Noisy Preferential Attachment and Language Evolution

Samarth Swarup[1] and Les Gasser[1,2]

[1] Dept. of Computer Science,
[2] Graduate School of Library and Information Science,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801, USA
{swarup, gasser}@uiuc.edu

Abstract. We study the role of the agent interaction topology in distributed language learning. In particular, we utilize the replicator-mutator framework of language evolution for the creation of an emergent agent interaction topology that leads to quick convergence. In our system, it is the links between agents that are treated as the units of selection and replication, rather than the languages themselves. We use the Noisy Preferential Attachment algorithm, which is a special case of the replicator-mutator process, for generating the topology. The advantage of the NPA algorithm is that, in the short-term, it produces a scale-free interaction network, which is helpful for rapid exploration of the space of languages present in the population. A change of parameter settings then ensures convergence because it guarantees the emergence of a single dominant node which is chosen as teacher almost always.

1 Introduction

The study of communication and language is an important aspect of the study of adaptive behavior. Predefined languages for multiagent systems may not be appropriate as they reflect the designer's viewpoint rather than the agents', and are unable to adapt to changing environmental conditions and task definitions. It is much more desirable for the agents to be able to create and maintain their own language. This is not an easy task, however, as the mechanisms of language evolution are far from being well understood.

The last decade or so has seen increasing application of computational methods to the study of language evolution [1], [2], [3]. The main mathematical approach, meanwhile, is to apply models of biological evolution to the evolution of language(s) [4]. In this case, the languages themselves are considered the units undergoing selection and mutation. These models have been used to address questions about convergence [5], and the emergence of syntax [6], for example.

One of the main problems in language evolution, which has received little attention so far, is how to get a population of agents to converge to a common language, without globally imposing some kind of hierarchy on the population. In

other words, how does the topology of agent interactions affect the convergence to a common language?

The topology clearly has an important role to play in convergence. For example, if the population is split into two disjoint subgroups, then they cannot converge onto a single language except by chance. Even if the topology consists of a single component, multiple languages might co-exist in the population, especially if the rate of change of the language (in response to environmental changes, for example) is high in relation to diameter of the network. In other words, languages might be changing faster than they can propagate across the network.

In this work we show that we can take advantage of evolutionary dynamics to actually construct the agent interaction topology on the fly. This is done by a subtle change of focus. Instead of the languages being treated as the units of selection and replication, we treat the interaction links between agents as the units undergoing selection and replication.

The rest of this paper is organized as follows. We first describe some recent work investigating the role of the interaction topology in the convergence of language and emergence of social conventions in multi-agent systems. This is followed by a discussion of our model for generating agent interaction topologies, which is based on the evolutionary framework described by the replicator-mutator equation. We show that this mathematical model is valid through some simple simulations. Then we go on to do a language learning experiment using simple recurrent neural networks. Finally we discuss the possibilities for expanding on this work to include situatedness and further numerical exploration of the theoretical model.

2 Related Work

There has been some significant work on the convergence of a population of agents to a particular language. Komarova et. al [7], and Lee et al. [8] have studied the problem from the point of view of population dynamics, while Dall'Asta et al. [9] have studied the dynamics of the naming game [10] on small-world networks, and Lieberman et al. [11] have introduced evolutionary graph theory, which is the study of evolutionary processes on graphs.

The model of Dall'Asta et al., while very interesting, is not really an evolutionary model since there is no notion of selection or variation in it. Therefore we will not discuss it further here.

Lee et. al studied the role of the interaction topology on the convergence of a population to a single language. This study looked at a set of specific interaction topologies, including fully connected, linear, von Neumannn lattice, and a bridge topology. Using the model of Komarova et. al, which assumes random pairwise interactions between all agents, they empirically studied the critical learning fidelity threshold for language convergence in the various topologies. Although several different interaction topologies were used, the topologies were not emergent and were specified beforehand by the creator of the experiments.

In addition, the agents did not learn a language from interactions with other agents, but rather neighbors of high fitness agents were transformed into copies of the high fitness agent with some fixed probability. Lieberman et al. put their work on a firmer theoretical basis by studying the probability of fixation (i.e. the probability that a fitter language, if it appears by mutation, will be adopted by the entire population) on a graph. They showed that some graphs can be selection amplifiers, in that the probability of fixation can be made as high as possible, and also that some graphs are selection suppressors.

Here we ask the question, can the agents generate the topology on the fly, while still ensuring that the emergent topology leads to rapid convergence?

3 Agent Interaction Topologies and Convergence

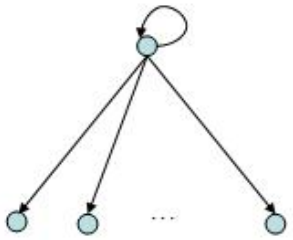

Fig. 1. The interaction graph for quickest convergence. One agent teaches the language to all the other agents. The circles represent agents, and the directed arrows represent the influence of one agent on the language of the other agent.

The agent interaction topology is a weighted directed *influence* graph which describes the influence of an agent on the language of another agent. Such a graph captures constraints such as spatial locality, agents' knowledge of each others' existence, interaction choice preferences, etc.

A simple strategy for rapid convergence would be to designate a special agent from which each of the other agents learn their language, as in figure 1. This corresponds essentially to a pre-imposed or designed language, which may or may not be of the highest objective quality. Such a centralized system is brittle in practice because a) the teacher agent has to be responsible for adapting the language to keep up with changing tasks, environments and needs of all the agents, b) communicative load on the teacher increases at least linearly with population size, reducing scalability, and c) the centralized teacher is a single failure point. Multi-agent systems are generally distributed and open, which means that there is no central control point, and agents may enter and leave the population at any time. This means that although desirable for its speed,

uniformity, and certainty, the interaction topology shown in figure 1 is both undesirable and unrealistic for a general multi-agent system.

It would be much better if agents could develop their interaction topologies on the fly, by selecting interaction partners autonomously. We would still like, however, to have some guarantee of convergence, and of rapid convergence. In this regard, we next discuss the Noisy Preferential Attachment algorithm which we can use initially to generate scale-free topologies, and also to guarantee convergence. The final emergent topology, as we will see, looks a lot like fig. 1, but the crucial distinction is that it is emergent. Thus, e.g., if the central agent left the population, another would emerge to take its place.

4 Noisy Preferential Attachment

We first derive a variant of the replicator-mutator equation (RME), the RME without Death (RME-WD). Then we show that the preferential attachment model of small-world network generation is a special case of the RME-WD. We then use this equivalence to give a version of the preferential attachment algorithm, called Noisy Preferential Attachment, which we will later use to generate the agent interaction topology.

The Replicator-Mutator Equation (RME) describes the rate of change of the proportion of *types* (genomes, languages) in a population undergoing replication and mutation.

Suppose there are N types in a population of n individuals. Let f_i be the fitness of an individual of type i. Since fitness includes both frequency-independent and frequency-dependent components, it is written as,

$$f_i = w_i + \sum_{j=1}^{N} a_{ij} x_j, \tag{1}$$

where x_j is the proportion of individuals of type j in the population, and w_i is a measure of intrinsic fitness of the language which might be related to learnability, expressiveness, etc.. The matrix $A = [a_{ij}]$ is known as the payoff matrix, and can be thought of as the payoff or reward achieved by an individual of type i in an interaction with an individual of type j. In the case of languages, A can be thought of as a measure of intelligibility, i.e. the degree to which a speaker of language i understands a speaker of language j.

The total number of individuals added to the population in a time step is $\sum_{j=1}^{N} f_j x_j n$. Further, replication is imperfect. With a small probability, replicating an individual of type i results in an individual of type j. This is quantified by a matrix $Q = [q_{ij}]$, where q_{ij} is the probability that replication of an individual of type i results in an individual of type j. In the case of languages, this corresponds to learning fidelity. In a limited interaction between individuals, the learner may not learn exactly the teacher's language.

Suppose also that the size of the population is held constant at n, by removing an equal number of individuals uniformly randomly, as are added to the

population. This is because, in the case of language learning, when an agent learns a new language, it necessarily replaces the old language of that agent. Then the number of individuals of type i that are removed in one time step is $x_i \sum_{j=1}^{N} f_j x_j n$.

Putting all these terms together, we get the rate of change of the proportion of individuals of type i in the population, as

$$\dot{x}_i = \sum_{j=1}^{N} f_j x_j q_{ij} - x_i \phi, \tag{2}$$

where $\phi = \sum_{j=1}^{N} f_j x_j$. This is the Replicator-Mutator Equation (RME).

4.1 Replication-Mutation without Death

If there is no death, the number of individuals of type i at the next time step is,

$$x_i^{'}(n + \sum_{j=1}^{N} f_j x_j n) = x_i n + \sum_{j=1}^{N} f_j x_j q_{ij} n.$$

Here $n + \sum_{j=1}^{N} f_j x_j n$ is the total size of the population at the next time step, and $\sum_{j=1}^{N} f_j x_j q_{ij} n$ is the number of new individuals of type i. Rearranging, and letting $\phi = \sum_{j=1}^{N} f_j x_j$, we get

$$x_i^{'}(1 + \phi) = x_i + \sum_{j=1}^{N} f_j x_j q_{ij}$$

$$x_i^{'}(1 + \phi) - x_i(1 + \phi) = \sum_{j=1}^{N} f_j x_j q_{ij} - x_i \phi$$

Thus the rate of change of the proportion of type i is,

$$\dot{x}_i = \frac{\sum_{j=1}^{N} f_j x_j q_{ij} - x_i \phi}{1 + \phi} \tag{3}$$

This is the Replicator-Mutator Equation without Death. Note that, since $f_j \geq 0 \; \forall \; j$, the denominator on the right-hand side is always positive. Therefore the critical points of the RME-WD are the same as those of the RME.

The general form of the RME is very difficult to study because of the large number of parameters (all the entries of the A and the Q matrices). Often a special symmetrical case is studied, where the A matrix is set to have diagonal values equal to 1, and off-diagonal values $a << 1$, and the Q matrix is similarly set to have diagonal values p (close to 1), and off-diagonal values $(1-p)/(N-1)$. A complete analysis of the critical points is possible in this *fully symmetric* case [12].

In particular, when p is less than a critical threshold, the system has only one attractor, where all types are present in equal proportion in the population. For large values of N and small values of a, this threshold is approximately 0.5. Above this value, the attractor turns into a repeller, and the only stable attractors that emerge correspond to the situation where a single type dominates the population. Note that in the absence of death, and presence of mutation, there will always be all types present in the population, but the *proportion* of one of the types goes towards one. Since the system is fully symmetric, it could be any one of the N types that eventually dominates the population, and which attractor the system falls into depends on the initial conditions, and statistical fluctuations.

We now shed some light on the transient behavior of the RME-WD, under certain special conditions, by showing its equivalence with the preferential attachment algorithm of small-world network generation. This means that, under the right initial conditions, if we create a network (as we will later describe) using the RME-WD, the network will be a scale-free network (in the short term).

4.2 Preferential Attachment and the Underlying Probabilistic Model

The Preferential Attachment algorithm is the most commonly cited model of small-world network generation [13]. Small-world networks are graphs which have three properties: a small diameter, a high clustering coefficient, and a power-law degree distribution. The clustering coefficient is defined as the average fraction of neighbors of a node that are also neighbors of each other. Barabasi and Albert showed that a small-world network can be generated by preferential attachment, as follows.

We start the network with a small number of nodes and links, say two of each, randomly connected. At each step, we add a node to the network and add a link from the new node to one of the pre-existing nodes with probability proportional to the number of in-links that node already has. Thus, the probability of node i acquiring a new link is,

$$P(i) = \alpha x_i^{\gamma}, \tag{4}$$

where x_i is the proportion of in-links that go to node i, γ is a constant, and α is a normalizing term. γ is generally set to 1, in which case α is also 1.

This process results in a small-world network. There are a couple of things worth noting here. First, since new nodes don't have any in-links, the probability of acquiring any in-links is zero for these nodes. To get around this problem, every node is assumed to have one pseudo-link, i.e. the number of in-links for each node for the purposes of preferential attachment, begins at 1. Second, since new nodes are added at every time step, the number of links remains approximately equal to the number of nodes in the network. In later work, Albert and Barabasi modified the preferential attachment algorithm to allow rewiring of links with some small probability, and also to allow adding links without adding nodes with some small probability [14], but the essential algorithm remains the same as that described above.

The underlying probabilistic model is an instance of a Polya's urn model, as described below (and also in [15]).

Imagine a set of N urns which are all empty except for one, which has one ball in it. We now add balls one by one. A ball is put into urn i with probability proportional to the number of balls already in that urn (plus one "pseudo-ball").

This process is clearly equivalent to the preferential attachment algorithm with the caveat that we have fixed the number of urns to be N. An urn represents a node and a ball represents an in-link. In the short-term, i.e. while the number of balls is of the same order as the number of urns, this probabilistic model represents the small-world network generation process.

We now add a further step to it to make it equivalent to the RME-WD. We introduce a *transfer matrix*, Q, which is the same as the mutation matrix in the RME. Suppose a ball is added to urn i at time step t. Then a ball is taken out of urn i and moved to any of the urns with probability q_{ij}. This is similar to later versions of the preferential attachment model which include rewiring.

This probabilistic model captures the RME-WD dynamics if we consider urns to correspond to types and balls to individuals in the population. Since it is the balls that correspond to the individuals undergoing replication and mutation, we have to set the payoff matrix, A, equal to the null matrix in this case to get the linear dependence of $P(i)$ on x_i. Note that in this case, the RME-WD loses its frequency-dependent aspect. If we set $A = I$, the identity matrix, $P(i)$ varies as the square of the proportion of individuals of type i. If the off-diagonal elements of A are set to be non-zero, then $P(i)$ acquires additional second-degree terms.

We call this extended (but still finite) version of preferential attachment, Noisy Preferential Attachment (NPA) [16], because of the introduction of the mutation matrix into the probabilistic model. A caveat is in order here too: there is no notion of pseudo-links (or pseudo-balls) in this model. New nodes (types) are introduced into the graph (population) by the mutation process. This means that the number of nodes increases much more slowly that it does in the preferential attachment case. Therefore to generate a large network, the initial state needs to include a fairly large number of nodes with non-zero number of in-links. Alternatively, the mutation rate needs to have a high value.

5 Using NPA to Generate Agent Interaction Topologies

We use the NPA algorithm to generate the agent interaction topology on the fly in two stages as follows. Initially the agents have no knowledge of (the quality of) each other's languages. Therefore the first stage is an exploration phase, which sets up the second convergence phase. In the exploration phase, an agent Alice chooses another agent, Bob, as a teacher with probability proportional to Bob's fitness. The fitness of an agent is equal to the number of times that agent has been chosen as a teacher. The fitness can also include a term that is independent of the frequency of selection as teacher, but for now, we ignore this term since our current simulations are ungrounded. With probability $(1 - p)$, Alice switches to a uniformly randomly chosen teacher. This is similar to the

notion of exploration-exploitation in reinforcement learning. The intuition is that if a lot of agents are choosing a particular agent as teacher, then choose that agent as a teacher because a lot of agents consider its language to be good. However, the proportions might be misleading near the beginning of the process because the actual counts will be low. Therefore it makes more sense to explore rather than exploit at the beginning of the distributed language learning process, i.e. it makes sense to start out with a high value of the mutation rate, $(1 - p)$, and switch to a low value when the process has been going on for a while.

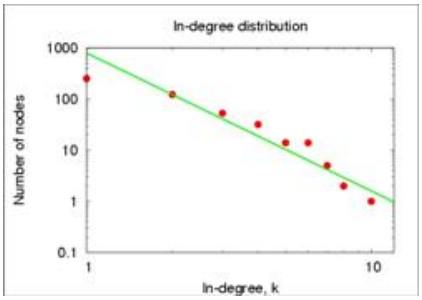

Fig. 2. The degree distribution for $p = 0.3$ and $\gamma = 1$, after 1000 links have been added. The number of nodes in the graph is 1000 as well. The distribution is clearly a power law.

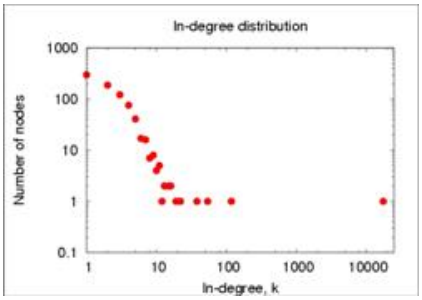

Fig. 3. The degree distribution after 20,000 links have been added to the graph. One node is clearly dominant.

Figure 2 shows a simulation in which we have a population of 1000 agents, i.e. a graph with 1000 nodes. Initially, one link is randomly added to start the process off. The initial value of p is 0.3, and γ, which is the exponent of the proportion in the preferential selection equation, is set to 1. We add one link at each time step, and figure 2 shows the in-degree distribution after 1000 links have been added. The graph is plotted on a log-linear scale, and the distribution is clearly a power-law. Therefore, at this stage, the graph is a scale free network.

At this point we start the second stage, by changing the value of p to 0.95, and the value of γ to 2. The intuition is that once the space of languages has been sufficiently explored, we can switch to the "convergence mode", where we trust the statistics of interactions that have been established in the first stage to guide us to a good overall language.

As we continue adding links, the node with highest degree becomes the dominant node. Figure 3 shows the degree distribution after 20,000 links (total) have been added. We can see that a single node has acquired a far larger proportion than the rest, and because of the frequency-dependent effect, the proportion of links acquired by this nodes will continue to increase towards 1 as we continue adding links to the graph. This means that the population will converge to the

language of this agent. If this agent later gets removed from the population, the next most "fit" agent will become the dominant agent. It may possibly have a different language, though.

6 A Language Learning Experiment

We now do a simulation where we have a population of agents trying to converge onto a common language by learning from each other. The agent interaction topology is generated as described above. The agents use simple recurrent neural networks to generate, parse, and learn sentences. Each simple recurrent network has 5 inputs, 3 hidden layer nodes, and 5 outputs. There are 5 symbols in the "languages", `{a,b,c,d,e}`, and we use a 1-of-n encoding, i.e. the symbol `a` is encoded as the vector $[1, 0, 0, 0, 0]$ at the input of the neural network. A sentence is generated from a simple recurrent network by setting its internal state to 0.5 and giving it a random initial input vector. The output of the neural net is then fed back to its input and this process is repeated until we have generated as many symbols as we want. The weights of the neural networks are initialized randomly in the range $[-0.5, 0.5]$.

The population size was set to 100, and the experiment was run for 1000 time steps. At each time step, an agent is selected in sequential order, and it chooses a teacher according to the NPA algorithm. It receives a sample of 100 sentences of length 10 from the teacher and trains on this sample to convergence or for 100 epochs, whichever comes first. Every 50th time step, we collect 5 randomly generated sentences from each agent to form a testing set and the one-step symbol prediction error is calculated for each agent on this testing set. These are summed up to indicate the error (the inverse of convergence) of the entire population. This value is plotted in figure 4.

The parameters for the NPA algorithm were set in a manner similar to the previous section. Since there are 100 agents, i.e. 100 nodes in the graph of the agent interaction topology, we set $p = 0.3$ and $\gamma = 1$ for the first 100 steps, and then changed these values to $p = 0.95$ and $\gamma = 2$ for the remainder of the simulation. As we see in figure 4, the error only starts dropping after time step 100. However, after that the error drops quite rapidly and reaches almost zero by time step 1000. As a comparison, we also plot the error with uniformly random teacher selection. We see that convergence is attained much faster with the NPA algorithm.

7 Discussion and Future Work

The two stages of the distributed convergence mechanism described in this work combine the ideas of convergence to a common social convention and convergence to a common language, through the mechanism of frequency-dependent (or preferential) selection.

The goal of the first, or exploratory, stage is to evaluate the languages that are present in the population and collectively decide on a single language, embodied

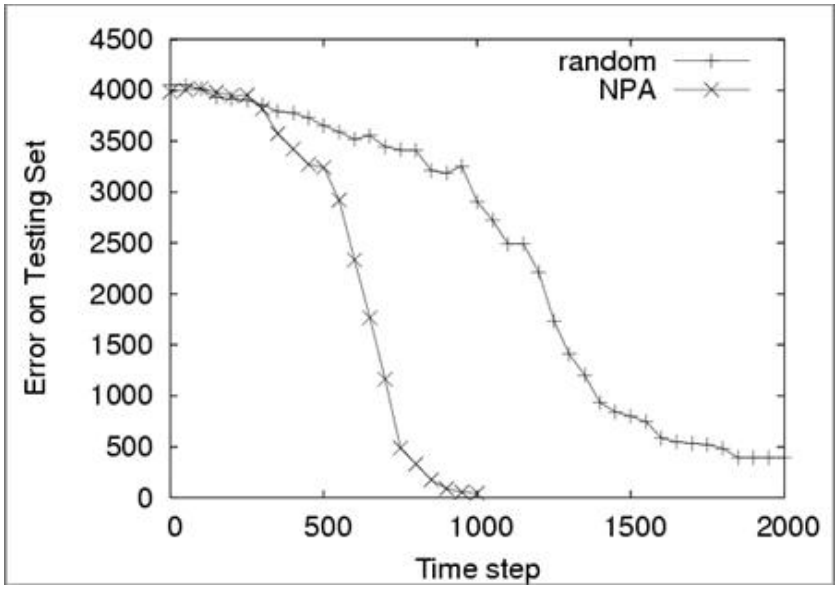

Fig. 4. The one step symbol prediction error summed over all the agents on the testing set. The testing set is generated by sampling 5 sentences from each of the agents at that particular time step. The low prediction error at the end ($\sim 1\%$) indicates almost perfect convergence of the language.

by a single teacher, as the language to converge upon. The second stage then focuses on all the agents learning this one language, again in a distributed way, by simply changing two global parameters of the system: the probability of switching to a random teacher rather than the preferentially selected one, and the exponent which determines how strongly the preferential selection mechanism works. The underlying theoretical model guarantees that the appropriate parameter values will result in both a power law initial distribution of links, and the dominance of one teacher in the second stage.

Another important point, which was not mentioned in the language learning experiment, is that learning is non-trivial. With all learning architectures and algorithms, including the simple recurrent networks trained with the delta rule and backpropagation that we have used, some languages are easier to learn than others. In the work of Lee et al., e.g., this is captured by the learning fidelity parameter [8]. However, the point is that in practice, this parameter is non-uniform across the language space. "Simple" languages can be learned with greater fidelity than more complex ones. For example, it is much easier to converge upon point attractors, i.e. languages consisting of a single symbol, with simple recurrent networks, than to converge upon other kinds of attractors: e.g. languages consisting of alternating symbols or other more complex languages. In the language learning experiment shown, it is much harder to attain convergence when the selected teacher has a complex language, and convergence depends on

having appropriate parameters settings for the neural networks (such as number of hidden layer nodes, learning rate, momentum, etc.).

There are many details that remain to be fleshed out in the model. The algorithm given is not truly a distributed algorithm since it hasn't really been specified from the point of view of a single agent. We need to explicate how the exploration phase works at the beginning when the agents have no information about each other. We can imagine that each agent maintains a list of its own estimate of all the other agents' fitness values. They can all be initialized to zero, and then a few of these get filled in by a "recommendation" mechanism corresponding to the preferential selection. In other words, an agent chooses a teacher randomly (or chooses a neighbor), and then gets referred to a better teacher by this one based on the teacher's knowledge about the population. This is where a scale-free (or small-world if possible) nature of the agent interaction topology helps. The small diameter of such a network means that it is easy for an agent to find a good teacher using such a referral mechanism.

In future work, we intend to explore a more grounded language learning case. As a first step, we need to investigate the effects of making the frequency-independent part of fitness non-zero. A second concern is the quality of language that is converged upon. As pointed out above, there is an inherent bias in the population towards learning simple languages, as learning fidelity is higher in this case. There has to be a corresponding pressure towards language *complexification*, perhaps from a task based reward, otherwise the learned language will almost surely be the simplest possible.

8 Conclusions

We have outlined a system capable of converging onto a common language in a distributed manner. It relies on the framework of language evolution, with a change of focus: instead of treating the languages themselves as the individuals undergoing replication, selection and mutation, we treat the links in the agent interaction topology as the evolutionary units.

We further showed that under certain special conditions, we can recover the preferential attachment algorithm of small-world network generation from the replicator-mutator system of evolution. This allowed us to give a two stage model of distributed language learning, based on our Noisy Preferential Attachment algorithm. In the first stage, the agents explore the languages present in the population and generate a scale-free network of interaction. In the second stage, the parameters are changed to allow rapid convergence by letting a dominant "teacher" node emerge through the same evolutionary dynamics.

We demonstrated through a simple language learning experiment using simple recurrent networks, that such a system can converge to a common language autonomously and rapidly.

In the near future we intend to explore the parameter space of the system more thoroughly, both numerically and theoretically, through more grounded simulations.

9 Acknowledgements

We thank Kiran Lakkaraju for help with the programming of the language learning experiment, and for very helpful feedback. We thank the UIUC Language Evolution Group for very helpful and stimulating discussions. This work was supported in part by NSF grant IIS-0340996.

References

1. Steels, L.: The evolution of communication systems by adaptive agents. In Alonso, E., D.K., Kazakov, D., eds.: Adaptive Agents and Multi-Agent Systems: Adaptation and Multi-Agent Learning. LNAI 2636, Springer Verlag, Berlin (2003) 125–140
2. Smith, K., Kirby, S., Brighton, H.: Iterated learning: A framework for the emergence of language. Artificial Life **9**(4) (2003) 371–386
3. Batali, J.: Computational Simulations of the Emergence of Grammar. In: Approaches to the Evolution of Language: Social and Cognitive Bases. Cambridge University Press, Cambridge (1998)
4. Nowak, M.A., Komarova, N.L.: Towards an evolutionary theory of language. Trends in Cognitive Sciences **5**(11) (2001) 288–295
5. Komarova, N.L.: Replicator-mutator equation, universality property and population dynamics of learning. Journal of Theoretical Biology **230** (2004) 227–239
6. Nowak, M.A., Plotkin, J.B., Jansen, V.A.A.: The evolution of syntactic commmunication. Nature **404** (2000) 495–498
7. Komarova, N., Niyogi, P., Nowak, M.: Evolutionary dynamics of grammar acquisition. Journal of Theoretical Biology **209**(1) (2002) 43–59
8. Lee, Y., Collier, T.C., Taylor, C.E., Stabler, E.E.: The role of population structure in language evolution. In: Proceedings of the 10th International Symposium on Artificial Life and Robotics. (2005)
9. Dall'Asta, L., Baronchelli, A., Barrat, A., Loreto, V.: Agreement dynamics on small-world networks. Europhysics Letters (2006)
10. Steels, L.: A self-organizing spatial vocabulary. Artificial Life **2**(3) (1996) 319–332
11. Lieberman, E., Hauert, C., Nowak, M.A.: Evolutionary dynamics on graphs. Nature **433** (2005) 312–316
12. Mitchener, W.G.: Bifurcation analysis of the fully symmetric language dynamical equation. Journal of Mathematical Biology **46**(3) (2003) 265–285
13. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286** (1999) 509–512
14. Albert, R., Barabási, A.L.: Topology of evolving networks: Local events and universality. Physical Review Letters **85**(24) (2000) 5234–5237
15. Chung, F., Handjani, S., Jungreis, D.: Generalizations of Polya's urn problem. Annals of Combinatorics **7** (2003) 141–153
16. Swarup, S., Gasser, L.: Unifying network and evolutionary dynamics. In Preparation (2006)