# TDT4310 Project

Jan Burak

April 26, 2019

**Abstract**

The study of language evolution concerns itself with modelling the mechanisms that led to the emergence of languages as we know them today. As opposed to the study of written sources, language evolution requires generative models that aim to test the hypotheses of the origins. Naming game is a type of computational model that simulates interactions between two agents that are trying to achieve common understanding. Lipowska's model adds an evolutionary element to the naming game by introducing a survival chance based on an agent's communication successfulness. Thus, the sum of interactions between simple agents drive the global emergence of system's behaviour. This project aims to implement the Lipowska's model to acquire an understanding of computational models of language evolution. The outcomes of the resulting simulations are presented and any potential discrepancies are discussed. Also, potential additions to the model are presented.

# 1 Introduction

The study of the origins of language is an interdisciplinary field with contributions from computer science, linguistics, and evolution. As opposed to the majority of language studies, the origins lack historical sources, due to speech having evolved before writing. Thus, to study the ways in which languages could form, computational models are developed to simulate the mechanisms that could have led to languages as we know them today. One such type of models is the naming game, which is described as "a simulation-based numerical study exploring the emergence and evolution of shared information in a population of communicating agents"[1]. The naming game is used to explore various hypotheses of language formation regarding how a consensus on a new word is formed. By simulating the simple interactions of multiple agents, it is possible to study the global emergent behaviour of a system. The agents interact with each other with relation to a certain communication topology, and thus it is based on reaction-diffusion processes and graph theory.

In its simplest form, the interactions are based on two neighbouring agents, where one is a speaker and the other is a hearer. The speaker utters a word to the hearer, and if the hearer is capable of comprehending the word, the interaction is considered a success. The uttered words are either retrieved from agents memory, or if the inventory is empty, it is randomly generated. The way in which communication successes and failures are handled decides how the system evolves. This process continues until convergence on a common word, or after a given number of iterations. In addition to language evolution, this model has been also used to study opinion spreading, cultural development, and community formation.

This project aims to explore the field of computation models of language evolution through the implementation of a particular model of the naming game. The model of Lipowska[2] was chosen due to being close to the basic concept of the naming game, as well as its evolutionary elements. Lipowska's model intends to specifically simulate language evolution through agent death and reproduction based on communication successfulness, as well as incorporating a mutation mechanism. It uses a square grid lattice as its neighbourhood topology, which makes it well suited as an introductory model. The following sections describe mechanisms of Lipowska's model, explain how the model was implemented for this project, and finally show and discuss the results.

# 2 Method

Lipowska's model is based on the basic naming game where agents are organized in a square lattice. This means that every agent can interact with its von Neumann neighbourhood with Manhattan distance of one. When interacting with another agent, the instigator picks a word from its inventory to communicate it. If the other agent has the same word in their inventory, the communication is successful. The conveying of communication is ideal, so misunderstandings are not modelled. The choice of a word from an agent's inventory is based on each words weight. Thus, a given word has a probability of being chosen equal to its weight divided by the sum of all weights of the inventory words. Initially every random word starts with a weight of one. The updating of weights is driven by the agent's learning ability $l$, which is a real number between zero and one. A successful communication increases the weights of the word for both agents by the values of their respective learning ability. Otherwise, the speaker subtracts the value of its learning ability from the uttered word's weight, whilst the hearer adds the word to its inventory with an initial weight of one. If a word ends up with a negative weight, it is removed from the inventory. The inventory represents an agent's memory, and it is defined to be infinite. This means that memory storage and retrieval is greatly simplified compared to human cognition.

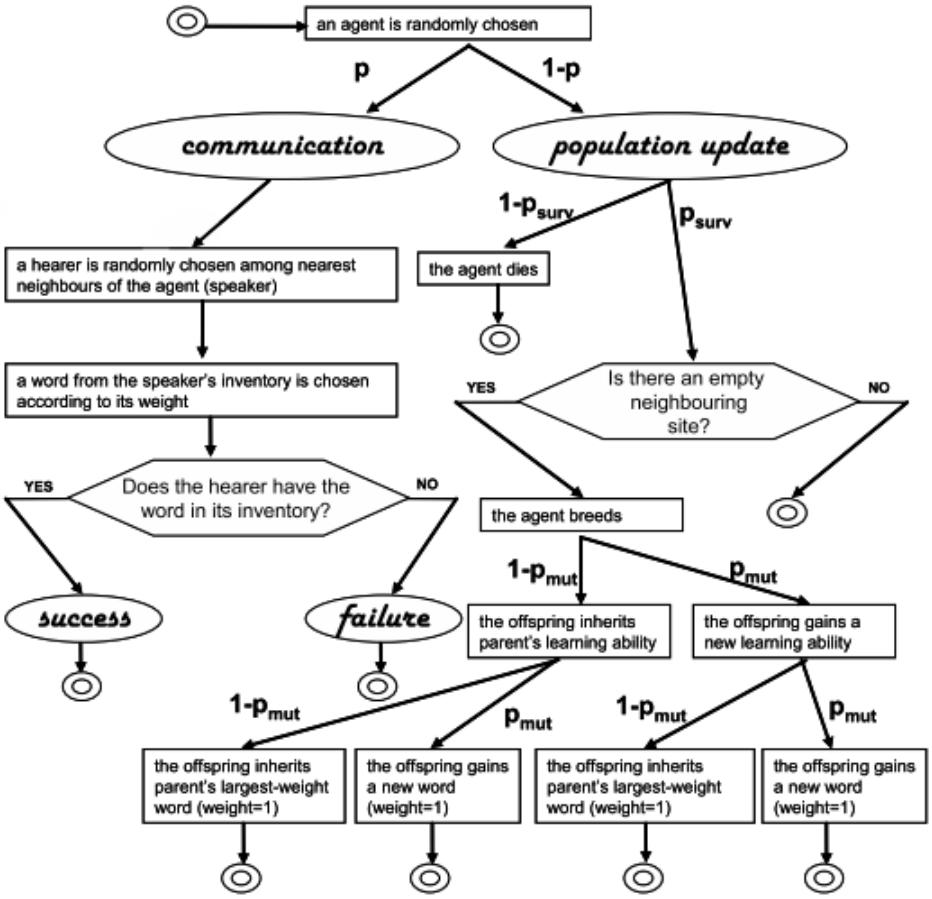



Figure 1: The actions that an agent in Lipowska's model performs. Taken from [2].

A main component of the model is communication probability. When an agent is chosen to act, the probability decides whether the agent will attempt to communicate, or compute its vital functions. This is summarized in figure 1. If the outcome is to communicate, a random neighbour will be chosen as the hearer. Otherwise, a different probability decides whether the agent will die or attempt to reproduce. The formula is $p_{surv} = exp(-at)[1 - exp(-b \sum_j w_j/\langle w \rangle)]$ and it is the driving component behind the evolution of the system. The constants a and b are responsible for population turnover and are equal to 0.05 and 5 respectively. $\sum_j w_j/\langle w \rangle)$ is the sum of inventory weights divided by the average of the total sum of the inventory weights of the population. The variable $t$ represents the age of the agent. No explicit aging mechanism is described in the paper. The agent dies with probability $1 - p_{surv}$, making it unable to act in any way anymore, and leaving an empty spot in its place in the lattice. Otherwise, with probability $p_{surv}$, the agent attempts to reproduce, choosing a random empty spot in its neighbourhood and spawning an offspring there. The offspring starts with either the parent's highest weighted word, or with a new word with probability $p_{mut}$. The highest weighted word also has a chance of $p_{mut}$ to become mutated. At the same time the learning ability of the offspring is random with probability $p_{mut}$, or otherwise inherited from the parent.

The model is used to investigate the spread of languages over time. In this context, a language is defined as the highest weighted word of an agent. Given different configurations of the described probabilities, the model produces varying results, which can be used to uncover the relationships between the variables. [2] explores among others the influence of the communication probability on communication success rate, average learning ability, and the fraction of agents using the most common language.

# 3 Implementation

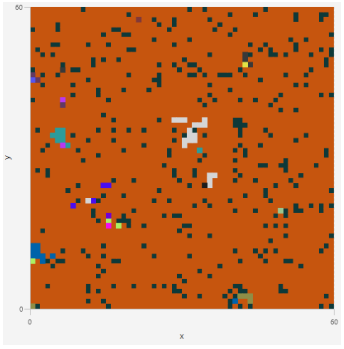
The implementation of Lipowska's model in this project was developed in Java, with JavaFx library for plotting. The main idea was to follow the description given in [2] during development and attempt to reproduce the results. Still, some ambiguity led to implementation choices that may differ from those of Lipowska. The implementation code has been uploaded to GitHub[3].

The main loop of the program is found in the class *Start*. This class contains most of the adjustable parameters that drive the simulation, as well as decide which data to collect during a run. A global random number generator is used to be able to reproduce a run when given its respective seed. The simulation starts with the generation of a lattice of agents that is based on the *latticeSize*. All the functions of an agent that are summarized in figure 1 are implemented in the class *Agent*. The agents are always initialized with a random learning ability and a single random word with weight one. In [2], different configurations of learning ability initialization were explored. The words are always generated based on the *wordSize*, which was set to four. Lipowska does not report any particular word size. Multiple runs showed that different values of word size do not have a significant influence above four, so it was left at this value for computational efficiency. The words are built from lowercase characters of the English alphabet with no additional rules. The words are implemented through the class *Word* that holds the string literal together with its weight. It also contains a function that mutates a given word by randomly changing one of its characters.

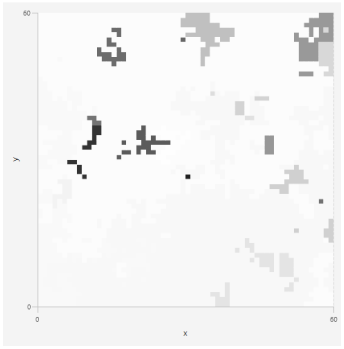
Lipowska writes that an agent is to be chosen randomly to either communicate or to perform the population update. The specifics of this mechanism in relation to iterations are not described, so two different mechanisms were implemented to compare the results. Each mechanism performs a number of updates equal to the total size of the lattice. One method computes all the updates in order, whilst the other picks random agents. In the random ways, some agents might be updated multiple times while the others are not updated at all in an iteration. Another ambiguity was how the agents age. The decision was to increase the age by one every time an agent is updated.

The *Agent* class contains the main functionality of the implementation. It holds the agent's learning ability, age, and word inventory. There are no restrictions on the inventory, so the memory is practically infinite. The function *trigger* is used to update an agent by with regards to the aforementioned probabilities and calls on the functions of the respective outcomes. The function *communicate* picks a word from the inventory based on the weighted probability, and chooses one of the neighbours. Furthermore, it calls the neighbours function *hear* to find out whether the communication attempt is successful, and adjusts the hearer's inventory accordingly. If an agent does not communicate, $p_{surv}$ is computed to find out whether the agent should reproduce or die. With probability $1 - p_{surv}$ the agent dies by setting its *dead* variable to false and clearing its memory. A dead agent acts as an empty spot that is available for reproduction. The function *breed* finds an empty spot in the neighbourhood and if it found, calls on the function *reproduce* to reinitialize an agent in that spot. Inheritance and mutation based on the constant $p_{mut}$ takes place here.

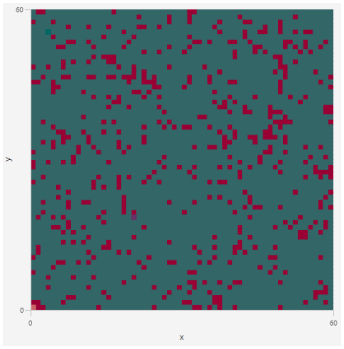
Additionally the program contains functionality for collecting and outputting various data from the simulations. Basic metrics such as births, death, and communication successes and failures are collected on the go. Furthermore, variables *languageCount* and *learningCount* decide respectively whether the average number of languages or the total average of the agent's learning abilities should collected every iteration. JavaFx's line charts are used to graph the changing values over iterations, whilst scatter chart is used to graph the lattice with relation to a given agent property.



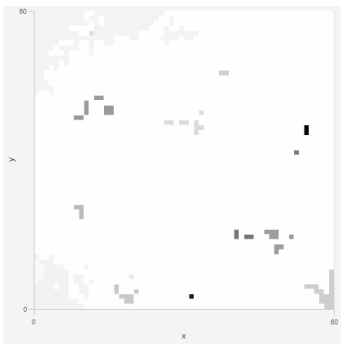
(a) Language scatter for $p_{com} = 0.15$ and lattice size 60.



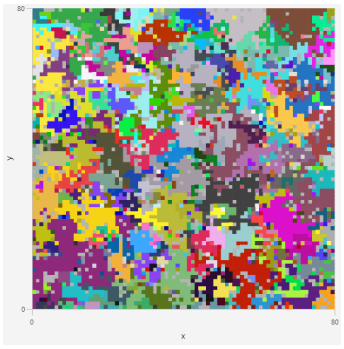
(b) Learning scatter for $p_{com} = 0.15$ and lattice size 60.



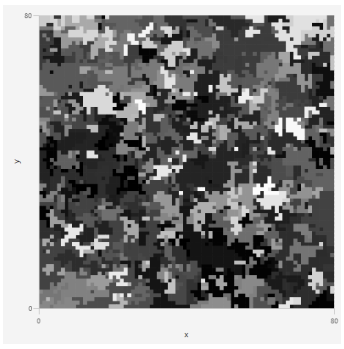
(c) Language scatter for $p_{com} = 0.30$ and lattice size 60.



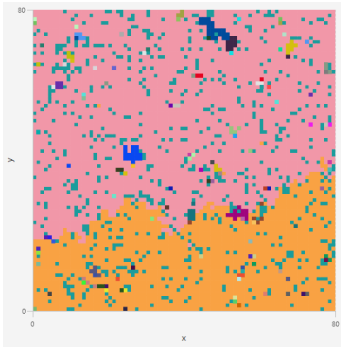
(d) Learning scatter for $p_{com} = 0.30$ and lattice size 60.



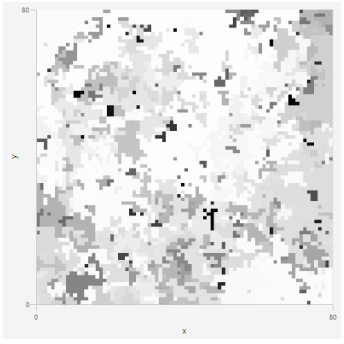
(e) Language scatter for $p_{com} = 0.15$ and lattice size 80.



(f) Learning scatter for $p_{com} = 0.15$ and lattice size 80.



(g) Language scatter for $p_{com} = 0.30$ and lattice size 80.



(h) Learning scatter for $p_{com} = 0.30$ and lattice size 80.

Figure 2: Visualisations of lattices with regards to languages and learning ability.

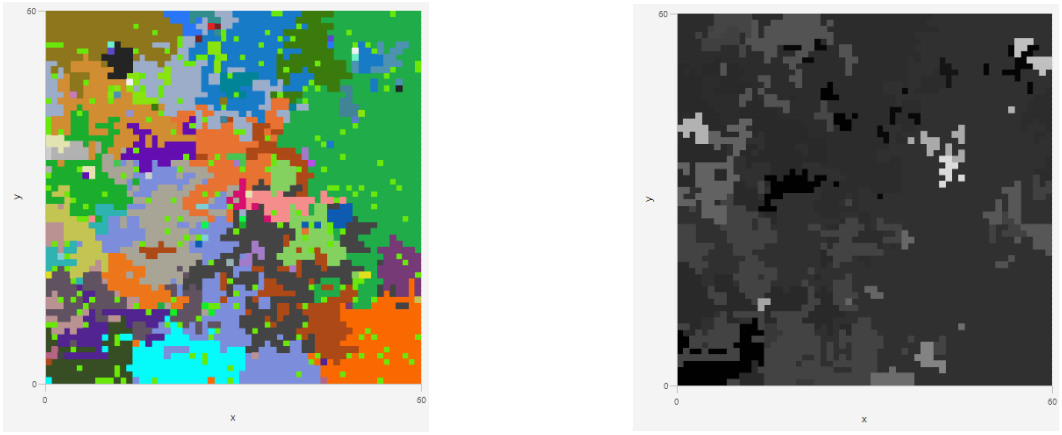
# 4 Results

Figure 2 shows the simulations for the configurations given in table 1, where $p_{com}$ is the communication probability, and seed is the input to the random number generator. The colors of the language scatter are arbitrary, whilst for the learning ability, white corresponds to value of one and black to zero, and respective shades of gray in between. The simulations were run for 100 000 iterations using the random mechanism for picking agents to update. The values in table 1 correspond to the configurations used by Lipowska (except the seed), and were used to test whether the simulations produce similar results. The expected behaviour would be that for a given value of $p_{com}$, a change in lattice size and $p_{mut}$ between the given value pairs would produce the same behaviour. Lipowska reports that for these configurations $p_{com} = 0.15$ does not produce a converging lattice with regards to language and learning ability spread. Meanwhile, for $p_{com} = 0.30$ the behaviour is reported to be converging. The biggest discrepancy in this project's results is between the language spreads of figures 2a and 2e. Whilst 2e does represent the expected behaviour of linguistic disorder, 2a shows linguistic coherence that does not agree with Lipowska's findings. The reason behind this can be the aforementioned ambiguity with regards the mechanism of picking an agent to update, as well as aging.

| Lattice size | 60 | 60 | 80 | 80 |
|---|---|---|---|---|
| $p_{com}$ | 0.15 | 0.30 | 0.15 | 0.30 |
| $p_{mut}$ | 0.001 | 0.001 | 0.01 | 0.01 |
| Seed | 2 | 2 | 1 | 1 |

Table 1: Parameters of the different simulation configurations.

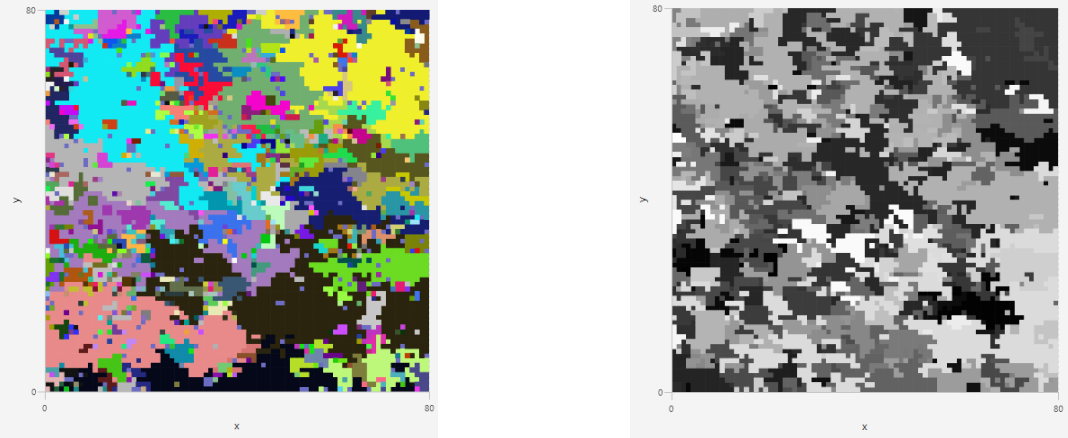
Attached in the appendix are the graphs that show the plots of average learning ability and number of languages with values taken every 10 iterations. Whilst every plot of the number of languages over iterations has the same form differing only in the stationary value, different behaviours can be seen in the average learning ability plots. It can be seen in figures 8 and 12 that the average learning ability converges relatively quickly to a value near one. These correspond to the simulations that were expected to, and mostly are, producing language cohesion. Figure 6 of the discrepant simulation 2a shows that it takes way longer for the learning ability to converge. Meanwhile, the change in the respective plots of the number of languages per generation in figures 5 and 7 is barely visible. Setting $p_{com} = 0.05$ for the lattice size 60 and $p_{mut} = 0.001$ does produce language decoherence as seen in figure 3a, and its evolution of learning ability in figure 13 does resemble that of figure 9.




(a) (b)

Figure 3: Language and learning scatter for $p_{com} = 0.05$ and lattice size 60.

Another interesting finding is that 2g does not produce a complete language cohesion, but instead there are two regions with distinct languages that together make up most of the lattice. The respective spread of learning ability in 2h shows that it is not as uniform as for the more coherent states of 2b and 2d. Furthermore, due to the ambiguity regarding the agent picking mechanism, the in-order method was tested to check whether it produced different results. It was tested with lattice size 80 and $p_{com} = 0.15$. Figure 4a shows that it does not produce language cohesion, but the continuous areas of languages are visibly larger than those of 2e. The same is true for the spread of learning ability. This shows that the choice of agents to update is a significant mechanism of the model.




(a) (b)

Figure 4: Language and learning scatter for $p_{com} = 0.15$ and lattice size 80 with in-order iterations

# 5 Conclusion

The discrepancies shown in the results section show the volatility of the simulations with regards to the implementations choices. Even though different kind of behaviours can be achieved by tuning the parameters, a thorough comprehensions of the used model is needed to be able to reproduce existing findings. The boundaries of different language groups produced by the simulations can be reminiscent of geographical entities such as countries. It would be interesting to apply similar mechanisms using different neighbourhood topologies. [1] explores alternatives to the square grid lattice, and mentions that "compared to regular lattices, the Erdös–Rényi random-graph network, Watts–Strogatz small-world network, and Barabási–Albert scale-free network models signicantly affect the achievements of such linguistic consensus". Lipowska's model on the other hand offers a simple introduction to computation models of language evolution, which seems to be a reasonable introductory model. Other mechanisms that may expand the model are limited memory size, modelling weights with a network depending on the success rate of past communications, as well as imperfect communication that models misunderstandings.

# References

[1] Yang Lou Guanrong Chen. Naming game, 2019.

[2] Dorota Lipowska. Naming game and computational modelling of language evolution, 2011.

[3] Jan Burak. Tdt4310 project 2019, github repository. `https://github.com/vinerothas/TDT4310_Project2019`.

# A Appendix

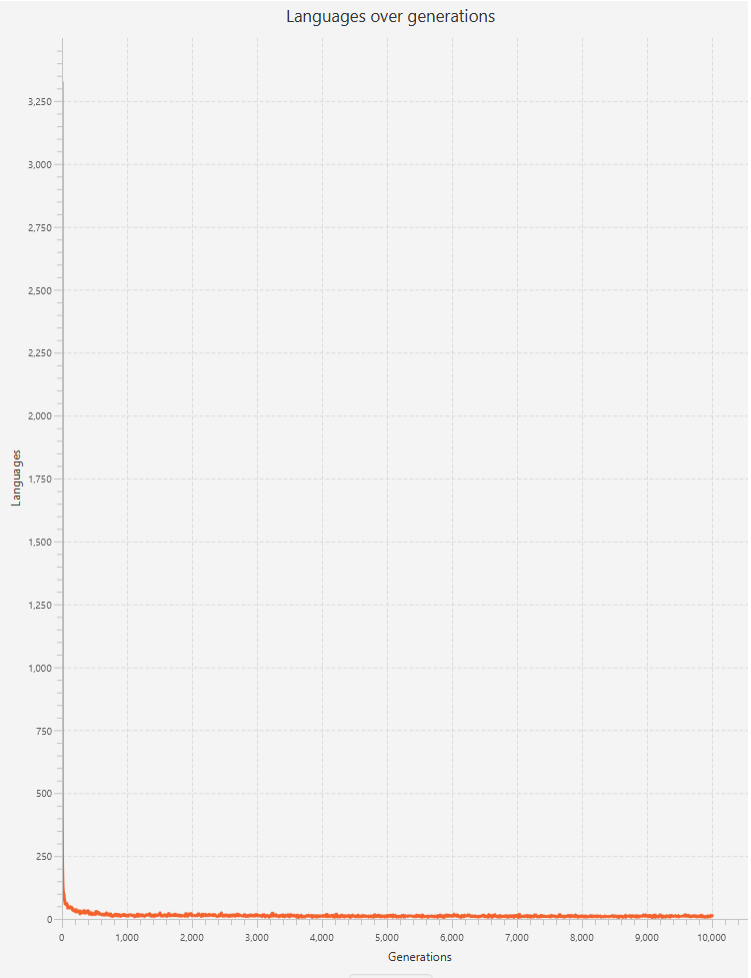



Figure 5: Language over iterations for $p_{com} = 0.15$ and lattice size 60.

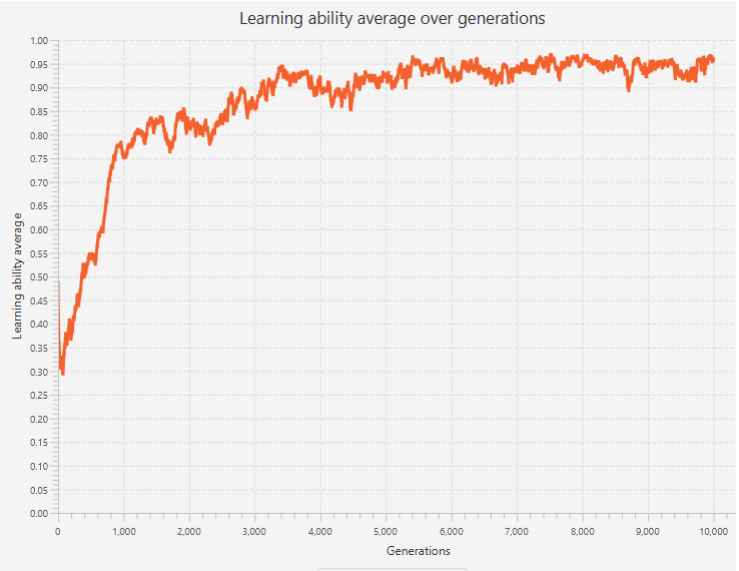



Figure 6: Average learning ability over iterations for $p_{com} = 0.15$ and lattice size 60.

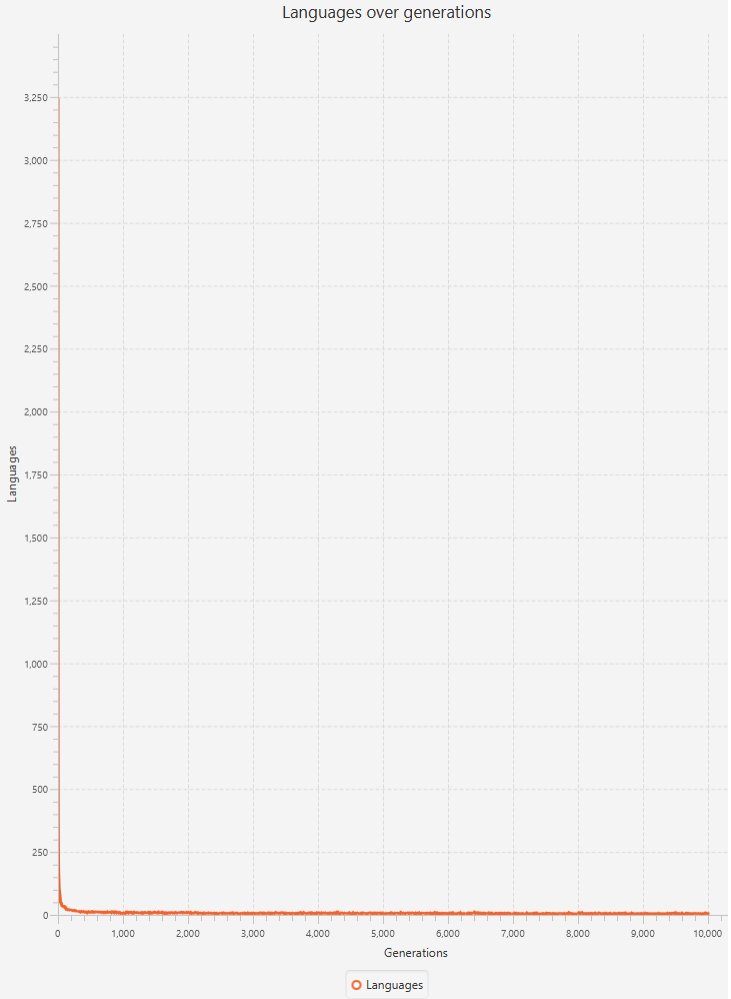



Figure 7: Language over iterations for $p_{com} = 0.30$ and lattice size 60.

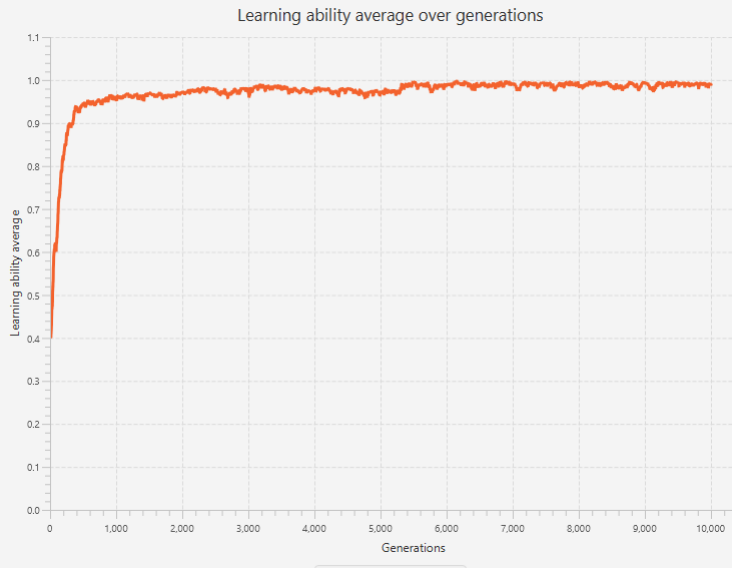



Figure 8: Average learning ability over iterations for $p_{com} = 0.30$ and lattice size 60.

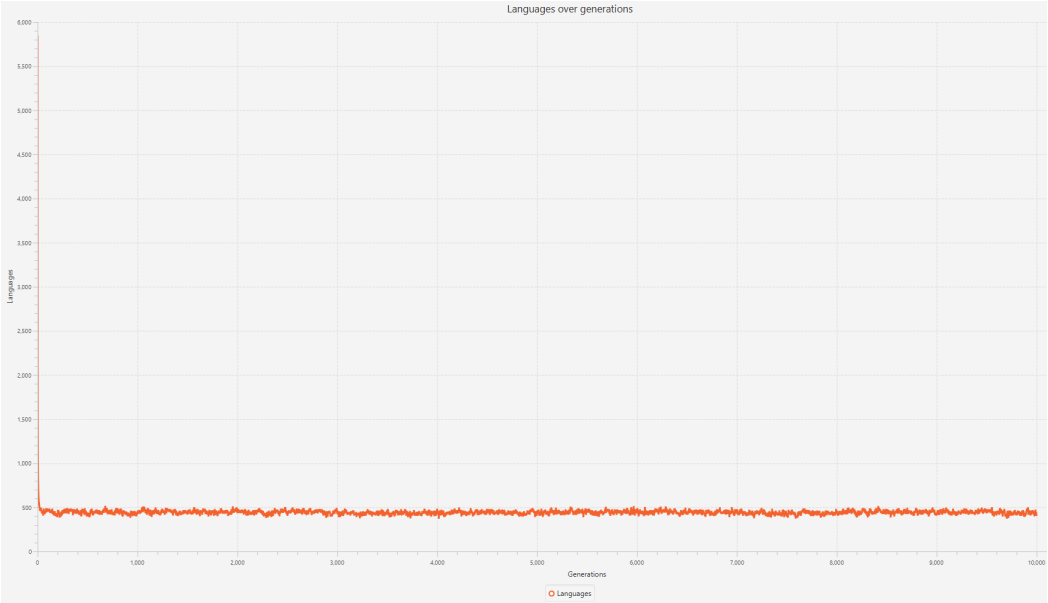



Figure 9: Language over iterations for $p_{com} = 0.15$ and lattice size 80.

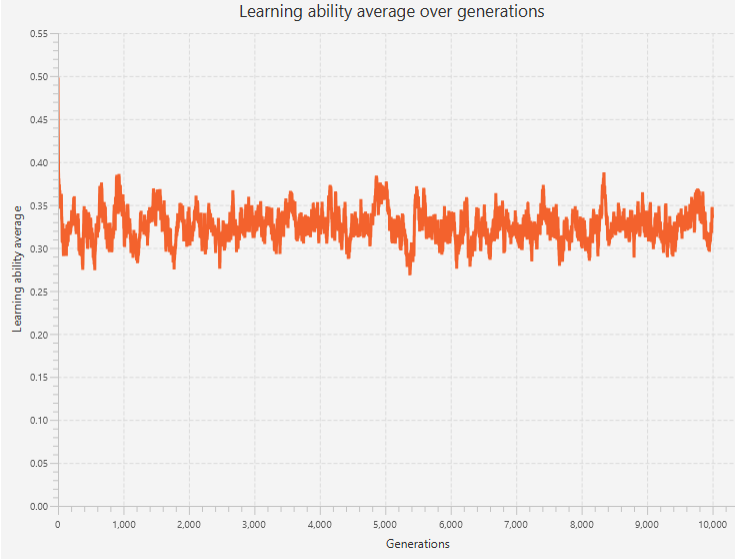



Figure 10: Average learning ability over iterations for $p_{com} = 0.15$ and lattice size 80.

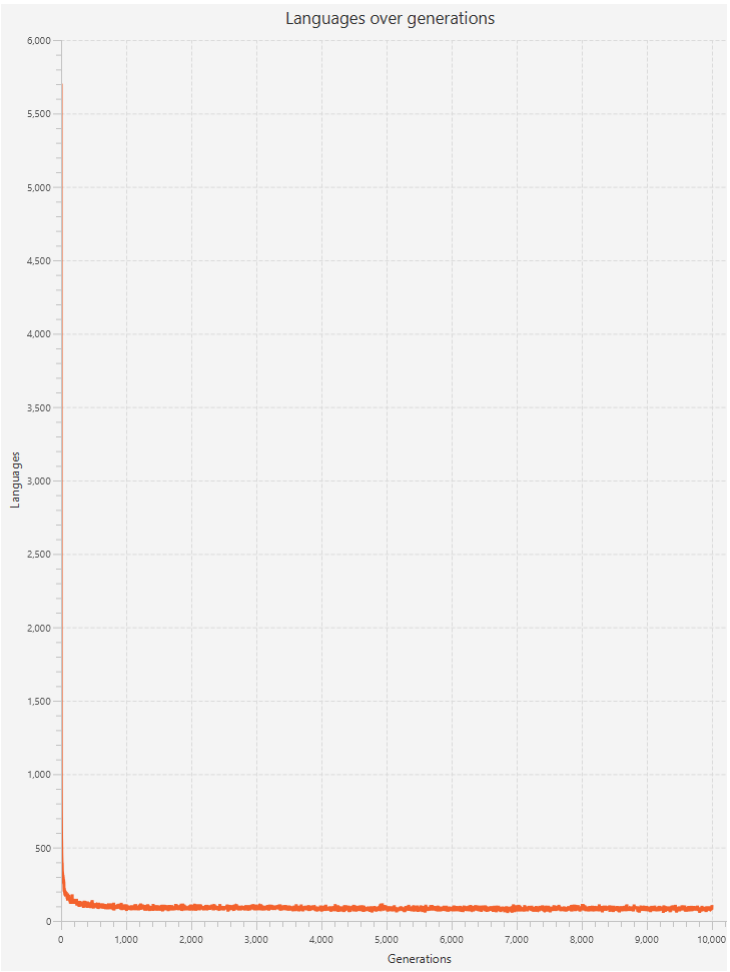



Figure 11: Language over iterations for $p_{com} = 0.30$ and lattice size 80.

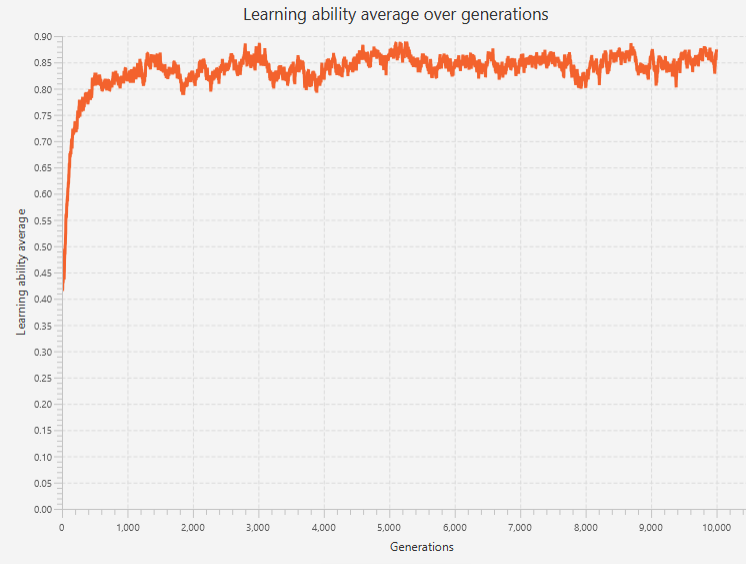



Figure 12: Average learning ability over iterations for $p_{com} = 0.30$ and lattice size 80.

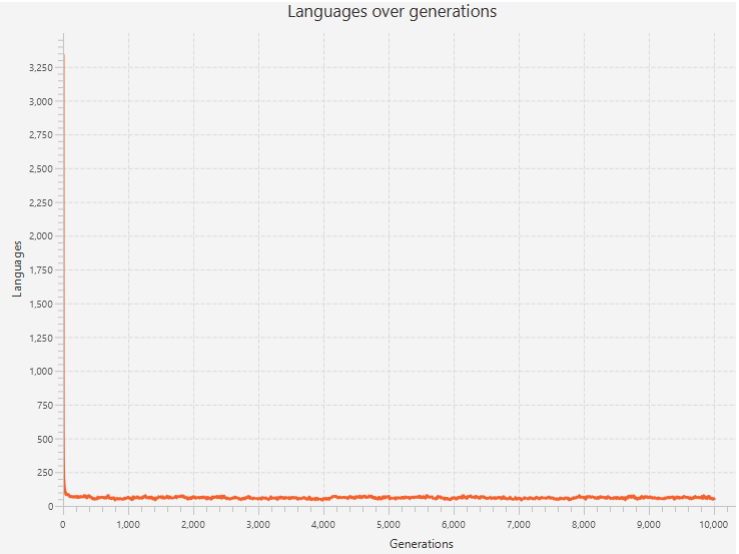



Figure 13: Language over iterations for $p_{com} = 0.05$ and lattice size 60.

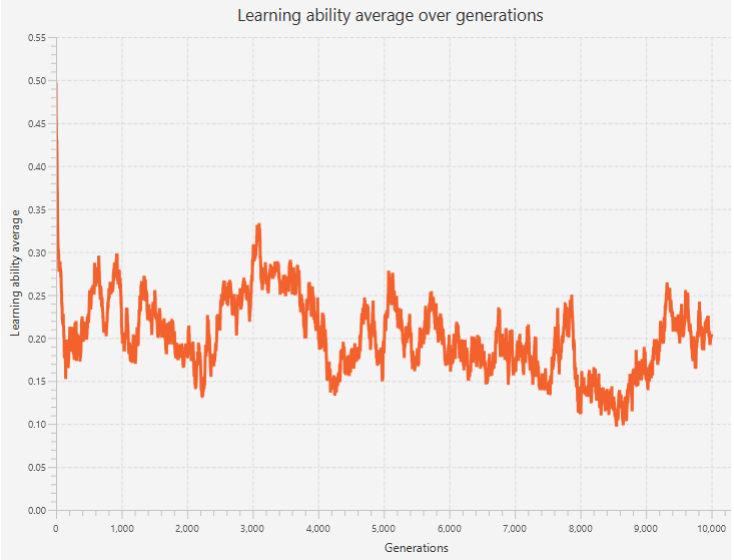



Figure 14: Average learning ability over iterations for $p_{com} = 0.05$ and lattice size 60.