{Meanings formed into } \phi',\ p \in M_a, x \in V_a \\
P & : & \text{A set of rules} \\
 & & \alpha \to \beta\ \ (\alpha \in N \times M, \beta \in ((N \times V_a) \cup T)^+) \\
S & : & \text{Start non-terminal symbol } S \in N
\end{array}
$$

Number of variables in α must be number of $(N \times V)$ in β.

M_a and V_a are not predefined and what an agent creates through learning. Therefore, size of M is possible to be infinite because syntax ϕ has recursion.

Extension of Learning

In ILM, after an agent has accepted a certain number of utterances, she tries to build a new set of generation rules. There are three operations *chunk*, *merge*, and *replace* based on the generalized method. In the conventional ILM, a rule is applicable when a pair of meaning and a part of a sequence of symbols in utterance matches. However, this condition is still ambiguous because the same local sequence of symbols may own different meanings. In our model, we define the applicable conditions by two interpretation of a meaning. Extended LCFG rule in agent's knowledge has the unique atomic meaning respectively. Thus, although every rule can possess a different meaning from others, these difference may not reflect the different sets of attributes. In our formalism, therefore, we can understand the sameness of the attributes by the interpretation.

- Chunk

We supposes the conditions on chunk as follows.

$$
\begin{gathered}
{m_a}^{\mathcal{I}} \cap {m_b}^{\mathcal{I}} \neq \emptyset \\
{m_1}^{\mathcal{I}} = {m_a}^{\mathcal{I}} \cap {m_b}^{\mathcal{I}} \\
{m_2}^{\mathcal{I}} = {m_a}^{\mathcal{I}} - {m_1}^{\mathcal{I}} \\
{m_3}^{\mathcal{I}} = {m_b}^{\mathcal{I}} - {m_1}^{\mathcal{I}}
\end{gathered}
$$

  - For two no compositional rules

$$
\begin{gathered}
S/m_a \to CDCCDEF^{\#}GC \\
S/m_b \to CDCDEEF^{\#}GC \\
\downarrow chunk \\
S/m_1\,(x_1) \to CDC\ \ N_1/x_1\ \ EF^{\#}GC \\
N_1/m_2 \to CD \\
N_1/m_3 \to DE
\end{gathered}
$$

– For compositional rule and no compositional one

$$\begin{aligned}
&S/m_b\,(x_1) \to C^{\#}D \;\; N_1/x_1 \;\; F^{\#}GC^{\#} \\
&S/m_a \to C^{\#}DCDEF^{\#}GC^{\#} \\
&\qquad\qquad \downarrow chunk \\
&S/m_1, x_1 \to C^{\#}D \;\; N_1/x_1 \;\; F^{\#}GC^{\#} \\
&N_1/m_2 \to C^{\#}DE
\end{aligned}$$

– The diff of generalized strings has some non-terminal symbol.

$$\begin{aligned}
&S/m_a\,(x_1, x_2) \to CD \;\; N_1/x_2 \;\; N_2/x_1 \\
&S/m_b\,(x_1, x_2) \to CDCDE \;\; N_3/x_1 \\
&\qquad\qquad \downarrow chunk \\
&S/m_1\,(x_1) \to CD \;\; N_4/x_1 \\
&N_4/m_2\,(x_1, x_2) \to N_1/x_2 \;\; N_2/x_1 \\
&N_4/m_3\,(x_1) \to CDE \;\; N_3/x_1
\end{aligned}$$

- Merge

We suppose the conditions on merge as follows.

$$\left({m_i}^{\mathcal{I}} \supseteq {m_j}^{\mathcal{I}}\right) \vee \left({m_j}^{\mathcal{I}} \supseteq {m_i}^{\mathcal{I}}\right)$$
$${m_1}^{\mathcal{I}} = {m_a}^{\mathcal{I}} \cup {m_b}^{\mathcal{I}}$$

– Merge two rules

$$\begin{aligned}
&S/m_2\,(x_1, x_2) \to N_1/x_1 \;\; E\flat FA \;\; N_2/x_2 \\
&N_1/m_a \to CD \\
&N_1/m_3 \to AB\flat \\
&N_2/m_b \to CD \\
&\qquad\qquad \downarrow merge \\
&S/m_2\,(x_1, x_2) \to N_3/x_1 \;\; E\flat FA \;\; N_3/x_2 \\
&N_3/m_1 \to CD \\
&N_3/m_3 \to AB\flat
\end{aligned}$$

– For start rule and not start rule

$$\begin{aligned}
&S/m_a\,(x_1, x_2) \to N_1/x_1 \;\; E\flat FA \;\; N_2/x_2 \\
&N_1/m_b\,(x_1, x_2) \to N_1/x_1 \;\; E\flat FA \;\; N_2/x_2 \\
&\qquad\qquad \downarrow merge \\
&S/m_1\,(x_1, x_2) \to N_1/x_1 \;\; E\flat FA \;\; N_2/x_2
\end{aligned}$$

- Replace

We suppose the conditions on replace as follows.

$${m_i}^{\mathcal{I}} \supseteq {m_j}^{\mathcal{I}}$$
$${m_1}^{\mathcal{I}} = {m_a}^{\mathcal{I}} - {m_b}^{\mathcal{I}}$$

– Replace a part of string in left side to the other rule.

$$
\begin{aligned}
&S/m_a \rightarrow F^{\#}GCEDAF^{\#} \\
&N_1/m_b \rightarrow ED \\
&\qquad\qquad \downarrow replace \\
&S/m_1\,(x_1) \rightarrow F^{\#}GC \;\; N_1/x_1 \;\; AF^{\#} \\
&N_1/m_b \rightarrow ED
\end{aligned}
$$

– For two compositional rules

$$
\begin{aligned}
&S/m_a\,(x_1) \rightarrow F^{\#}GC \;\; N_1/x_1 \;\; AF^{\#} \\
&N_2/m_b\,(x_2) \rightarrow F^{\#}GC \;\; N_1/x_2 \\
&\qquad\qquad \downarrow replace \\
&S/m_1\,(x_1) \rightarrow N_2/x_1 \;\; AF^{\#} \\
&N_2/m_b\,(x_2) \rightarrow F^{\#}GC \;\; N_2/x_2
\end{aligned}
$$

4.3 Subjective Meaning

In our model, a unique label has each meaning independently. Therefore, when given a label, a composition of music is represented by a straightforward sequence; the composite label structure represents the tree directly, being different from other generative models like deep learning or probabilistic CFG.

4.4 Procedural Generation of Music

In the following example, we can easily understand that m_1 is the top node and has two branches of m_2 and $m_4(m_1(\cdots))$.

- Target label: $m_1\,(m_2, m_4\,(m_1\,(m_3, m_5\,(m_2))))$

$$
\begin{aligned}
&\qquad\text{Agent's Knowledge} \\
&S/m_1\,(x_1, x_2) \rightarrow N_1/x_1 \;\; N_2/x_2 \\
&N_1/m_2 \rightarrow F^{\#}GD \\
&N_1/m_3 \rightarrow CDC \\
&N_2/m_4\,(x_3) \rightarrow F^{\#}GC \;\; S/x_3 \\
&N_2/m_5\,(x_4) \rightarrow CD \;\; N_1/x_4
\end{aligned}
$$

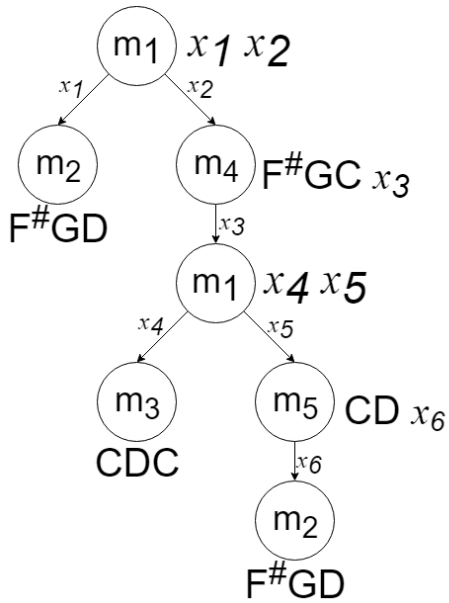


Figure 4.1: Representation of parse tree in the case of section 4.4

$$
\begin{aligned}
&S/m_1\,(x_1, x_2) \rightarrow N_1/x_1 \;\; N_2/x_2 \\
&S/m_1\,(m_2, x_2) \rightarrow F^{\#}GD \;\; N_2/x_2 \\
&S/m_1\,(m_2, m_4\,(x_3)) \rightarrow F^{\#}GDF^{\#}GC \;\; S/x_3 \\
&S/m_1\,(m_2, m_4\,(m_1\,(x_4, x_5))) \\
&\rightarrow F^{\#}GDF^{\#}GC \;\; N_1/x_4 \;\; N_2/x_5 \\
&S/m_1\,(m_2, m_4\,(m_1\,(m_3, x_5))) \\
&\rightarrow F^{\#}GDF^{\#}GCCDC \;\; N_2/x_5 \\
&S/m_1\,(m_2, m_4\,(m_1\,(m_3, m_5\,(x_6)))) \\
&\rightarrow F^{\#}GDF^{\#}GCCDCCD \;\; N_1/x_6 \\
&S/m_1\,(m_2, m_4\,(m_1\,(m_3, m_5\,(m_2)))) \\
&\rightarrow F^{\#}GDF^{\#}GCCDCCDF^{\#}GD
\end{aligned}
$$

4.5 Parsing a Generated Sequence

We can calculate a tree structure if given a set of rules as expression of an utterance. For example, when we get a set of rules as section 4.4, a tree is calculated as Figure 4.1.

4.6 Evaluation Method for Syntactic Structure

In natural language processing, when we evaluate the validity of the process, we compose a parse tree of sentences and investigate if the tree reflects the adequate structure or not because we believe the tree reflects the meaning of a sentence. In the similar way, we consider the adequacy of the tree structure of music piece even though music does not own explicit meaning like natural language.

However, in general, we may not be able to enumerate all the probable trees for a given music piece since our LCFG has a property of recursion and the combination of applicable generation rules can be infinite. In order to avoid this combinatorial explosion of computational complexity of $O(2^n)$, we restrict the number of probable trees to one for each music piece.

Based on the newly acquired generative rules, we can compose the exactly same music since every generation process can be preserved in those rules. In addition, the new rules compose a new piece of music. If we can record the history of the generation process, we can detect how the new rules were applied differently from the original one; that is, the original piece is decomposed by the rules and the each part is recomposed in a different way. Then, we give an analysis system of the logs of composition.

A rule of agent's learned knowledge has a condition that the order of variables in the left-hand side is the same with one in the right-hand side. m_a and m_b are atomic meanings before apply a learning method, m_1, m_2 and m_3 are used in new rules. We can denote a rewriting rule as follows.

$$
\begin{gathered}
args : x, y, z, x', z' \\
m_a, m_b, m_1, m_2, m_3 \in M
\end{gathered}
$$

- Chunk

$$
\begin{gathered}
m_a(x, y, z) \Rightarrow m_1(x, m_2(y), z) \\
m_b(x', y, z') \Rightarrow m_1(x', m_3(y), z')
\end{gathered}
$$

- Merge

$$
\begin{gathered}
m_a \Rightarrow m_1 \\
m_b \Rightarrow m_1
\end{gathered}
$$

- Replace

$$
m_a(x, y, z) \Rightarrow m_1(x, m_2(y), z)
$$

4.7 Experimental Result

We experimented to find grammar in the 25 études of Burgmüller (Figure 4.2). At first, we divide each piece beat by beat, and we have grouped a set of notes in one beat. Then, we translate a piece of music to a sequence of sets of simultaneously sounding notes, and, an agent tries to find generative rules in these pieces.

In this translation process we have embedded the notion of *measure*. We have explicitly defined that the sequence of a certain number of beats composes a measure (Figure 4.3), giving a single atomic label on each as terminal symbols. Therefore, each rule which represents a measure has four terminal symbols if the piece has four beats music.

A whole piece of music is represented by a rule with the start non-terminal symbol. The rule has a string of non-terminal symbols in the right-hand side.

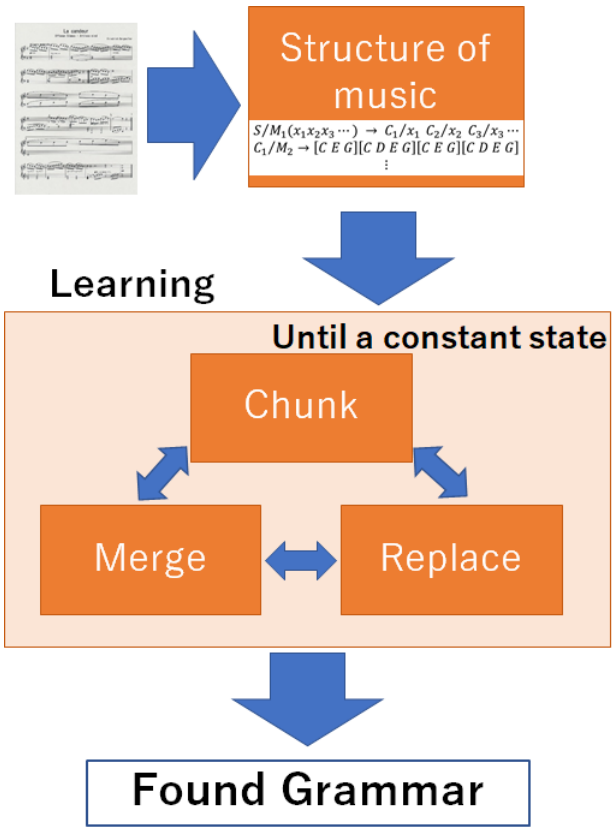
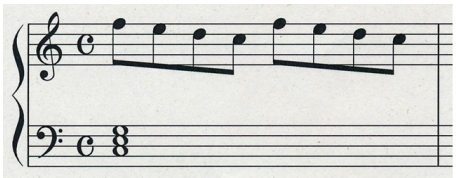


Figure 4.2: Illustration of the Simulation



$C_1/M_1 \rightarrow [C\ E\ F\ G]\ [C\ D\ E\ G]\ [C\ E\ F\ G]\ [C\ D\ E\ G]$

Figure 4.3: Translation of Measure to Symbol String

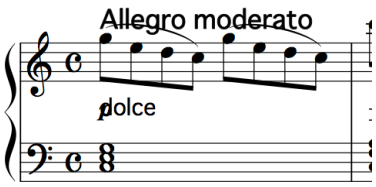


Figure 4.4: Part of a Piece: Recursion to a stable chord in the 25 études of Burgmüller

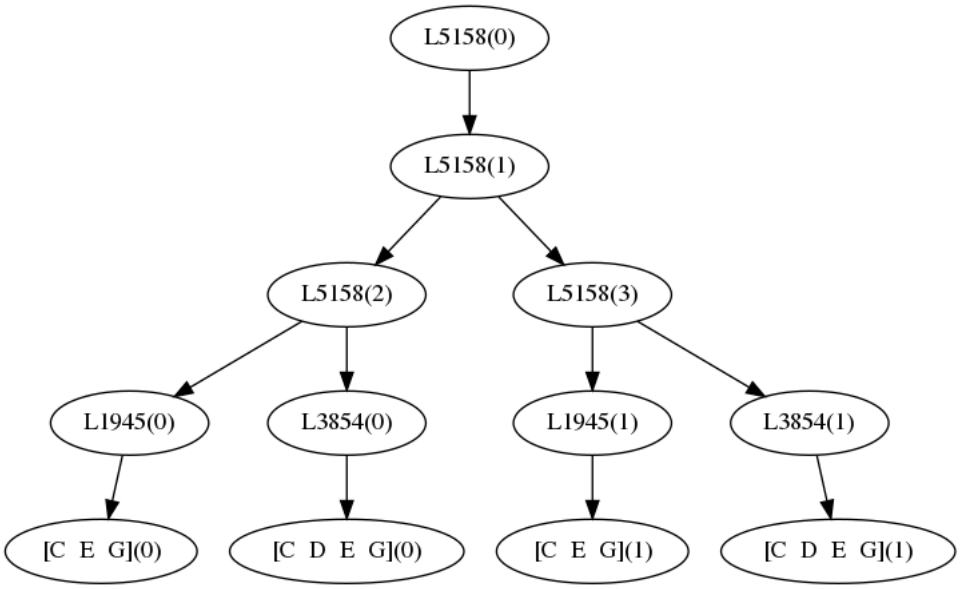


Figure 4.5: Parse Tree: Recursion to a stable chord in the 25 études of Burgmüller

4.7.1 Detection of Music Knowledge

We got two salient results as follows by analyzing the parse tree of existing scores using agent's knowledge.

Progression to stable chord

In Figure 4.5, we can find the solution to a harmonic chord from a set of dissonant notes. Figure 4.4 shows the first bar of *Le Candeur* of Burgmüller. The second beat includes a passing note of D, but is resolved to C in the third beat. We can find a corresponding generation rule that represents a progression from the dissonant to consonant chord.

Phrase detection

In early classicist music, the music phrase and motif is articulated by 4 bars, 8 bars, and 16 bars. *La Candeur* also suggests us the phrasing by 8 bars. Our rule has found that the articulation after the 17-th bar after the first repetition consisting of eight-bars, though we have not explicitly put the punctuation here (Figure 4.6, 4.7).

Figure 4.6: Part of Piece: Phrase detection in the 25 études of Burgmüller

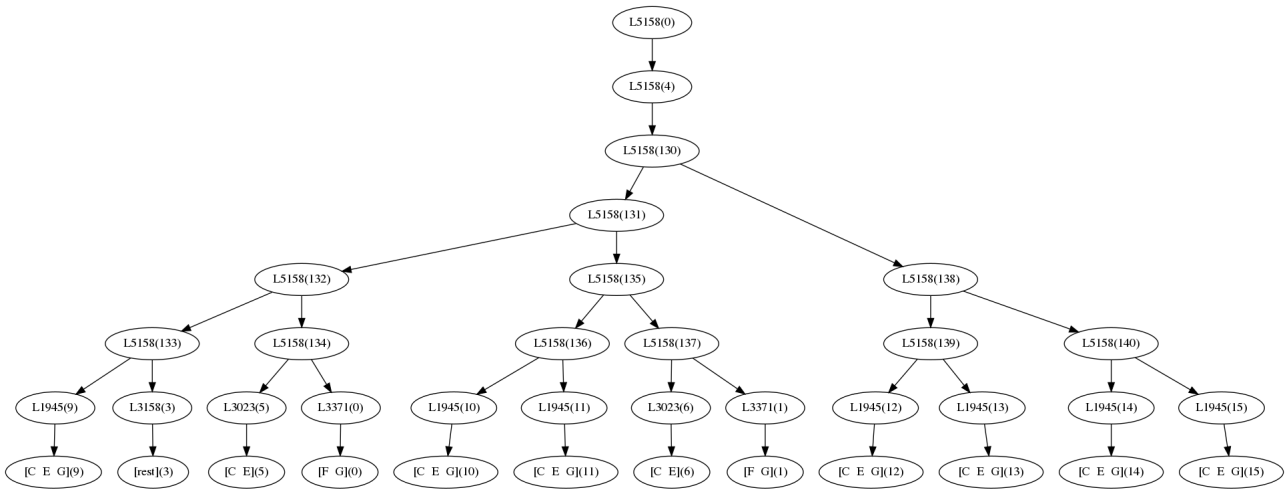


Figure 4.7: Parse Tree: Phrase detection in the 25 études of Burgmüller

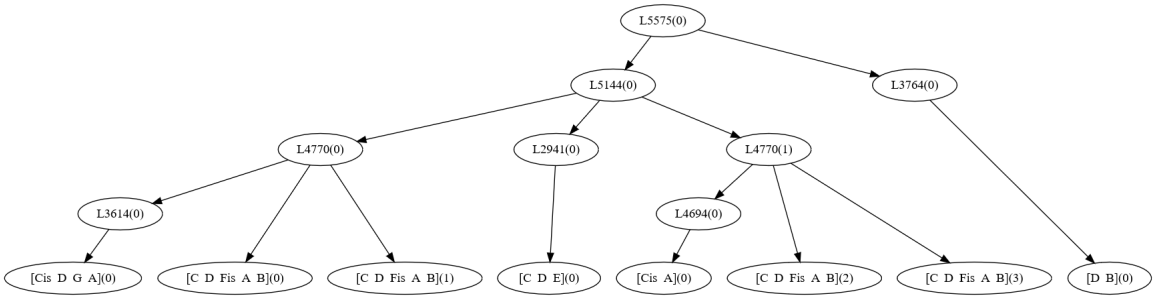


Figure 4.8: Parce Tree: Creation of a piece

4.7.2 Composition of Music Score

After learning the generative rules, an agent can compose a piece of music. For an example, Figure 4.8 is created by a learned agent.

4.8 Usefulness of Self-organizing Meanings

In this research, we have modified the ILM by Kirby, and have applied it to grammar detection in music. We have chopped off the 25 études of Burgmüller by beat, and have found the statistically plausible connections between them. Although many of those nodes in tree, that are the categories of the syntax, seem redundant and futile, we still found that some of them reflected the probable progression, which well matched with our human intuition, *e.g.*, the progression from a set of dissonant notes including suspension or appoggiatura to a chordal set.

This experiment has several virtues compared with other grammar-based formalism for music. One is that we do not need to provide a dictionary beforehand. The other is that we can exclude the human-biased intuition, which had hindered the definition of creativity.

Chapter 5

Conclusions and Further Directions

5.1 Achievement

We showed the evaluation of the language distance between two language knowledge, and verified the language distance in ILM. We have evaluated the language distance by the mean of expressivity and the number of rules. And furthermore, We would have accelerated the simulation in ILM using the clipped utterances. The evaluations with the clipped utterances were the same as one without it.

We investigated language evolution in ambiguous environment using symmetry bias. According to our result, the aspect of linguistic evolution depends on the implementation of symmetry bias. PMSB which is one of implementation of symmetry bias disturbs language evolution, moreover IMSB which is another implementation of symmetry bias accelerates language evolution.

We applied the ILM to music to analyze music scores, employing the methodology to self-organize a meaning. We obtained the parse trees of music score, and analyzed the parse trees manually. As a result of the experiment, we have found several meaningful nodes (rules); a progression from dissonant chord to harmonic chord, or dominant motion (V-I), but still there are many meaningless rules.

5.2 Further Directions

We would like to claim that the three operations of ILM, viz., chunk, merge, and replace, are quite human-like language process. However, in order to verify this, we need to conduct laboratory experiments, with a more realistic situation.

We extended the ILM to ambiguous environment such as music scores. A further direction of our model will be machine learning for general symbolic strings, e.g., the automatic generation for a parser of programming languages, the analysis of undeciphered documents like Voynich manuscript[19]. In our model of Chapter 4, we do not need the predefined meanings and meaning structure, and they will be self-organized corresponding to the learned results. The model can find the structure of syntax and meanings, independent of the parameters given by the experimenter.

Appendix A

The Implementation of Chunk, Merge, Replace (C++)

```
bool Knowledge::learning_loop(void) {
  bool flag = true;
  std::array<int, LEARNING_TYPE::ALL_METHOD> ar, tmp;
  std::iota(std::begin(ar), std::end(ar), 0);
  if (LOGGING_FLAG) LogBox::push_log("\n\n!!LEARNING!!");
  while (flag) {
    flag = false;
    tmp = ar;
    std::shuffle(std::begin(tmp), std::end(tmp),
                 MT19937::igen);
    std::for_each(std::begin(tmp), std::end(tmp),
                  [this, &flag](int i) {
                    switch (i) {
                      case LEARNING_TYPE::CHUNK: {
                        flag = chunk() || flag;
                        break;
                      }
                      case LEARNING_TYPE::MERGE: {
                        if ((flag = merge() || flag)) {
                          unique(input_box);
                        }
                        break;
                      }
                      case LEARNING_TYPE::REPLACE: {
                        flag = replace() || flag;
                        break;
                      }
                      default:
                        std::cerr << "Learning Type Error"
                                  << std::endl;
                        exit(1);
                    }
```

```
                    });

      if (LOGGING_FLAG) {
        LogBox::refresh_log();
      }
    }

    if (LOGGING_FLAG) {
      LogBox::push_log("\n\n!!LEARNING FIN!!");
      LogBox::push_log("Knowledge Size: " +
                        std::to_string(ruleDB.size() +
                                       box_buffer.size() +
                                       input_box.size()));
    }

    send_db(input_box);
    unique(ruleDB);

    if (box_buffer.size() != 0) {
      std::cerr << "Still remaining data(box_buffer): "
                << box_buffer.size() << std::endl;
      exit(1);
    }
    if (input_box.size() != 0) {
      std::cerr << "Still remaining data(input_box): "
                << input_box.size() << std::endl;
      exit(1);
    }
    build_word_index();

    return true;
  }

  bool Knowledge::chunk(void) {
    std::shuffle(std::begin(input_box), std::end(input_box),
                 MT19937::igen);
    bool is_chunked;
    RuleDBType::iterator it;
    while ((it = std::begin(input_box)) !=
           std::end(input_box)) {
      Rule r = *it;
      input_box.erase(it);
      is_chunked = chunking_loop(r, ruleDB);
      if (!is_chunked)
        is_chunked =
            is_chunked || chunking_loop(r, box_buffer);
      if (is_chunked) {
```

```
      break;
    } else {
      box_buffer.push_back(r);
    }
  }
  send_box(box_buffer);

  return is_chunked;
}

bool Knowledge::chunking_loop(Rule &unchecked_sent,
                              RuleDBType &checked_rules) {
  RuleDBType buffer;
  bool is_chunked = false;
  auto it = std::begin(checked_rules);
  while (!is_chunked && it != std::end(checked_rules)) {
    Rule r = *it;
    buffer.clear();
    buffer = chunking(unchecked_sent, r);
    if (buffer.size() == 0) {
      it++;
    } else {
      if (LOGGING_FLAG) {
        LogBox::push_log("\n-->>CHUNK:");
        LogBox::push_log("**FROM");
        LogBox::push_log(unchecked_sent.to_s());
        LogBox::push_log(r.to_s());
        LogBox::push_log("**TO");
        std::for_each(std::begin(buffer), std::end(buffer),
                      [](Rule &temp) {
                        LogBox::push_log(temp.to_s());
                      });
        LogBox::push_log("<<--CHUNK");
      }
      send_box(buffer);
      checked_rules.erase(it);
      is_chunked = true;
      break;
    }
  }
  return is_chunked;
}

Knowledge::RuleDBType Knowledge::chunking(Rule &src,
                                          Rule &dst) {
  enum CHUNK_TYPE { UNABLE = 0, TYPE1, TYPE2 };
  RuleDBType buf;
```

```
bool multi_cat;
if (src.get_internal().get_cat() !=
    dst.get_internal().get_cat()) {
  if (intention.chunk_equal(
          src.get_internal().get_base(),
          dst.get_internal().get_base())) {
    multi_cat = true;
  } else {
    return buf;
  }
}

int fmatch_length = 0;
for (auto src_it = src.get_external().begin(),
          dst_it = dst.get_external().begin();
     src_it != src.get_external().end() &&
     dst_it != dst.get_external().end() &&
     *src_it == *dst_it;
     fmatch_length++, src_it++, dst_it++) {
}

int rmatch_length = 0;
for (auto src_rit = src.get_external().rbegin(),
          dst_rit = dst.get_external().rbegin();
     src_rit != src.get_external().rend() &&
     dst_rit != dst.get_external().rend() &&
     *src_rit == *dst_rit;
     rmatch_length++, src_rit++, dst_rit++) {
}

if (fmatch_length + rmatch_length == 0 ||
    fmatch_length + rmatch_length >=
        src.get_external().size() ||
    fmatch_length + rmatch_length >=
        dst.get_external().size())
  return buf;

CHUNK_TYPE chunk_type = UNABLE;
Rule base, targ;
std::vector<SymbolElement> noun1_ex, noun2_ex;
std::copy(
    std::next(src.get_external().begin(), fmatch_length),
    std::prev(src.get_external().end(), rmatch_length),
    std::back_inserter(noun1_ex));
std::copy(
    std::next(dst.get_external().begin(), fmatch_length),
```

```
    std::prev(dst.get_external().end(), rmatch_length),
    std::back_inserter(noun2_ex));
if (noun1_ex.size() == 1 &&
    noun1_ex.front().type() == ELEM_TYPE::NT_TYPE) {
  if (noun2_ex.size() == 1 &&
      noun2_ex.front().type() == ELEM_TYPE::NT_TYPE) {
    return buf;
  } else {
    //        base=src, targ=dst
    base = src;
    targ = dst;
    chunk_type = TYPE2;
  }
} else {
  if (noun2_ex.size() == 1 &&
      noun2_ex.front().type() == ELEM_TYPE::NT_TYPE) {
    base = dst;
    targ = src;
    chunk_type = TYPE2;
    noun1_ex.swap(noun2_ex);
  } else {
    base = src;
    targ = dst;
    chunk_type = TYPE1;
  }
}
switch (chunk_type) {
  case TYPE1: {
    int new_cat_id;          // category
    int new_var_id;          // variable
    int new_sent_ind_id1;    // id
    int new_sent_ind_id2;    // id
    int new_ind_id1;         // id
    int new_ind_id2;         // id

    // generate
    new_cat_id = cat_indexer.generate();
    new_var_id = var_indexer.generate();
    new_sent_ind_id1 = ind_indexer.generate();
    new_sent_ind_id2 = ind_indexer.generate();
    new_ind_id1 = ind_indexer.generate();
    new_ind_id2 = ind_indexer.generate();

    std::list<MeaningElement> var_vector1, var_vector2;

    std::for_each(
        std::begin(noun1_ex), std::end(noun1_ex),
```

```
    [&d_size, &var_vector1](SymbolElement &se) {
      if (se.type() == ELEM_TYPE::NT_TYPE) {
        MeaningElement mel =
            se.get<RightNonterminal>().get_var();
        var_vector1.push_back(mel);
        d_size++;
      }
    });

std::for_each(
    std::begin(noun2_ex), std::end(noun2_ex),
    [&e_size, &var_vector2](SymbolElement &se) {
      if (se.type() == ELEM_TYPE::NT_TYPE) {
        MeaningElement mel =
            se.get<RightNonterminal>().get_var();
        var_vector2.push_back(mel);
        e_size++;
      }
    });

// noun1
Rule noun1(LeftNonterminal(Category(new_cat_id),
                           Meaning(AMean(new_ind_id1),
                                   var_vector1)),
           noun1_ex);
// noun2
Rule noun2(LeftNonterminal(Category(new_cat_id),
                           Meaning(AMean(new_ind_id2),
                                   var_vector2)),
           noun2_ex);

// insert
std::for_each(
    std::begin(targ.get_external()),
    std::next(std::begin(targ.get_external()),
              fmatch_length),
    [&in_pos](SymbolElement &sel) {
      if (sel.type() == ELEM_TYPE::NT_TYPE) {
        in_pos++;
      }
    });
in_pos++;

Meaning new_meaning =
    base.get_internal().get_means().replaced(
        in_pos, d_size, Variable(new_var_id));
new_meaning = new_meaning.replaced(
```

```
        0, 1, Meaning(AMean(new_sent_ind_id1)));
    std::vector<SymbolElement> vec_sel;
    std::copy_n(std::begin(base.get_external()),
                fmatch_length,
                std::back_inserter(vec_sel));
    vec_sel.push_back(RightNonterminal(
        Category(new_cat_id), Variable(new_var_id)));
    std::copy(std::prev(std::end(base.get_external()),
                        rmatch_length),
              std::end(base.get_external()),
              std::back_inserter(vec_sel));

    Rule sent{LeftNonterminal{
                  Category{base.get_internal().get_cat()},
                  new_meaning},
              vec_sel};
    Rule sent2;
    if (multi_cat) {
      sent2 = Rule{
          LeftNonterminal{
              Category{targ.get_internal().get_cat()},
              new_meaning.replaced(
                  0, 1,
                  Meaning(AMean(new_sent_ind_id2)))},
          vec_sel};
    }
    buf.push_back(sent);
    if (multi_cat) {
      buf.push_back(sent2);
    }
    buf.push_back(noun1);
    buf.push_back(noun2);

    intention.chunk(
        base.get_internal().get_means().get_base(),
        targ.get_internal().get_means().get_base(),
        sent.get_internal().get_means().get_base(),
        sent2.get_internal().get_means().get_base(),
        noun1.get_internal().get_means().get_base(),
        noun2.get_internal().get_means().get_base(),
        in_pos, d_size, e_size, chunk_type);

    break;
  }

  case TYPE2: {
    int new_ind_id_targ;
```

```
int new_sent_ind_id1;
int new_sent_ind_id2;

// generate
new_sent_ind_id1 = ind_indexer.generate();
new_sent_ind_id2 = ind_indexer.generate();
new_ind_id_targ = ind_indexer.generate();

std::list<MeaningElement> var_vector;

// var_vector
std::for_each(
    std::begin(noun2_ex), std::end(noun2_ex),
    [&d_size, &var_vector](SymbolElement &se) {
      if (se.type() == ELEM_TYPE::NT_TYPE) {
        var_vector.push_back(Variable(
            se.get<RightNonterminal>().get_var()));
        d_size++;
      }
    });

// noun
Rule noun(
    LeftNonterminal(
        Category{RightNonterminal(
                    noun1_ex.front()
                        .get<RightNonterminal>())
                    .get_cat()},
        Meaning(AMean(new_ind_id_targ), var_vector)),
    noun2_ex);
Rule sent{
    LeftNonterminal(
        Category{base.get_internal().get_cat()},
        Meaning{
            AMean(new_sent_ind_id1),
            base.get_internal().get_followings()}),
    base.get_external()};
Rule sent2;
if (multi_cat) {
  sent2 = Rule{
      LeftNonterminal{
          Category{targ.get_internal().get_cat()},
          Meaning{
              AMean(new_sent_ind_id2),
              sent.get_internal().get_followings()}},
      base.get_external()};
}
```

```
      buf.push_back(sent);
      if (multi_cat) {
        buf.push_back(sent2);
      }
      buf.push_back(noun);

      // insert
      std::for_each(
          std::begin(targ.get_external()),
          std::next(std::begin(targ.get_external()),
                    fmatch_length),
          [&in_pos](SymbolElement &sel) {
            if (sel.type() == ELEM_TYPE::NT_TYPE) {
              in_pos++;
            }
          });
      in_pos++;

      intention.chunk(
          targ.get_internal().get_means().get_base(),
          base.get_internal().get_means().get_base(),
          sent.get_internal().get_means().get_base(),
          sent2.get_internal().get_means().get_base(),
          noun.get_internal().get_means().get_base(),
          AMean(), in_pos, d_size, e_size, chunk_type);

      break;
    }
    default:
      std::cerr << "CHUNK PROC ERROR" << std::endl;
      throw "CHUNK PROC ERROR";
  }

  return buf;
}

bool Knowledge::merge(void) {
  std::shuffle(std::begin(input_box), std::end(input_box),
               MT19937::igen);
  bool is_merged;
  RuleDBType::iterator it;
  while ((it = std::begin(input_box)) !=
         std::end(input_box)) {
    Rule r = *it;
    input_box.erase(it);
    is_merged = merging(r);
    if (is_merged) {
```

```
      break;
    } else {
      box_buffer.push_back(r);
    }
  }
  send_box(box_buffer);

  return is_merged;
}

bool Knowledge::merging(Rule &src) {
  RuleDBType buf, sub_buf;

  std::set<Category> unified_cat;
  std::set<AMean> unified_mean;

  collect_merge(src, input_box, unified_cat, unified_mean);

  collect_merge(src, ruleDB, unified_cat, unified_mean);

  collect_merge(src, box_buffer, unified_cat, unified_mean);

  if (unified_cat.size() == 0 && unified_mean.size() == 0) {
    return false;
  }

  Category base_cat{cat_indexer.generate()};
  AMean base_mean{ind_indexer.generate()};
  unified_cat.insert(src.get_internal().get_cat());
  unified_mean.insert(src.get_internal().get_base());
  buf.push_back(src);

  if (LOGGING_FLAG)
    LogBox::push_log("MEAN base rule " + src.to_s());
  merge_mean_proc_buffer(base_mean, buf, unified_mean);

  if (LOGGING_FLAG)
    LogBox::push_log("MEAN input_box " + src.to_s());
  sub_buf =
      merge_mean_proc(base_mean, input_box, unified_mean);
  std::copy(std::begin(sub_buf), std::end(sub_buf),
            std::back_inserter(buf));
  sub_buf.clear();
  if (LOGGING_FLAG)
    LogBox::push_log("MEAN ruleDB " + src.to_s());
  sub_buf =
      merge_mean_proc(base_mean, ruleDB, unified_mean);
```

```
  std::copy(std::begin(sub_buf), std::end(sub_buf),
            std::back_inserter(buf));
  sub_buf.clear();
  if (LOGGING_FLAG)
    LogBox::push_log("MEAN box_buffer " + src.to_s());
  sub_buf =
      merge_mean_proc(base_mean, box_buffer, unified_mean);
  std::copy(std::begin(sub_buf), std::end(sub_buf),
            std::back_inserter(buf));
  sub_buf.clear();

  intention.merge(src.get_internal().get_base(),
                  unified_mean, base_mean);

  if (LOGGING_FLAG)
    LogBox::push_log("CAT processed about MEAN " +
                     src.to_s());
  merge_cat_proc_buffer(base_cat, buf, unified_cat);

  if (LOGGING_FLAG)
    LogBox::push_log("CAT input_box " + src.to_s());
  sub_buf =
      merge_cat_proc(base_cat, input_box, unified_cat);
  std::copy(std::begin(sub_buf), std::end(sub_buf),
            std::back_inserter(buf));
  sub_buf.clear();
  if (LOGGING_FLAG)
    LogBox::push_log("CAT ruleDB " + src.to_s());
  sub_buf = merge_cat_proc(base_cat, ruleDB, unified_cat);
  std::copy(std::begin(sub_buf), std::end(sub_buf),
            std::back_inserter(buf));
  sub_buf.clear();
  if (LOGGING_FLAG)
    LogBox::push_log("CAT box_buffer " + src.to_s());
  sub_buf =
      merge_cat_proc(base_cat, box_buffer, unified_cat);
  std::copy(std::begin(sub_buf), std::end(sub_buf),
            std::back_inserter(buf));
  sub_buf.clear();
  send_box(buf);

  if (LOGGING_FLAG) {
    LogBox::push_log("<<--MERGE");
  }

  return true;
}
```

```
void Knowledge::collect_merge(
    Rule &src, RuleDBType &rule_db,
    std::set<Category> &unified_cat,
    std::set<AMean> &unified_mean) {
  std::for_each(
      std::begin(rule_db), std::end(rule_db),
      [this, &src, &unified_cat, &unified_mean](Rule &r) {
        if (src.get_external() == r.get_external() &&
            intention.merge_equal(
                src.get_internal().get_base(),
                r.get_internal().get_base())) {
          if (src.get_internal().get_cat() !=
              r.get_internal().get_cat()) {
            unified_cat.insert(
                Category{r.get_internal().get_cat()});
          }
          if (src.get_internal().get_base() !=
              r.get_internal().get_base()) {
            unified_mean.insert(
                AMean{r.get_internal().get_base()});
          }
        }
      });
}

void Knowledge::merge_cat_proc_buffer(
    const Category &base_cat, RuleDBType &buffer,
    std::set<Category> &unified_cat) {
  std::for_each(
      std::begin(buffer), std::end(buffer),
      [&base_cat, &unified_cat](Rule &r) {
        Rule tmp = r;
        bool is_modified = false;
        if (unified_cat.find(r.get_internal().get_cat()) !=
            std::end(unified_cat)) {
          r.get_internal() =
              LeftNonterminal{Category{base_cat},
                              r.get_internal().get_means()};
          is_modified = is_modified || true;
        }
        std::for_each(
            std::begin(r.get_external()),
            std::end(r.get_external()),
            [&base_cat, &unified_cat,
             &is_modified](SymbolElement &sel) {
              if (sel.type() == ELEM_TYPE::NT_TYPE &&
```

```
                    unified_cat.find(
                        sel.get<RightNonterminal>()
                            .get_cat()) !=
                        std::end(unified_cat)) {
                  sel = RightNonterminal{
                      Category{base_cat},
                      sel.get<RightNonterminal>().get_var()};
                  is_modified = true;
                }
              });
          if (is_modified && LOGGING_FLAG) {
            LogBox::push_log("CAT MERGE-> " + tmp.to_s());
            LogBox::push_log("CAT MERGE<- " + r.to_s());
          }
        });
}
Knowledge::RuleDBType Knowledge::merge_cat_proc(
    const Category &base_cat, RuleDBType &DB,
    std::set<Category> &unified_cat) {
  RuleDBType buf, swapDB;
  std::for_each(
      std::begin(DB), std::end(DB),
      [&base_cat, &unified_cat, &buf, &swapDB](Rule &r) {
        Rule tmp = r;
        bool is_modified = false;
        if (unified_cat.find(r.get_internal().get_cat()) !=
            std::end(unified_cat)) {
          r.get_internal() =
              LeftNonterminal{Category{base_cat},
                              r.get_internal().get_means()};
          is_modified = true;
        }
        std::for_each(
            std::begin(r.get_external()),
            std::end(r.get_external()),
            [&base_cat, &unified_cat,
             &is_modified](SymbolElement &sel) {
              if (sel.type() == ELEM_TYPE::NT_TYPE &&
                  unified_cat.find(
                      sel.get<RightNonterminal>()
                          .get_cat()) !=
                      std::end(unified_cat)) {
                sel = RightNonterminal{
                    Category{base_cat},
                    sel.get<RightNonterminal>().get_var()};
                is_modified = true;
              }
```

```
            });
        if (is_modified) {
          if (LOGGING_FLAG) {
            LogBox::push_log("CAT MERGE-> " + tmp.to_s());
            LogBox::push_log("CAT MERGE<- " + r.to_s());
          }
          buf.push_back(r);
        } else {
          swapDB.push_back(r);
        }
      });
  DB.swap(swapDB);

  return buf;
}
void Knowledge::merge_mean_proc_buffer(
    const AMean &base_mean, RuleDBType &buffer,
    std::set<AMean> &unified_mean) {
  std::for_each(
      std::begin(buffer), std::end(buffer),
      [&base_mean, &unified_mean](Rule &r) {
        Rule tmp = r;
        bool is_modified = false;
        if (unified_mean.find(
                r.get_internal().get_base()) !=
            std::end(unified_mean)) {
          r.get_internal() = LeftNonterminal{
              r.get_internal().get_cat(),
              Meaning{AMean{base_mean},
                      r.get_internal().get_followings()}};
          is_modified = true;
        }
        if (is_modified && LOGGING_FLAG) {
          LogBox::push_log("MEAN MERGE-> " + tmp.to_s());
          LogBox::push_log("MEAN MERGE<- " + r.to_s());
        }
      });
}

Knowledge::RuleDBType Knowledge::merge_mean_proc(
    const AMean &base_mean, RuleDBType &DB,
    std::set<AMean> &unified_mean) {
  RuleDBType buf, swapDB;
  std::for_each(
      std::begin(DB), std::end(DB),
      [&base_mean, &unified_mean, &buf, &swapDB](Rule &r) {
        Rule tmp = r;
```

```
        bool is_modified = false;
        if (unified_mean.find(
                r.get_internal().get_base()) !=
            std::end(unified_mean)) {
          r.get_internal() = LeftNonterminal{
              r.get_internal().get_cat(),
              Meaning{AMean{base_mean},
                      r.get_internal().get_followings()}};
          is_modified = true;
        }
        if (is_modified) {
          if (LOGGING_FLAG) {
            LogBox::push_log("MEAN MERGE-> " + tmp.to_s());
            LogBox::push_log("MEAN MERGE<- " + r.to_s());
          }
          buf.push_back(r);
        } else {
          swapDB.push_back(r);
        }
      });
  DB.swap(swapDB);

  return buf;
}

bool Knowledge::replace(void) {
  std::shuffle(std::begin(input_box), std::end(input_box),
               MT19937::igen);
  bool is_replaced;
  RuleDBType::iterator it;
  while ((it = std::begin(input_box)) !=
         std::end(input_box)) {
    if (LOGGING_FLAG) {
      LogBox::push_log("\n-->>REPLACE");
    }
    Rule r = *it;
    input_box.erase(it);
    is_replaced = replacing(r, input_box);
    is_replaced = is_replaced || replacing(r, box_buffer);
    is_replaced = is_replaced || replacing(r, ruleDB);
    if (is_replaced) {
      if (LOGGING_FLAG) {
        LogBox::push_log("USED WORD:");
        LogBox::push_log(r.to_s());
        LogBox::push_log("<<--REPLACE");
      }
      send_box(r);
```

```
        break;
      } else {
        if (LOGGING_FLAG) LogBox::pop_log(1);
        box_buffer.push_back(r);
      }
    }
    send_box(box_buffer);

    return is_replaced;
  }

  bool Knowledge::replacing(Rule &word,
                            RuleDBType &checking_sents) {
    bool is_replaced = false;
    RuleDBType buf, swapDB;
    std::boyer_moore_searcher search_word{
        std::begin(word.get_external()),
        std::end(word.get_external())};
    std::for_each(
        std::begin(checking_sents), std::end(checking_sents),
        [this, &word, &is_replaced, &buf, &swapDB,
         &search_word](Rule &r) {
          if (r.get_external().size() >
                  word.get_external().size() &&
              intention.replace_equal(
                  r.get_internal().get_base(),
                  word.get_internal().get_base())) {
            auto it = std::search(
                std::begin(r.get_external()),
                std::end(r.get_external()), search_word);
            if (it != std::end(r.get_external())) {
              if (LOGGING_FLAG) {
                LogBox::push_log("REPLACE-> " + r.to_s());
              }
              int new_var_id, new_mean_id;
              int b_pos, b_size;
              b_pos = b_size = 0;

              new_var_id = var_indexer.generate();
              new_mean_id = ind_indexer.generate();
              std::for_each(
                  std::begin(word.get_external()),
                  std::end(word.get_external()),
                  [&b_size](SymbolElement &se) {
                    if (se.type() == ELEM_TYPE::NT_TYPE) {
                      b_size++;
                    }
```

```
            });
          // insert
          std::for_each(
              std::begin(r.get_external()), it,
              [&b_pos](SymbolElement &sel) {
                if (sel.type() == ELEM_TYPE::NT_TYPE) {
                  b_pos++;
                }
              });
          b_pos++;
          Meaning new_meaning =
              r.get_internal().get_means().replaced(
                  b_pos, b_size, Variable(new_var_id));
          new_meaning = new_meaning.replaced(
              0, 1, Meaning(AMean(new_mean_id)));

          it = r.get_external().erase(
              it,
              std::next(it, word.get_external().size()));
          r.get_external().insert(
              it, RightNonterminal(
                      word.get_internal().get_cat(),
                      Variable(new_var_id)));

          Rule sent{
              LeftNonterminal{
                  Category{r.get_internal().get_cat()},
                  new_meaning},
              r.get_external()};
          buf.push_back(sent);
          intention.replace(
              r.get_internal().get_base(),
              word.get_internal().get_base(),
              sent.get_internal().get_base(), b_pos,
              b_size);
          if (LOGGING_FLAG) {
            LogBox::push_log("REPLACE<- " + sent.to_s());
          }
          is_replaced = true;
        } else {
          swapDB.push_back(r);
        }
      } else {
        swapDB.push_back(r);
      }
    });
checking_sents.swap(swapDB);
```

```
    send_box(buf);

    return is_replaced;
}
```

Bibliography

[1] Philip Ball. *The music instinct : how music works and why we can't do without it / Philip Ball.* Bodley Head London, 2010.

[2] Robert C. Berwick, Kazuo Okanoya, Gabriel J.L. Beckers, and Johan J. Bolhuis. Songs to syntax: the linguistics of birdsong. *Trends in Cognitive Sciences*, 15(3):113 – 121, 2011.

[3] Ronnie Cann. *Formal Semantics: An Introduction.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1993.

[4] N. Chomsky and M.P. Schtzenberger. The algebraic theory of context-free languages*. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, volume 35 of *Studies in Logic and the Foundations of Mathematics*, pages 118 – 161. Elsevier, 1963.

[5] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137 – 167, 1959.

[6] Noam. Chomsky. *Knowledge of language : its nature, origin, and use / Noam Chomsky.* Praeger New York, 1986.

[7] Noam A. Chomsky. Rules and representations. *Behavioral and Brain Sciences*, 3(127):1–61, 1980.

[8] David Cope. *Experiments in musical intelligence.* A-R Editions, Inc., 1996.

[9] David Cope. *Computer Models of Musical Creativity.* The MIT Press, 2005.

[10] E. Dupoux and J. Mehler. *Language, Brain, and Cognitive Development: Essays in Honor of Jacques Mehler.* Bradford book. MIT Press, 2001.

[11] Ray S. Jackendoff Fred Lerdahl. *A Generative Theory of Tonal Music.* MIT Press, 1982.

[12] T. Fukunari, S. Arn, and S. Tojo. Ccg analyzer with tonal pitch space for non-classical chords. In *2016 Eighth International Conference on Knowledge and Systems Engineering (KSE)*, pages 239–246, Oct 2016.

[13] Mark Granroth-Wilding and Mark Steedman. Statistical parsing for harmonic analysis of jazz chord sequences. In *ICMC*, 2012.

[14] Mark Granroth-Wilding and Mark Steedman. A robust parser-interpreter for jazz chord sequences. *Journal of New Music Research*, 43(4):355–374, 2014.

[15] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Implementing a generative theory of tonal music. *Journal of New Music Research*, 35(4):249–277, 2006.

[16] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. deepgttm-i&ii: Local boundary and metrical structure analyzer based on deep learning technique. In *Bridging People and Sound - 12th International Symposium, CMMR 2016, São Paulo, Brazil, July 5-8, 2016, Revised Selected Papers*, pages 3–21, 2016.

[17] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. deepgttm-iii: Multi-task learning with grouping and metrical structures. In *Music Technology with Swing - 13th International Symposium, CMMR 2017, Matosinhos, Portugal, September 25-28, 2017, Revised Selected Papers*, pages 238–251, 2017.

[18] M. HATTORI. Adaptive heuristics of covariation detection : A model of causal induction. *Proceedings of the 4th International Conference on Cognitive Science and the 7th Australian Society for Cognitive Science Joint Conference (ICCS/ASCS 2003)*, 1:163–168, 2003.

[19] Bradley Hauer and Grzegorz Kondrak. Decoding anagrammed texts written in an unknown language and script. *Transactions of the Association for Computational Linguistics*, 4:75–86, 2016.

[20] Marc D. Hauser, Noam Chomsky, and W. Tecumseh Fitch. *The faculty of language: what is it, who has it, and how did it evolve?*, page 1442. Approaches to the Evolution of Language. Cambridge University Press, 2010.

[21] Marc D. Hauser and Jeffrey Watumull. The universal generative faculty: The source of our expressive power in language, mathematics, morality, and music. *Journal of Neurolinguistics*, 43:78 – 94, 2017. Language Evolution: On the Origin of Lexical and Syntactic Structures.

[22] Jane Heal. Joint attention and understanding the mind. In N. Elian, Christoph Hoerl, Teresa McCormack, and Johannes Roessler, editors, *Oxford University PressJoint Attention: Communication and Other Minds*, pages 34–44. Oxford University Press, 2005.

[23] Timo Honkela. Self-organizing maps in natural language processing. Technical report, Espoo, Finland, 1997.

[24] James R. Hurford. Biological evolution of the saussurean sign as a component of the language acquisition device. *Lingua*, 77(2):187 – 222, 1989.

[25] Mutsumi Imai and Dedre Gentner. A cross-linguistic study of early word meaning: universal ontology and linguistic influence. *Cognition*, 62(2):169 – 200, 1997.

[26] Mutsumi Imai, Dedre Gentner, and Nobuko Uchida. Children's theories of word meaning: The role of shape similarity in early acquisition. *Cognitive Development*, 9(1):45 – 75, 1994.

[27] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling, 2016.

[28] Michael N. Katehakis and Arthur F. Veinott. The multi-armed bandit problem: Decomposition and computation. *Mathematics of Operations Research*, 12(2):262–268, 1987.

[29] Masatoshi Hamanaka Kenji Hirata, Satoshi Tojo. *An Algebraic Approach to Time-Span Reduction*, pages 251–270. Springer International Publishing, Cham, 2016.

[30] Simon Kirby. Learning , bottlenecks and the evolution of recursive syntax. *Ted Briscoe(Ed.) Linguistic Evolution through Language Acquisition: Formal and Computational Models*, pages 173 – 204, 2002.

[31] Takuya Koumura and Kazuo Okanoya. Automatic recognition of element classes and boundaries in the birdsong with variable sequences. *PLOS ONE*, 11(7):1–24, 07 2016.

[32] Barbara Landau, Linda B. Smith, and Susan S. Jones. The importance of shape in early lexical learning. *Cognitive Development*, 3(3):299 – 321, 1988.

[33] Satoshi Tojo Masatoshi Hamanaka, Kenji Hirata. *Implementing Methods for Analysing Music Based on Lerdahl and Jackendoff's Generative Theory of Tonal Music*, pages 221–249. Springer International Publishing, Cham, 2016.

[34] Ryuichi Matoba, Makoto Nakamura, and Satoshi Tojo. Efficiency of the symmetry bias in grammar acquisition. *Information and Computation*, 209(3):536 – 547, 2011. Special Issue: 3rd International Conference on Language and Automata Theory and Applications (LATA 2009).

[35] Sudo H. Hagiwara S. Matoba, R. and S. Tojo. Evaluation of efficiency of the symmetry bias in grammar acquisition. In *Proceedings of the 18th International Symposium on Artificial Life and Robotics(AROB2013)*, pages 444–447, 2013.

[36] Etsuko Haryu Mutsumi Imai. The nature of word learning biases: From a cross linguistic perspective. *D. G. Hall & S. Waxman (Eds.) Weaving a lexicon*, pages 411 – 444, 2004.

[37] Michael Oliphant. The dilemma of saussurean communication. *Biosystems*, 37(1):31 – 38, 1996.

[38] Sergio Oramas, F. Barbieri, Oriol Nieto, and Xavier Serra. Multimodal deep learning for music genre classification. *Transactions of the International Society for Music Information Retrieval*, 1:4–21, 2018.

[39] Salim Perchy and Gerardo Sarria. Musical Composition with Stochastic Context-Free Grammars. In *8th Mexican International Conference on Artificial Intelligence (MICAI 2009)*, Guanajuato, Mexico, November 2009.

[40] Olivia E. Kang Phil N. Johnson-Laird and Yuan Chang Leong. On musical dissonance. *Music Perception: An Interdisciplinary Journal*, 30(1):19 – 35, 2012.

[41] W. V. Quine. Word and object. *Philosophy and Phenomenological Research*, 22(1):115–116, 1961.

[42] Andrea Ravignani, Bill Thompson, Thomas Grossi, Tania Delgado, and Simon Kirby. Evolving building blocks of rhythm: How human cognition creates music via cultural transmission. *bioRxiv*, 2017.

[43] J.C. Scholtes. Resolving linguistic ambiguities with a neural data-oriented parsing (dop) system. In Igor ALEKSANDER and John TAYLOR, editors, *Artificial Neural Networks*, pages 1347 – 1350. North-Holland, Amsterdam, 1992.

[44] Murray Sidman, Ricki Rauzin, Ronald Lazar, Sharon Cunningham, William Tailby, and Philip Carrigan. A search for symmetry in the conditional discriminations of rhesus monkeys, baboons, and children. *Journal of the Experimental Analysis of Behavior*, 37(1):23–44, 1982.

[45] Christina Wallin Steven Brown, Bjrn Merker. *The Origins of Music*. MIT Press, 1993.

[46] Toshitaka N. Suzuki, David Wheatcroft, and Michael Griesser. Experimental evidence for compositional syntax in bird calls. *Nature Communications*, 7:10986, Mar 2016. Article.

[47] Masaya Nakatsuka Takashi Hashimoto. Reconsidering kirby's compositionality model toward modelling grammaticalisation. *The Evolution of Language : Proceedings of the 6th International Conference (EVOLANG6)*, pages 415 – 416, 03 2006.

[48] Satoshi Tojo, Yoshinori Oka, and Masafumi Nishida. Analysis of chord progression by hpsg. In *Proceedings of the 24th IASTED International Conference on Artificial Intelligence and Applications*, AIA'06, pages 305–310, Anaheim, CA, USA, 2006. ACTA Press.

[49] Tessa Verhoef. The origins of duality of patterning in artificial whistled languages. *Lang Cogn*, 4(4):357–380, Nov 2012. 23637710[pmid].

[50] Terry Winograd. Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory*, 12:2, 1968.

Publications

[1] H. Sudo, M. Taniguchi and S. Tojo: "Finding Grammar in Music by Evolutionary Linguistics", "The 13th International Conference on Knowledge, Information and Creativity Support Systems", Thai, Pattaya, November, 2018.

[2] H. Sudo, R. Matoba, T. Cooper and A. Tsukada: "Effect of symmetry bias on linguistic evolution", Journal of Artificial Life and Robotics, Springer Japan, Vol. 21, No. 2, pp. 207-214, 2016.

[3] R. Matoba, H. Sudo, M. Nakamura, S. Hagiwara and S. Tojo: "Process Acceleration in the Iterated Learning Model by String Clipping", International Journal of Computer and Communication Engineering, International Academy Publishing, Vol. 4, No. 2, pp100-106, 2015.

[4] H. Sudo, R. Matoba, T. Cooper and A. Tsukada: "Effect of the Symmetry Bias on Linguistic Evolution", International Symposium on Artificial Life and Robotics, Beppu, Oita, January 2015.

[5] R. Matoba, H. Sudo, M. Nakamura and S. Tojo: "Application of Loose Symmetry Bias to Multiple Meaning Environment", The Seventh International Conference on Advanced Cognitive Technologies and Applications, Nice, France, March, 2015.

[6] R. Matoba, H. Sudo, M. Nakamura, S. Hagiwara and S. Tojo: "Process Acceleration in the Iterated Learning Model by String Clipping", International Conference on Computational Collective Intelligence Technologies and Applications, Seoul, Korea, September, 2014.

[7] R. Matoba, H. Sudo, S. Hagiwara and S. Tojo: "Evaluation of the Symmetry Bias in Grammar Acquisition", International Symposium on Artificial Life and Robotics, Daejeon, Korea, January, 2013.