COMPUTATIONAL APPROACHES TO LINGUISTIC CONSENSUS

BY

JUN WANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Library and Information Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2006

Urbana, Illinois

Abstract

The main question we ask is how a common language might come about in complex adaptive language systems comprising many agents. Our primary objective is to analyze and design complex language models so that a group of agents can converge on a common language from their initially different languages by using reinforcement learning, a minimal information approach. Towards this end, we present a game-based self-organizing language framework, and study three important cases of reaching linguistic consensus: word consensus, coherent communication, and grammar consensus.

The study of word consensus concerns how agents can converge to a common word to reliably express a single shared meaning from their initially different words. We have proposed a win-stay lose-shift learning model, and have shown by computer simulation and mathematical analysis the conditions under which the agents in the model can converge to a common word.

The study of coherent communication concerns how agents can converge on a communication system in which the word used by a sender to represent some meaning can be interpreted correctly by a receiver to extract the same meaning. We have proposed a minimum reinforcement learning model comprising two agents (a sender and a receiver), and have shown by computer simulation and mathematical analysis the conditions under which agents in the model can converge to a coherent communication system.

The study of grammar consensus concerns how agents can converge to a common grammar. In the converged state, the sentences generated by one agent using his grammar can be recognized by another agent using her grammar. We have proposed a mutual perceptron learning model in which grammars are modeled as Boolean functions that can be used to classify or recognize Boolean

instances (sentences), and have shown by mathematical analysis the conditions under which agents in this model can converge to a common grammar (i.e., a common Boolean function).

This work has important implications for many kinds of distributed semantic systems, such as shared web ontologies, agent communication protocols, collaborative tagging, database schema integration, and biological networks.

Acknowledgments

I wish to express my greatest thanks to my advisor, Professor Les Gasser. This thesis would not have been possible without his guidance throughout my long journey of doctoral study. Since I came to GSLIS, Les has treated me as an independent researcher. He supported my exploration of various thesis topics and spent a lot of time helping me shape my ideas with numerous discussions. I also improved my scientific writing skills significantly during the experience of co-authoring papers with him.

I am also grateful to my other committee members—Linda Smith, J. Stephen Downie, and Dave Dubin. Their valuable, detailed comments and assistance help me immensely in clarifying my presentation.

Finally, thanks to my wife Bei and our son Albert for their love and support. This thesis is dedicated to them.

Table of Contents

Chapter 1

Introduction

1.1 The problem

The use of complex language is one of humanity's most useful and unique characteristics. Language allows people to exchange knowledge and to pass information across time and space—it represents the principal non-genetic information system for the human species (Maynard-Smith and Szathmary, 1997). The technical and social development of human groups cannot be fully understood without understanding the development and impact of language. The ability to coordinate activity, divide labor, evaluate goals and options, learn skills, and many other capacities that have fostered the emergence of human societies all depend on the emergence of languages with adequate complexity and scope.

Two of the principal issues in language evolution are explaining its simultaneous properties of *collectivity* and *adaptivity*. First, there is no single-agent language. Since a primary purpose of a language is communication, language is only useful if it is shared by several individuals, that is, by a *population*. Language is a collective distributed information system, and it makes no sense as a individualized concept. Second, even highly developed human languages are not static; all natural languages are *adaptive* systems that evolved from earlier possibly more primitive communication systems, and that continue to evolve as communicative needs and situations change.

Thus two of the central, complementary issues in language evolution research are 1) how a communicative population comprising autonomous individuals converges to a language that is common enough for mutual understanding, and 2) how the population maintains this coherence while evolving its language.

Beyond understanding the origins and dynamics of human languages, many researchers are interested in how *artificial agents* and other intentionally designed software programs might develop, reconcile, and sustain their own sophisticated representation and communication regimes from the ground up (Steels, 1997, 2003, 2006). There are two reasons for this interest. First, the study of language evolution informs many central problems of semantically oriented distributed information systems such as distributed database systems; collaborative tagging systems and "folksonomies" (Mathes, 2004; Sen et al., 2006; Golder and Huberman, 2006); shared ontologies envisioned as critical underpinnings for semantic web markup and information query expansion; information integration and search; self-adapting communication languages for web services and software agents; and so on. We can model each of these application areas as a collection of "agents" (defined below) that interact by using some form of language. Little theory of the communication dynamics of these types of information system exists, and they are currently very difficult to manage and make robust. Thus there are many potential practical impacts of understanding the collective dynamic properties of language.

Second, the ability to work with artificial agents promises more general insights than studies based entirely on human language development. Human language arose in a context of unique environments and survival problems, over a specific historical period, in organisms with particular genetic makeup and development. This limits data and theorizing. With artificial agents, on the other hand, research explorations are limited only by theoretical constraints on agent representation and performance, so there is a much wider range of conceptual (descriptive) and practical (normative) options to explore than those built from natural historical human scenarios. Indeed, the collective dynamics of human language should be a *specific case* within a general theory of collective language dynamics.

In this thesis, then, we will engage the research issue of collective language dynamics by studying how a population of artificial agents can converge to a common language from a set of initially different languages through pairwise interaction and learning. We will call this process *reaching linguistic consensus*.

It is no accident that a growing body of researchers is working to discover the underlying mathematical, computational, and implementation principles that explain how languages *in general*—both human and artificial—emerge, how they change, and how language-using populations maintain stable communicativity while languages are evolving. These studies have naturally focused the work of researchers from many different disciplines, including evolutionary biology, psychology, cognitive sciences, anthropology, and artificial intelligence to name a few (Christiansen and Kirby, 2003). See (Wagner et al., 2003; Steels, 2003; Wang and Minett, 2005; Brighton et al., 2005) for typical overviews of this work, which will not be surveyed in depth in this thesis given the existing literature.

Because the complexity of the collective dynamics problems of language, most work to date has been exploratory, using computer simulations and intuitively interpreting them. While simulation is an important, suggestive, and insightful methodology, to achieve a thorough understanding of the emergence of shared languages, we need to augment simulation studies with mathematical characterizations of specific conditions and limitations under which agents can converge to shared languages. This thesis aims filling some of this gap. We study the emergence of common languages using mathematical modeling and theoretical analysis. Our models and theories have sometimes been inspired by our computational simulation models, but we generally use simulations to discover and illustrate interesting phenomena. In contrast, we use mathematical theory to define these phenomena exactly and then to derive and express their dynamics and limits with precision.

1.2 The general research question

As mentioned above, the class of distributed semantic information systems we study can be characterized as systems of "agents". Shalizi has provided a useful, general definition of "agent" as "a persistent thing which has some state we find worth representing, and which interacts with other agents, mutually modifying each others' states" (Shalizi, 2003). For an agent who has the capa-

bility of language, the state that is worth representing is the agent's language knowledge such as the mechanisms (including parameters) of sentence production, interpretation, and grammar rule acquisition. Note that people can be seen in a very abstract sense as agents under this definition, if we take people as having states of mind, interacting with others, and having influence on each other. Of course people are presently much more complex intellectual, social, sentimental, and political beings than the artificial, mathematical, or software agents we model here, and we are not claiming that our mathematical models provide a full account of all processes of human language development and convergence. Nonetheless, considering language *as a collective dynamic information system*, we do believe that clear mathematical/computational treatments of language dynamics will be useful for understanding the potential landscape of human language dynamics (for example, tractability limits). Such new models may also help as foundations for the design of application-specific languages and language processing approaches that exhibit greater representational parsimony, communicative efficiency, simplicity, or adaptivity than naturally occurring human languages.

Given this agent-based viewpoint, there are four factors in the design of complex adaptive language systems that may affect whether the agents can (or cannot) converge to a common language:

- Population size: the number of agents in a population;
- Language complexity: the complexity of a language such as the number of words in a vocabulary;
- Learning mechanism: how agents adapt or learn their language through pairwise interactions;
- Interaction structure: who interacts with whom.

The general question we ask in this thesis is how these factors affect the ways agents can converge to a common language. In other words, we want to know what ranges of "settings" of these factors will lead to *convergence success or failure* for a set of agents, and how the settings affect

the *convergence speed*. We want to understand these relationships in three linguistic consensus cases, detailed in the next section.

1.3 Research scope

In this thesis, we study the above questions of how four factors affect convergence success and speed in each of three important cases of linguistic consensus:

1. **Word consensus:** Converging on an agreement about which word, out of a number of possible words, will be used to reliably express a single shared meaning.

2. **Coherent communication:** Converging on a coherent communication system in which the word used by a sender to represent some meaning can be interpreted correctly by a receiver to extract the same meaning.

3. **Grammar consensus:** Grammar is a shared structure for accurately interpreting a sequence of words, namely, a sentence. Grammar consensus refers to how a group of agents can converge to a common grammar by which the sentences generated by one agent using his grammar can be recognized by another agent using her grammar.

These three problems play an important role in the study of the emergence of language (Steels, 1997; Wagner et al., 2003). Among them, the first two are related to developing a reliable shared lexicon, and the third is related to developing a shared grammar. Lexicon and grammar are the two most basic components of any language. The lexicon enables an agent to represent meanings (objects in the world or mental concepts) with symbols (words) and to interpret words back into meanings. Grammar improves the efficiency and accuracy of coding complex meanings (Nowak et al., 2000; Plotkin and Nowak, 2000) and enables an agent to express a large (possibly infinite) number of meanings using only a finite number of words.

Existing work on these three problems can be classified into two paradigms, *observational learning* or *reinforcement learning*, according to what kind of feedback information is available for

a learning algorithm. In the case of the observational learning paradigm, an agent can have much information available to it for learning, including any information it can observe; for example, an agent may observe the linguistic behavior of other agents, enabling it to imitate the entire observed behavior or generalize from some specific observed behaviors. In contrast, for the reinforcement learning paradigm, the feedback information that is available to an agent is limited to a feedback score or *reward* from its interaction with other agents; for example, the agent may only know whether a communication is "successful" or not.

This thesis is devoted to the reinforcement learning paradigm. This is a choice with important consequences. Since reinforcement learning requires only a *minimum* of feedback information, by studying using this paradigm as our theoretical basis, we can obtain lower bounds on the effectiveness and efficiency of complex adaptive language systems.

As we mentioned earlier, because we are pursuing general results, in this thesis we are mainly interested in the study of abstract language models rather than real human languages. The assumptions made in the abstract models, though unrealistic, can provide a starting point for further development as well as theoretical values on guiding the design of artificial agents.

1.4 Research methods

Our approach to the study of the above questions is to use both computer simulation and mathematical analysis. With the tools of computer simulation, we can gather data on the effects of a range of different parameter inputs over multiple runs. In the words of Robert Axelrod (1997), simulation is "driving a model of a system with suitable inputs and observing the corresponding outputs." By running a simulation many times with different parameter settings, we can observe the range of these settings under which the agents reach linguistic consensus or fail to do so.

For complex adaptive systems with many agents such as the one used here, computer simulations are a common research approach because many mathematical analyses seem harder than computational experiments. Though simulation is a convenient tool that can quickly provide intu-

itive results, there are several disadvantages to simulation. First, a complex system usually involves many parameters and the combinatorial parameter space may be huge. In many situations we can gain this intuitive understanding of the behavior of a modeled system by examining relatively few special cases. Nonetheless, there are many types of problems that would require us to systematically explore the whole parameter space to understand system behavior fully, and this can be a formidable task. In addition, it is always hard to guarantee that an implemented computational model accurately reflects the system modeled and has no software bugs.

To have a solid understanding of a complex system, the best approach is to combine computer simulation and mathematical analysis. By using mathematical analysis, we can obtain a complete description on the relationship between different parameters, without the need to run numerous simulations to explore the whole parameter space. More importantly, we can understand clearly the phenomenon observed in the simulations. In addition, if results obtained by simulation and analysis fail to adequately match, that in itself is an interesting finding that can lead to identification of software bugs or mathematical errors.

There is an important third point that we need to mention when deriving mathematical equations from complex computational models. A complex computational model often involves many details that prevent a researcher from converting it directly into a precise mathematical description. To be able to derive a mathematical equation from a computational model, we often need to make some assumptions. The role of these assumptions is to eliminate unimportant details while keeping essential elements of a model. However, if the assumptions are incorrect about what details are unimportant, then even when the simulation code is bug-free the simulation results may not agree with the results of mathematical analysis. In this case, we have to examine whether the assumptions are really catching the necessary causal aspects of the computational model. On the other hand, if the simulation and analytical results match well, we have strong evidence that we have made accurate assumptions about what details are important.

The following procedure gives a summary of our research method of combining simulation and mathematical analysis.

1. Design a computational model that includes a collection of parameter settings and gives some output.

2. Run simulations, and observe what happens for different parameter inputs to the model.

3. Try to derive mathematical equations from the computational model.

4. If the results obtained by mathematical analysis and simulation can explain each other, done!

5. Otherwise, check these possibilities:

   (a) Check the program for bugs. If bugs exist, then revise the code, go to (2), and try again.

   (b) Check the derivation of mathematical equations. If it is not correct or the assumptions made in the derivation are inaccurate, then re-derive the equations or revise the assumptions, and go to (3) and try it again.

1.5 Contributions

The main question we ask is how a common language might come about in complex adaptive language systems. Our primary objective is to design and analyze complex language models so that a group of agents can converge on a common language from their initially different languages by using reinforcement learning. Towards this end, we present a game-based self-organizing language framework into which reinforcement learning can be naturally fit. Using the framework, we study three important cases of reaching linguistic consensus: word consensus, coherent communication, and grammar consensus.

In general, we have made the following contributions.

1. We have proposed a general game-based self-organizing language framework, and have introduced two general language game models (the simultaneous and sequential models). These well-defined models can be easily used to frame the three linguistic consensus problems studied in this thesis as well as existing models studied before by others.

2. We have studied three cases of linguistic consensus.

   (a) In the case of word consensus, we have proposed a word consensus model in which agents make adaptations using the *Win-Stay, Lose-Shift (WSLS)* learning rule (Matsen and Nowak, 2004). We have shown by computer simulation and by mathematical analysis that the agents in the model can converge to a common word under certain conditions, and we give those conditions. We also give a dynamics equation on how coherence changes over time when the convergence condition is satisfied. Our work was motivated by (Shoham and Tennenholtz, 1997) and (Matsen and Nowak, 2004), but compared to the Shoham-Tennenholtz model, ours requires a minimum memory load on the agents, and compared to the Matsen-Nowak model, our analytical result gives a comprehensive description of the relationship among all related parameters.

   (b) In the case of coherent communication, we have proposed a minimum model which consists of two agents (a sender and a receiver) who have the task of reaching coherent communication using simple reinforcement learning (Lenaerts et al., 2005). Again, we have shown by computer simulation and by mathematical analysis that agents in the model can converge to a coherent communication system under some conditions, and we give those conditions. We also give a dynamics equation on how coherence changes over time when the convergence condition is satisfied. Compared with existing work, ours is the first to give an analytical result on the conditions under which the agents can reach coherent communication using reinforcement learning, which is, as noted above, a minimum-information approach, and hence provides a lower bound.

   (c) In the case of grammar consensus, we have proposed a mutual perceptron learning model in which grammars are modeled as Boolean functions that can be used to classify or recognize Boolean instances (sentences). We have shown by mathematical analysis that agents in this model can converge to a common grammar (i.e., a common classification function) under some conditions, and we have given those conditions. Compared

with existing work on the grammar consensus problem (which has mainly been done within the observational learning paradigm), ours is the first that uses reinforcement learning as the learning mechanism of agents.

1.6 Outline

The organization of this thesis is as follows. In Chapter 2, we present the game-based self-organizing language framework, including game model, agent learning model, interaction structure, and population coherence that measures the consensus level of a population. Considering the significant difference between two basic types of games: simultaneous and sequential games, we introduce two general self-organizing language models, called the *simultaneous language game* and the *sequential language game* models. The simultaneous model will be used in Chapter 3 for the case study of word consensus, and the sequential model will be used in Chapter 4 for the case study of coherent communication and in Chapter 5 for grammar consensus. From Chapter 3 to 5, we study the three linguistic consensus cases. We close with a summary of the thesis and then we point out some future directions.

Chapter 2

A Game-Based Self-Organizing Language Framework

2.1 Introduction

Language is a game that it takes (at least) two to play. A critical question is how a shared common language might emerge from repeated pairwise interactions between communicators (Pinker, 2000). The primary objective of this study is to design and analyze a mechanism under which a group of adaptive agents, using reinforcement learning that requires only information about the payoff received from an instance of a language game, can converge to a common language from their initially different languages.

In this chapter, we present a game-based self-organizing language framework as a foundation for studying how a common language can come about from repeated pairwise game play. In the successive three chapters we will use this framework to study three cases of reaching linguistic consensus: word consensus, coherent communication, and grammar consensus.

Our framework for self-organizing language systems, as shown in Fig. 2.1, consists of

- a population of agents;
- a *coherence* measure used to represent the degree of consensus in a population of agents;
- a 2-player language game model that formalizes the pairwise interaction between two agents;
- an agent learning model that specifies how agents make decisions, and especially how they learn to improve decision making based on the payoff received from games they play; and,
- an interaction structure that is used to specify which pairs of agents interact.

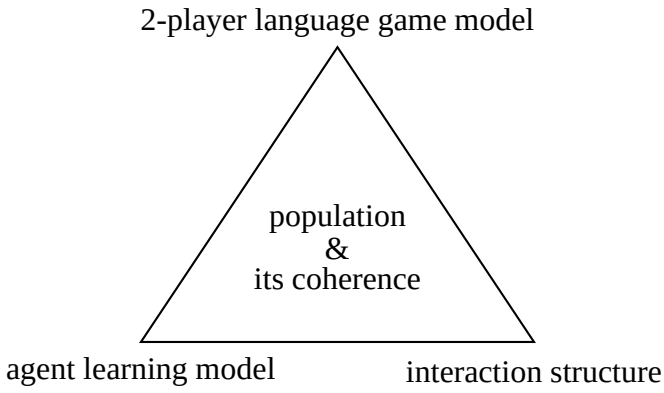


Figure 2.1: A framework of self-organizing language systems.

In the following sections, we will describe in depth the 2-player game models, agent learning models, interaction structures, and the coherence measure.

2.2 2-player language games

To best situate our research, we will first give a brief introduction to some basic game theory concepts that we will use later, including a general definition of games and two representation forms of games (normal form and extensive form). We will then introduce two types of language games: simultaneous and sequential language games.

2.2.1 Definition and representation of games

In game theory, a game consists of the following three components (Osborne and Rubinstein, 1994):

- a set of players or agents $\{1, 2, \cdots, K\}$;
- a set of actions (also called "strategies") A_i available to each player $i \in \{1, 2, \cdots, K\}$;
- a payoff function defined on each combination of strategies for each player, which is given by $\pi_i(a_1, \cdot\cdot, a_i, \cdot\cdot, a_K)$ where $a_i \in A_i$ is the action of player i.

A game can be represented in normal form or extensive form (ibid.). In normal form, a game is usually represented by a payoff matrix which shows the players, actions, and payoffs. Fig. 2.2 illustrates an example of a 2-player game in which each player has three actions (scissors, rock, and paper) and the payoff of a player is shown in the cells of the matrix. The first number of a cell is the payoff received by the row player (i.e., Player 1); the second is the payoff for the column player (i.e., Player 2). Suppose that Player 1 plays "scissors" and that Player 2 plays "rock", then Player 1 gets a payoff of -1, and Player 2 gets 1.

		Player 2 (π_2)		
		scissors	rock	paper
Player 1 (π_1)	scissors	0,0	-1,1	1,-1
	rock	1,-1	0,0	-1,1
	paper	-1,1	1,-1	0,0

Figure 2.2: Normal form (payoff matrix) of the 2-player "scissors, rock, paper" game.

When a game is represented in normal form, it is usually assumed that each player acts simultaneously or, equivalently for purposes of analysis, acts without any information about the actions of other players. Otherwise, the game is usually represented in extensive form. Extensive form games are often presented as trees, showing that there is some important ordering to the sequence of actions and to the information players obtain from them. In a game tree, each node represents a time point of action for one of the players, and the lines from the node represent a possible action for that player. The payoffs are specified at the bottom of the tree. Fig. 2.3 illustrates an example of 2-player extensive form game in which Player 1 moves first and chooses either L or R, and Player 2 sees Player 1's move and then chooses U or D. For example, in the case where Player 1 chooses L and then Player 2 chooses D, Player 1 gets a payoff of 3 and Player 2 gets 0.

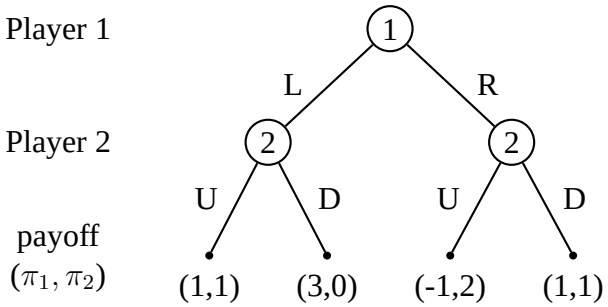


Figure 2.3: Extensive form of a 2-player game.

2.2.2 Simultaneous language games

In game theory, a game can be classified into *simultaneous games* or *sequential games* (ibid.). Simultaneous games are games where both players move simultaneously, or if they do not move simultaneously, the later players are unaware of the earlier players' actions—making them effectively simultaneous in terms of information available to agents. Sequential games are games where later players have some knowledge about earlier actions. Normal form is typically used to represent simultaneous games, and extensive form is typically used to represent sequential ones.

Applying this classification scheme, we can then distinguish two types of language games: *simultaneous language games* and *sequential language games*. One example is the ESP game[1] (von Ahn, 2006), a game in which two players label online images with words. Players choose and submit their label words without seeing those chosen by their partner player. Players earn points when the label words they have chosen match those of their partner player. This is a simultaneous language game, because the two players submit their words to the ESP game system without knowledge of each other's word choices. In contrast, a communication event that takes place between a speaker and a hearer can be seen as a sequential game, because the speaker moves (speaks) first and thus the hearer is aware of the utterance produced by the speaker.

Fig. 2.4 shows a specific procedure for playing simultaneous games with two agents. Fig. 2.5 illustrates an example of the payoff matrix of an ESP game. In this ESP game, we suppose there is an online image of a red car in a street. Both players want to label the image using one of the

[1] http://www.espgame.org/

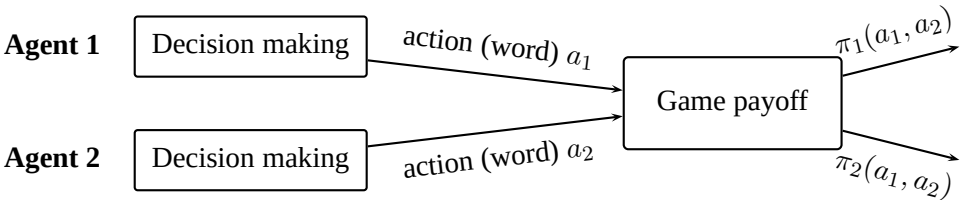


Figure 2.4: Procedure of a simultaneous language game.

		Player 2 (π_2)		
		car	red	street
Player 1 (π_1)	car	1,1	0,0	0,0
	red	0,0	1,1	0,0
	street	0,0	0,0	1,1

Figure 2.5: Payoff matrix of a 2-player simultaneous language game.

three words "car", "red", and "street". Suppose each player is allowed to submit only one word as the label. Then each word represents an action (the action of choosing that word and submitting it). In the payoff matrix for this game, each cell has two numbers, indicating the payoff for each of the two players. For example, if Player 1 uses "car" and Player 2 uses "street" as the label, the payoff for Player 1 will be $\pi_1(car, street) = 0$, which in this case is also the payoff for Player 2. The specific structure of this payoff matrix also represents the information that the two players both need to choose the same word to get the best payoff.

In Chapter 3, Reaching Word Consensus, the simultaneous language game will be used as our 2-player game model.

2.2.3 Sequential language games

Since a primary function of language is for communication between speakers and hearers, in a typical situation of playing language games, there is an order of play between the two players. To illustrate the concept, we present a cartoon example of a sequential language game. Suppose

there is a baby and a mother. The baby cannot speak any real words, but he can utter two different sounds: “ah” and “oh”. Sometimes the baby is hungry (H) and sometimes he has a dirty diaper (D). He may use “ah” to mean he is hungry and “oh” to mean that he has a dirty diaper; but he may also use “ah” for having a dirty diaper and “oh” for being hungry. So his mother needs to learn the true meaning of a sound. This baby-mother communication process can be modeled as a sequential game, as shown in Fig. 2.6. The need or meaning that the baby wants to communicate—being hungry or having a dirty diaper—is decided by some factor external to the communication game (i.e., the baby’s internal systems). A typical treatment for this in game theory (Kreps, 1990; Osborne and Rubinstein, 1994) is to add an additional player called *nature* that moves first to decide what need to create for the baby, according to some probability distribution. In the figure, the number $\frac{1}{2}$ indicates that half of the time nature makes the baby hungry and half of the time nature produces a dirty diaper. The dashed lines in the tree indicate that there are two different paths to “ah” (or to “oh’), but the mother cannot tell the difference because she does not know to which meaning the baby refers by the utterance “ah” (or “oh”). At the bottom of the figure are the payoffs of the players; this example shows that when the mother correctly interprets the meaning of the baby, both get a positive payoff.

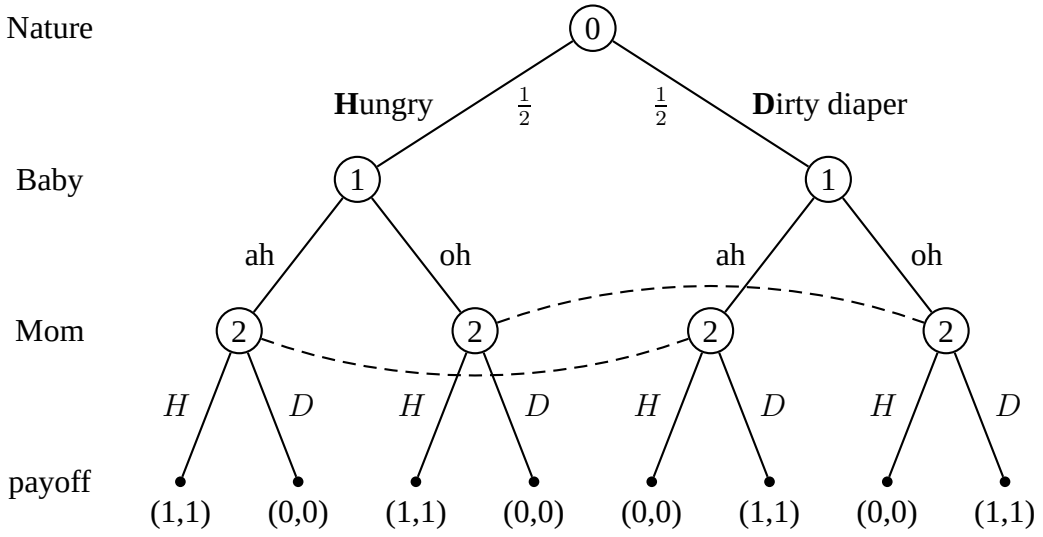


Figure 2.6: Payoff function of a sequential language game.

Fig. 2.7 shows a general procedure for carrying out such sequential game between a speaker

and a hearer. In this procedure model, nature decides what meaning the speaker wants to communicate to the hearer. This decision of nature is represented as a probability distribution over all possible meanings (i.e. the actions of nature). Note that the meaning in the head of the speaker is information only known to the speaker himself, and unknown to the hearer. In game theory, the particular nature-driven action for a player is called the *type* of the player (ibid.). For example, in the example of baby and mother, there are two types of baby: a hungry baby and baby with a dirty diaper. Later in this section, for the sake of generality, we use term *type* for describing a specific meaning choice in the general sequential language game model.

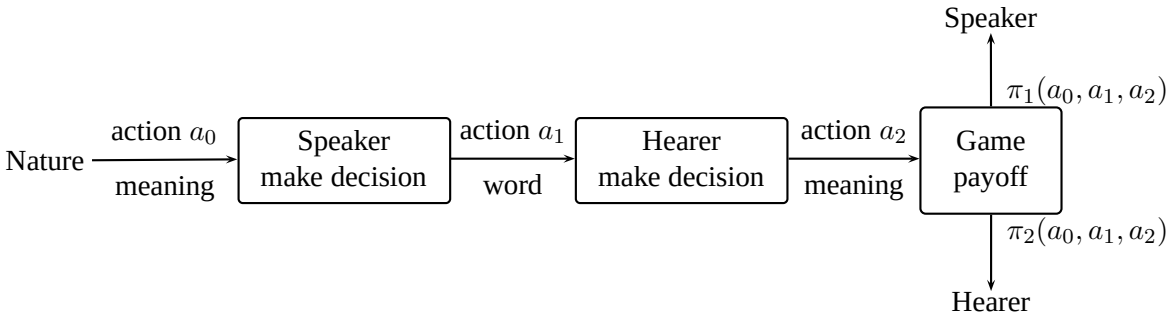


Figure 2.7: Procedure of a sequential language game.

In Chapter 4 Reaching Coherent Communication and Chapter 5 Reaching Grammar Consensus, sequential language games will be used as our 2-player game model.

2.3 Agent learning model

By specifying the payoffs in a game that agents or players will receive after their moves, games provide a formal model of how to characterize the rational behavior of players. A state of *Nash equilibrium* is the most classical model of the best set of moves for a set of rational players in a game. The basic idea is that at Nash equilibrium every player has an action that yields the highest payoff it can get given the action choices of others. Thus no unilateral action switch will yield a higher payoff for that agent, and this is true of all agents, so no agent has a unilateral incentive to switch—hence the equilibrium (ibid.). A critical problem of Nash equilibrium is that some

games may have many equilibria. When there are many equilibria in a game, which equilibrium (i.e. which action choices of others) should a player assume? For example, in the game shown in Fig. 2.5, there are three Nash equilibria: (car, car), (red, red), and $(street, street)$; in each the two players should use the same word, but which word to use?

One solution to the problem is to extend a one-shot game to a (possibly infinite) set of repeated games in which agents are allowed to change their actions until they reach some equilibrium (Fudenberg and Levine, 1998). But here the question is what procedure should the agents use to decide how to change their actions? This is the problem of agent learning in a repeated game. Here we present several agent learning models for specifying how agents make decisions on which actions to take, and how they learn to change actions to improve decision making. In our model, each agent has three components:

1. *an internal state* b;

2. *a decision function* f that specifies how an agent makes decision on which action to take based on the agent's current internal state; and,

3. *a state update function* g that specifies how an agent updates its state based on the payoff received from past games.

Fig. 2.8 illustrates a learning model for simultaneous language games, and Fig. 2.9 a learning model for sequential language games. In both figures, there are three types of functions: decision making, state update, and game payoff functions. Among them, the decision making and state update functions reflect the behavior of individual agents, and the payoff function represents the pairwise interaction behavior between the agents.

Formally speaking, denote by b_i the state of agent i and a_i the action of agent i. Then the decision making function of agent i is given by the form $f_i : b_i \mapsto a_i$ for simultaneous games, or $f_i : (b_i, a_{i-1}) \mapsto a_i$ for sequential games. In addition, the state update function of agent i is given by the form $g_i : (b_i, a_i, \pi_i) \mapsto b_i'$ for simultaneous games, or $g_i : (b_i, a_{i-1}, a_i, \pi_i) \mapsto b_i'$ for

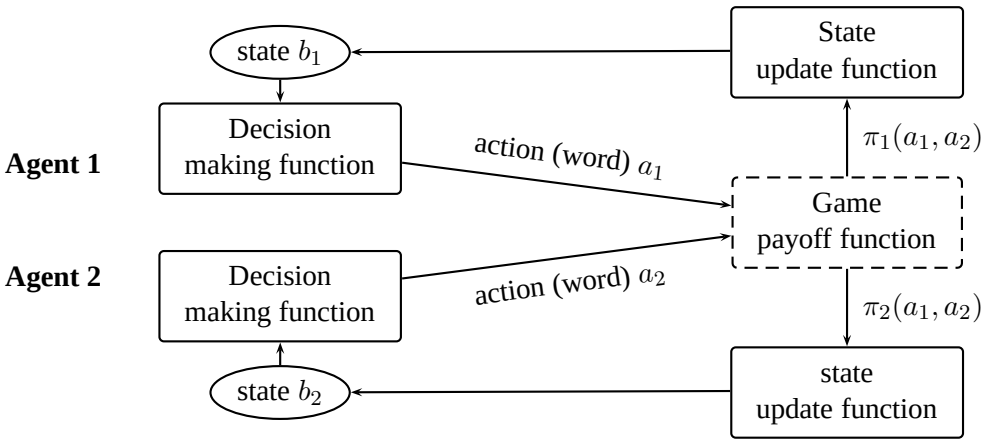


Figure 2.8: Agent learning model in simultaneous language games.

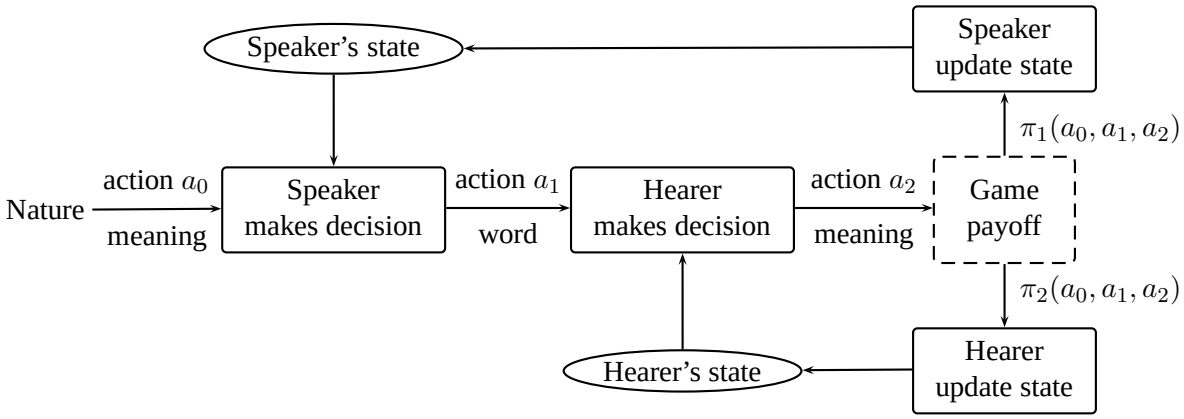


Figure 2.9: Agent learning model in sequential language games.

sequential games. Note that state b does not need to be a scalar variable. Actually, in many cases, such as in the following example, the state of an agent is represented as a vector.

Here we give a concrete example to illustrate the concepts of state, decision function, and state update function. This is an example based on the ESP word-guessing game (Fig. 2.5). In this example, the state of a player is modeled as a probability distribution over all the words: $b = (p(w_1), \cdots, p(w_n))$, where $w_1, \cdots, w_n$ are the all n available words. Then, the decision function is modeled as choosing the word with the largest probability: $a = f(b) = \arg\max_w p(w)$. Suppose the recently-chosen word is w. Then the update of a state or a distribution can be implemented as

follows using *linear reward-penalty scheme* (Bush, 1958; Narendra and Thathachar, 1989). If the payoff is positive, then

$$\begin{cases} p(w) &= p(w) + \alpha(1 - p(w)) \\ p(w') &= p(w') - \alpha p(w'), \text{ for all } w' \neq w \end{cases}$$

where $0 \leq \alpha \leq 1$ is an update parameter. Otherwise,

$$\begin{cases} p(w) &= p(w) - \beta p(w) \\ p(w') &= p(w') + \beta(\frac{1}{n-1} - p(w')), \text{ for all } w' \neq w \end{cases}$$

where $0 \leq \beta \leq 1$ is another update parameter.

2.4 Interaction structure

The 2-player language game models specified above only impose some constraints on the interaction behavior of *two* agents. For a population of $N > 2$ agents, this poses a question: who plays with whom and when? The job of interaction structure is to address this question.

A general approach is to use a joint probability distribution to specify when two agents play a game. The idea behind this approach is that at any given point in time, any pair has some chance to play a game, but that chance depends on a joint probability defined on the pair. Let there be N agents, then p_{ij}, ($\sum_{1 \leq i,j \leq N} p_{ij} = 1$), can be used to represent the probability of agent i and j being paired to play a game. Note that for simultaneous games, the probability matrix should be symmetrical, because the chance of agent i playing with agent j is the same as that of agent j playing with agent i. But for sequential games, the matrix does not need to be symmetrical. It is possible that for two agents i and j, agent i is always a speaker while agent j is always a hearer, as was the case in the baby-and-mother communication game described above.

Fig. 2.10 and 2.11 shows two cases of interaction structures among $N = 4$ agents. The first

case is for both simultaneous and sequential games in which every agent is equally likely to play a game with every other agent. The second case is only used for sequential games in which some pairs of agents have a better chance to play a game than other pairs, and more specifically, for the same pair of agents, one agent might play the role of speaker more often than it plays the role of hearer. For example, agent 2 has a chance of 0.2 of speaking to agent 4, but agent 4 never has a chance of speaking to agent 2.

	Agt 1	Agt 2	Agt 3	Agt 4
Agt 1	0	1/12	1/12	1/12
Agt 2	1/12	0	1/12	1/12
Agt 3	1/12	1/12	0	1/12
Agt 4	1/12	1/12	1/12	0

Figure 2.10: Symmetrical joint probability matrix of an interaction structure.

	Agt 1	Agt 2	Agt 3	Agt 4
Agt 1	0	.1	.1	.1
Agt 2	.2	0	0	.2
Agt 3	0	0	0	.2
Agt 4	0	0	.1	0

Figure 2.11: Asymmetrical joint probability matrix of an interaction structure. Rows stand for speaker and columns for hearer.

The approach we take in this thesis. Since we are generally concerned with the overall degree of linguistic consensus in a population of interacting agents, another interesting modeling question is when do we choose to measure the overall coherence of the population in an ongoing sequence of agent-agent games? For convenience of analysis (see the next section), we will measure population coherence only after every agent has already played exactly one game with every other agent. This means that our model gives every agent the chance to be paired with every other agent in the manner illustrated by Fig. 2.10.

Specifically, for N agents, in the case of simultaneous games, the population's coherence will be measured every $L = \binom{N}{2} = \frac{N(N-1)}{2}$ games, and in the case of sequential games, the coherence will be measured every $L = N(N-1)$ games. As we will show in later chapters, we will call L the length of an *iteration*; that is, during an iteration, there are L games or interactions.

2.5 Coherence as a measure of linguistic consensus

Given N agents, suppose their states are $b_1, \cdots, b_N$. Then a general definition of population coherence can be given by

$$\phi(b_1, \cdots, b_N) = \frac{1}{N(N-1)} \sum_{i \neq j} sim(b_i, b_j)$$

where $0 \leq sim(b_i, b_j) \leq 1$ is the similarity between the states of two agents i and j. Depending on contexts, the definition of $sim(b_i, b_j)$ will be different, as we will show in later chapters. For example, in the case that state b is a vector such as a probability distribution which was illustrated at the end of Section 2.3, $sim(b_i, b_j)$ can be defined as the cosine between two vectors b_i and b_j, namely, $sim(b_i, b_j) = \cos(b_i, b_j)$.

Given a specific definition of coherence, we can characterize how much consensus a population has over time. Denote by $\phi^{(t)}$ the coherence at time step t (i.e., at the t-th iteration, see the previous section), then the collective dynamics of the whole population can be given by

$$\phi^{(t)} \mapsto \phi^{(t+1)}.$$

The procedure of a population of agents reaching or approximating its maximum linguistic coherence is called *self-organization* of language. Characterizing the collective dynamics in various self-organizing language models, in terms of coherence, will be the main task of this thesis.

2.6 Simultaneous and sequential language game models

We have presented the details of the components in the self-organizing language framework, and now we can put them together to form two general self-organizing language models: simultaneous and sequential models. These are shown by Fig. 2.12 and 2.13, and they constitute the basic computational models for our studies in this thesis. Specifically, the simultaneous model will be used in Chapter 3, Reaching Word Consensus, and the sequential model will be used in Chapter 4, Reaching Coherent Communication, and Chapter 5, Reaching Grammar Consensus.

Settings

(1) population: N agents $\{1, \cdots, N\}$

(2) game model: a 2-player simultaneous game that includes

- two agents: s and r
- n available actions for both agents: $\{a_1, \cdots, a_n\}$
- a payoff function: $\pi_i(a_s, a_r)$ for each agent $i \in \{s, r\}$

(3) agent learning model

- an internal state b
- a decision function $a = f(b)$
- a state update function $b' = g(b, a, \pi)$

(4) a coherence measure on the consensus of the population (see Section 2.5): ϕ

Initialization

all agents' initial states are randomized

Iterations (each iteration includes $N(N-1)/2$ interactions (see Section 2.4 for details).)

During each interaction

1. two agents, s and r, are paired according to Section 2.4
2. they play the 2-player game by each taking an action a_i, $i \in \{s, r\}$, based on: $a_i = f(b_i)$
3. each agent receives a payoff $\pi_i(a_s, a_r)$
4. each agent updates its state based on the received payoff $b_i' = g(b_i, a_i, \pi_i)$

After each iteration, take a snapshot of the population coherence $\phi(t)$

Figure 2.12: A general simultaneous language game model.

2.7 Discussions

The presented self-organizing language framework and the corresponding simultaneous and sequential models are very general. When the 2-player language game in the framework is replaced

Settings

(1) population: N agents $\{1, \cdots, N\}$

(2) game model: a 2-player sequential game that includes

- two agents: a speaker, denoted by s, and a hearer, denoted by r
- m types for the speaker: $x_1, \cdots, x_m$, each with a probability $p(x)$
- n actions for the speaker: $\{y_1, \cdots, y_n\}$
- l actions for the hearer: $\{z_1, \cdots, z_l\}$
- a payoff function: $\pi_i(x, y, z)$ for each agent $i \in \{s, r\}$,
  where x, y, z are the speaker's type, the speaker's action and the hearer's action, respectively

(3) agent learning model

a. as a speaker, the agent has
- an internal state b_s
- a production function $f_s : (x, b_s) \mapsto y$
- a state update function $g_s : (b_s, x, y, \pi_s) \mapsto b'_s$

b. as a hearer, the agent has
- an internal state b_r
- an interpretation function $f_r : (y, b_r) \mapsto z$
- a state update function $g_r : (b_r, y, z, \pi_r) \mapsto b'_r$

(4) a coherence measure on the consensus of the population (see Section 2.5): ϕ

Initialization

all agents' initial states are randomized

Iterations (each iteration includes $N(N-1)$ interactions (see Section 2.4 for details).)

During each interaction

1. two agents are paired according to Section 2.4, one as speaker and one as hearer
2. they play the 2-player sequential game
   a. nature moves first to determine the type x of the speaker
   b. the speaker makes an action y based on its state b_s and its type x: $y = f_s(x, b_s)$
   c. the hearer makes an action z based on its state b_r and the speaker's action y: $z = f_r(y, b_r)$
3. each agent receives a payoff $\pi_i(x, y, z)$, $i \in \{s, r\}$
4. each agent updates its state based on the received payoff
   a. speaker's new state $b'_s = g_s(b_s, x, y, \pi_s)$
   b. hearer's new state $b'_r = g_r(b_r, y, z, \pi_r)$

After each iteration, namely $N(N-1)$ interactions, take a snapshot of the population coherence $\phi(t)$

Figure 2.13: A general sequential language game model.

with a different game, the framework or models can be applied to contexts that are beyond self-organizing language systems. For example, the simultaneous model can be used in the study of the evolution of cooperation, by specifying the 2-player game as the Prisoner's Dilemma game (Axelrod, 1984; Nowak and May, 1992). Studying other games such as the Prisoner's Dilemma might result in a totally different collective dynamics. We want to emphasize that the games to

be studied in this thesis have a common feature: the agents try to reach an agreement on using a common language aspect (word, vocabulary, or grammar). To see this, recall that in the payoff specification illustrated in the examples of both simultaneous and sequential games (Fig. 2.5 and 2.6), both agents get the best payoff when they agree on the usage of words or meanings. In addition, it is this common feature that makes sense of the notion *coherence* or *consensus*.

We also want to mention that there are some self-organizing language systems that are not covered by this framework. The currently widely used social tagging systems (e.g., Flickr[2] and Del.icio.us[3]) are examples of such systems. When a user tags his documents in a social tagging system, usually he will not modify those tags in the future. In addition, there is no notion of playing games and receiving payoffs. However, social tagging systems do demonstrate self-organization properties (Mathes, 2004; Golder and Huberman, 2006). This kind of self-organization behavior can be explained by other models. For example, considering that when a user tags a document he can tag the document using words already familiar to him, or using common tags shared by other users, then we may adopt models such as preferential attachment network models (Barabasi and Albert, 1999) or source-item model (Egghe and Rousseau, 1990). Such alternative models for different phenomena as described here are not treated further in this thesis.

2.8 Summary

In this chapter we presented a game-based self-organizing language framework, including game model, agent learning model, interaction structure, and population coherence. To distinguish the significant difference between two basic types of games: simultaneous and sequential games, we also introduced two general self-organizing language models, called *simultaneous language game* and *sequential language game* models. The simultaneous model will be used in Chapter 3 for the case study of word consensus, and the sequential model will be used in Chapter 4 for the case study of coherent communication and in Chapter 5 for grammar consensus.

[2]http://www.flickr.com/
[3]http://del.icio.us/

Chapter 3

Case 1: Reaching Word Consensus

3.1 Introduction

In this chapter we present our study of the first case of linguistic consensus: reaching word consensus. In the word consensus problem, a group of agents each need to describe an object (or concept, or meaning) using a single word from their vocabulary (which is a set of words). All the agents have the same vocabulary, but of course they may initially use different words in the vocabulary to describe the object. The agents' job is to reach an agreement on using one common word to represent the object.

The agents are adaptive—they can change their chosen word based on interactions they have with each other. So a key question is how can we design adaptive mechanisms for such agents so that they can converge from their initially different word choices to using one common word. In the design of the agents, there are four factors that may affect the convergence:

1. Population size: the number of agents;
2. Vocabulary size: the number of words in the vocabulary;
3. Learning mechanism: how agents change their description word;
4. Interaction structure: who interacts with whom.

This chapter aims at studying how the population size and vocabulary size affect the convergence under a minimalist learning mechanism called the *win-stay lose-shift* (WSLS) rule (Matsen and Nowak, 2004) and an all-to-all interaction network. We will construct a computational model

using the self-organizing language framework presented in Chapter 2, and then conduct computer simulation and mathematical analysis to find the conditions under what the agents can converge to a common word and to understand their convergence speed if the convergence conditions are satisfied.

In the next section we give a brief review of related work. Then we set up a WSLS learning model and specify our research questions. Next we concentrate presenting our computer simulations and mathematical analyses of the questions. We close with a summary of the chapter.

3.2 Related work

The word consensus problem studied in this chapter originated from the computational study of the emergence of *social conventions*, which studies how a group of agents can come to reach a global agreement on a common strategy of action (such as which side of the road to drive on) by using only locally available information (Shoham and Tennenholtz, 1993; 1997). Obviously, in the context of word consensus, words are just kinds of social conventions that are used in communication activities.

The study of conventions can be traced back to the work by Lewis (1969), who proposed using game theoretical frameworks to study the conventional aspects of language and meaning. In the last decade, various computational models of the emergence of social conventions have been introduced to show that a population of agents can converge to adopting one social convention (Shoham and Tennenholtz, 1993; 1997; Kittock, 1993; Walker and Wooldridge, 1995; Delgado, 2002). In relation to the four factors presented in the introduction (population size, vocabulary size, learning mechanism, and interaction structure), all of these studies assumed that the space of possible conventions over which the agents must agree (called the *potential agreement space* in a recent general model of multi-agent agreement (Lakkaraju and Gasser, 2006)) is limited to only two possible conventions. This translates into a two word limitation on the combined vocabulary of all agents in the case of word consensus, which is a very limiting constraint. Of course, all

studies supposed there are many agents, namely, a large population.

The principal difference among these studies lies in the learning mechanisms and interaction network they employed. For example, the work by Shoham and Tennenholtz (1993,1997) and by Walker and Wooldridge (1995) focused on studying various learning mechanisms while keeping the interaction network as simple as possible (they use an all-to-all connection network). Shoham and Tennenholtz were able to show the agents using a learning algorithm called *highest cumulative reward* (HCR) can converge to using a common convention. Walker and Wooldridge studied sixteen specific mechanisms using computer simulation and found some interesting unexpected results which implied how much further we need to pursue our understanding of this complex topic. Along dimensions other than variance in the learning mechanism, Kittock (1993) and Delgado (2002) have begun investigating the role played by interaction structure. Kittock showed that there is an important effect of using interaction networks that are not fully connected. Delgado then investigated the case of complex social networks such as small-world networks (Watts and Strogatz, 1998) and scale-free networks (Barabasi and Albert, 1999) and compared the efficiency of convention emergence in the two-convention case under these complex networks.

The most influential of these studies[1] is the model of Shoham and Tennenholtz (1997). Our model is based on theirs. In their design of a mechanism for the HCR learning rule, an agent has to remember a vector of payoffs, each entry of which represents the total payoff that the agent has received on a word in the past interactions with other agents. When there are many words this vector is large and the effort of maintaining statistics on all words is high. (This is not a problem for existing work because, as we mentioned above, existing studies on the emergence of social conventions have a common feature: they only studied the case of two conventions.)

To overcome the limitations in this their model, in this thesis we propose to use the WSLS learning rule—a simplest stochastic learning strategy (Posch et al., 1999)—in which each agent only needs to remember three things: its most recently-used word, the number of times that word

[1] According to search results from Google Scholar with query "emergence convention" on December 10th 2006, Shoham and Tennenholtz (1997) has the most citations (91 citations).

has been used since it was chosen, and the number of successes from using the word since it was chosen. The use and success statistics yield the success ratio of the word during the period since it was chosen, and they are re-initialized when a new word is chosen. The next section gives the details of the WSLS learning model.

3.3 Reaching word consensus using win-stay lose-shift learning

In terms of the self-organizing language framework given in Chapter 2, here we present a word consensus model that focuses on the following two components: (1) a 2-player word consensus game, and (2) an agent learning model (WSLS model).

Game model. The 2-player word consensus game is designed as a simultaneous game between two agents. All agents have the same vocabulary (i.e., action set, in terms of game theory) that consists of n words (or actions). Let the words be (w_1,$\cdots$,w_n). The payoff in the game is defined as follows: if the agents use the same word, then both receive a positive payoff of 1, otherwise 0. Fig. 3.1 gives the payoff matrix of the 2-player game. (We use font w_j to mean a word in the vocabulary, and font w_i to indicate a word submitted by agent i.) In the payoff matrix, each cell has two numbers (π_1, π_2), indicating the payoffs of the two agents. The payoff matrix in the figure tells that the two agents need choose the same word to get the best payoff.

A general agent model. Here we present a general agent model, and later we will present the specific WSLS model. In a general agent model, every agent has three components: a state b, a decision function f, and a state update function g. We suppose that all agents have the same form of decision and state update functions. The difference among the agents is reflected in their state.

Fig. 3.2 shows how these components work together in a word consensus game played by two

		Agent 2 (π_2)			
		w_1	w_2	$\cdots$	w_n
Agent 1 (π_1)	w_1	1,1	0,0	$\cdots$	0,0
	w_2	0,0	1,1	$\cdots$	0,0
	$\vdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
	w_n	0,0	0,0	$\cdots$	1,1

Figure 3.1: Payoff matrix of a 2-player word consensus game.

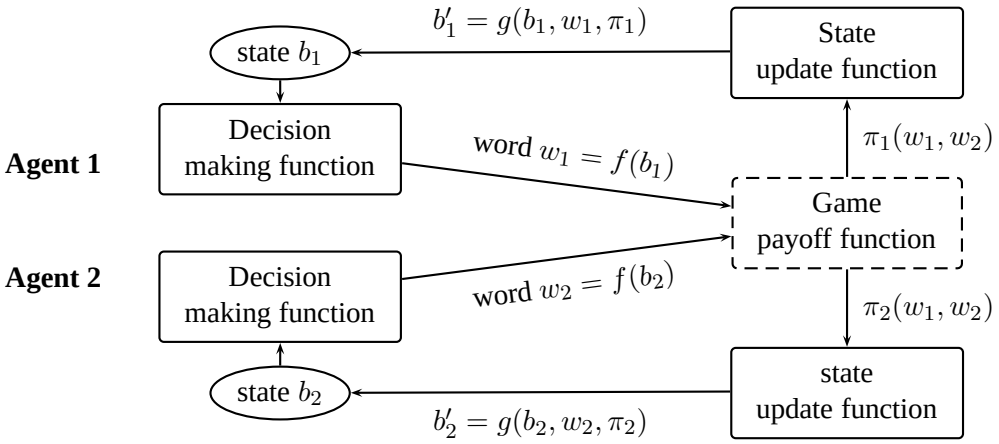


Figure 3.2: Agent model in a word consensus game.

agents. Denote the states of the two agents by b_i, $i = 1, 2$, then their actions (i.e., words) will be

$$w_i = f(b_i), \qquad i = 1, 2.$$

After the two agent submit their words, they each receive a payoff given by

$$\pi_i = \pi_i(w_1, w_2) = I(w_1 = w_2), \qquad i = 1, 2, \tag{3.1}$$

where $I(w_1 = w_2)$ is the indicator function that outputs 1 if $w_1 = w_2$, otherwise 0 (Cover and Thomas, 1990).

Then the two agents update their states from b_i to b_i' based on the received payoff in the form of

$$b_i' = g(b_i, w_i, \pi_i), \qquad i = 1, 2.$$

The WSLS learning model. Here we describe how to implement the above general agent model using the WSLS rule. In this model, the agent state b is represented as a triple

$$b =< w, u, v >,$$

where w is the word used by the agent in the previous game and will be called *recently-used word*, u and v are the number of *uses* and *successes* of the word w since w has been used consecutively by the agent. (In the first game that an agent participates, its u and v are set to 0, and there is no recently-used word.)

According to the WSLS rule, the decision function f which decides which word to use is stated as follows. If the success ratio (defined as the ratio of the successes to the uses, $\frac{v}{u}$) of the recently-used word is above some *threshold*, the agent will keep using the word; otherwise the agent will choose a random word from the vocabulary. In WSLS learning, the threshold is called *aspiration level*. Denote by α the aspiration level, then the decision function f can be formally represented as

$$\begin{aligned} f(b) &= f(< w, u, v >) \\ &= \begin{cases} w & \text{if } \frac{v}{u} \geq \alpha, \\ \text{a randomly chosen word} & \text{else (including the case of } u = 0). \end{cases} \end{aligned} \tag{3.2}$$

And, the state update function g is stated as follows. Suppose w is an agent's recently-used word, and u and v are the uses and successes of the word. After a game, the agent will increase the uses u by 1. If the agent receives from the game a positive payoff π, it will increase the successes v by 1; otherwise, if the ratio of the successes to the uses (i.e., $\frac{v}{u}$) is below the aspiration level α,

the agent will set both numbers to 0. Formally, the agent state is updated based on the following rule

$$(u', v') = \begin{cases} (u+1, v+1) & \text{if } \pi > 0, \\ (u+1, v) & \text{if } \frac{v}{u+1} \geq \alpha, \\ (0, 0) & \text{else.} \end{cases} \tag{3.3}$$

Coherence. In Section 2.5 of Chapter 2, we stated that: for N agents, suppose their states are $b_1, \cdots, b_N$, then a general definition of population coherence can be given by

$$\phi(b_1, \cdots, b_N) = \frac{1}{N(N-1)} \sum_{i \neq j} sim(b_i, b_j) \tag{3.4}$$

where $0 \leq sim(b_i, b_j) \leq 1$ is the similarity between the states of two agents i and j and its definition depends on contexts.

In the context of the WSLS model, the definition of the similarity is given as follows

$$\begin{aligned} sim(b_i, b_j) &= sim(< w_i, u_i, v_i >, < w_j, u_j, v_j >) \\ &= \begin{cases} 1 & \text{if } w_i = w_j, \\ 0 & \text{else.} \end{cases} \end{aligned} \tag{3.5}$$

A computational model of reaching word consensus. Fig. 3.3 gives the computation model of reaching word consensus using the WSLS rule. This model is based on the general one given in Fig. 2.12 of Section 2.6 of Chapter 2. The simulations conducted in the next section will be based on this word consensus model.

3.4 Specific research questions

From the concrete computational WSLS model given in Fig. 3.3, we can see that there are three input parameters: the number of available words n, the number of agents N, and the aspiration level α; and one output: population coherence $\phi(t)$ at some time point t. Now, we ask the following

Settings

population: N agents
language complexity: n available words
agent state: each agent i has a state $b_i = (w_i, u_i, v_i)$
learning parameter: aspiration level α

Initialization

each agent's recently-used word w is randomly chosen from the n words, and $(u, v) = (0, 0)$

Iterations (each iteration includes $N(N-1)/2$ interactions (see Section 2.4 of Chapter 2 for details).)

During each interaction

1. two agents, denoted by 1 and 2, are paired according to Section 2.4
2. they play the word consensus game by each using a word $w_i = f(b_i)$, $i = 1, 2$ (see Eq (3.2))
3. each agent receives a payoff $\pi_i(w_1, w_2)$ (see Eq (3.1))
4. each agent updates its state based on the received payoff $b_i' = g(b_i, w_i, \pi_i)$ (see Eq (3.3))

After each iteration, record the population coherence $\phi(t)$ (see Eqs (3.4) and (3.5))

Figure 3.3: The WSLS computational model.

questions.

1. Is it possible for the agents to eventually reach word consensus?

2. What are the conditions that can make the agents converge to using one common word? Since in the design of WSLS rules, a critical question is how to set the aspiration level α (if α is too low, it is very likely for the agents to diverge into using different words; if α is too high, the agents may never converge to using one common word), therefore, in particular, we ask, for a given number of words n and a number of agents N, what is the minimum aspiration level α that can make the agents converge to one common word with at least a coherence level of $\rho = \lim_{t\to\infty} \phi(t)$?

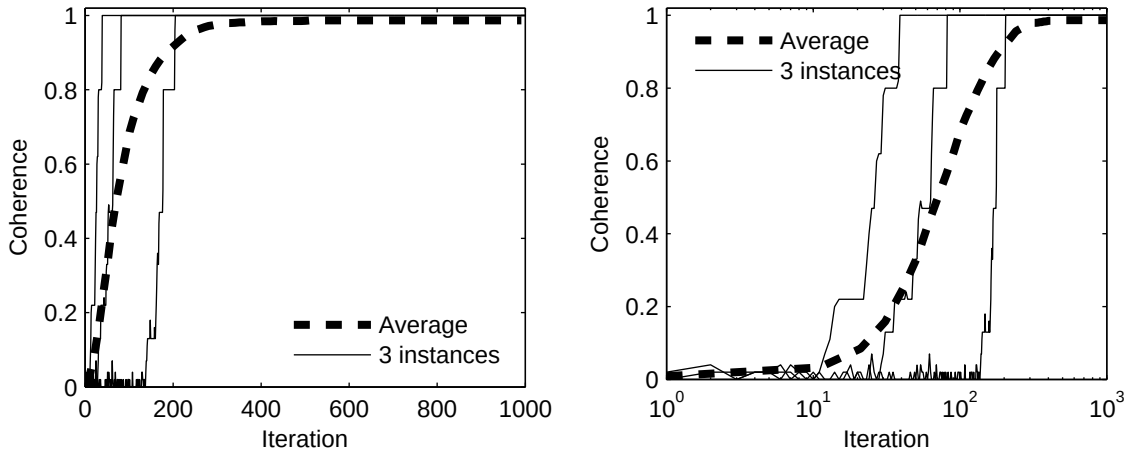
3. How much time is needed for achieving a given level of coherence if the above conditions are satisfied?

Research methods. We will use both computer simulation and mathematical analysis to study the above questions. For the details on the limitations and advantages of using simulation and

mathematical analysis, and the procedure of how to combine the two approaches, please see Section 1.4 of Chapter 1.

3.5 Simulations

In this section, we show that the agents in the WSLS model can converge to some common word when the aspiration level α is chosen appropriately. To show this, four experiments are made. The first experiment aims to show how the coherence among agents changes over time, for a given aspiration level (such as $\alpha = 0.15$). Fig. 3.4 shows the dynamics of the model, for a setting of 10 agents, 100 words, and an aspiration level 0.15. The simulation shows the agents using the WSLS rule can converge to the same word. In the figure, the dashed thick lines show the simulation result obtained by averaging 1000 runs, 3 of which are shown in thin lines. A log timescale in subfigure (b) is used for clarifying the detail of the dynamics in the initial period (during the first 10 or 20 iterations).



(a) Linear timescale (b) Log timescale

Figure 3.4: Simulated dynamics of the WSLS model for 10 agents, 100 words, and an aspiration level of 0.15.

The second experiment is designed to explore how different values of aspiration level affect the convergence properties (convergence speed and eventual coherence level). The simulation results

are shown in Fig. 3.5. From the figure, we can see that when the aspiration level is set appropriately (such as $\alpha = 0.15$ or $\alpha = 0.2$), the agents can achieve a coherence level of at least 0.95 at the 500-th iteration. When the aspiration level is set too low (such as $\alpha = 0.05$ or $\alpha = 0.1$), the agents are easily satisfied and so can quickly converge to a non-optimal coherence state (such as 0.45). When the aspiration level is set too high (such as $\alpha = 0.3$), the agents find it difficult to get satisfied and thus their words are switched back and forth. As a consequence, they may never converge or it may take very long time to converge to a common word.

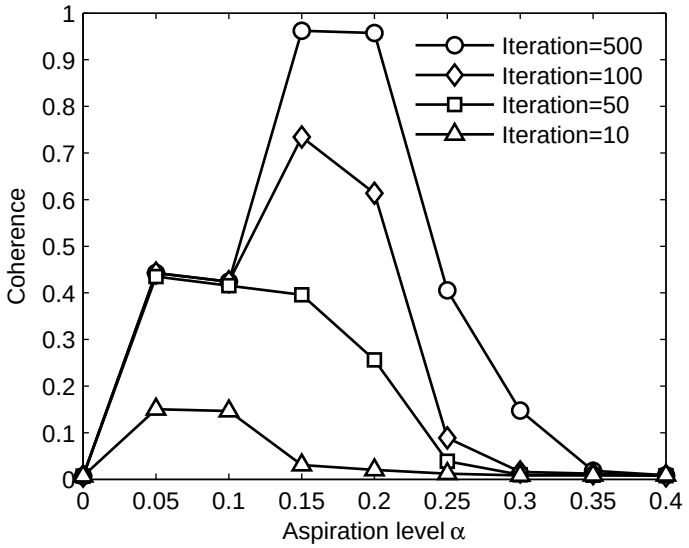


Figure 3.5: Coherence vs. aspiration level at different iteration points. Parameter setting: $N = 10$ agents and $n = 100$ words.

The third experiment is designed to explore how different numbers of agents affect the best aspiration level. By "best aspiration level", we mean the agents can converge to some common word with high probability (such as at least 0.85, i.e., at least a coherence level of 0.85) in a short time. For two different aspiration levels α_1 and α_2, suppose the agents can converge to the same coherence level of 0.85. Then if with the level of α_1, the agents can converge faster, we say α_1 is a better "aspiration level" than α_2. Fig. 3.6 shows that when the number of agents is given by $N = 20$ while the number of words is held constant (i.e., $n = 100$), the best aspiration level is $\alpha = 0.1$, compared to $\alpha = 0.15$ in the case of $N = 10$.

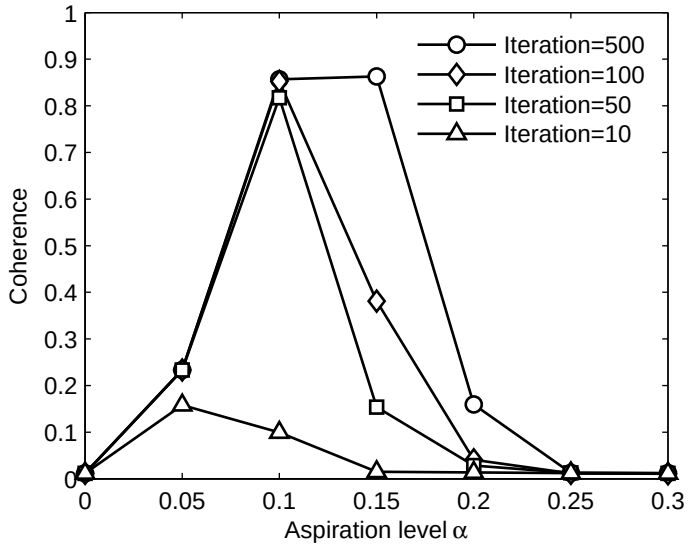


Figure 3.6: Coherence vs. aspiration level at different iteration points. Parameter setting: $N = 20$ agents and $n = 100$ words.

The fourth experiment is designed to explore how different numbers of words affect the best aspiration level. Fig. 3.7 shows that when the number of words, n, is small (e.g., $n < 50$), lowering the number of words increases the best aspiration level. For example, when there are only two words, it is obvious that the success ratio has to exceed 0.5 (i.e., at least half of population shares the same word). When the number of words is large enough (e.g., $n > 50$), there is no significant difference in the best aspiration level for $n = 50$ or $n = 1000$. We say "no significant difference" because in our simulations the best aspiration level is obtained at a coarse scale of with an interval of 0.05.

From the simulations results obtained in the four experiments. We have the following general observations:

1. The agents can converge to some common word when the aspiration level is chosen appropriately;

2. There is a relationship between the best aspiration level, the number of agents N and the number of words n. When the number of words is fixed, the best aspiration level depends on the numbers of agents. However, when the number of agents is fixed, the best aspiration

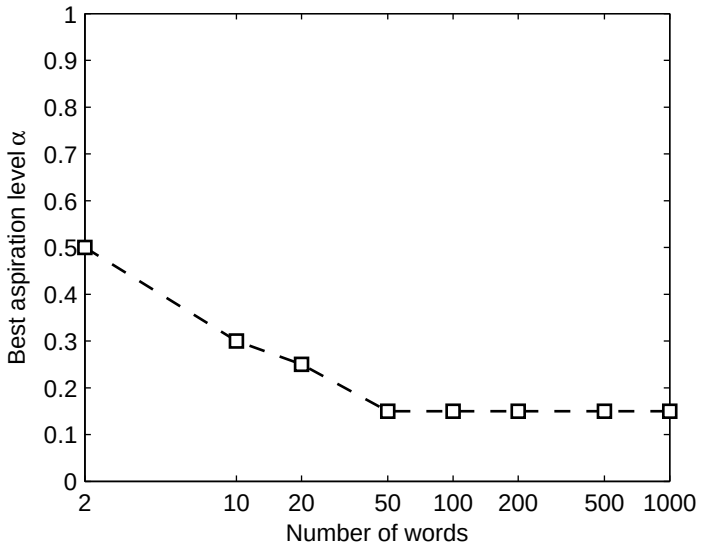


Figure 3.7: Best aspiration level vs. number of words n. Parameter setting: $N = 10$ agents.

level is similar when n is large enough.

Obviously, the above experiments are only suggestive, not exhaustive. The analysis given in the next section aims at giving a comprehensive theoretical account of the relationship between these parameters.

3.6 Analysis

We will analyze the conditions for the agents to reach word consensus and how much time is needed for the consensus.

3.6.1 Settings for the analysis

The WSLS model presented above is based on a 2-player game model. For several reasons to be shown below, we convert it into an N-player game model (N is the population size, namely the number of agents). In a N-player model, all agents submit their words at the same time, and each

agent i receives a payoff that is defined as follows

$$\pi_i(w_1, \cdots, w_N) = \frac{\sum_{j \neq i} I(w_i = w_j)}{N-1}, \qquad i = 1, 2, \cdots, N \tag{3.6}$$

where w_k is the word submitted by agent k, and $I(w_i = w_j)$ is the indicator function. Obviously, in an N-player game, the payoff that an agent can receive has a much richer range (from 0, $\frac{1}{N-1}$, $\cdots$, to 1) than the payoff an agent can receive in a 2-player game (either 0 or 1).

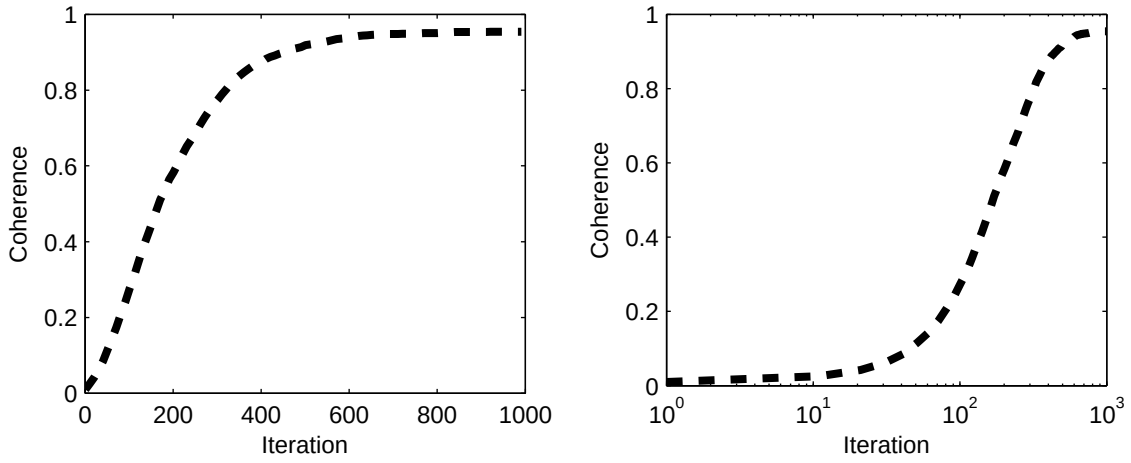
In addition to the advantage of having a rich payoff range, in the N-player game model, there is no need to specify an interaction structure that defines who plays with whom as in the 2-player game, because all the N agents participate the N-player game at the same time. These advantages of the N-player game model make it convenient to conduct analysis. In fact, Matsen and Nowak (2004) have already shown (though the authors did not explicitly state this in their article) that when all agents play the N-player game, they can converge to using one word. We will use some of their analysis techniques to explore the full relationship between the parameters specified in our questions (Section 3.4).

Before we do the analysis using the N-game model, however, we must (1) make sure that the N-player game model has a qualitatively similar behavior to the WSLS model based on the 2-player game; and (2) set up the conversion from the 2-player model to the N-player model.

Comparison between 2-player and N-player game models. To make sure that the N-player game model has a qualitatively similar behavior to the WSLS model based on the 2-player game, we build a computational model for the N-player game which is the same as the 2-player one give in Fig. 3.3 but with two exceptions. One exception is that now each iteration only contains one interaction. The other one is that the success ratio $\frac{v}{u}$ in the 2-player model now becomes the payoff (see Eq (3.6)).

Then, we run 1000 simulations for the N-player model under the same parameter settings that we used before for the 2-player model. By averaging the 1000 simulations, we obtain two graphs

that are shown in Fig. 3.8: one is on linear timescale and the other on log timescale. By comparing these graphs with the ones in Fig. 3.4 on the 2-player model, we can see that the dynamics of the N-player model has similar behavior to the 2-player model. The main difference is that the convergence time for the N-player model is much longer. This is because in the N-player model each agent only updates once in each iteration, while in the 2-player model each agent updates $N-1$ times in each iteration because an agent has to interact with every other agent.



(a) Linear timescale (b) Log timescale

Figure 3.8: Simulated dynamics of the N-player model for 10 agents, 100 words, and an aspiration level of 0.15.

Redefining coherence and aspiration level. In the Matsen-Nowak model, the measure used for analysis is different from our coherence measure given by Eqs (3.4) and (3.5). To situate their measure in our word consensus problem, we will call their measure the *largest number of agents sharing the same word*, or for brevity *largest cluster size*. Formally, the largest cluster size can be defined as

$$\psi(\mathbf{w}) = \psi(w_1, \cdots, w_N) = \max_i \sum_{1 \le j \le N} I(w_i = w_j).$$

For example, suppose among the 20 agents, 16 agents use the same word and 4 agents use some other word, then the largest cluster size is 16.

To use the analysis technique from their work, we will use largest cluster size as a measure

of coherence. The measure of largest cluster size is strongly related to the previous definition of coherence $\phi(\mathbf{w})$. To see this, let us normalize largest cluster size to

$$\bar{\psi}(\mathbf{w}) = \frac{\psi(\mathbf{w}) - 1}{N - 1}.$$

We can see that when all the words in $\mathbf{w}$ are the same, both measures $\phi(\mathbf{w})$ and $\bar{\psi}(\mathbf{w})$ can reach their maximum 1; when all the words in $\mathbf{w}$ are unique, both measures reach their minimum 0. The introduction of the new measurement ψ does make some small difference on the collective dynamics—we can show that $\phi(\mathbf{w}) \leq \bar{\psi}(\mathbf{w})$ always holds; however, as far as the qualitative dynamics behavior is concerned, there is no difference.

With the change of the measure from coherence ϕ to largest cluster size ψ, correspondingly, we need to change the aspiration level from a fractional number ($0 < \alpha < 1$) to an integer number denoted by K, as well as change the payoff from the fractional number given by

$$\pi_i(w_1, \cdots, w_N) = \frac{\sum_{j \neq i} I(w_i = w_j)}{N - 1}, \qquad i = 1, 2, \cdots, N$$

to an integer number given by

$$\pi_i(w_1, \cdots, w_N) = \sum_{j \neq i} I(w_i = w_j), \qquad i = 1, 2, \cdots, N.$$

After these conversions, the WSLS rule for the N-player game model will read as follows. An agent will keep using its recently-used word if the payoff from the word is at least K (namely, the agent shares its word with at least K other agents); otherwise uses a randomly chosen word.

3.6.2 Conditions for reaching consensus

After the conversion from 2-player model to N-player model and other corresponding changes given above, our first question turns into the following: for a given number of words n, a number

of agents N, and a desired coherence level of ρ, what is the minimum aspiration level K that can make the agents converge to a common word? Theorem 3.6.1 gives an analytical result to the question.

Theorem 3.6.1. *For a given number of words n and a number of agents N, to reach a desired coherence level ρ, the aspiration level K should be at least*

$$\min\{K : \frac{\binom{N-K-2}{K}}{(K+1)n^{K-1}} \leq 1/\rho - 1\}. \tag{3.7}$$

Before we give the proof to the theorem, it is worth giving some geometrical illustration to Eq (3.7). The two graphs shown in Fig. 3.9 give such an illustration about how the aspiration level K depends on the values of parameters N, n, and ρ. From the graphs, we have the following observations: (1) when the coherence level required for consensus is higher (e.g., $\rho = 0.95$ vs. 0.90), we may need to set the aspiration level K to be higher; (2) when there are many more words than agents, $K = 2$ will be sufficient for the agents to reach a high level of coherence.

In addition, to make connections between the aspiration level in the form of α and the aspiration in the form of K, we normalize K to

$$\bar{\alpha} = \frac{K-1}{N-1}.$$

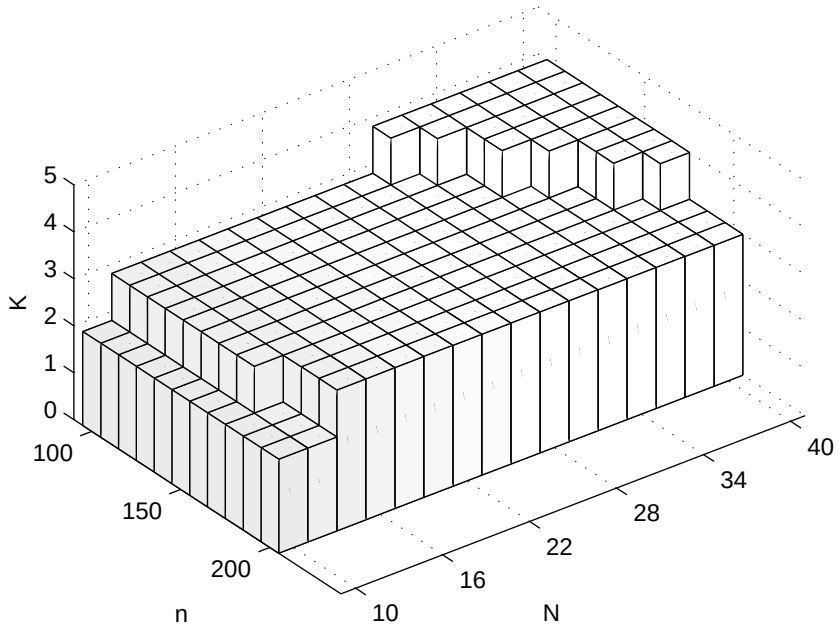
With the normalized $\bar{\alpha}$, Fig. 3.9(a) becomes Fig. 3.10. From this new graph, we can see that for the setting of $n = 100$, $N = 10$, and $\rho = 0.95$, the normalized aspiration level is $\bar{\alpha} = 0.11$. Now, take a look at the value of α in Fig. 3.5 for the same setting in the previous section, we can see that, $\bar{\alpha} = 0.11$ is the lower bound for the aspiration level α. If α is lower than 0.11, then the agents cannot reach consensus.

Now it is time to prove the theorem.

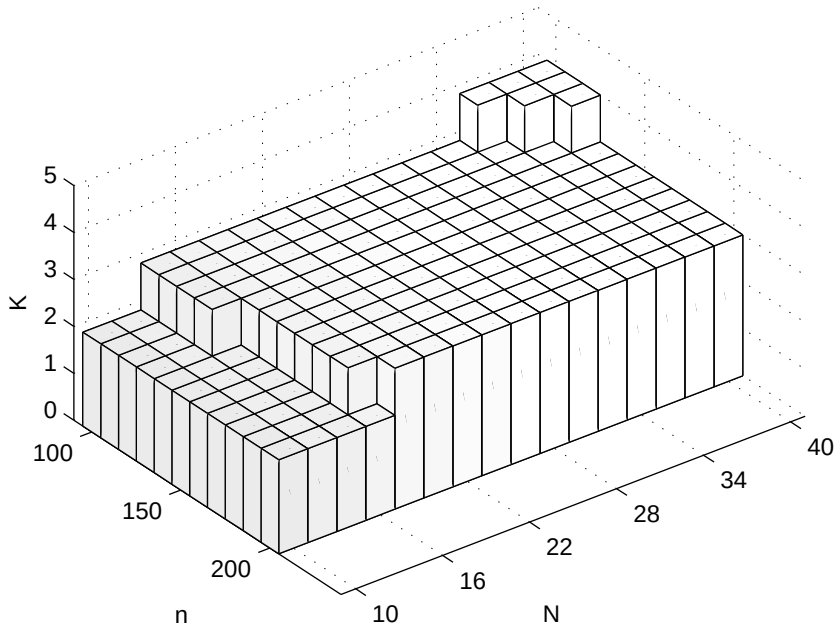
Proof. For convenience, we call a group of agents that share the same word a *cluster*. According to the WSLS rule in the N-player game model, if the word used by an agent is shared by at least K other agents then the agent will use the word forever (because all the agent in this cluster will

not change their word). A cluster with size of $K + 1$ or more—recall that K is the aspiration level—will be called an *aspired cluster*.

If the aspiration level K is too small (such as $K = 1$), it is very easy to form many aspired



(a) $\rho = 0.95$



(b) $\rho = 0.90$

Figure 3.9: How aspiration level K depends on the number of agents, N, the number of words, n, and the coherence level ρ.

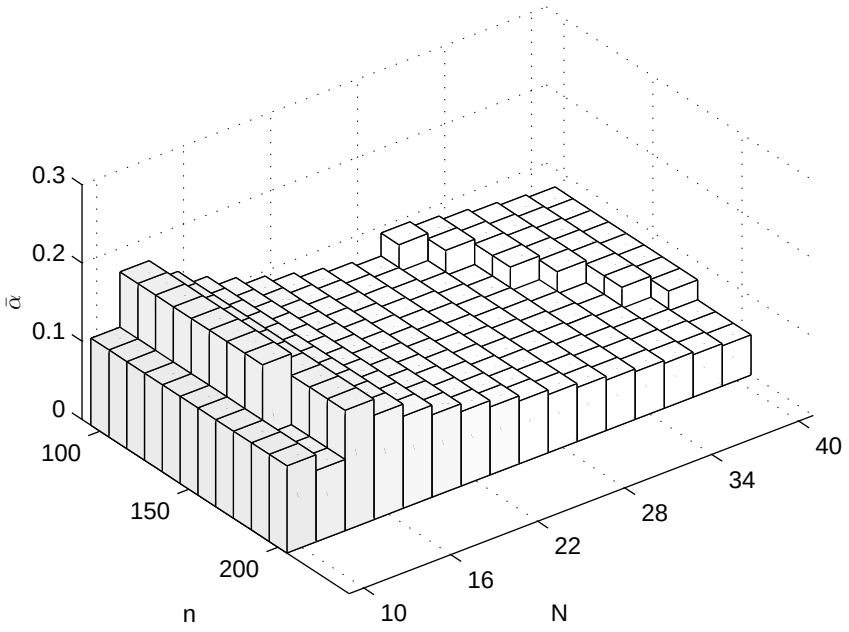


Figure 3.10: How aspiration level, in the form of $\bar{\alpha}$, depends on the number of agents N, the number of words n, when the coherence level ρ is held to 0.95.

clusters when there are many agents. If the aspiration level K is big enough (such as $K \geq 2$), it is not easy to form even one aspired cluster. Now suppose K is big enough, then according to the WSLS rule, once the first aspired cluster is formed[2], the agents in that cluster will stay in the cluster at the next time step. An agent that is not in the aspired cluster, which we call an *outsider*, will randomly choose another word at the next time step. There are three possible outcomes when an outsider agent randomly chooses a word:

1. the word is the same word shared by the agents in the existing aspired cluster; in this case, we say the agent is absorbed into the cluster;

2. this agent and some other outsiders form a second aspired cluster whose members share a word that is different from the one shared by the first cluster;

3. neither outcome 1 nor outcome 2.

[2]It is not easy to form one aspired cluster, so we can safely suppose that it is unlikely that two or more aspired clusters can be formed simultaneously.

Now, the critical point is that in order for all the agents to reach a consensus, the chance that the outsiders of the first aspired cluster to be absorbed into that cluster, denoted by p_1, should be much larger than the chance that they form a second aspired cluster, denoted by p_2. The calculation of p_1 can be given by

$$p_1 = 1 - \left(1 - \frac{1}{n}\right)^{N-K-1},$$

where $\left(1 - \frac{1}{n}\right)^{N-K-1}$ indicates the probability that no outsider—there are $N-K-1$ outsiders—uses the word shared by the first cluster. And, the calculation of p_2 can be approximately given by

$$p_2 = \binom{N-K-1}{K+1}\left(\frac{1}{n}\right)^{K+1}(n-1),$$

where $\binom{N-K-1}{K+1}$ means the number of the ways to form a second cluster of size $K+1$ from the $N-K-1$ outsiders, and $(\frac{1}{n})^{K+1}(n-1)$ means the chance of the $K+1$ "lucky" outsiders to fall in any one of the remaining $n-1$ words.

Given p_1 and p_2, the eventual coherence ρ that the agents can achieve can be approximately estimated by (the larger the difference between p_1 and p_2 ($p_1 - p_2$), the more accurate the estimation, because we neglect many other possible outcomes such as a third cluster in the future)

$$\rho \approx \frac{p_1}{p_1 + p_2}.$$

So, to achieve a coherence level of at least ρ, the following inequality should hold

$$\frac{p_1}{p_1 + p_2} \geq \rho.$$

Plugging p_1 and p_2 into the above equation, and making some simplifications such as replacing $(n-1)$ by n, we have the aspiration level K should be at least $\min\{K : \frac{\binom{N-K-2}{K}}{(K+1)n^{K-1}} \leq 1/\rho - 1\}$. ■

3.6.3 Time for reaching consensus

Above we have given the conditions for the agents to reach consensus. A general conclusion from the condition theorem is that if the number of words is much larger than the number of agents, the aspiration level K can be set to quite low such as $K = 2$ or $K = 3$. Here supposing the condition is satisfied, we want to know how much time is needed for reaching a given level of coherence. One motivation is that we know that a high value of the aspiration level K can guarantee that the agents will reach a consensus, but this may take infinite time to reach. For example, in the simulations given earlier in this chapter (e.g. Fig. 3.5), we can see that when the aspiration level is too high, then the agents can not reach any consensus within some limited number of time steps such as 500 iterations.

The computation of the time for reaching a given level of coherence is approached by deriving a dynamics equation that specifies on the average how much coherence can be obtained at a given time step. This is given in the following theorem.

Theorem 3.6.2. *In the N-player game model, if the conditions for reaching consensus is satisfied (see Theorem 3.6.1), then the expected dynamics of the coherence is described by the following equation:*

$$\psi(t) = \psi(0) + \frac{\binom{N}{K+1}}{n^K}\Big(1-\frac{1}{n}\Big)^{N-K-1} \times \sum_{i=1}^{t}\Bigg[\Big[1-\frac{\binom{N}{K+1}}{n^K}\Big(1-\frac{1}{n}\Big)^{N-K-1}\Big]^{i-1}\Big[K+1-\psi(0) + (N-K-1)\Big(1-(1-\frac{1}{n})\Big)^{t-i}\Big]\Bigg] \quad (3.8)$$

where $\psi(t)$ is the expected largest number of agents that share the same word at t-th iteration (i.e., at time step t). (The computation of $\psi(0)$ is given in the next theorem.)

Proof. For convenience, we call a group of agents that share the same word a *cluster*. Then $\psi(t)$ is the expected size of the largest cluster of agents that share the same word at t-th time step.

According to the WSLS rule in the N-player model, an agent will use a word forever if the word is shared by at least K other agents. In other words, there is an aspired cluster of size $K+1$.

Obviously, the time point when an aspired cluster is formed is a random variable, which can take any value from $1, 2, \cdots, t$. Denote this random variable by T, and denote by $P(T = i)$, $1 \leq i \leq t$, the probability that an aspired cluster is formed at time step i.

If p is the probability that the agents can form an aspired cluster in one time step for the first time before there is any other aspired cluster, then $P(T = i)$, the probability that an aspired cluster is formed at time step i, can be computed as follows

$$P(T = i) = p(1 - p)^{i-1}.$$

As to the computation of p, it can be calculated by

$$\begin{aligned} p &= \binom{N}{K+1} n(1/n)^{K+1}(1 - 1/n)^{N-K-1} \\ &= \frac{\binom{N}{K+1}}{n^K}(1 - 1/n)^{N-K-1}, \end{aligned} \tag{3.9}$$

where in Eq (3.9), $\binom{N}{K+1}$ indicates the number of the ways to form a cluster of size $K + 1$ from a population of N agents, $(1/n)^{K+1}(1-1/n)^{N-K-1}$ indicates the probability that exactly the $K+1$ agents in the cluster share the same word (considering that each agent randomly chooses a word from n words), and n means there are n words that can be shared.

When such an aspired cluster is formed, the agents in the cluster will keep their word forever. The agents outside the cluster (if not in another aspired cluster) will randomly update their word until they join the cluster (or another aspired cluster). Therefore, we want to know how the size of the aspired cluster will change once the aspired cluster has been formed. Note that, when the aspiration level K is large enough (such as 3), it is very unlikely that two or more aspired clusters will be formed.

Denote by $g(\tau)$ the cluster size at τ time steps after that the aspired cluster has already been formed. In other words, if the aspired cluster is formed at time step t_0, then $g(\tau)$ means the cluster size at time step $\tau + t_0$. For example, if the aspired cluster is formed at time $t_0 = 100$, and $\tau = 30$,

then $g(\tau = 30)$ means the cluster size at time step 130 from the beginning.

The computation of $g(\tau)$ is as follows. In each time step, an agent that is outside the aspired cluster will have a chance of $1/n$ to be absorbed into the cluster, given that the agent chooses a word at random from n words. So, during τ time steps, the probability for an outsider agent to join the cluster is $1 - (1 - 1/n)^\tau$. From $g(0) = K + 1$ and there are $N - K - 1$ outsider agents, we have $g(\tau) = (K + 1) + (N - K - 1)(1 - (1 - 1/n)^\tau$.

According to strong Markov property (Norris, 1997), the expected cluster size $\psi(t)$ should be

$$\begin{aligned}\psi(t) &= \psi(0) + \sum_{i=1}^{t} P(T = i)\Big(g(t-i) - \psi(0)\Big) \\ &= \psi(0) + \frac{\binom{N}{K+1}}{n^K}\Big(1 - \frac{1}{n}\Big)^{N-K-1} \times \sum_{i=1}^{t}\Bigg[\Big[1 - \frac{\binom{N}{K+1}}{n^K}\Big(1 - \frac{1}{n}\Big)^{N-K-1}\Big]^{i-1}\Big[K + 1 - \psi(0) \\ &\qquad + (N - K - 1)\Big(1 - (1 - \frac{1}{n})\Big)^{t-i}\Big]\Bigg] \quad (3.10)\end{aligned}$$

The proof is almost completed except that we have not shown the computation of $\psi(0)$, which will be given in the next theorem. ■

Theorem 3.6.3. *For N agents, n words, the expected largest number of agents that share the same word at the beginning, $\psi(0)$, is*

$$\psi(0) = N - \sum_{u=1}^{N-1} \sum_{\substack{0 \le k_i \le u \\ k_1 + \cdots + k_n = N}} \frac{N!}{k_1! k_2! \cdots k_n!\ n^N}.$$

Proof. The setting of N agents each choosing at random a word from the n available words $\{w_1, \cdots, w_n\}$, can be converted to a classical urn model, called *Maxwell-Boltzman urn model* (Rosen et al., 2000), which is about placing N distinguishable balls in n distinguishable urns. The result of the N agents each choosing a word can be characterized as such an event $(k_1, k_2, \cdots, k_n)$, in which k_1 agents share word w_1, k_2 agents share word w_2, . . ., and k_n agents share word w_n, with the restriction $\sum_{i=1}^{n} k_i = N$, $k_i \ge 0$. Or, in terms of the urn model, the event $(k_1, k_2, \cdots, k_n)$

means k_1 balls in urn 1, k_2 balls in urn 2, ..., and k_n balls in urn n.

The probability of the event $(k_1, k_2, \cdots, k_n)$ is given by

$$p(k_1, k_2, \cdots, k_n) = \binom{N}{k_1\ k_2\ \cdots\ k_n} / n^N = \frac{N!}{k_1! k_2! \cdots k_n!\ n^N},$$

where $\binom{N}{k_1 k_2 \cdots k_n}$ is a multinomial combinatorial number that gives the number of the ways that result in the event $(k_1, k_2, \cdots, k_n)$.

Let X_i be the number of agents that share word c_i. Then $(X_1, X_2, \cdots, X_n)$ is a random vector, from which we can give a definition to the largest number of agents that share the same word—also a random variable, denoted by U

$$U = \max\{X_1, X_2, \cdots, X_n\}$$

Then we have the cumulative distribution of U

$$P(U \leq u) = \sum_{\substack{0 \leq k_i \leq u \\ k_1 + \cdots + k_n = N}} p(k_1, k_2, \cdots, k_n)$$

And finally, the expected value of the largest number of agents that share the same word, is as follows

$$\begin{aligned}
E[U] &= \sum_{u=1}^{N} P(U = u) \\
&= 1 \times P(U = 1) + 2 \times P(U = 2) + \cdots + N \times P(U = N) \\
&= 1 \times \big(P(U \leq 1) - P(U \leq 0)\big) + 2 \times \big(P(U \leq 2) - P(U \leq 1)\big) + \\
&\quad + \cdots + N \times \big(P(U \leq N) - P(U \leq N - 1)\big) \\
&= NP(U \leq N) - P(U \leq 0) - P(U \leq 1) - \cdots - P(U \leq N - 1) \\
&= N - \sum_{u=1}^{N-1} P(U \leq u) \qquad \textit{since } P(U \leq N) = 1 \textit{ and } P(U \leq 0) = 0
\end{aligned}$$

$$
\begin{aligned}
&= N - \sum_{u=1}^{N-1} \sum_{\substack{0 \le k_i \le u \\ k_1+\cdots+k_n=N}} p(k_1, k_2, \cdots, k_n) \\
&= N - \sum_{u=1}^{N-1} \sum_{\substack{0 \le k_i \le u \\ k_1+\cdots+k_n=N}} \frac{N!}{k_1!k_2!\cdots k_n!\, n^N}.
\end{aligned}
$$

Thus complete the proof. ■

Remark. Though we give an explicit formula for $\psi(0)$, it is very difficult to compute it, especially when N and n are large numbers. For the case of $N = 20$ and $n = 100$, by using an approximate algorithm to the above theorem, we have obtained an approximation $\psi(0) = 1.964$. Actually, the average of 1000 runs of simulation shows that $\psi(0) = 1.962$.

Comparison between simulation and analytical results on dynamics. Fig. 3.11 gives the comparison between the simulation and analytical results. The parameter setting for obtaining the shown results is: $K = 3$, $N = 20$, and $n = 100$. The plot of the dashed simulation line is obtained by averaging 1000 simulations. From the figure, we can see that there is a trend for the theoretical result (as shown by smooth line) to eventually have higher coherence values than the simulation result. This is because the theoretical result is computed based on Theorem 3.6.2, and the theorem has an assumption that there are not two or more aspired clusters. When there is a chance for two or more aspired clusters to occur, the coherence will decrease (the maximum coherence is achieved when there is only one cluster). Clearly, the assumption does not always hold in simulations, thus the simulation result will eventually show lower coherence than the theoretical (ideal case) result. In addition, this difference only becomes significant when time tends to infinity, because the theoretical coherence will, according to our assumption, tend to 1 while the simulation result cannot.

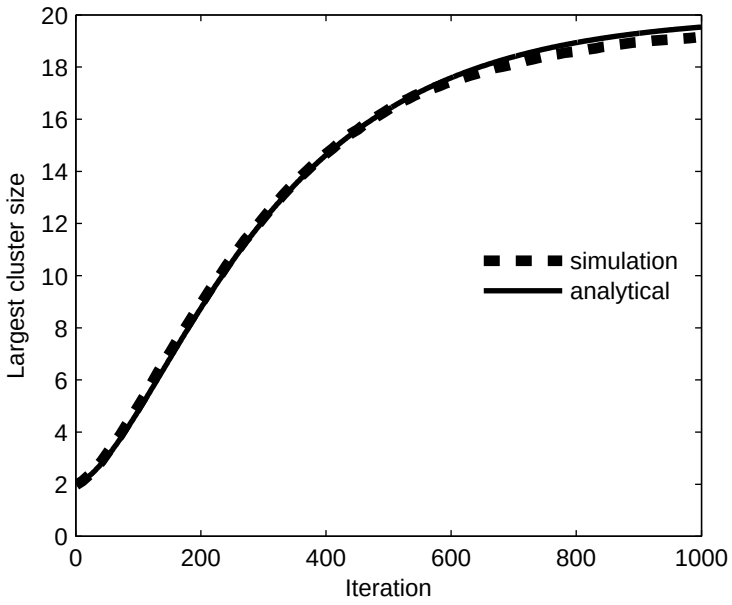


Figure 3.11: Comparison between the simulation and analytical results.

3.7 Summary

In this chapter we studied our first linguistic consensus case: reaching word consensus. The word consensus problem concerns how a group of agents can converge to a common word, out of a number of possible words, for representing a single shared meaning.

In terms of the self-organizing framework given in Chapter 2, we designed a word consensus model that focuses on the following two components: (1) the 2-player word consensus game, and (2) the agent learning model. In the design of the game, we defined its payoff function as follows: if two agents use the same word, both receive a positive payoff; otherwise 0. In the design of agents, the win-stay lose-shift (WSLS) learning rule was used. According to the WSLS rule, an agent will keep using its recently-used word unless the success ratio of the word is below some threshold.

Our work has the following contributions. First, we proposed a word consensus model in which agents make adaptations using the WSLS learning rule. We showed by computer simulation and mathematical analysis that agents in our model can converge to a common word under certain conditions, and we gave those conditions. We also gave a dynamics equation on how coherence

changes over time when the convergence conditions are satisfied. Second, though our work was motivated by (Shoham and Tennenholtz, 1997) and (Matsen and Nowak, 2004), compared to the Shoham-Tennenholtz model, ours requires a minimum memory load on the agents, and compared to the Matsen-Nowak model, our analytical result gives a comprehensive description of the relationship among all related parameters.

Chapter 4

Case 2: Reaching Coherent Communication

4.1 Introduction

In the word consensus problem we studied in the last chapter, we supposed there is only one meaning involved and the task of the agents is to reach an agreement on using one common word for representing the meaning. However, what will happen if there are multiple meanings involved? For example, suppose there are two meanings $\{hungry, thirsty\}$ and two words $\{a, b\}$. It is possible for the agents to use the same word, say a, to represent both meanings. If this happens, then when a speaker uses the word a to represent meaning $hungry$, it is possible for a hearer to interpret a as the other meaning $thirsty$.

In this chapter, to address the above situation where there are many meanings, we study another linguistic consensus case: coherent communication. In the coherent communication problem, there are a set of meanings and a set of words. A group of agents each have an encoding function that can map a meaning to a word, and a decoding function that can map a word to a meaning. The agents may have incoherent (or incompatible) encoding and decoding functions and thus they cannot communicate effectively. The job of the agents is to develop a coherent communication system so that the intended meaning of one agent can be correctly conveyed to another agent through their encoding and decoding functions.

The agents are supposed to be adaptive—they can change their coding functions based on interactions they have with each other. So a key question is how can we design adaptive mechanisms for such agents so that they can converge to a coherent communication system.

In the design of the agents, there are four factors that may affect the convergence which are

listed as follows.

1. Population size: the number of agents;
2. Vocabulary scale: the number of meanings and the number of words;
3. Learning mechanism: how agents change their coding function;
4. Interaction structure: who interacts with whom.

In this chapter, we will study how two agents (a sender and a receiver) can converge to coherent communication using a learning mechanism called *simple reinforcement learning* (SRL) rule. We aim at studying how the vocabulary scale and SRL learning parameters affect the convergence. We will construct a computational model using the self-organizing language framework presented in Chapter 2, and then conduct computer simulation and mathematical analysis to find the conditions under what the agents can converge to a coherent communication system and to understand their convergence speed if the convergence conditions are satisfied.

In the next section we give a brief review of related work. Then we set up a simple reinforcement learning model and specify our research questions. Next we concentrate on presenting our computer simulations and mathematical analyses of the questions. We close with a summary of this chapter.

4.2 Related work

The coherent communication problem studied in this chapter was pioneered by James Hurford (1989) in his seminal work on the computational study of the *evolution of vocabulary*. Since then, many studies has been conducted on the *evolution of vocabulary* or the *emergence of communication*, including, to cite a few of them, (Oliphant and Batali, 1997; Nowak et al., 1999; Smith, 2004; Lenaerts et al., 2005).

According to the learning mechanism, the existing work can be classified into two paradigms: *observational learning* and *reinforcement learning*. The observational learning mechanism is

mainly used in the study of vocabulary evolution through cultural transmission (Hurford, 1989; Oliphant and Batali, 1997; Nowak et al, 1999; Smith, 2004). In such cultural transmission models, it is supposed that there are two kinds of agents: parents and children, and vocabulary can be transmitted from parents to children. During the process of transmission, children acquire their vocabulary (the encoding and decoding functions) by learning (i.e., observing and generalizing) how adults produce and interpret words, and then they become the parents of the next generation. In this way the collective vocabulary of the agents and the corresponding communication coherence change over generations.

On the other hand, the reinforcement learning is mainly used in the study of the emergence of communication through self-organization (Steels, 1996; Kaplan, 2000, 2005; De Jong and Steels, 2003; Lenaerts et al., 2005). The mechanism of self-organization is similar to our framework presented in Chapter 2 (the difference is that ours is based on a game theoretical framework).

In this thesis we are particularly interested in reinforcement learning which only requires a minimum of feedback information for learning. Though the formulation of communication models underlying the studies using reinforcement learning is abstract and straightforward, there is no theoretical proof to show why or not a group of agents can converge to a coherent communication system. For example, a recent article by Lenaerts et al. (2005) has done a comprehensive study on using reinforcement learning for reaching coherent communication, but all the results were obtained using computer simulation. The difficulty of giving a theoretical proof might be explained by that each agent has dual functions (encoding and decoding functions) which make things complicated.

To make progress, we propose a minimum communication model in which there are only two agents: a sender and a receiver. Different from existing work, in our minimum model, the sender only has an encoding function and the receiver only has a decoding function. With this reduction, we hope we can achieve a thorough understanding of the theoretical aspects of the reinforcement learning model.

4.3 A formal communication model

The objective of this chapter is to study how agents can reach a coherent communication. First we need to clarify what we mean by communication. Our model of communication is adapted from Hurford (1989), which is the basis for most work in studying the emergence of communication or the evolution of vocabulary. In a simple communication model (see Fig. 4.1), there are m meanings $X = \{x_1, \cdots, x_m\}$, n words $Y = \{y_1, \cdots, y_n\}$, and two agents: sender and receiver. The sender has an encoding (or production) function $f_s : X \mapsto Y$ that can produce a word to represent a meaning, and the receiver has a decoding (or interpretation) function $f_r : Y \mapsto X$ that can output a meaning as an interpretation to a word. A basic assumption in the model is that the sender cannot convey his meaning to the receiver directly; rather, he can only convey a meaning via some word. To communicate a meaning, say x, the sender needs to call his encoding function f_s to convert the meaning into a word y, in the form of $y = f_s(x)$. For the receiver, once receiving the word y, she needs to call her decoding function f_r to convert the word back to a meaning $\hat{x}$, in the form of $\hat{x} = f_r(y)$. If the interpreted meaning $\hat{x}$ is the same as the original meaning x, then we say the communication between the sender and receiver over the meaning x is successful; otherwise it is failed. Later we will frame this communication process into a communication game.

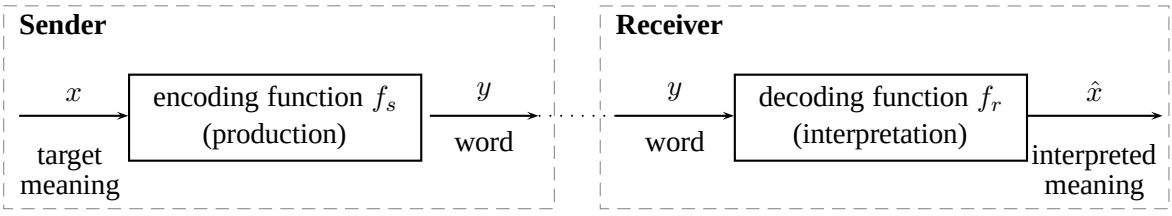


Figure 4.1: A communication model. Given a meaning x, the sender encodes it into a word. When receiving a word, the receiver decodes it to a meaning $\hat{x}$.

Remark. The meaning set is a discrete set, which implies that each meaning is atomic or primitive. The word set is also a discrete set. Another assumption is that the sender and receiver have the same meaning set X and the same word set Y.

Communication coherence. Here we give a definition of *communication coherence* between a sender and a receiver, which is the probability of having successful communication between the sender and receiver over any meaning. To be precise, let f_s be the encoding function of the sender, f_r be the decoding function of the receiver, and $p(x)$ be the probability of meaning x being used. Then we can define the communication coherence between the sender and receiver as

$$\phi(f_s, f_r) = \sum_{x \in X} p(x) I\Big(x = f_r(f_s(x))\Big),$$

where $I(x = x')$ is the indicator function, and thus $I\big(x = f_r(f_s(x))\big)$ indicates the communication coherence over one meaning x.

This coherence measurement will be used as the definition of population coherence in our minimum model that is to be given later (see Fig. 4.6).

A probabilistic implementation of coding functions. In the above communication model, the encoding and decoding functions f_s and f_r have not been specified yet. There are various ways to implement the two functions. An elegant mathematical treatment is to generalize the encoding function $y = f_s(x)$ to be a conditional distribution $p(y|x)$, which allows each word y to be used for representing a given meaning x with some probability. Similarly, the decoding function $x = f_r(y)$ can be generalized to another conditional distribution $q(x|y)$, which allows each meaning x to be used as an interpretation for a received word y. For convenience, the distribution $p(y|x)$ will be called the *encoding distribution* of the sender, and the distribution $q(x|y)$ will be called the *decoding distribution* of the receiver. With this treatment, we can easily define the communication coherence between a sender and a receiver over meaning x as

$$\phi_x(f_s, f_r) = \phi_x(p, q) = \sum_y p(y|x) q(x|y)$$

where $p(y|x)q(x|y)$ indicates the probability that meaning x can be successfully communicated via word y, and thus $\sum_y p(y|x)q(x|y)$ is the probability that meaning x can be successfully com-

municated via all possible words. Then, the expected or average communication coherence over all meanings is:

$$\phi(f_s, f_r) = \sum_x \Big[p(x)\phi_x(f_s, f_r)\Big] = \sum_x \left[p(x) \sum_y p(y|x)q(x|y)\right].$$

As an example, consider the case that there are three meanings $X = \{x_1, x_2, x_3\}$, and three words $Y = \{y_1, y_2, y_3\}$. The encoding and decoding distributions $p(y|x)$ and $q(x|y)$ are given in Fig. 4.2. The success ratio of communicating meaning x_1 via word y_1 will be: $p(y_1|x_1) \times q(x_1|y_1) = 0.5 \times 0.2 = 0.1$, and the total success ratio on communicating meaning x_1 is: $\phi_{x_1} = \sum_y p(y|x_1)q(x_1|y) = 0.1 + 0.4 + 0 = 0.5$. If we suppose that all meanings are uniformly distributed; i.e., $p(x_1) = p(x_2) = p(x_3) = 1/3$, then the communication coherence between such a sender and receiver is:

$$\phi = 1/3 \times (\phi_{x_1} + \phi_{x_2} + \phi_{x_3}) = 1/3 \times (0.5 + 0.2 + 0.42) = 0.37.$$

	y_1	y_2	y_3
x_1	.5	.4	.1
x_2	.4	.3	.3
x_3	0	.3	.7

(a) Sender's encoding distribution $p(y|x)$

	x_1	x_2	x_3
y_1	.2	.2	.6
y_2	1	0	0
y_3	0	.4	.6

(b) Receiver's decoding distribution $q(x|y)$

Figure 4.2: Encoding and decoding distributions. Note that the sum of any row is 1, indicating a distribution.

Coherent communication. For different encoding and decoding functions or distributions, the communication coherence $\phi(f_s, f_r)$ could be very different: it could be as low as 0 or as high as 1. When there are m meanings and n words, it is clear that the maximum coherence that any pair of encoding and decoding functions can achieve will be

$$\min\{1, \frac{n}{m}\}.$$

When a sender and a receiver can achieve their maximum coherence, the communication between them is called *coherent communication.* For example, Fig. 4.3 shows such a pair of encoding/decoding distributions by which the sender and receiver can achieve coherent communication.

	y_1	y_2	y_3
x_1	1	0	0
x_2	0	0	1
x_3	0	1	0

(a) Sender's encoding distribution $p(y|x)$

	x_1	x_2	x_3
y_1	1	0	0
y_2	0	0	1
y_3	0	1	0

(b) Receiver's decoding distribution $q(x|y)$

Figure 4.3: Illustration of a coherent communication system.

4.4 Reaching coherent communication using simple reinforcement learning

In terms of the self-organizing language framework given in Chapter 2, here we present a model for reaching coherent communication that focuses on the following two components: (1) a 2-player communication game, and (2) an agent learning model called *simple reinforcement learning* model.

Game model. The 2-player communication game is designed as a sequential language game which is illustrated by Fig. 4.4. From the figure, we can see that there are three players: nature, sender (he), and receiver (she). The player nature, who moves first, is added here for modeling that the sender has some private information that is unknown or uncertain to the receiver. (See Section 2.2.3 of Chapter 2 for details.) In our case of communication games, the private information of the sender is the meaning he wants to convey. The sender knows the meaning, but the receiver only knows the word produced by the sender. For example, in the figure, the left dashed line illustrates that the receiver cannot tell which path, x_1 or x_2, the word y_1 comes from. When the receiver interprets a word correctly, both agents get a positive payoff of 1; otherwise 0.

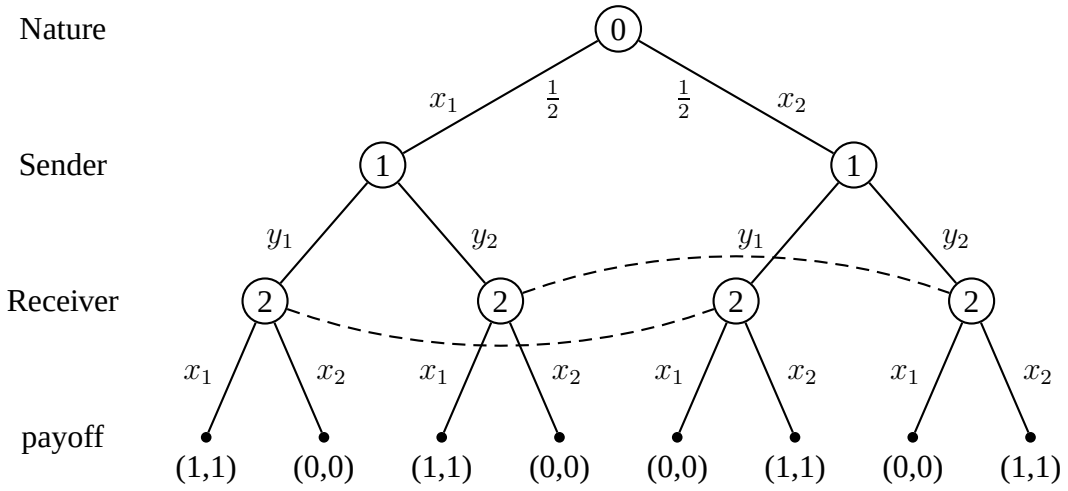


Figure 4.4: Communication game tree.

A general agent model. Here we present a general agent model, and later we will present the specific simple reinforcement learning model. In a general agent model, every agent has three components: a state b, a decision function f, and a state update function g. We suppose that all senders have the same form of decision and state update functions, and all receivers have the same form of functions too. Since an agent can play the role of sender as well as the role of receiver, so it is required that an agent has dual states and dual functions.

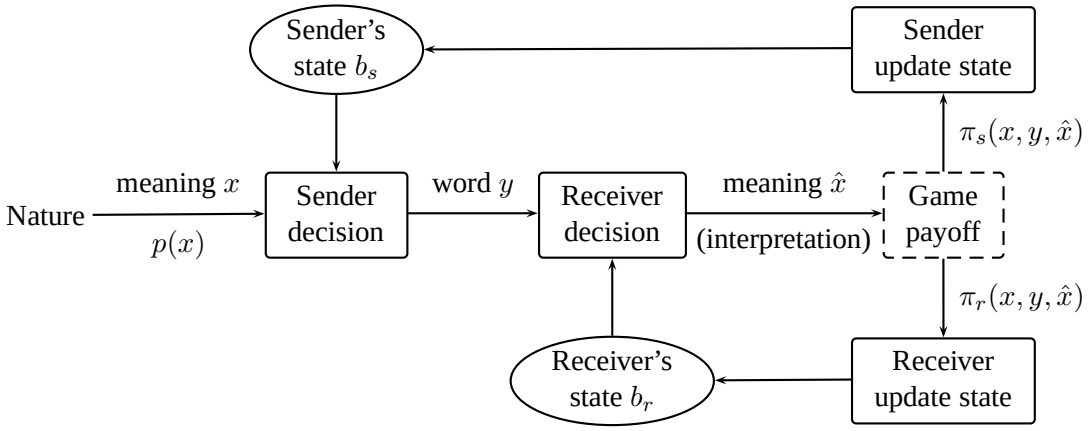


Figure 4.5: Agent model in a communication game.

Fig. 4.5 shows how these components work together in a communication game played by a

sender s and a receiver r. In a communication game, nature moves first, deciding which meaning the sender wants to convey, according to probability $p(x)$. When a meaning x is given, the sender will call his decision function f_s to encode the meaning x to a word y based on its internal state b_s in the form of

$$y = f_s(x, b_s).$$

Then, the receiver, when receiving the word y, will call her decision function f_r to decode the word y to a meaning $\hat{x}$ based on her internal state b_r in the form of

$$\hat{x} = f_r(y, b_r).$$

After the receiver gives an interpretation, both the sender and receiver receive a payoff that is given by

$$\pi_i = \pi_i(x, y, \hat{x}), \qquad i \in \{s, r\}.$$

And then, both agents update their state from b_i to b_i' based on the received payoff. For the sender, its update function g_s is used to update its current state b_s to a new one b_s' based on its current encoding $x \mapsto y$ and the received payoff π_s, in the form of

$$b_s' = g_s(b_s, x, y, \pi_s).$$

For the receiver, its update function g_r is used to update its current state b_r to a new one b_r' based on its current decoding $y \mapsto \hat{x}$ and the received payoff π_r, in the form of

$$b_r' = g_r(b_r, \hat{x}, y, \pi_r).$$

The simple reinforcement learning model. Here we present a learning model called *simple reinforcement learning*, which is the most widely used one in the study of reaching coherent communication (Steels, 1996; Kaplan, 2000; De Jong and Steels, 2003; Lenearts et al., 2005). Specif-

ically, the learning model we use here is adapted from Lenearts et al. (2005). In this learning model, the agent state is represented as an association matrix between meanings and words with $b(x, y)$ indicating the association strength or weight between meaning x and word y. Denote by $b_s(x, y)$ the sender's association matrix, and by $b_r(x, y)$ the receiver's matrix.

As shown in the above general agent learning model, the decision functions of the sender and receiver are somewhat different. For the sender, his decision function is given by

$$y = f_s(x, b_s) = \arg\max_{y'} b_s(x, y'), \tag{4.1}$$

which indicates for a given meaning x, he will choose the word y that has the strongest association with the meaning x.

For the receiver, her decision function is given by

$$\hat{x} = f_r(y, b_r) = \arg\max_{x'} b_r(x', y), \tag{4.2}$$

which indicates when receiving word y, she will interpret it as the meaning $\hat{x}$ that has the strongest association with the word y.

Suppose in a communication game, x is the intended meaning of the sender (i.e., the meaning decided by the external player nature), y is the word produced by the sender, and $\hat{x}$ is the meaning interpreted by the receiver, then the state update functions of the two agents are as follows. For the sender, his state update rule is

$$b_s(x, y) = \begin{cases} b_s(x, y) + \alpha & \text{if the payoff received from the game is positive} \\ b_s(x, y) - \beta & \text{else.} \end{cases} \tag{4.3}$$

And for the receiver, her update rule is

$$b_r(\hat{x}, y) = \begin{cases} b_r(\hat{x}, y) + \alpha & \text{if the payoff received from the game is positive} \\ b_r(\hat{x}, y) - \beta & \text{else.} \end{cases} \tag{4.4}$$

In these rules, the parameter $\alpha \geq 0$ is called *reward rate*, and the parameters $\beta \geq 0$ is called *punishment rate*. Note that the two agents have the same behavior.

A minimum simple reinforcement learning model. All computer simulations in various existing work have shown, though not systematically, that a group of agents designed as above can converge to a coherent communication system. However, there is no theoretical proof to show why (or not) a group of agents can converge to a coherent communication system. To make progress, we only study a minimum model in which there are only a sender and a receiver.

The minimum model is described in Fig. 4.6. In this model, there are four parameters: the number of meanings m, the number of words n, reward rate α, and punishment rate β. For convenience of analysis, we will set agents' punishment rate β to be larger than 1, which is the largest difference between any two initial association weights, to allow the agent to choose a different word (or meaning) in the next game in the case of failing to communicate in the current game. In addition, for convenience, we define the length of each iteration as m interactions. Note that m is the number of meanings.

4.5 Specific research questions

As we mentioned before, the minimum model given in Fig. 4.6 has four parameters: the number of meanings m, the number of words n, reward rate α, and punishment rate β. There is also an output from the model: the communication coherence $\phi(t)$ at the t-th iteration.

Now, we ask the following questions:

1. What are the conditions that can lead the agents to reach coherent communication? Specif-

Settings

population: 2 agents—a sender and a receiver
language complexity: m meanings $\{x_1, \cdots, x_m\}$, and n words $\{y_1, \cdots, y_n\}$
agent state: the association matrices of the sender and receiver are $b_s(x, y)$ and $b_r(x, y)$, respectively
learning parameter: reward rate α, and punishment rate β

Initialization

the association weights in both $b_s(x, y)$ and $b_r(x, y)$, are randomized to fall in the range $[0, 1]$

Iterations (each iteration includes m interactions).

During each interaction
1. a meaning x is chosen from $\{x_1, \cdots, x_m\}$ with probability $p(x)$
2. the sender represents the meaning by word $y = \arg\max_{y'} b_s(x, y')$
3. the receiver interprets the word as meaning $\hat{x} = \arg\max_{x'} b_r(x', y)$
4. if the interpreted meaning $\hat{x}$ is correct (i.e., $\hat{x} = x$)
   $b_s(x, y) = b_s(x, y) + \alpha$
   $b_r(\hat{x}, y) = b_r(\hat{x}, y) + \alpha$
   otherwise (i.e., $\hat{x} \neq x$)
   $b_s(x, y) = b_s(x, y) - \beta$
   $b_r(\hat{x}, y) = b_r(\hat{x}, y) - \beta$

After each iteration, namely m interactions, take a snapshot of the communication coherence $\phi(t)$

Figure 4.6: The minimum computational model of coherent communication using the simple reinforcement learning.

ically, for a given number of meanings, m, and a given number of words, n, what kind of values of reward rate α and punishment rate β can make the agents reach coherent communication?

2. How fast can the agents reach coherent communication if the conditions are satisfied?

A careful reader might notice that there is one more (kind of) parameter in the model. That is, the initial association matrices of the sender and receiver, b_s and b_r. The initial matrices could be parameters, but we will not consider them as parameters in our study, for the following reasons. First, we aim at studying the average or expected dynamics behavior of the agents. Second, we want to know for any given initial association matrices, what ranges of the settings of (m, n, α, β) would make the agents converge to a coherent communication. Third, the initial matrices could be easily set to the ones that are already converged.

Research methods. Like what we did in the last chapter, we will use both computer simulation and mathematical analysis to study the above questions. For the details on the limitations and advantages of using simulation and mathematical analysis, and the procedure of how to combine the two approaches, please see Section 1.4 of Chapter 1.

4.6 Simulations

Our goal in this section is to get some intuitions about the above questions by conducting simulation on the minimum model given in Fig. 4.6. Without loss of generality, we fix the number of meanings m and the number of words n both to be 30, and the punishment rate $\beta = 1$. (For other settings of m, n, and $\beta \geq 1$, the simulation results are qualitatively similar). Also, we suppose that all meanings are uniformly distributed; in other words, $p(x_i) = 1/m,\ i = 1, \cdots, m$ because there are m meanings. For each value of reward rate α in $\{0.1, 0.2, \cdots, 1.8, 1.9, 2\}$, we run 100 times of the procedure given in Fig. 4.6. After 500 iterations, stop and record the communication coherence, as the approximation of the *eventual* coherence. Plotting the eventual coherence by averaging the results of the 100 runs of simulation as a function of reward rate α, we obtain a graph shown in Fig. 4.7.

The graph shows that when the reward rate α is around some critical value $\alpha^* = 0.8$, there is a phase transition. When α is above α^*, the two agents can eventually reach coherent communication; otherwise, they cannot. We want to mention that the critical value of α depends on the other three parameters (as we will show later in Section 4.7.1). In this case where $m = n = 30$ and $\beta = 1$, the critical value of α happens to be 0.8.

The results shown in Fig. 4.7 only illustrates the eventual coherence (snapshot at iteration 500). To get a sense of how the communication coherence changes over time, Fig. 4.8 shows the dynamics for the case of reward rate $\alpha = 2$, which is above the critical value $\alpha^* = 0.8$. For other settings of the number of meanings and words, the results are similar as long as the reward rate is *large* enough. From the graph, we can see that the communication coherence has a trend to

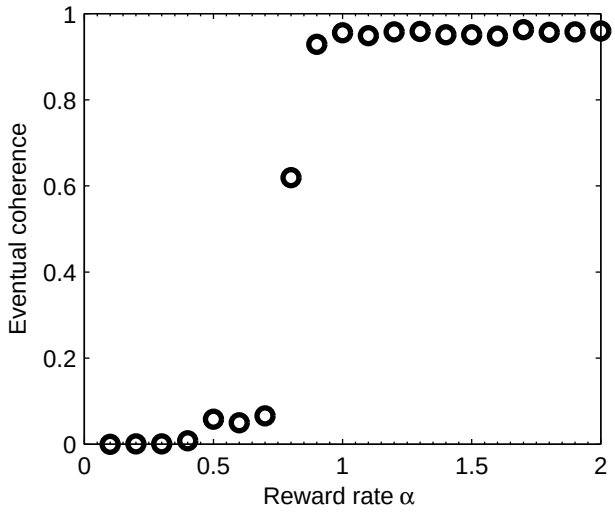


Figure 4.7: Phase transition.

converge to 1.

Fig. 4.9 shows the dynamics for the case of reward rate $\alpha = 0.5$ which is below the critical value $\alpha^* = 0.8$. For other settings of the number of meanings and words, the results are similar as long as the reward rate is *small* enough. From the graph, we can see that the communication coherence cannot get improved over time using the reinforcement learning—it oscillates around some very low coherence value.

4.7 Analysis

4.7.1 Conditions for reaching coherent communication

In the last section, we have shown by simulation that whether or not the agents can reach coherent communication depends on the settings of the parameters. In the case of the number of meanings $m = 30$, the number of words $n = 30$, and punishment rate $\beta = 1$, when reward rate α is above 0.8, the two agents can eventually reach coherent communication; otherwise, they cannot. This section aims at finding out what ranges of settings of the four parameters will lead to coherent communication.

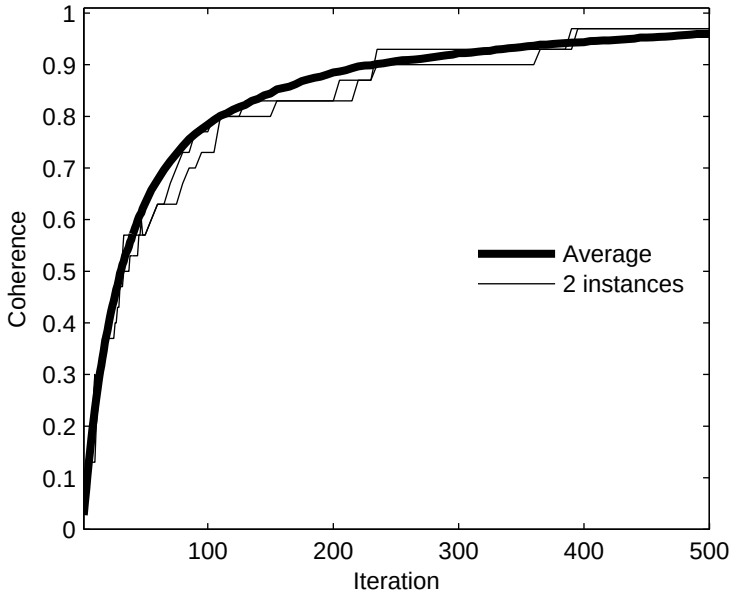


Figure 4.8: Simulated dynamics of the reinforcement learning model for large reward rate $\alpha = 2$.

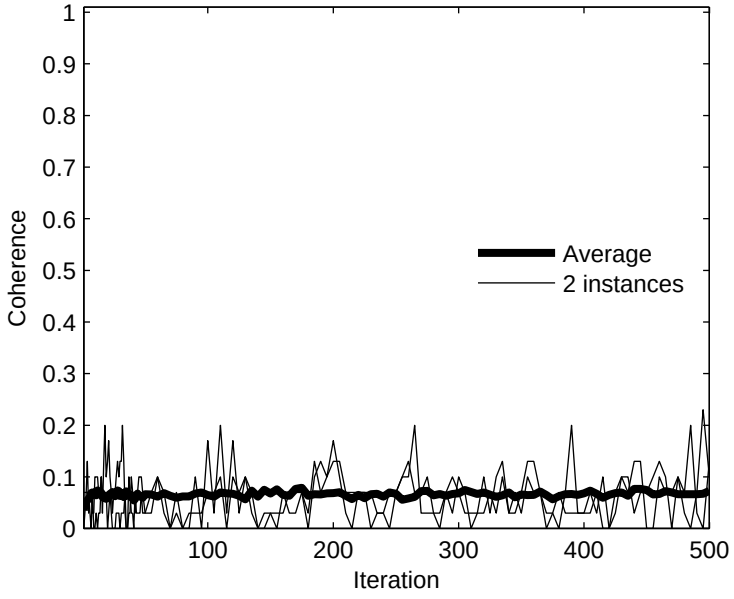


Figure 4.9: Simulated dynamics of the reinforcement learning model for small reward rate $\alpha = 0.5$.

The idea behind is that if the parameters are appropriatedly set, a temporary (or lucky) meaning-word agreement between the sender and receiver will have a greater chance to be reinforced by the reward rule than to be weakened by the punishment rule. Once a temporary agreement has a better chance to be reinforced than to be weakened, random walk theory (Norris, 1997) can tell us that

the temporary agreement will have a positive probability of becoming a stable agreement.

To facilitate our arguments and analysis, we give the following informal notations, which will be defined formally later.

Reward probability. The probability that a temporary agreement is reinforced by some reward rule.

Punishment probability. The probability that a temporary agreement is weakened by some punishment rule.

Without loss of generality, let the first temporary agreement be on the meaning-word pair (x_1, y_1). This means for the sender, it will use word y_1 to represent the meaning x_1, i.e., $y_1 = f_s(x_1)$; for the receiver, it will use meaning x_1 to interpret the word y_1, i.e., $x_1 = f_r(y_1)$. In terms of association matrix, according to the decision function, we have, for the sender $b_s(x_1, y_1) = \max_k b_s(x_1, y_k)$, and for the receiver $b_r(x_1, y_1) = \max_k b_r(x_k, y_1)$. For convenience, we denote by $s_{11} = b_s(x_1, y_1)$ and by $r_{11} = b_r(x_1, y_1)$. That the agreement on (x_1, y_1) is reinforced means both s_{11} and r_{11} will increase by α, and that the agreement is weakened means at least one of s_{11} and r_{11} will decrease by β.

If at the next round what is played is again the meaning x_1, then the communication on meaning x_1 will succeed, so the reward rule will be applied to both s_{11} and r_{11}. Since the probability of playing meaning x_1 is p_1, the *reward probability* of the agreement will be p_1, and then the *expected reward amount* of the agreement will be

$$\alpha p_1.$$

If at the next round what is played is not the meaning x_1, let it be x_2 for example, then there is a chance that the word produced by the sender to represent the meaning x_2 happens to be the word y_1, and this chance can be written as the probability $\Pr(f_s(x_2) = y_1)$. If this does not happen, then there is neither reward nor punishment, so it is of no interest. If this happens, the communication will fail, since the interpreted meaning by the receiver for y_1 must be x_1, according to the decision function given by Eq (4.2). And then, the association weight r_{11} between word y_1 and meaning

x_1 in the receiver's matrix will decrease by β, according to the punishment rule of the receiver. Similarly, at the same time, the weight s_{21} between meaning x_2 and word y_1 in the sender's matrix will decrease by β. (This means the punishment rule of the sender has no effect on the weight s_{11}.) Therefore, the *punishment probability* can be represented as $(1-p_1)\text{Pr}(f_s(x_2)=y_1)$, where $(1-p_1)$ is the probability of playing a meaning other than x_1, and $\text{Pr}(f_s(x_2)=y_1)$ is the probability that the sender will use y_1 to represent the other meaning. As a result, the *expected punishment amount* of the agreement will be

$$\beta(1-p_1)\text{Pr}(f_s(x_2)=y_1).$$

Now we want to compute $\text{Pr}(f_s(x_2)=y_1)$, the probability that the word y_1 is the one used by the sender for the meaning x_2. Considering that in our model words are produced by Eq (4.1), a nonlinear function, it is hard to give a precise estimation. So, we make the following assumption.

Assumption 4.7.1. *(Random word production assumption.) The word produced by the sender for a meaning that is not in any (temporary or stable) agreement is assumed to be randomly generated.*

With the assumption, we can suppose that the probability of the word y_1 being the one produced by the sender for meaning x_2, $\text{Pr}(f_s(x_2)=y_1)$, can be approximated by $\frac{1}{n}$, provided that there are totally n words. (We will show by simulations that this is a good approximation, or the assumption is a good one.)

Then, our task is to solve the following inequality, where αp_1 is the expected reward amount, and $\beta(1-p_1)\frac{1}{n}$ is the expected punishment amount:

$$\alpha p_1 > \beta(1-p_1)\frac{1}{n}. \tag{4.5}$$

Eq (4.5) is just the condition for the agents to develop the first stable agreement. (Actually it is very likely that more than one agreements can be established parallel.) Now we need to find the conditions for the other agreements to be established. Before we go further, we make the following

assumption about the order of developing agreements.

Assumption 4.7.2. *(Sequential agreement establishment assumption.) Agreements are established in the sequential order of the frequency that their meanings are used. If the probabilities of the meanings have the relationship:* $p_1 \geq p_2 \geq \cdots$*, then we assume that the agreements will be established in the order of* $(x_1, y_{l_1}), (x_2, y_{l_2}), \cdots$.

Now suppose we already have $l-1$ established stable agreements that satisfy the above sequential assumption. Then we need to find the condition for a new agreement, the l^{th} agreement, to be established. Note that $l = 1$ is the case we have just discussed above. Let the probabilities of those meanings associated with the already established agreements be $p_1 \geq \cdots \geq p_{l-1}$. By applying the similar arguments as above, for the l^{th} temporary agreement to become stable, we require the following inequality to hold

$$\alpha p_l > \beta(1 - \sum_{k=1}^{l} p_k)\frac{1}{n}, \tag{4.6}$$

where the term $\sum_{k=1}^{l} p_k$ in the inequality comes from the fact that these l meanings can guarantee their sender will not generate a word that would cause the temporary agreement to be weakened.

In general, for the agents to establish L agreements, we must require the following L inequalities to hold

$$\alpha p_l > \beta(1 - \sum_{k=1}^{l} p_k)\frac{1}{n}, \qquad (l = 1, 2, \cdots, L),$$

which are equivalent to

$$\frac{\alpha}{\beta} > (1 - \sum_{k=1}^{l} p_k)\frac{1}{np_l}, \qquad (l = 1, 2, \cdots, L), \tag{4.7}$$

When the agents can develop L stable agreements on meanings $x_1, \cdots, x_L$, the communicative coherence of the system will be

$$\phi(L) = \sum_{k=1}^{L} p_k.$$

When meanings are uniformly distributed. Above we have shown that, for the agents to establish stable communication on L meanings $x_1, \cdots, x_L$, the inequalities in Eq (4.7) should be satisfied for $l = 1, 2, \cdots, L$. Now we are ready to solve the inequalities to obtain the convergence conditions for some specific distributions over meanings. Here we only consider the uniform distribution.

When the meanings are uniformly distributed, we have $p_i = \frac{1}{m}$ for $i = 1, ..., m$. For the agents to establish L agreements, by substituting $p_l = \frac{1}{m}$ into Eq (4.7), we have

$$\frac{\alpha}{\beta} > \frac{m-l}{n}, \qquad (l = 1, 2, \cdots, L).$$

It is clear that if $\frac{\alpha}{\beta} > \frac{m-1}{n}$, then for all $l \geq 1$, the above L inequalities will hold. In summary, the condition for reaching coherent communication under uniform distribution of meanings is:

$$\frac{\alpha}{\beta} > \frac{m-1}{n}. \tag{4.8}$$

Specifically, if we fix $m = n = 30$ and $\beta = 1$ (the same setting as in our simulation), then $\alpha^* = \frac{m-1}{n} \approx 0.97$ will be the critical value of reward rate, above which the agents can develop a communication system. This value is larger than the one in the simulation where $\alpha^* \approx 0.8$. This is because we have made two assumptions in the analysis, the *random word production* assumption and the *sequential agreement establishment* assumption. Of course, in the simulations, there is no such assumptions. The difference between simulation and analytical conditions also indicates that in the simulation, it is easier for the agents to reach coherent communication.

Clearly, the two agents can develop as many as $\min\{m, n\}$ stable agreements, since there are at most $\min\{m, n\}$ distinct pairs of meaning-word, and thus, the maximum converged communication coherence is $\min\{m, n\}\frac{1}{m}$ or equivalently $\min\{\frac{n}{m}, 1\}$.

4.7.2 Time for reaching coherent communication

In this section, we aim at giving an analytical answer to our second question: how fast can the agents reach coherent communication if the conditions are satisfied? Our approach is to build a dynamics equation describing how communication coherence changes over time as a result of the agents updating their coding function by reinforcement learning, and then to infer from the equation how much time is needed for a given level of coherence.

Luckily, it turns out that the dynamics equation can be constructed explicitly using the following theorem.

Theorem 4.7.1. *If all the meanings are uniformly distributed and the convergence condition (given in Eq* (4.8)*) is satisfied, then the expected communication coherence at iteration t is (each iteration contains m interactions between the sender and receiver):*

$$\begin{cases} \phi(0) & = & 0 \\ \phi(t+1) & = & \phi(t) + \big(1-\phi(t)\big)\Big(\frac{1}{m} - \frac{\phi(t)}{n}\Big) \end{cases} \tag{4.9}$$

where m is the number of meanings, and n is the number of words.

Proof. In the above recursive relation, $\phi(t)$ represents the current expected communication coherence, and $\phi(t+1)$ the expected coherence at the next iteration. For $(1-\phi(t))(1-\frac{m}{n}\phi(t))\frac{1}{m}$, it represents the probability that a meaning will be communicated successfully for the first time by the agents at the next iteration—the meaning will be called *new lucky meaning*. Within this part, the term $(1-\frac{m}{n}\phi(t))$ represents the probability that a randomly chosen word can be used by the sender for representing the new lucky meaning. That current communication coherence is $\phi(t)$ means there are as many as $m\phi(t)$ words that have already been successfully and thus permanently associated with some meanings, we can see that the number of remaining "fresh" words is $n - m\phi(t)$. Therefore, for a word which is randomly chosen from n words to be a fresh word, its probability must be $(n-m\phi(t))/n = 1-\frac{m}{n}\phi(t)$. Another term $\frac{1}{m}$ represents the probability that a word will be correctly interpreted by the receiver. Recall that there are m meanings, so by random

guess the probability of hitting the correct meaning is $\frac{1}{m}$. ■

Corollary 4.7.1. *The expected communication coherence at $t \to \infty$ is:*

$$\lim_{t\to\infty} \phi(t) = \min\{1, \frac{n}{m}\}$$

Proof. The Eq (4.9) has two fixed points: 1 and $\frac{n}{m}$. As long as $\phi(t) < 1$ and $\phi < \frac{n}{m}$, $\phi(t)$ will increase monotonely as t increases. Considering that the initial condition is $\phi(0) = 0 < \min\{1, \frac{n}{m}\}$, therefore, we have $\lim_{t\to\infty} \phi(t) = \min\{1, \frac{n}{m}\}$. ■

Comparison between simulation and analytical results. Fig. 4.10 shows the comparison between the simulation and analytical results. The parameter setting for obtaining the results is: $(m, n, \alpha, \beta) = (30, 30, 2, 1)$. It is the same setting as in the simulation given in Fig. 4.8. In the figure, the simulation result, shown by the dashed line, was obtained by averaging 100 simulations, and is the same as Fig. 4.8. The theoretical result, shown by the smooth line, is computed by the dynamics equation which is given by Eq (4.9).

It turns out that the simulation result has a slightly higher eventual communication coherence than the analytical one (compare the two coherence values at the 500th iteration). This can be explained by the comparison on the simulation and analytical results on the convergence condition (see the end part of Section 4.7.1). There, the simulation result, compared with the analytical result, shows a lower threshold of the reward rate α which makes the agents in the simulation context easier to reach higher coherence than that in the analysis context.

How much time is needed for achieving a given level of coherence? Suppose we can solve the above recursive equation (Eq (4.9)) by a closed form such as $coh = \psi(t)$, then we can represent time t as a function of coherence coh in the form of $t = \psi^{-1}(coh)$. For example, from Fig. 4.10, we can estimate that to achieve a communication coherence of 90%, the agents need around 250 iterations. Unfortunately, since there is no closed form for solving the recursive equation except in

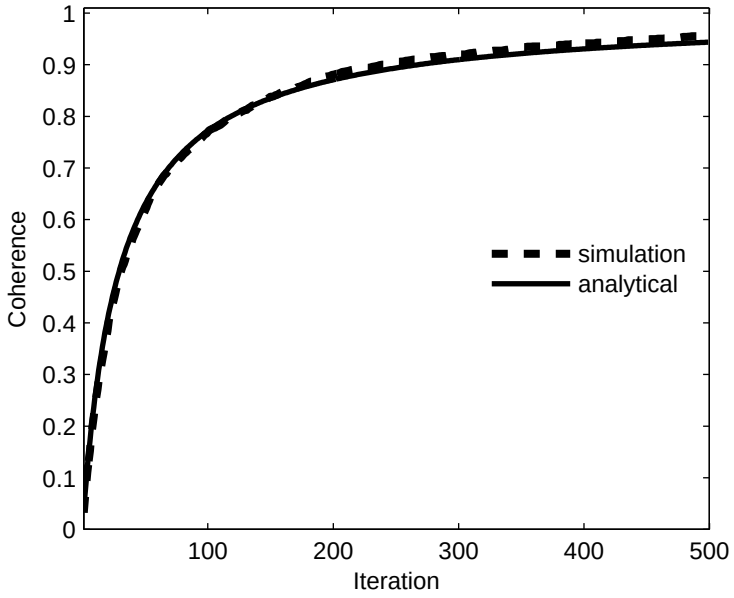


Figure 4.10: Comparison between the simulation and analytical dynamics.

very special cases such as $m = n = 1$, we cannot calculate explicitly the number of iterations that is needed for a given level of coherence.

4.8 Summary

In this chapter we studied our second linguistic consensus case: coherent communication. The coherent communication problem concerns how a group of agents can converge to a coherent communication system in which the word used by a sender to represent some meaning can be interpreted correctly by a receiver to extract the same meaning.

In terms of the self-organizing framework given in Chapter 2, we designed a reaching coherent communication model that focuses on the following two components: (1) the 2-player communication game, and (2) the agent learning model. In the design of the game, we defined the payoff function as follows: if the sender's intended meaning can be correctly conveyed to the receiver, both agents get a positive payoff; otherwise 0. In the design of agents, we implemented the agent state as an association matrix between words and meanings. Thus, for a sender to represent an

intended meaning, he will choose the word with the strongest association to the meaning; and for a receiver to interpret a word, she will choose the meaning with the strongest association to the word. Furthermore, agents were designed to update their matrices using the *simple reinforcement learning rule*. If the communication between a sender and a receiver is successful, both agents will increase the association strength between their respective pair of word and meaning by a number called *reward rate*; otherwise decrease the strength by a number called *punishment rate*.

Our work has the following contributions. First, we proposed a minimum model which only consists of two agents (a sender and a receiver). We found by computer simulations and mathematical analysis that, for a given number of meanings and words, there exists a critical value of the reward-punishment ratio, above which the agents can converge to a coherent communication system, below which the agents cannot. We also gave a dynamics equation on how coherence changes over time when the convergence condition is satisfied. Second, compared with existing work, ours is the first to give an analytical result on the conditions under which the agents can reach coherent communication using reinforcement learning.

Chapter 5

Case 3: Reaching Grammar Consensus

5.1 Introduction

In this chapter we present our study of the third case of linguistic consensus: reaching grammar consensus. In our grammar consensus problem, each agent has a grammar which is modeled as a function by which the agent can generate and recognize grammatical sentences. When two agents have different grammars, it is possible that the sentences generated by one agent cannot be recognized by the other agent. The task of the agents, then, is to reach a common grammar so that the sentences generated by one agent can be recognized by every other agent.

The agents are supposed to be adaptive—they can change their grammar based on interactions they have with each other. So a key question is how can we design adaptive mechanisms for such agents so that they can converge from their initially different grammars to a common grammar. In the design of the agents, there are four factors that may affect the convergence:

1. Population size: the number of agents;
2. Grammar complexity: the number of variables in a grammar function;
3. Learning mechanism: how agents change their description word;
4. Interaction structure: who interacts with whom.

This chapter aims at studying how the population size and grammar complexity affect the convergence under a learning mechanism called *perceptron learning rule* (Rosenblatt, 1958) and an all-to-all interaction network. We will construct a computational model using the self-organizing

language framework presented in Chapter 2. Because of the complexity of grammar adaptation that prevents from systematic simulation experiments, we will mainly focus on mathematical analysis to find the conditions under what the agents can converge to one common grammar.

In the next section we give a brief review of related work. Then we set up a mutual perceptron learning model and specify our research questions. Next we concentrate on presenting our mathematical analysis of the questions. We close with a summary of this chapter.

5.2 Related work

Our grammar consensus problem can be seen as a special case of the study of the emergence of grammar. Various computational models have been proposed to study the emergence of grammar. A most classical one is Batali's (1998) computational model that is based on simple recurrent network (SRN). In his model, a group of agents take turns to play the role of speaker (he) and hearer (she). The speaker converts a vector of meanings into a sequence of characters of $\{a, b, c\}$ using his SRN, and sends the sequence to the hearer. The hearer then converts the sequence to a meaning vector using her SRN. If the hearer's interpreted meaning vector is wrong, the hearer will update the connection weights in her SRN according to the correct meaning vector provided by the speaker. Batali's simulation shows that the agents can develop a shared grammar that resembles in many ways to human languages. However, because of the complexity of the SRN, it is difficult to prove the convergence of the agents in Batali's model. (This is also true for most of the work on the emergence of grammar.)

Recently, Cucker, Smale, and Zhou (2004) proposed an elegant abstract model (CSZ model) in which grammars are modeled as convex functions that take the form of $f : X \mapsto Y$, [1] and the grammar adaptation of an agent is modeled as learning from the data (sentences) generated from other agents. They have shown some very pleasant analytical properties in their model. For example, the agents can converge to a shared grammar if one agent can learn (directly or

[1] In function $f : X \mapsto Y$, X is a convex subset of n-dimensional Euclidean space $\Re^n$, and Y is a convex subset of $\Re$.

indirectly) from every other agent using Regularization learning algorithm (Neumaier, 1998), and the convergence speed depends on the eigenvalues of the connection matrix (network) among the agents.

Like the CSZ model, our work on reaching grammar consensus is also along the direction of building abstract models with a focus on modeling the grammar adaptation of an agent as learning from sentences generated from other agents. However, different from the CSZ model, we focus on building a grammar consensus model that has the following two important properties: (1) the learning algorithm is online learning (rather than learning from the scratch as in the CSZ model); (2) the grammar can be used by an agent to generate or recognize grammatical sentences. As we will see later, our mutual perceptron learning model meets these two properties.

5.3 Modeling grammars and sentences

The objective of this chapter is to study how agents can reach a common grammar. Towards this end, first we have to clarify what we mean by grammar (and sentence). In formal language theory, a sentence is defined as a string of symbols, a language is a set of sentences, and a grammar is a finite list of rules that define a language. For example, the set of all sentences over the binary alphabet $\{0, 1\}$ is $\{0, 1, 00, 01, 10, 11, 000, 001, \cdots\}$. A list of four rewriting rules $\{S \rightarrow 0S; \quad S \rightarrow A; \quad A \rightarrow 1A; \quad A \rightarrow \epsilon\}$ where ϵ is a null element, is a grammar that defines a regular language $L = 0^m 1^n$. Whether a sentence belongs to a language or not depends on whether the sentence can be generated by the grammar of the language or recognized by its corresponding grammar recognition machine (Lewis and Papadimitriou, 1997). For example, the above grammar can tell us that sentence "001" is grammatical while sentence "0010" is not.

In a general sense, we can think of grammars as functions that take the form of $f : X \mapsto \{0, 1\}$, where f indicates a grammar, X indicates the space of all possible sentences, and 1 or 0 means whether a sentence is grammatical or not.

In this thesis, a sentence will be modeled as a data point, called *instance* henceforth, in an n-

dimensional Boolean space $\{0,1\}^n$, and a grammar will be modeled as a Boolean (classification) function which can classify an instance as grammatical (positive) or not (negative). For example, for $\mathbf{x} = (x_1, \cdots, x_6) \in \{0,1\}^6$, if a classification function $f(\mathbf{x})$ is defined as:

$$f(x_1, \cdots, x_6) = 1 \quad \text{if and only if } \; x_2 = 1 \;\&\; x_4 = 0,$$

then $\mathbf{x} = (1,1,1,0,0,0)$ is a positive instance while $\mathbf{x} = (1,1,1,1,0,0)$ is not. Obviously, this treatment is very different from the traditional formal language theory.

Though not realistic, our treatment has several advantages. First, it conforms to a fundamental property of grammar: grammar can be used to generate grammatical sentences and can also be acquired or adapted by learning from sentences. Similarly, a classification function can be used to generate positive (or negative) instances and can also be acquired or adapted by learning from instances.

Second, modeling grammars as Boolean functions would bring up a theoretical possibility of using the principles and parameters (P&P) framework (Chomsky, 1981)—a framework that is regarded by many linguists as the dominant form of mainstream linguistics—to interpret the spontaneous emergence of human languages such as the Nicaragua sign language (Senghas et al., 2004). In the principles and parameters framework, the set of grammatical hypothesis is generated by k binary parameters. At any time, a learner holds a particular hypothesis (or grammar) given by a particular setting of these parameters. The learner does not change his grammar as long as the received sentences can be recognized by the grammar. If a sentence arrives that is not recognizable, then the learner might change some of the parameters (Matsen and Nowak, 2004).

5.4 Reaching grammar consensus using mutual perceptron learning

In terms of the game-based self-organizing language framework given in Chapter 2, here we present a grammar consensus model that focuses on the following two components: (1) a 2-player grammar game, and (2) an agent learning model called *mutual perceptron learning* model.

Game model. The 2-player grammar game is designed as a sequential language game which is illustrated by Fig. 5.1. From the figure, we can see that there are three players: nature, speaker (he), and hearer (she). The player nature is added here for modeling that the speaker has some private information that is unknown or uncertain to the hearer. (See Section 2.2.3 of Chapter 2 for details.) In our case of grammar game, the private information of the speaker is the class of a sentence. The speaker knows the class, but the hearer only knows the sentence produced by the speaker, and her job is to predict the class of the sentence. (In the figure, the left dashed line illustrates that the hearer cannot tell which path, x_1 or x_2, the sentence y_1 comes from.) Nature moves first and determines the class of a sentence to be produced by the speaker. There are two classes: grammatical (G) and ungrammatical (U), and nature determines a class according to the probability of $p(G)$ or $p(U) = 1 - p(G)$. The game tree in Fig. 5.1 shows $p(G) = 0.9$ and $p(U) = 0.1$. When the hearer predicts correctly the class, both agents will receive a positive payoff of 1, otherwise 0.

A general agent model. Here we present a general agent model, and later we will present the specific mutual perceptron learning model. In a general agent model, every agent has three components: a state b, a decision function f, and a state update function g. We suppose that all speakers have the same form of decision and state update functions, and all hearers have the same form of functions too. (As we will show in the specific model, we suppose that an agent use a unified function for the role of speaker and hearer.)

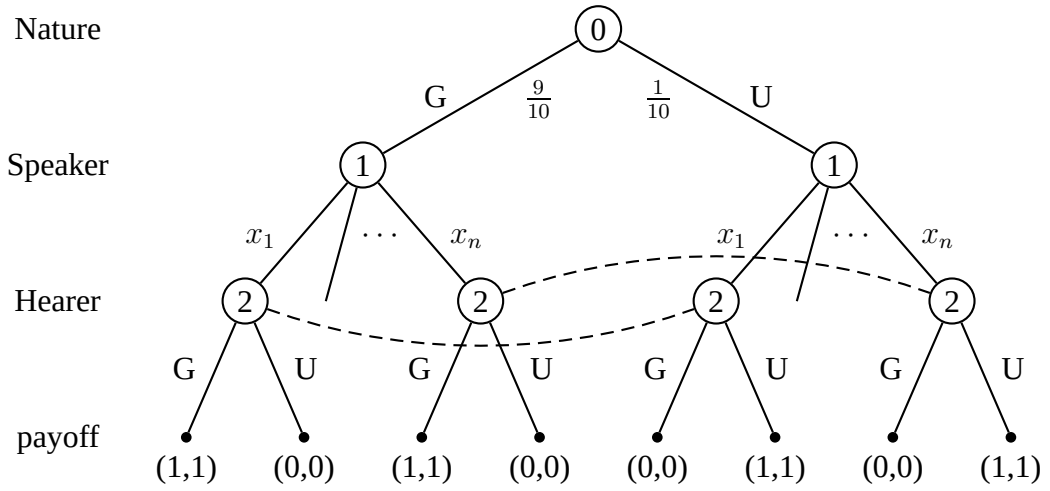


Figure 5.1: Grammar game tree.

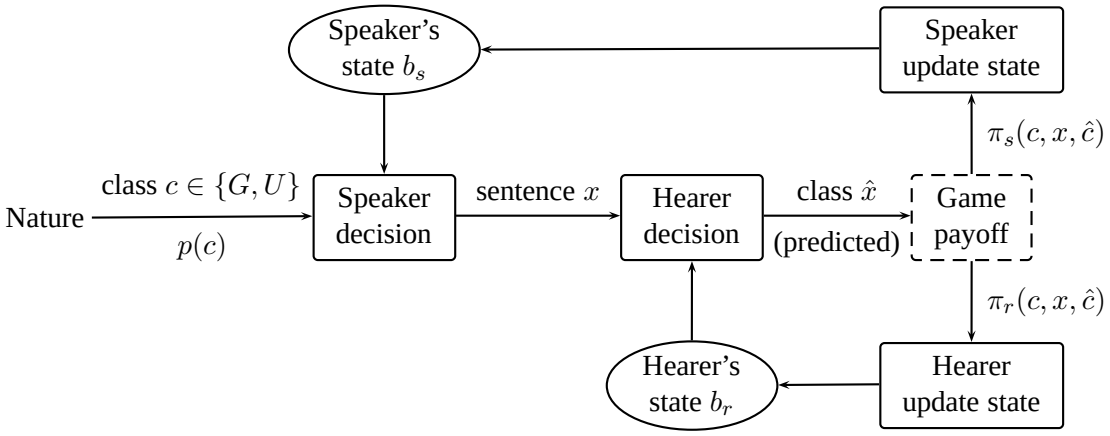


Figure 5.2: Agent model in a grammar game.

Fig. 5.2 shows how these components work together in a grammar game played by a speaker s and a hearer r. In a grammar game, nature moves first, deciding the class c, G or U, of a sentence that the speaker will produce, according to probability $p(c)$. When class c is given, the speaker will call his decision function f_s to generate a sentence x that has the class c, based on its state b_s, in the form of

$$x = f_s(c, b_s).$$

The hearer, after receiving the sentence x, will call her decision function f_r to predict a class $\hat{c}$ for

sentence x, based on

$$\hat{c} = f_r(x, b_r).$$

After the hearer predicts the class of the sentence, both agents receive a payoff that is given by

$$\pi_i = \pi_i(c, x, \hat{c}).$$

And then, both agents update their state from b_i to b_i' based on the received payoff. For the speaker, its update function g_s is used to update its current state b_s to a new one b_s' based on its current sentence production $c \mapsto x$ and the received payoff π_s, in the form of

$$b_s' = g_s(b_s, c, x, \pi_s).$$

For the hearer, its update function g_r is used to update its current state b_r to a new one b_r' based on its current prediction $x \mapsto \hat{c}$ and the received payoff π_r, in the form of

$$b_r' = g_r(b_r, \hat{c}, x, \pi_r).$$

The mutual perceptron learning model. As we stated in Section 5.3, a sentence is modeled as an instance in an n-dimensional Boolean space $\mathbf{X} = \{0, 1\}^n$, and a grammar is modeled as a classification function which can classify an instance as 1 or -1 (i.e., classify a sentence as grammatical or not).

We use linear threshold function to implement the decision function of an agent (Duda et al., 2000). A linear threshold function is a function that uses a linear combination of the components of its input instances $\mathbf{x} \in \mathbf{X}$ for making its decision. It can be written as

$$y = f(\mathbf{w}, \mathbf{x}) = \Theta(\mathbf{w} \cdot \mathbf{x}) = \Theta(\sum_i w_i x_i),$$

where $\mathbf{w} = (w_1, \cdots, w_n) \in \Re^n$ represents the state of the agent, and $\Theta(z)$ is a threshold function

that outputs 1 if z is above some threshold, otherwise -1.

This definition of decision function is straightforward for hearer, who just needs to call the function to output whether a received instance $\mathbf{x}$ is 1 or -1. But for the speaker, it is a little tricky. If he wants to produce a positive (or negative) instance, the way to implement it is to repeatedly sample an instance $\mathbf{x}$ from the instance space $\mathbf{X}$ until the instance satisfies $f(\mathbf{w}, \mathbf{x}) = 1$ (or -1), where $\mathbf{w}$ is the hearer's state.

The state update function is as follows. If both agents get a positive payoff, they will keep their current state. Otherwise, they will update their state $\mathbf{w}$ according to the following perceptron learning rule:

$$\mathbf{w} = \mathbf{w} - \lambda y \mathbf{x},$$

where $\lambda > 0$ is a parameter called *learning rate*. In addition, for the hearer, y is the predicted class value (1 or -1), and for the speaker, y is the intended class of the sentence $\mathbf{x}$. Note that in the case of zero payoff, the speaker and hearer's class value are opposite (i.e., $y_{speaker} = -y_{hearer}$).

To distinguish our perceptron learning from the traditional one (Rosenblatt, 1958), which involves only one agent (learner) updating its function to fit to another agent (teacher), we call ours *mutual perceptron learning*, because both speaker and hearer can update their functions.

More about perceptron learning. To be prepared for the analysis of our model in the next section, we introduce some concepts here about the traditional, basic perceptron learning model in which a *learner* learns from a *teacher*.

The basic perceptron algorithm was introduced first by Rosenblatt (1958) to solve the linear threshold learning problem. A perceptron takes a vector of real-valued inputs[2], calculates a linear combination of these inputs, and outputs a 1 if the result is greater than some threshold and -1 otherwise. More precisely, given an input instance $\mathbf{x} = (x_1, ..., x_n)$, the output $f(\mathbf{x})$ computed by

[2]The instance space $\{0, 1\}^n$ is a special case of $\Re^n$.

the perceptron is:

$$f(\mathbf{x}) = \begin{cases} +1 & \text{if} \quad \sum_{i=1}^{n} w_i x_i > \theta \\ -1 & \text{otherwise} \end{cases} \tag{5.1}$$

where $\mathbf{w} = (w_1, ..., w_n) \in \Re^n$ is the current weight vector maintained by the perceptron. For convenience, usually the threshold θ is set to 0. The reason is that we can add an additional constant input $x_0 = 1$ with a weight variable w_0.

For brevity, we will sometimes write the perceptron function as:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x})$$

where

$$\text{sgn}(y) = \begin{cases} +1 & \text{if} \quad y > 0 \\ -1 & \text{otherwise} \end{cases}$$

Note that each weight vector defines a perceptron function or decision function. Learning a perceptron involves choosing values for the weight vector $\mathbf{w} = (w_1, ..., w_n)$. Initially the algorithm starts with a weight vector $\mathbf{w} = (0, ..., 0)$. Upon receiving an instance $\mathbf{x} = (x_1, ..., x_n)$, the learner predicts the label of $\mathbf{x}$ to be $f(\mathbf{x})$.

If the predicted label is correct, then there are no changes in the weight vector. However, if the prediction is wrong, the weight vector of the learner is updated using the *perceptron learning rule*:

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda f(\mathbf{x}) \cdot \mathbf{x} \tag{5.2}$$

where λ is the learning rate.

The perceptron Convergence Theorem was proven in (Novikoff, 1962; Minsky 1969); we sketch it here because we draw upon it in later proofs of mutual perceptron convergence.

Theorem 5.4.1. Perceptron Convergence Theorem *If all instances are linearly separable, then a learner which uses the perceptron algorithm will only make a finite number of mistakes. That is,*

the learning procedure converges.

Proof. The basic idea of the proof is to show that on each mistake made by the learner, the distance between the currently maintained weight vector (of the function f) and the target weight vector (of the target perceptron function f^*) becomes smaller after the update using the perceptron learning rule. ■

Coherence. In Section 2.5 of Chapter 2, we stated that: for N agents, suppose their states are $b_1, \cdots, b_N$, then a general definition of population coherence can be given by

$$\phi(b_1, \cdots, b_N) = \frac{1}{N(N-1)} \sum_{i \neq j} sim(b_i, b_j)$$

where $0 \leq sim(b_i, b_j) \leq 1$ is the similarity between the states of two agents i and j and its definition depends on contexts.

In the context of the mutual perceptron learning model, the definition of the similarity is given as follows. Since the state of an agent is a weight vector $\mathbf{w}$, we can define the similarity between two states $\mathbf{w}_1$ and $\mathbf{w}_2$ as follows:

$$sim(\mathbf{w}_1, \mathbf{w}_2) = \cos(\mathbf{w}_1, \mathbf{w}_2) = \frac{\mathbf{w}_1 \cdot \mathbf{w}_2}{|\mathbf{w}_1||\mathbf{w}_2|},$$

where $|\mathbf{w}|$ denotes the length of vector $\mathbf{w}$ and is given by $|\mathbf{w}| = \sqrt{\sum_{k=1}^{n} w_k^2}$.

Then the population coherence over N agents can be given by

$$\phi(\mathbf{w}_1, \cdots, \mathbf{w}_N) = \frac{2}{N(N-1)} \sum_{1 \leq i < j \leq N} \cos(\mathbf{w}_i, \mathbf{w}_j). \tag{5.3}$$

A computational model of reaching grammar consensus. Fig. 5.3 gives the whole computational model of reaching grammar consensus using the mutual perceptron learning rule. This model is based on the general model that presented in Fig. 2.13 of Section 2.6 of Chapter 2. In the model, the dimensions of the instance space, n, is the number of grammar parameters in terms of

the P&P framework (see Section 5.3). The analysis and simulation conducted in later sections will be based on this model.

Settings

population: N agents
language complexity: n dimensions of instance space
agent state: each agent i has a weight vector $\mathbf{w}_i$
learning parameter: learning rate λ

Initialization

each agent's weight vector is randomized

Iterations (each iteration includes $N(N-1)$ interactions (see Section 2.4 for details).

During each interaction

1. a class c is chosen from $\{U, G\}$ with probability $p(c)$
2. the speaker generates a sentence x whose class is c
3. the hearer predicts the class of the sentence x as $\hat{c}$
4. if the predicted class $\hat{c}$ is wrong (i.e., both agents receive a payoff of 0)
   each agent updates its state based on perceptron learning rule

After each iteration, namely $N(N-1)$ interactions, take a snapshot of the population coherence $\phi(t)$

Figure 5.3: The mutual perceptron learning model.

5.5 Specific research questions

The traditional perceptron learning model, which involves only one agent (learner) updating its function to fit to another fixed agent (teacher), is the first learning algorithm that was shown to converge under some conditions (Novikoff, 1962; Minsky 1969; Vapnik, 1995). However, we have no knowledge about the convergence property of mutual perceptron learning because this is a new learning model.

We ask the following questions:

1. Will the agents in our proposed mutual perceptron learning model be able to converge to some common grammar?

2. What are the conditions that can lead the agents to reach grammar consensus? Concretely speaking, for a given dimensions of instance space, n, a given number of agents, N, and

learning rate λ, what kind of settings of these parameters will make the agents reach grammar consensus?

Research methods. Unlike in the cases of the previous two chapters where we did computer simulation first and then explored the problem by mathematical analysis, here we will mainly focus on using mathematical analysis to study the above questions and then use computer simulation to partly test the analytical result. (See Section 5.7 for the detailed reasons.)

5.6 Convergence analysis

To conform to the terms used in traditional perceptron learning, in this section of analysis, we will use "teacher" to stand for "speaker", "learner" for "hearer", "mistake" for "nonpositive payoff", "instance" for "sentence", "classification function" for "grammar", and "label" for "class of sentence".

The purpose of this section is to show that under some conditions the perceptron learning model will converge to a common classification function. The basic idea is as follows. If the learner makes a mistake on the instance given by the teacher, we want to show that the "distance" between the weight vectors of the two agents will become smaller after weight updating using perceptron learning rule.

However, if the number of agents is greater than two, we also need to consider the new "distances" between functions of the learner/teacher agent and other agents in the population—we want the entire population to converge. We show that under some conditions, the *sum of the new distances* can be less or equal than the old distance, in which case the algorithm is guaranteed to globally converge.

Before going ahead to prove the convergence, we need some assumptions, some of which are also used in the traditional perceptron convergence theorem proof (Thm. 5.4.1).

Assumption 5.6.1. *For any instance* $\mathbf{x}$*, there exists a positive constant* κ *such that* $\|\mathbf{x}\| \leq \kappa$.

Assumption 5.6.2. *For any weight vector* $\mathbf{w}$ *maintained by any agent, suppose for any instance and label pair* $(\mathbf{x}, y)$ *generated according to* $y = sgn(\mathbf{w} \cdot \mathbf{x})$*, there exists a positive constant* γ *such that* $y\mathbf{w} \cdot \mathbf{x} \geq \gamma > 0$.

To make the proof readable, we will first introduce some definitions as follows.

Definition 5.6.1. *Given two agents with weight vectors* $\mathbf{w}_A$ *and* $\mathbf{w}_B$*, the distance between them is defined as:*

$$d(\mathbf{w}_A, \mathbf{w}_B) = \|\mathbf{w}_A - \mathbf{w}_B\|^2$$

where $\|\cdot\|$ *is the 2-norm (i.e., the Euclidean norm).*

Definition 5.6.2. *At time step* t*, the weight vectors of two agents* A *and* B *are* $\mathbf{w}_A$ *and* $\mathbf{w}_B$*. At time step* $t+1$*, the weight vectors become* $\mathbf{w}'_A$ *and* $\mathbf{w}'_B$*. The variance of distance between the two agents from time* t *to* $t+1$ *is defined as:*

$$\Delta(\mathbf{w}_A, \mathbf{w}_B) = d(\mathbf{w}'_A, \mathbf{w}'_B) - d(\mathbf{w}_A, \mathbf{w}_B) = \Delta$$

Definition 5.6.3. *Distance reduction is defined as* $-\Delta$ *if* $\Delta < 0$.

Definition 5.6.4. *Distance introduction is defined as* Δ *if* $\Delta > 0$.

Lemma 5.6.1. *Suppose on a round of the game, a learner* L *makes a mistake on the instance* $\mathbf{x}$ *given by a teacher* T*, and both agents use the perceptron learning rule to update their weight vectors* $\mathbf{w}_L$ *and* $\mathbf{w}_T$*. Then there exists a positive constant* $\delta = \frac{\gamma^2}{\kappa^2}$ *such that:*

$$\Delta(\mathbf{w}_T, \mathbf{w}_L) \leq -\delta. \tag{5.4}$$

when the learning rate $\lambda = \frac{\gamma}{2\kappa^2}$.

Proof. When the learner L makes a wrong prediction for the label of the instance $\mathbf{x}$ sent by the teacher T, it will modify its weight vector from $\mathbf{w}_L$ to $\mathbf{w}'_L$. According to the definition of label

generation and prediction, we have:

$$\begin{cases} y &= \text{sgn}(\mathbf{w}_T \cdot \mathbf{x}) \\ -y &= \text{sgn}(\mathbf{w}_L \cdot \mathbf{x}) \end{cases} \tag{5.5}$$

Then we have the following inequalities

$$\begin{cases} y \cdot \mathbf{w}_T \cdot \mathbf{x} > 0 \\ y \cdot \mathbf{w}_L \cdot \mathbf{x} < 0 \end{cases} \tag{5.6}$$

With the above Assumptions 5.6.1 and 5.6.2, we have

$$\begin{aligned} \Delta(\mathbf{w}_T, \mathbf{w}_L) &= d(\mathbf{w}_T^{'}, \mathbf{w}_L^{'}) - d(\mathbf{w}_T, \mathbf{w}_L) \\ &= \|(\mathbf{w}_T - \lambda y\mathbf{x}) - (\mathbf{w}_L + \lambda y\mathbf{x})\|^2 - \|\mathbf{w}_T - \mathbf{w}_L\|^2 \\ &= -4\lambda y\mathbf{w}_T\mathbf{x} + 4\lambda y\mathbf{w}_L\mathbf{x} + 4\lambda^2\|\mathbf{x}\|^2 \\ &\leq -4\lambda(\gamma - \lambda\kappa^2) \end{aligned}$$

When the learning rate $\lambda = \frac{\gamma}{2\kappa^2}$, we have

$$\Delta(\mathbf{w}_T, \mathbf{w}_L) \leq -\frac{\gamma^2}{\kappa^2}.$$

Let $\delta = \frac{\gamma^2}{\kappa^2}$, then $\Delta(\mathbf{w}_T, \mathbf{w}_L) \leq -\delta$ holds. ■

The above lemma shows that two agents can move their functions closer to each other through adaptation after making mistakes. However, for the situation of more than two agents, it is possible that moving one agent's function toward that of another agent will cause it to move farther away from the function of a third agent. Fig. 5.4 shows a very simple illustration.

We want to show that under some conditions, the total distance *reduction* (see Definition 5.6.3) is larger than the total distance *introduction* (see Definition 5.6.4) after updating the weight vectors

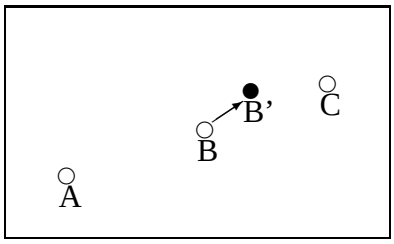


Figure 5.4: An illustration of the complexity of mutual perceptron learning. Agent B moves to B' which is closer to C, but the distance between B' and A becomes larger than before.

of the two agents (teacher and learner) in a game round. The following lemma says that when the learning rate is set to the positive constant described in Lemma 5.6.1, then the *reduction* in the distance between the learner/teacher and other agents' functions has a lower bound—there will be at least that much reduction in distance. Thus the point is that with such a learning rate, it is possible (but not yet certain) that the functions may converge.

Lemma 5.6.2. *Given an instance* $\mathbf{x}$*, and two agents* A *and* B *with weight vector* $\mathbf{w}_A$ *and* $\mathbf{w}_B$*. Suppose* A*'s label* y_A *on* $\mathbf{x}$ *is different from* B*'s label* y_B*, and* A *changes its weight vector* $\mathbf{w}_A \leftarrow \mathbf{w}_A + y_B\mathbf{w}_A\mathbf{x}$*, while* B *keeps its weight vector* $\mathbf{w}_B$ *unchanged. Let learning rate* $\lambda = \frac{\gamma}{2\kappa^2}$*, then there exists a constant* $\Delta_r = 3\gamma^2/4\kappa^2 > 0$ *such that*

$$\Delta(\mathbf{w}_A, \mathbf{w}_B) \leq -\Delta_r.$$

Δ_r *is called the lower bound of distance reduction.*

Proof.

$$\begin{aligned}
\Delta(\mathbf{w}_A, \mathbf{w}_B) &= \|(\mathbf{w}_A + \lambda y_B\mathbf{x}) - \mathbf{w}_B\|^2 - \|\mathbf{w}_A - \mathbf{w}_B\|^2 \\
&= 2\lambda y_B\mathbf{w}_A\mathbf{x} - 2\lambda y_B\mathbf{w}_B\mathbf{x} + \lambda^2\|\mathbf{x}\|^2 \\
&\leq -2\lambda\gamma + \lambda^2\kappa^2 \\
&= -3\gamma^2/4\kappa^2 = -\Delta_r \qquad (5.7)
\end{aligned}$$

■

Assumption 5.6.3. *For any weight vector* $\mathbf{w}$ *maintained by any agent, suppose for any instance and label pair* $(\mathbf{x}, y)$ *generated according to* $y = sgn(\mathbf{w} \cdot \mathbf{x})$*, there exists a positive constant* β *such that* $y\mathbf{w} \cdot \mathbf{x} \leq \beta$.

Next, Lemma 5.6.3 says that the distance introduced between the functions of the learner-teacher pair and other agents has an upper bound.

Lemma 5.6.3. *Given an instance* $\mathbf{x}$*, and two agents* A *and* B *with weight vector* $\mathbf{w}_A$ *and* $\mathbf{w}_B$*. Suppose* A*'s label* y_A *on* $\mathbf{x}$ *is the same as* B*'s label* y_B*, and* A *changes its weight vector* $\mathbf{w}_A \leftarrow \mathbf{w}_A - y_B\mathbf{w}_A\mathbf{x}$ *(note, this is a mis-adaptation), while* B *keeps its weight vector* $\mathbf{w}_B$ *unchanged. Let learning rate* $\lambda = \frac{\gamma}{2\kappa^2}$*, then there exists a constant* $\Delta_i = \frac{\gamma}{\kappa^2}(\beta - \frac{3\gamma}{4}) > 0$ *such that*

$$\Delta(\mathbf{w}_A, \mathbf{w}_B) \leq \Delta_i.$$

Δ_i *is called the upper bound of distance introduction.*

Proof. It is obvious that $\Delta_i = \frac{\gamma}{\kappa^2}(\beta - \frac{3\gamma}{4}) > 0$ from the fact $\beta \geq \gamma$.

$$\begin{aligned}
\Delta(\mathbf{w}_A, \mathbf{w}_B) &= \|(\mathbf{w}_A - \lambda y_B\mathbf{x}) - \mathbf{w}_B\|^2 - \|\mathbf{w}_A - \mathbf{w}_B\|^2 \\
&= 2\lambda y_B\mathbf{w}_A\mathbf{x} - 2\lambda y_B\mathbf{w}_B\mathbf{x} + \lambda^2\|\mathbf{x}\|^2 \\
&\leq 2\lambda\beta - 2\lambda\gamma + \lambda^2\kappa^2 \\
&= \frac{\gamma}{\kappa^2}(\beta - \frac{3\gamma}{4}) = \Delta_i
\end{aligned} \tag{5.8}$$

■

Now we are ready to give the proof of the convergence of the perceptron learning model.

Theorem 5.6.4. *If* $\frac{\beta}{\gamma} \leq \frac{3}{2}$ *holds, then a population of agents which use the perceptron learning rule can converge to a shared function after making a finite number of mistakes.*

We call $\frac{\beta}{\gamma} \leq \frac{3}{2}$ as the *convergence condition*.

Proof. The idea is simple. After a learner makes a mistake, we hope that by using our mutual perceptron learning rule for each weight vector update, the sum of new inter-function distances among all agents is smaller than before. Suppose there are N agents. Among these N agents, there are two agents, learner and teacher respectively, whose corresponding weight vectors are $\mathbf{w}_L, \mathbf{w}_T$. The weight vectors of other agents are $\mathbf{w}_1, ..., \mathbf{w}_{N-2}$. (To be consistent, let $\mathbf{w}_L = \mathbf{w}_{N-1}$, and $\mathbf{w}_T = \mathbf{w}_N$.) Then the sum of distances can be written as:

$$\begin{aligned} \sum_{i,j=1}^{N} d(\mathbf{w}_i, \mathbf{w}_j) &= d(\mathbf{w}_T, \mathbf{w}_L) + \sum_{i,j=1}^{N-2} d(\mathbf{w}_i, \mathbf{w}_j) \\ &+ \sum_{i=1}^{N-2} d(\mathbf{w}_T, \mathbf{w}_i) + \sum_{i=1}^{N-2} d(\mathbf{w}_L, \mathbf{w}_i) \end{aligned} \tag{5.9}$$

And the sum of new distances after the learning is written as:

$$\begin{aligned} \sum_{i,j=1}^{N} d^{'}(\mathbf{w}_i, \mathbf{w}_j) &= d^{'}(\mathbf{w}_T, \mathbf{w}_L) + \sum_{i,j=1}^{N-2} d^{'}(\mathbf{w}_i, \mathbf{w}_j) \\ &+ \sum_{i=1}^{N-2} d^{'}(\mathbf{w}_T, \mathbf{w}_i) + \sum_{i=1}^{N-2} d^{'}(\mathbf{w}_L, \mathbf{w}_i) \end{aligned} \tag{5.10}$$

where $d^{'}(\mathbf{w}_i, \mathbf{w}_j)$ is the brief notation of $d(\mathbf{w}'_i, \mathbf{w}'_j)$.

Now let us compute the new total distances. Let $\mathbf{x}$ be the instance on which the learner made a mistake. Denote by y_L the label computed by the learner, and by y_T the label from the teacher. On the instance $\mathbf{x}$, other $N-2$ agents also have their own labels, denoted by $y_1, ..., y_{N-2}$. Among these $N-2$ agents, suppose there are p agents whose labels are the same as y_T, and q agents whose labels are the same as y_L, where $p + q = N - 2$. Without loss of generality, suppose $y_1 = \cdots = y_p = y_T$ and $y_{p+1} = \cdots = y_{p+q} = y_L$.

So, according to the Lemma 5.6.2 and 5.6.3, the total distance variance between the teacher's function and those of the other agents (excluding the learner) is at most $-p\Delta_r + q\Delta_i$. This is because we have

$$\begin{aligned} \sum_{i=1}^{N-2} d^{'}(\mathbf{w}_T, \mathbf{w}_i) &= \sum_{i=1}^{p} d^{'}(\mathbf{w}_T, vwi) + \sum_{i=p+1}^{N-2} d^{'}(\mathbf{w}_T, \mathbf{w}_i) \\ &\leq \sum_{i=1}^{p} d(\mathbf{w}_T, \mathbf{w}_i) - p \cdot \Delta_r \\ &+ \sum_{i=p+1}^{N-2} d(\mathbf{w}_T, \mathbf{w}_i) + q \cdot \Delta_i \end{aligned}$$

and similarly, the total distance variance between the learner's function and those of the other agents (excluding the teacher) also is at most $p\Delta_i - q\Delta_r$. Again, this is because we have:

$$\begin{aligned}\sum_{i=1}^{N-2} d^{'}(\mathbf{w}_L, \mathbf{w}_i) &= \sum_{i=1}^{p} d^{'}(\mathbf{w}_L, \mathbf{w}_i) + \sum_{i=p+1}^{N-2} d^{'}(\mathbf{w}_L, \mathbf{w}_i) \\ &\leq \sum_{i=1}^{p} d(\mathbf{w}_L, \mathbf{w}_i) + p \cdot \Delta_i \\ &+ \sum_{i=p+1}^{N-2} d(\mathbf{w}_L, \mathbf{w}_i) - q \cdot \Delta_r\end{aligned}$$

Putting these two inequalities together, plus

$$\sum_{i,j=1}^{N-2} d^{'}(\mathbf{w}_i, \mathbf{w}_j) = \sum_{i,j=1}^{N-2} d(\mathbf{w}_i, \mathbf{w}_j)$$

and Eq (5.4):

$$d^{'}(\mathbf{w}_T, \mathbf{w}_L) \leq d(\mathbf{w}_T, \mathbf{w}_L) - \delta,$$

we can express the total distance between all agents as follows (following Eqs (5.9) and (5.10)):

$$\sum_{i,j=1}^{N} d^{'}(\mathbf{w}_i, \mathbf{w}_j) \leq \sum_{i,j=1}^{N} d(\mathbf{w}_i, \mathbf{w}_j) + (p+q) \cdot (\Delta_i - \Delta_r) - \delta$$

If the convergence condition $\frac{\beta}{\gamma} \leq \frac{3}{2}$ holds, then the amount of *reduction* in distance is equal or greater than the *introduction* in distance, i.e., the difference is less than or equal to zero:

$$\begin{aligned}\Delta_i - \Delta_r &= \tfrac{\gamma}{\kappa^2}(\beta - \tfrac{3\gamma}{4}) - \tfrac{3\gamma^2}{4\kappa^2} \\ &= \tfrac{\gamma}{\kappa^2}(\beta - \tfrac{3}{2}\gamma) \\ &\leq 0\end{aligned}$$

Therefore, the sum of new inter-function distances among all agents *is* smaller than before:

$$\sum_{i,j=1}^{N} d^{'}(\mathbf{w}_i, \mathbf{w}_j) \leq \sum_{i,j=1}^{N} d(\mathbf{w}_i, \mathbf{w}_j) - \delta$$

where $\delta = \frac{\gamma^2}{\kappa^2}$.

Suppose the initial total distances among all agents is Δ_σ, then the algorithm will converge after making at most $\frac{\kappa^2}{\gamma^2}\Delta_\sigma$ mistakes. ■

Discussion of the convergence condition. What is the interpretation of $\frac{\beta}{\gamma} \leq \frac{3}{2}$?

We want to show that there is a relationship between the quality of the function instance produced by the teacher, and the possibility of convergence. From Assumptions 5.6.2 and 5.6.3 we know that,

$$\gamma = \min_{y,\mathbf{w},\mathbf{x}} \|y\mathbf{w}\mathbf{x}\| \tag{5.11}$$

$$\beta = \max_{y,\mathbf{w},\mathbf{x}} \|y\mathbf{w}\mathbf{x}\| \tag{5.12}$$

Without loss of generality, suppose all instances have unit length. So, rewriting, we have:

$$\|y\mathbf{w}\mathbf{x}\| = |y| \cdot \|\mathbf{w}\| \cdot \|\mathbf{x}\| \cdot |cos(\mathbf{w}, \mathbf{x})| = \|\mathbf{w}\| \cdot |cos(\mathbf{w}, \mathbf{x})|$$

Then, the ratio between γ and β can be rewritten as:

$$\frac{\gamma}{\beta} = \frac{\min_{\mathbf{w},\mathbf{x}} \|\mathbf{w}\| \, |\cos(\mathbf{w}, \mathbf{x})|}{\max_{\mathbf{w},\mathbf{x}} \|\mathbf{w}\| \, |\cos(\mathbf{w}, \mathbf{x})|}$$

Suppose

$$\max \, |\cos(\mathbf{w}, \mathbf{x})| = 1$$

and let

$$\max \|\mathbf{w}\| = \rho \, \min \|\mathbf{w}\|$$

where $\rho \geq 1$.

Now we have

$$\gamma/\beta = \frac{1}{\rho} \min_{\mathbf{w},\mathbf{x}} \, |\cos(\mathbf{w}, \mathbf{x})|$$

Given all above assumptions, $\frac{\beta}{\gamma} \leq \frac{3}{2}$ is equivalent to:

$$\min_{\mathbf{w},\mathbf{x}} |\cos(\mathbf{w}, \mathbf{x})| \geq \frac{2}{3}\rho$$

where $1 \leq \rho \leq \frac{3}{2}$.

This analysis means that a sufficient condition for convergence is that the instance generated by a teacher falls within a certain distance region within the space of all possible instances. This idea is expressed visually in Fig. 5.5.

The *most* representative instance, represented by the dark vector in Fig. 5.5, occurs when $\mathbf{x} = \eta\mathbf{w}$, where $\eta \in \Re$ and $\eta \neq 0$, that is, when the instance coincides with the weight vector or negative weight vector exactly.

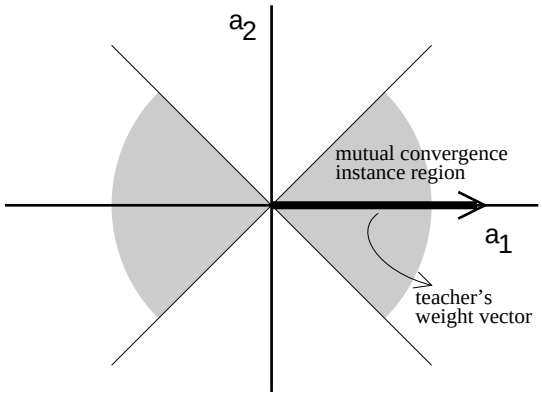


Figure 5.5: Region of instances satisfying the mutual convergence condition ($\rho = 1.06$)

The shaded area of Fig. 5.5 shows the region of instances that, when generated by the teacher, will satisfy the mutual convergence condition given $\rho = \frac{3\sqrt{2}}{4} \approx 1.06$. Happily, this region occupies half of the entire instance space.

5.7 Simulations

Above we have given the convergence analysis to the mutual perceptron learning model. In this section we want to test by computer simulation if the agents using the mutual perceptron learning

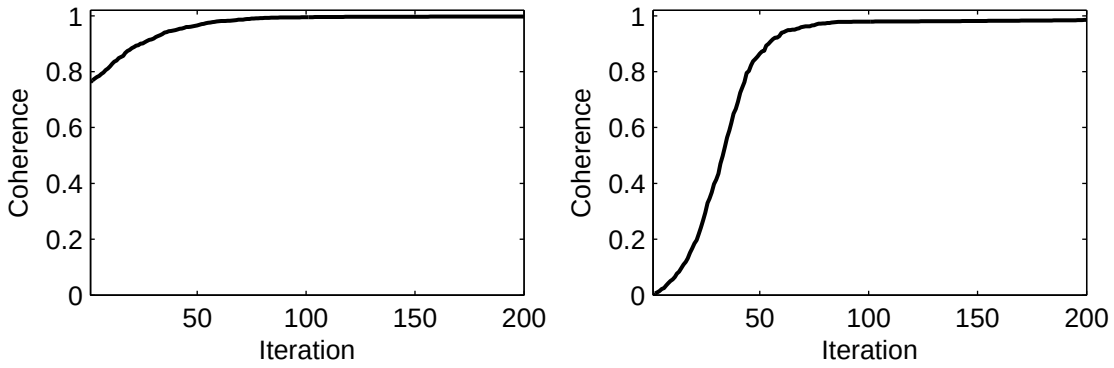
rule can actually converge to a common function. To this end, we design two experiments. In the first experiment, like the experiments we did in previous chapters, we suppose the initial states (weight vectors) of the agents are randomized. In the second experiment, we set the initial weight vectors of the agents to be orthogonal so that the population coherence could be 0 at the beginning of the self-organization.

The settings for both experiments are as follows. There are 10 agents, the instance space is $\{0,1\}^{10}$, and the learning rate is $\lambda = 0.1$. In each iteration, every agent has a chance to interact with every other agent twice, once as speaker (teacher) and once as hearer (learner). Together, there are 90 games in an iteration. After each iteration, population coherence is calculated according to Eq (5.3) and is plotted. Fig. 5.6 shows the result of the first experiment with random initial weight vectors, and Fig. 5.6 shows the result of the second experiment with orthogonal initial weight vectors. An example of orthogonal vectors is

$$(1,0,0,\cdots,0)$$
$$(0,1,0,\cdots,0)$$
$$\cdots$$
$$(0,0,\cdots,0,1)$$

In both experiments, the convergence condition $\beta/\gamma \leq 3/2$ is satisfied. The values of γ and β are obtained during the simulation according to their definitions in Assumptions 5.6.2 and 5.6.3. From the figures, we can see that the agents can converge to a common weight vector (i.e., linear threshold function) with high coherence. We can also see that there is a difference between the two experiments: the one with random initial states has much higher coherence at the beginning, and the experiment with orthogonal initial states has a zero coherence. This difference can be easily interpreted according to the definition of coherence given in Eq (5.3).

So far we can only show by simulation that in some cases when the agents converge to a common classification function, the analytical convergence condition ($\beta/\gamma \leq 3/2$) is satisfied. We



(a) Random initial weight vectors. (b) Orthogonal initial weight vectors.

Figure 5.6: Simulation on the convergence of 10 agents.

have not figured out how to test whether the convergence condition holds true or not by conducting systematic simulations on the whole space of β/γ. This is because the values of β and γ, according to their definitions in Assumptions 5.6.2 and 5.6.3, depend on the initial weights and the instances generated during the self-organization. Recall that in the previous two chapters, we did not have this problem because the learning behavior in those models are much simpler than that in the mutual perceptron learning model in this chapter.

5.8 Summary

In this chapter we studied our third linguistic consensus case: grammar consensus. The grammar consensus problem concerns how a group of agents can converge to a common grammar by which the sentences generated by one agent using his grammar can be recognized by another agent using her grammar.

In terms of the game-based self-organizing language framework given in Chapter 2, we designed a grammar consensus model that focuses on the following two components: (1) the 2-player grammar game, and (2) the agent learning model. In the design of the grammar game, grammar is modeled as a Boolean classification function and sentences are modeled as instances, and the payoff function was defined as follows: if the hearer can correctly predict the class of the instance produced by the speaker, both agents receive a positive payoff; otherwise 0. In the design of agents,

perceptron learning algorithm was used.

Our work has the following contributions. First, we showed by mathematical analysis that agents in our model can converge to a common grammar (i.e., a common classification function) under some conditions, and we gave those conditions. Second, compared with existing work on the grammar consensus problem (which has mainly been done within the observational learning paradigm), ours is the first that uses reinforcement learning as the learning mechanism of agents.

Chapter 6

Conclusions

6.1 Summary

The primary objective of this study has been to design and analyze mechanisms that allow a group of agents to converge to a common language from a set of initially different languages. Specifically, our research aimed at studying how agents can converge to a common language using reinforcement learning, a learning mechanism that only requires a minimum of feedback information.

We presented a game-based self-organizing language framework into which reinforcement learning can be naturally fit. Using the framework, we studied three cases of reaching linguistic consensus: word consensus, coherent communication, and grammar consensus. The word consensus problem concerns how agents can converge to a common word out of many different words to represent a single shared meaning. The coherent communication problem concerns how agents can converge to a communication system in which the word used by a sender to represent some meaning can be interpreted correctly by a receiver to extract the same meaning. The grammar consensus problem concerns how agents can converge to a common grammar, so that the sentences generated by one agent using his grammar can be recognized by another agent using her grammar.

In the case study of word consensus, we proposed a win-stay lose-shift (WSLS) model that was based on the original social convention model introduced by Shoham and Tennenholtz (1997) and the WSLS learning rule (Matsen and Nowak, 2004). The main results obtained from computer simulation and mathematical analysis include the following:

1. Agents in the model can converge to a common word when their aspiration level—basically

the ratio of successes to uses in the history of word agreement—is set to some appropriate values (see Section 3.5);

2. We obtained an analytical result on the convergence conditions, from which we can tell for a given number of agents and words how high the aspiration level should be in order for the agents to reach word consensus (see Section 3.6.2);

3. We obtained a dynamics equation that captures how coherence changes over time when the convergence condition is satisfied. From this equation, we can tell how much time is needed to reach a given level of coherence (see Section 3.6.3).

In the case study of coherent communication, we proposed a minimum reinforcement learning model which consists of two agents (a sender and a receiver). The main results obtained from computer simulation and mathematical analysis include:

1. Unlike existing work on coherent communication which showed that the learning rates (i.e., reward and punishment rates) do not have critical effects on agents converging to a shared communication system, we found that there exists a critical reward-punishment ratio above which the agents can converge to a coherent communication system, below which the agents cannot (see Section 4.6);

2. We obtained an approximate analytical result on the convergence condition that tells how the critical reward-punishment ratio depends on the number of words and the number of meanings in a communication or vocabulary system (see Section 4.7.1);

3. We also obtained a dynamics equation that captures how communication coherence changes over time when the convergence condition is satisfied. From this equation, we can tell how much time is needed to reach a given level of communication coherence (see Section 4.7.2).

In the case study of grammar consensus, we have proposed a mutual perceptron learning model in which grammars are modeled as Boolean functions that can be used to classify or recognize Boolean instances (sentences). We have presented the following main results:

1. We obtained an analytical result on the convergence condition, which implies that a speaker should generate representative instances, under which the agents can converge to a common grammar using perceptron learning rule. (see Section 5.6);

2. We showed by computer simulation that the agents can converge to a common grammar when the convergence condition is satisfied (see Section 5.7).

6.2 Limitations and future research

First, in all of our models, we used an interaction model under which the likelihood on any agent interacting with any other agent is equal. This means that the underlying interaction network is a completely connected graph. However, it is more realistic to assume some restrictions in the interaction pattern an agent may have. For example, we could assume that the interaction networks are regular graphs, or complex networks such as small-world networks (Watts and Strogatz, 1998) or scale-free networks (Barabasi and Albert, 1999). The new interaction patterns of agents under these networks will provide a rich source for future work. Will the agents constrained by these networks be able to converge or not? Will the convergence speed be faster or not? Many interesting questions await exploration along this dimension.

Second, the models studied in this thesis are very abstract. Though of theoretical value, they are far from realistic. To overcome this limitation, one future direction would be to use data from real human-based language games such as the ESP image labeling game, collaborative tagging systems, or historical linguistics. For example, it would be interesting to build a straightforward model of the real ESP game and then analyze its dynamics. We could also imagine other kinds of realistic language games such as games that allow more than two players to play. Thinking along the direction of realistic language games would open up many chances for the future research.

Third, the studies presented here assume that agents stay with their converged language once the agents reach a consensus. However, in reality, agents' environments are in a flux of constant change; for example, new agents may be introduced into the population, the meaning distribu-

tion may change to reflect newly-important topics for agent conversations, etc. All of these situations imply there has to be in the long term a way for agents to innovate, de-converging and re-converging on new languages. We must explore the extent to which our current models do capture these dynamics, find ways to better characterize this sort of dynamics, and explore additional approaches to modeling continuous agent-environment-language relationships.

Last, we believe the study of self-organizing languages will open up many new possibilities of developing novel approaches to information organization and access, as already seen by the successful applications of social tagging systems, the ESP game, and other realms of "semiotic dynamics" (Staab et al., 2002). The theoretical analysis of lexicon and grammar consensus in this thesis represents only a small step in exploring the full potential of self-organizing language systems.

References

Axelrod, R. (1984). *The Evolution of Cooperation*. New York: Basic Books.

Axelrod, R. (1997). Advancing the art of simulation in the social sciences. In Conte, R., Hegselmann, R., and Terna, P., editors, *Simulating Social Phenomena*, pages 21–40. Berlin: Springer-Verlag.

Barabasi, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286:509–512.

Batali, J. (1998). Computational simulations of the emergence of grammar. In Hurford, J. R., Studdert-Kennedy, M., and C., K., editors, *Approaches to the Evolution of Language - Social and Cognitive Bases*. Cambridge University Press, Cambridge.

Brighton, H., Smith, K., and Kirby, S. (2005). Language as an evolutionary system. *Physics of Life Reviews*, 2(3):177–226.

Cangelosi, A. and Parisi, D., editors (2002). *Simulating the evolution of language*. Springer-Verlag.

Chomsky, N. (1981). *Lectures on government and binding*. Foris: Dordrecht.

Christiansen, M. H. and Kirby, S., editors (2003). *Language Evolution: The States of the Art*. Oxford University Press.

Cover, T. M. and Thomas, J. A. (1990). *Elements of Information Theory*. John Wiley & Sons.

Cucker, F., Smale, S., and Zhou, D.-X. (2004). Modeling language evolution. *Foundations of Computational Mathematics*, 4(3):315–343.

De Jong, E. D. and Steels, L. (2003). A distributed learning algorithm for communication development. *Complex Systems*, 14(4).

Delgado, J. (2002). Emergence of social conventions in complex networks. *Artificial Intelligence*, 141:171–185.

Duda, R. O., Hart., P. E., and Stork, D. G. (2000). *Pattern Classification*. John Wiley & Sons.

Egghe, L. and Rousseau, R. (1990). *Introduction to Informetrics*. Elsevier.

Golder, S. and Huberman, B. A. (2006). Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208.

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

Kaplan, F. (2000). Semiotic schemata: Selection units for linguistic cultural evolution. In Bedau, M., McCaskill, J., Packard, N., and Rasmussen, S., editors, *Artificial Life VII*. The MIT Press.

Kaplan, F. (2005). Simple models of distributed co-ordination. *Connection Science*, 17(3-4):249–270.

Kittock, J. (1993). Emergent conventions and the structure of multi-agent systems. In L. Nadel, D. S., editor, *Lectures in Complex Systems, SFI Studies in the Sciences of Complexity*. Addison-Wesley, Reading, MA.

Kreps, D. M. (1990). *A Course in Microeconomic Theory*. Princeton University Press.

Lakkaraju, K. and Gasser, L. (2006). A unified framework for multi-agent agreement. Technical Report UIUCDCS-R-2006-2803, Department of Computer Science, University of Illinois at Urbana-Champsign.

Lenaerts, T., Jansen, B., Tuyls, K., and de Vylder, B. (2005). The evolutionary language game: An orthogonal approach. *Journal of Theoretical Biology*, 235(4):566–582.

Lewis, D. (1969). *Convention: A Philosophical Study*. Cambridge, MA: Harvard University Press.

Lewis, H. R. and Papadimitriou, C. H. (1997). *Elements of the Theory of Computation*. Prentice Hall.

Mathes, A. (2004). Folksonomies - cooperative classification and communication through shared metadata. Computer Mediated Communication - LIS590CMC, Graduate School of Library and Information Science, University of Illinois Urbana-Champaign.

Matsen, F. A. and Nowak, M. A. (2004). Win-stay, lose-shift in language learning from peers. *PNAS*, 101(52):18053–18057.

Maynard-Smith, J. and Szathmary, E. (1997). *The Major Transitions in Evolution*. New York: Oxford University Press.

Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press.

Narendra, K. S. and Thathachar, M. A. (1989). *Learning automata: An introduction*. Prentice Hall.

Neumaier, A. (1998). Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM Review*, 40:636–666.

Norris, J. R. (1997). *Markov Chains*. Cambridge University Press.

Novikoff, A. B. (1962). On convergence proofs on perceptrons. In *Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622. Polytechnic Institute of Brooklyn.

Nowak, M. A. and May, R. M. (1992). Evolutionary games and spatial chaos. *Nature*, 359:826–829.

Nowak, M. A., Plotkin, J. B., and Jansen, V. A. A. (2000). The evolution of syntactic communication. *Nature*, 404:495–498.

Nowak, M. A., Plotkin, J. B., and Krakauer, D. C. (1999). The evolutionary language game. *Journal of Theoretical Biology*, 200:147–162.

Oliphant, M. and Batali, J. (1997). Learning and the emergence of coordinated communication. *The newsletter of the Center for Research in Language*, 11(1).

Osborne, M. and Rubinstein, A. (1994). *A Course in Game Theory*. The MIT Press.

Pinker, S. (2000). Survival of the clearest. *Nature*, 404:441–442.

Plotkin, J. B. and Nowak, M. A. (2000). Language evolution and information theory. *Journal of Theoretical Biology*, 205(1):147–159.

Posch, M., Pichler, A., and Sigmund, K. (1999). The efficiency of adapting aspiration levels. *Proc. R. Soc. Lond. B*, 266(1427):1427–1435.

Rosen, K. H., Michaels, J. G., Gross, J. L., Grossman, J. W., and Shier, D. R. (2000). *Handbook of Discrete and Combinatorial Mathematics*. CRC Press.

Rosenblatt, F. (1958). The perceptron: A probabilitic model for information storage and organization in the brain. *Psychological Review*, 65:386–407.

Sen, S., Lam, S. K., Rashid, A. M., Cosley, D., Frankowski, D., Osterhouse, J., Harper, M. F., and Riedl, J. (2006). Tagging, communities, vocabulary, evolution. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 181–190, New York, NY, USA. ACM Press.

Senghas, A., Kita, S., and Ozyurek, A. (2004). Children creating core properties of language: Evidence from an emerging sign language in nicaragua. *Science*, 305(5691):1779–1782.

Shalizi, C. R. (2003). Methods and techniques of complex systems science: An overview. In Deisboeck, T. S., Kresh, J. Y., and Kepler, T. B., editors, *Complex Systems Science in Biomedicine*. Kluwer.

Shoham, Y. and Tennenholtz, M. (1993). Co-learning and the evolution of social activity. Technical report, Computer Science Department, Stanford University.

Shoham, Y. and Tennenholtz, M. (1997). On the emergence of social conventions: modeling, analysis, and simulations. *Artificial Intelligence*, 94:139–166.

Smith, K. (2004). The evolution of vocabulary. *Journal of Theoretical Biology*, 228(1):127–142.

Staab, S., Santini, S., Nack, F., Steels, L., and Maedche, A. (2002). Emergent semantics. *IEEE Intelligent Systems*, 17(1):78–86.

Steels, L. (1996). Self-organizing vocabularies. In Langton, C. and Shimohara, T., editors, *Artificial Life V*, Nara, Japan.

Steels, L. (1997). The synthetic modeling of language origins. *Evolution of Communication*, 1(1):1–34.

Steels, L. (2003). The evolution of communication systems by adaptive agents. In Alonso, E., D. K. and Kazakov, D., editors, *Adaptive Agents and Multi-Agent Systems: Adaptation and Multi-Agent Learning. LNAI 2636*, pages 125–140. Springer Verlag, Berlin.

Steels, L. (2006). Semiotic dynamics for embodied agents. *IEEE Intelligent Systems*, 21(3):32–38.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

von Ahn, L. (2006). Games with a purpose. *IEEE Computer*, 39(6):92–94.

Wagner, K., Reggia, J., Uriagereka, J., and Wilkinson, G. (2003). Progress in the simulation of emergent communication and language. *Adaptive Behavior*, 11(1):37–69.

Walker, A. and Wooldridge, M. J. (1995). Understanding the emergence of conventions in multi-agent systems. In *ICMAS95*, pages 384–389, San Francisco, CA.

Wang, W. S.-Y. and Minett, J. W. (2005). Vertical and horizontal transmission in language evolution. *Transactions of the Philological Society*, 103(2):121–146.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393:440–442.