Evolving Artificial Language through Evolutionary Reinforcement Learning

Xun Li[1] and Risto Miikkulainen[2]

[1,2]University of Texas at Austin
xun@cs.utexas.edu

Abstract

Computational simulation of language evolution provides valuable insights into the origin of language. Simulating the evolution of language among agents in an artificial world also presents an interesting challenge in evolutionary computation and machine learning. In this paper, a "jungle world" is constructed where agents must accomplish different tasks such as hunting and mating by evolving their own language to coordinate their actions. In addition, all agents must acquire the language during their lifetime through interaction with other agents. This paper proposes the algorithm of Evolutionary Reinforcement Learning with Potentiation and Memory (ERL-POM) as a computational approach for achieving this goal. Experimental results show that ERL-POM is effective in situated simulation of language evolution, demonstrating that languages can be evolved in the artificial environment when communication is necessary for some or all of the tasks the agents perform.

Introduction

Highly efficient and low-cost computer systems have made computational simulation possible at an unprecedented scale in recent decades. In the specific field of language evolution, computational simulation provides a complementary methodology that can help researchers develop detailed hypotheses on language origins and evolution and test these hypotheses in the virtual laboratory of simulation (Cangelosi and Parisi, 2002). Furthermore, from a technical perspective, an understanding of the fundamental principles in language evolution may lead to innovative machine learning algorithms and communication methods that are applicable to interactive software agents and multi-agent systems (Wagner, Reggia, Uriagereka, and Wilkinson, 2003).

Language is a powerful tool that helps humans coordinate actions to accomplish various tasks. It is also a skill that is acquired through lifetime learning. The purpose of this paper is therefore to establish a simulation framework, i.e. an artificial world and a computational method that captures these important features into a simulation of language evolution. Such a framework should then make it possible to gain new insights into evolution of natural and artificial language.

The first part of the simulation framework: "the jungle world", is an artificial environment in which agents attempt to hunt and mate through coordinated actions. Initially, the agent population have neither any knowledge on the rules of the world nor any existing code of communication. Through generations of evolution, the agent population must develop their own language and learn to use that language to coordinate their hunting and mating efforts. Additionally, for each agent, the language and the behavioral policy in the artificial world must be acquired through interaction with other agents and the environment during lifetime. Thus, the goal of evolution is to (1) evolve a language, (2) evolve it in service of coordinated behavior, and (3) evolve the ability for individuals to acquire it during their lifetime.

To allow efficient simulation of language evolution in the jungle world, a biologically-inspired algorithm, Evolutionary Reinforcement Learning with Potentiation and Memory (ERL-POM) is proposed. This approach utilizes a genetic algorithm to configure reinforcement learners units. State-action memory and potentiation are introduced to balance exploitation with exploration and improve interactive learning ability.

Using the proposed algorithm, language evolution and acquisition is simulated under a variety of settings of the jungle world. These settings include the scenario where communication is necessary for both tasks, one of the tasks, or neither of the tasks. The paper also presents and analyzes samples of the artificial languages evolved in different settings. Experimental results and analysis show that ERL-POM is effective in simulating language evolution and acquisition, demonstrating that languages can be evolved and acquired in the artificial world if communication is necessary for one or both of the tasks.

The remaining sections of the paper are organized as follows. The next section gives a brief review on prior work in computational simulation of language evolution. The third section introduces rules and settings of the jungle world, and the fourth section provides details on the algorithm. The fifth section presents and analyzes experimental results, and the sixth section points out potential directions for future work.

Prior Work

In a typical simulation of language evolution, a multi-agent system is created to simulate an entire population of agents. Each agent acquires a shared communication system either by using machine learning methods and/or through simulated evolutionary process.

Simulations of language evolution can be divided into situated and non-situated simulations. In a non-situated simulation, an agent's actions consist solely of sending and receiving signals. Such non-embodied agents perceive objects

and events, but do not change the state of the environment. Usually, the agents aim at encoding an arbitrary meaning as a signal and send it to another agent, who decodes the signal back to a meaning. In such simulations, neural networks (Batali, 1998; Kvasnicka and Pospichal, 1999; Smith 2002), lookup tables (Kaplan 2000; Smith 2001), associative memories (De Boer and Vogt, 1999; Steels and Oudeyer, 2000) and finite state machines (MacLennan and Burghardt, 1993; Brighton 2002) are the most commonly used models to represent the behaviors. While they have been employed to demonstrate many interesting properties of communication systems, non-situated simulations are unrealistic in that they do not associate external tasks with communication actions. In contrast, the evolution of language in nature is strongly linked to the need to perform various tasks in which communication helps.

To address this problem, situated simulations can be built. In such a simulation, agents are embodied in an artificial world. Their goal is usually to accomplish tasks that require cooperation or competition among multiple agents. Thus, language serves as a necessary or beneficial tool to achieve higher performance in multi-agent tasks. Situated simulations can be used to test specific assumptions on the role of certain behaviors or environmental factors in the evolution of language (Quinn 2001; Mirolli and Parisi, 2010; De Greeff and Nolfi, 2010; Mitri, 2010; Rawal, Boughman, and Miikkulainen, 2014).

However, prior work on situated simulations is limited in two ways. First, most of them focus on a single task. More specifically, the rewards of actions, be it communicative or non-communicative, do not change throughout the lifetime of agents in all generations. In contrast, in nature, language is used for numerous tasks, and the rewards of actions depend on multiple factors. Second, the language is usually encoded genetically and passed on to the next generation through genotypes. In contrast, language in nature is acquired during lifetime learning and passed on to the next generation through interaction among individuals in the environment. While some of the existing work addresses one of the above problems, to our best knowledge, no prior work on situated simulations evolves artificial languages that are both applicable to different tasks and acquired through lifetime learning. Therefore, the purpose of this work is to introduce a simulation framework that achieves both goals. Such a framework makes the simulation more realistic and should be helpful in discovering deeper insights into the origin and evolution of language.

Simulation Environment

This section introduces the rules and settings of the jungle world – the artificial environment used in the experiments. The goal is to establish a paradigm of situated simulation environment where languages evolved are used in different tasks at different stages of an agent's life, and knowledge must be acquired through lifetime learning. While only a few variations of the jungle world are used in the experiments, the simulation environment can be modified to serve many other experimental goals. In addition, the jungle world does not impose any requirement for the artificial controller of the agents except for an interface that specifies inputs and outputs. Hence, it can be viewed as a general test environment for evaluating performances of genetic based machine learning algorithms.

Life in the Jungle World

This subsection presents basic concepts and rules in the virtual world. The focus is on actions and rewards during a single generation. Concepts and rules are organized into entries with short definitions and descriptions.

Step and Trial. Time is discretized into steps. At each step, agents receive inputs from the artificial environment including messages from their partner, and take actions accordingly. A trial is a 10-step experiment with two agents. It terminates early if any of the participants receive a positive or negative reward.

Jungle. The jungle is the place where agents hunt for prey and feed themselves. However, if an imprudent agent enters the jungle without its partner at any step, it will be hurt and receive a negative reward.

Agent. An agent has two integer states: fitness and position. Fitness ranges from 10 to 200, with 10 the initial value for new-born agents. Fitness increases by 10 after a successful hunt (defined later) and decreases by 1 after each trial. Position ranges from 0 to 5, indicating the distance between an agent and the jungle.

An agent senses its proximity to the jungle and becomes alert if its position is 1. It becomes ready for mating if its fitness is greater than 100.

At each step, an agent may decide to take the following actions: (1) move towards the jungle, (2) attempt to mate, and (3) send a two-bit message.

Thus, in the typical setting of the jungle world, an agent's brain, i.e. the controller, receives a four-bit input at each time step: position alert, mating readiness, message bit 1 and 2. Based on the input, the controller makes a four-bit decision, indicating whether the agent decides to move towards the jungle, to mate with its partner, or to set message bits to one.

Hunting. An agent succeeds in hunting if it enters the jungle with its trial partner at the same time step. A successful hunt gives a positive reward and increases fitness by 10 for both agents.

Mating. If a pair of agents decides to mate at the same step, and both of them are ready, they succeed in mating and receive a positive reward equal to 1/10 of their partner's fitness. Thus, successful mating always claims more rewards than hunting, especially so when fitness is high.

If an agent decides to mate when its fitness does not exceed 100, it receives a negative reward for cheating its partner. If an agent decides to mate while its partner is not ready, it is embarrassed and receives a negative reward, too.

Idling. If a pair of agents claim no reward at the end of a trial, they receive a negative reward for wasting time.

Population. The population in all generations contains 50 agents with 25 seniors and 25 juniors (except for the first generation in which no seniors exist). A senior is an agent who survived the selection process after the previous generation. A

junior is a newborn agent whose parents are a pair of senior agents.

Generation. A generation contains two phases: parenting and socializing (except for the first generation). In the parenting phase, junior agents pair with each of their parents for 100 trials. Since every junior has two parents, a total of 5000 trials are conducted in the parenting phase. In the socializing phase, every pair of agents in the current generation participates in 100 trials with random ordering of partners. Thus, a total of 122,500 trials are conducted in the socializing phase. Rewards accumulated from the socializing trials are used as the performance measure for all agents.

Selection. After each generation, agents are ranked based on their performance. Top 25 eligible agents survive the selection and become the senior agents in the next generation. An agent is eligible as long as its life spans fewer than four generations.

Reproduction. Before the next generation starts, selected agents must participate in the reproduction process to produce junior agents for the next generation. While the artificial world does not impose any requirement on the mechanisms in which genotypes of selected agents are used to generate new agents, the following approach is employed in the experiments.

Each of the 25 selected agents is paired with a randomly chosen partner. Junior agents are then constructed from them through mutation and crossover. Thus, every selected agent has at least one child, but may have two or more children due to random pairing. Details on genotypes, mutation, and crossover is presented in the "Method" section.

Language Acquisition Requirement (LAR). Junior agents in all generations as well as agents in the first generation must NOT have any knowledge on the rules of the world and the language that is used in it. All agents must acquire such knowledge by participating in trials. The genotype of an agent defines how it learns, rather than representing such knowledge directly.

Implications to Language Evolution

As defined in the previous subsection, the jungle world is an artificial environment for situated simulations of language evolution. This subsection gives a brief discussion on two important features of this artificial environment and points out their implications.

Multitasking. Jungle world requires artificial agents to evolve languages that are applicable to different tasks from three perspectives.

First and most obviously, there are two tasks (actions), namely hunting, i.e. moving into the jungle, and mating, i.e. attempting to mate with a partner, that need to be coordinated through communication. Each action receives negative or positive rewards under different environmental state.

Second, if the long-term reward of these actions remains constant in a changing environment, the so-called "different tasks" are actually a single task interpreted subjectively with the semantics of multiple tasks. After all, the core challenge of multitasking is that possible actions pursuing different tasks must be properly prioritized and coordinated in order to achieve higher rewards in the long run.

Indeed, agents in the jungle world are required to coordinate and prioritize their tasks throughout their lifetime via the language they evolve. For instance, agents in the socializing phase usually have a fitness score higher than 100, which gives them legitimate choices of hunting and mating. Successful mating always yields a higher reward than hunting. However, fitness decreases at the end of each trial. Therefore the agents have to hunt regularly to maintain a high fitness level. Since both actions are always available to the agents, they must learn to prioritize the actions and coordinate with partners in pursuing each task. Note that the messages sent and received are part of the environmental state, and in some variance of the jungle world, messaging is the only way in which agents can communicate state or convey intention. In this sense, the jungle world presents a true multi-task challenge.

Third, due to LAR and the fitness requirement for mating, agents in the jungle world must adapt their strategy at different stages of their life. As in nature, junior agents must first learn to interpret the environment and hunt successfully through communicating with their parents in the parenting phase. Before their fitness score can be maintained at a high level (i.e. greater than 100), mating is not a viable choice. However, as agents proceed into the socializing phase, they must adjust their strategy and learn to balance mating and hunting in order to maximize their cumulative rewards.

Language Acquisition. LAR ensures that agents have to learn to survive the jungle world during their lifetime rather than relying solely on the information encoded in their genes. LAR thus makes simulation of language evolution more realistic: in nature, language and knowledge is largely acquired through lifetime learning rather than genetically encoded. While some theories suggest that the human genotype encodes the universal grammar (Chomsky and DiNozzi 1972), it is commonly accepted that any particular language should be learned.

From a technical perspective, LAR presents interesting challenges. Many powerful methods such as neural networks are usually not directly applicable to learning in real time. Meanwhile, traditional reinforcement learning algorithms such as Q learning require accurate and full observation of environmental state. In contrast, in typical settings of the jungle world, states are only partially observable, i.e. agents cannot observe the fitness or the position of their trial partners. While communication can be leveraged to compensate partial observation, at the early stage of evolution, semantics of the messages are rather unreliable and frequently changing. Even after a language is established among the senior population, messages from junior agents in the parenting phase can mislead or confuse their parents and in the worst case, reverse the progress of language evolution in previous generations.

While reflecting important features of language in nature, both multitasking and LAR present interesting challenges to the method used in such simulations.

Method

This section presents details on Evolutionary Reinforcement Learning with Potentiation and Memory (ERL-POM) – the method adopted to allow efficient and effective simulation in

the jungle world. The reinforcement learner serves as the brain of the agents. In each generation, every individual in the population has its own reinforcement learner. Through lifetime interaction with trial partners, these controllers adjust their policy gradually to achieve higher cumulative rewards.

Controller Structure

This subsection introduces the structure of the controller.

Inputs and Outputs. The controller assumes that both the inputs (i.e. environmental states) and the outputs, (i.e. action decisions) are binary, or can be converted to binary.

Learner Unit. Similar to neurons in a neural network, learner units are basic functional units of a controller. They take binary inputs (multiple bits) and give a single bit output. Each unit consists a policy map and a memory. The policy map is implemented as a Hashmap whose keys are the input patterns and values *activation parameters* (AP).

Memory. Memory is a queue of input-output pairs with a certain size. Inspired by the short term memory in nature, old items in the queue are replaced by new items once the number of items reaches memory size. Memory provides a record on decisions given input patterns and helps the learner unit adjust APs to maximize long term reward.

Expansion. Since both inputs and outputs are binary, learner units can be connected similarly to neurons in a neural network to form various complex structures. They also adjust their weights based on memory and rewards while interacting with the environment. Thus, a controller is essentially a neural network designed for real-time learning.

However, it is worth pointing out that learner units are more powerful and complicated than neurons in typical neural networks. Therefore, it usually takes fewer learner units than neurons to solve a problem. As a matter of fact, given the typical setting of the jungle world, merely four learner units are needed to achieve high average cumulative rewards, which is similar to a single layer neural network if all learner units are replaced by neurons.

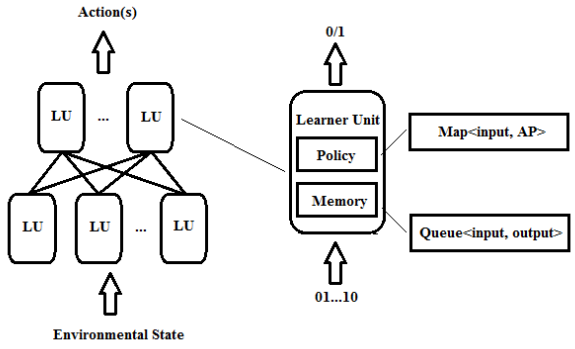

Figure 1: Controller Structure. A controller may contain one or multiple layers of learner units, connected similarly to neurons in a neural network. Each learner unit consists a policy and a memory. The policy is a Hashmap whose keys are input patterns and values activation parameters, and the memory is a queue of input-output pairs.

Real Time Learning and Evolution

Initially, all policy maps are empty. As an agent explores the artificial world, learner units receive inputs and insert a new entry to their policy map for each previously unseen input. The AP in each new entry is set to 0. The probability of activation P_A given the activation parameter AP of an input is computed according to formula 1.

$$P_A = \sigma(\mathrm{AP}) = \frac{1}{1+e^{-\mathrm{AP}}} \quad (1)$$

Here, σ is the sigmoid function. Note that for each new entry, the probability of activation is 0.5, i.e. the learner units performs random exploration.

Learning with Memory. Whenever a learner unit makes a decision given an input, the event (input-output pair) is pushed into the memory queue. After the queue is full, oldest records are replaced by new ones.

When the agent collects a reward r, each learner unit looks into its memory. For each event (i.e. input-output pair) in the queue, the learner unit updates the AP of the corresponding input in the policy map according to formula 2.

$$\mathrm{AP}' = \alpha \cdot \mathrm{AP} + r \cdot d^n(2x - 1) \quad (2)$$

Here, AP and AP' are the activation parameter before and after the update, respectively, α is the *decay rate* ($0 \le \alpha \le 1$), r is the normalized reward ($-1 \le r \le 1$), d is *discount factor* ($0 \le d \le 1$), n is event index in the memory queue, with 0 representing the most recent, and x is the binary output.

Intuitively, learner units increase the AP mapped to an input pattern (thus the activation probability given that input) if (1) a positive reward is received, and the unit outputs 1, or (2) a negative reward is received, and the unit outputs 0. The decay rate balances the influence of knowledge from the past with the most recent experience. When $\alpha = 0$, only the latest experience is taken into consideration. The discount factor reflects the contribution of decisions in the past. If $d = 0$, only the last decision is assumed to be the cause of the reward, and if $d = 1$, the reward is assumed to be equally attributable to all events in the memory.

Potentiation. Because junior partner behave randomly, it is possible that senior partners are confused during the parenting phase. Therefore, potentiation is introduced to retain long-term memory, i.e. well-tested knowledge and rules learned in the past.

As in nature, if the brain is confident enough on a decision for a certain input, that decision is fixed. Specifically, if AP of an input satisfies the condition in formula 3, the learner unit fixes the decision on that input to 1 if AP is positive, and to 0 if it is negative.

$$\frac{|\sigma(\mathrm{AP}) - 0.5|}{0.5} \ge \mathrm{PT} \quad (3)$$

Here PT ($0 \le \mathrm{PT} \le 1$) is *potentiation threshold*. Intuitively, PT specifies how confident the controller must be in order to fix its decision. If PT = 0, the controller will fix its decision after learning from a single event, and if PT = 1, the controller will never fix its decision.

Evolution. As introduced in the previous subsections, each learner unit is defined by the following parameters: (1) the potentiation threshold, (2) the discount factor, (3) the decay

rate, and (4) the memory size. Thus, a controller with m learner units and a fixed topology has a genome with $4m$ numbers, which include m positive integers with a maximum (i.e. the maximum size of the memory), and $3m$ real numbers [0..1]. Since all controllers have numeric genomes with uniformly defined structure, mutation and crossover can be directly applied to producing controllers for a new generation.

In the above mechanism, the role of evolution is to explore the learner space and optimize parameters for units in the controller. In other words, evolution aims to improve learning ability rather than encode policies. Junior agents in a generation are equipped with potentially better learning tools. Nevertheless, they have no specific knowledge of the rules of the world or the language among the senior agents. Thus, this method satisfies LAR as defined in the previous section.

Experimental Results

This section presents and discusses the experimental results of the situated simulation under four different settings of the jungle world. In the first setting, communication channels are disabled, and the environment is fully observable, i.e. agents can observe the fitness and position of their partners directly. The second settings allows full observation while enabling communication. In the third setting, communication channels are enabled, but agents cannot observe the position or fitness of their partner. The fourth setting is the same as the third, except that partner position is observable. The above four settings aim to address the following questions:

1. When communication channels are disabled and environmental states fully observable, what kind of behaviors emerge as a baseline?
2. If environmental states are fully observable, i.e. communication is enabled but unnecessary, will any language emerge?
3. When communication is necessary for all tasks, can the agents evolve a language to coordinate their actions?
4. If communication is necessary for some of the tasks but not for the others, will any language emerge?

In each of the experiments, any language that emerges is analyzed to understand what it is and how it helps agents to perform their tasks.

Experimental Setup

Table 1 shows the parameter settings used in all simulations. The normal distributions (ND) have a standard deviation of 0.1, with a mean of zero. Mutation rate is applied to each gene independently. A special rule (SR) is applied to the mutation of memory size: it increases or decreases by one with 0.5 probability for each.

After the seniors are selected from the previous generation, each senior has one chance to be paired with another senior randomly to produce a junior for the next generation. The genome of the first senior is used as the initial genome of the child. With a probability equal to crossover rate, each gene (i.e. number) has a chance to be replaced by the corresponding gene of the random spouse. In addition, it can mutate based on the mutation rate and rules in table 1. Crossover always takes place before mutation.

Item	Value/Setting	
Population Size	50	
Senior/Junior	25/25	
Mutation Rate	0.1	
Mutation Rule	Gene	Initial Value (Method)
	MS	1 (SR, $1 \le MS \le 10$)
	PT	1.0 (ND, $0 \le PT \le 1$)
	α	1.0 (ND, $0 \le \alpha \le 1$)
	d	0.0 (ND, $0 \le d \le 1$)
Crossover Rate	0.1	

Table 1: Parameters and Mutation Rules

Note that in table 1, Initial Value refers to the values of parameters of learner units in the first generation. Memory size is set to one so that agents memorize only the last event. Potentiation threshold is 1.0, thus the agents never fix their decisions. Discount factor is 0.0, i.e. reward is assumed to be attributable to only the latest action. In other words, memory and potentiation do NOT exist for the first generation. It is up to evolution to decide if they are desirable. Such a setting serves two purposes: (1) it demonstrates the benefit of memory and potentiation through evolution, and (2) it avoids unnecessary structural and algorithmic complication. However, it has a downside of potentially delaying the emergence of language in the artificial world since such a simple starting point may be far from the best settings.

Experiment 1 (Baseline)

In this experiment, the communication channels are blocked and the environment is fully observable to the agents. Thus, agents cannot send messages to their partner, but they can observe the fitness and the position of their partner directly. The results in this group serve as a reference to evaluate the performance in other experiments. Figure 2 shows the results.

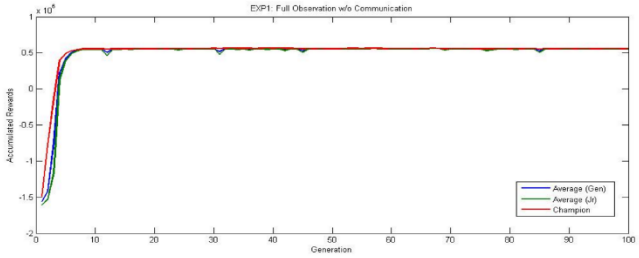

Figure 2: Experiment 1. In approximately eight generations, accumulated rewards become stabilized, and the performance does not differentiate much between juniors, seniors, and the champion. All results, including those of the other experiments are averaged from 20 runs.

Table 2 presents a sample policy of a champion in the 100th generation. According to the policy, the champion mates with its partner whenever they are both ready to mate (i.e. fitness $\ge$ 100). Until then, it moves towards the jungle as long as its position is greater than one. When getting close to the jungle, the champion waits for the partner if it is not in position, and

jumps into the jungle as soon as both of their positions equal to one.

F'	P'	F	P	**A**	**M**
0	0	0	0	1	0
0	0	0	1	0	0
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	1
1	1	1	1	1	1

Table 2: Champion Policy – Experiment 1. Regular letters are input states, and bold letters are actions. F represents fitness, i.e. whether the fitness score is greater than 100; P is position, i.e. whether position equals 1; F' and P' indicate partner's fitness and position, respectively; A is the action to approach the jungle; and M is the action to mate.

Lastly, memory size after ten generations averages 1.28, with a majority of learner units having no memory beyond the last decision. This result can be explained by the fact that in a fully observable environment, with a majority of senior agent policies like that in table 2, the challenge faced by a junior agent is largely a Markovian problem. On the other hand, the average potentiation threshold is 0.965 – the agents do fix their actions, but only when they are very confident in their decisions.

Experiment 2 (Unnecessary Communication)

Experiment 2 is different from the Experiment 1 in that agents are allowed to send and receive messages. Since the position and fitness of their partner are still observable, communication is possible but unnecessary in achieving any of the tasks.

Figure 3 presents the results of Experiment 2. Memory size after ten generations averages 1.42, with average potentiation thresholds as 0.971.

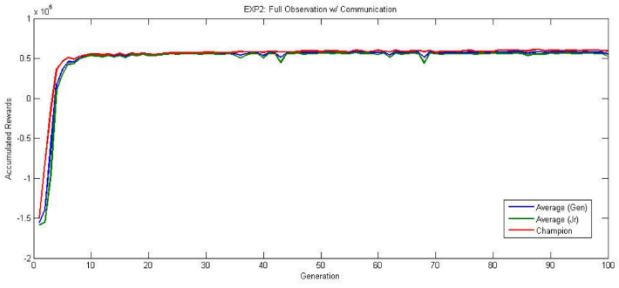

Figure 3: Experiment 2. Average cumulative rewards of each generation are nearly identical to those in Experiment 1.

The champion policies in Experiment 2 are characterized by the following three observations:

1. Given same observation of fitness and positions, actions are the same regardless of the message received.
2. Messaging policies vary from generation to generation, while having no influence on the stability of performance.
3. There is no clear correlation between messages and environmental states in most champion policies.

The results in Experiment 2 suggest that if communication is unnecessary for any of the tasks, messaging policy plays no role in agent performance. Agents evolve no language even when the communication channels are available.

Experiment 3 (Necessary Communication)

In this experiment, agents cannot observe the position and fitness of their trial partners. Therefore, the only means by which agents can coordinate their efforts in mating and hunting is through communication.

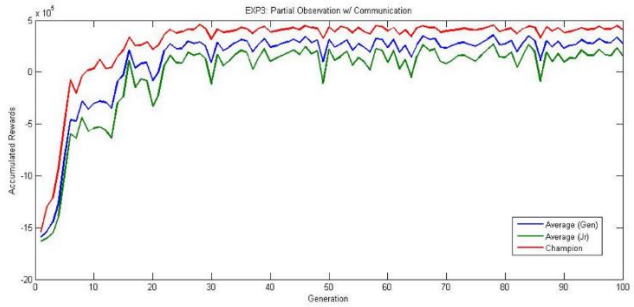

Figure 4: Experiment 3. Champions in the agent population achieve similar performance to that of Experiment 1 after approximately 25 generations. Champion performance (red) stabilizes afterwards. However, compared to Experiment 1, average cumulated rewards among entire generations (blue) and among the juniors (green) are lower and have bigger gaps in between.

The reasons for lower average performance is that multiple languages may occur simultaneously in one generation, causing confusion among the juniors in parenting phase, thus lower their performance in the socializing phase.

Table 3 presents a sample policy of a champion in the last generation. Champion policies after thirty generations encode the fitness and position accurately and consistently in 19 out of the 20 runs. However, in almost all generations, more than 20% of the seniors have a messaging policy that either fails to encode fitness or position states accurately, or differs from the champion policy.

Also, average memory size after 30 generations is 7.10, and the average discount factor is 0.722. The reason for the long memory is that past decisions can be used to complement partial observation and improve learning efficiency. For instance, if an agent keeps sending wrong messages while being in position for hunt, its trial partner can never know that the agent is ready, resulting in a punishment for idling to both

agents in the trial. While potentiation may keep the seniors from "second guessing" their correct policy, a long memory of the past can help the juniors learn faster in such cases.

F	P	R2	R1	**A**	**M**	**S2**	**S1**
0	0	0	0	1	0	1	1
0	0	0	1	1	0		
0	0	1	0	1	0		
0	0	1	1	1	0		
0	1	0	0	0	0	1	0
0	1	0	1	1	0		
0	1	1	0	1	0		
0	1	1	1	0	0		
1	0	0	0	1	1	0	0
1	0	0	1	1	1		
1	0	1	0	1	0		
1	0	1	1	1	0		
1	1	0	0	0	1	0	1
1	1	0	1	1	0		
1	1	1	0	1	0		
1	1	1	1	0	0		

Table 3: Champion Policy – Experiment 3. S1 and S2 represent the first and second message bit; R1 and R2 are the message bits received from the partner. All other letters have the same meaning as those in Table 2. The messages encode fitness and position values.

Although the average potentiation threshold is close to one (0.981), potentiation is crucial in the simulation because the parenting phase is sufficiently long to generate enough confusing interactions to reverse knowledge encoded in the seniors' controllers. In fact, if potentiation threshold is fixed to one (i.e. potentiation does not exist), language simply cannot be established among the agent population, rendering cumulative rewards consistently negative in all generations.

Experiment 4 (Partially Necessary Communication)

In this experiment, the position of a trial partner is observable while the fitness of the partner is hidden. Communication is possible between partners in a trial, however, it is necessary only for mating.

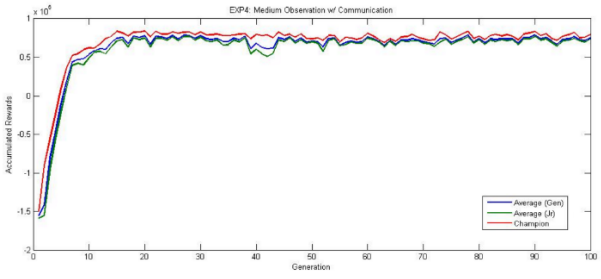

Figure 5: Experiment 4. Average performance in the first 10 generations is similar to that in Experiment 1. After 15 to 20 generations, rewards stabilize at a level approximately 50% higher than that in Experiment 1.

Typical messaging policies in the first ten generations of each run encode fitness in two bits (e.g. "11" for "ready to mate" and "01" otherwise). As evolution proceeds, more advanced policies may emerge, leading to better performance than the full observation baseline. Table 4 shows a messaging policy from a champion in the 100th generation.

F	P	P'	R2	R1	**S2**	**S1**
0	0	-	-	-	1	0
0	1	-	-	-	1	0
1	0	-	-	-	0	0
1	1	-	-	-	1	1

Table 4: Champion Messaging Policy – Experiment 4. Dash indicates that an input has no effect on the outputs.

While the messaging policy in table 4 looks confusing by itself, interestingly, it exploits the setting of the jungle world effectively if combined with corresponding action policy. Although such a policy can be expressed by a table as before, it is translated into the following rule-based policy due to limited space.

1. Run towards the jungle until position is one.
2. Wait until partner position becomes one.
3. Enter the jungle if positions of both agents are one.
4. If "11" is received before entering the jungle, mate and enter the jungle at the same time step.

Note that Rule 1 to 3 are based purely on observation, i.e. they have nothing to do with the messages received because partner position is directly observable. In fact, the only rule that relies on communication is Rule 4. It can be triggered only under the circumstance where both agents are one step away from the jungle and "11" is received; and according to Table 4, "11" is sent whenever an agent is ready for both hunting and mating. Since messages are ignored in all other scenarios, the only messaging rule that a junior agent needs to learn is to send "11" when F and P are both one. This rule has two positive effects. First, the juniors learn the language faster because it tolerates faults on all but a few inputs (i.e. inputs with F and P equal to one). Second, combined with the action policy, it allows agents to mate and hunt successfully in a single trial, making the average fitness of the population close to the maximum (200) all the time.

Among the 20 runs in Experiment 4, approximately half of them (11/20) discovered policies with similar principles, thus achieving higher performance than Experiment 1 and 2. Note that without the emergence of meaningful language, agents in Experiment 1 or 2 cannot discover behaviors that accomplish hunt and mating in a single trial. The fact that languages can be evolved to get around deceptive local optima (e.g. policies that decide to mate whenever both agents are ready) is intriguing.

Additionally, results in Experiment 2 and 4 suggest that for language to emerge, it is essential that language is indeed necessary to perform some of the tasks. However, as long as language appears, it can be evolved into a beneficial tool for all tasks.

Future Work

Situated simulation of language evolution provides interesting insights on the origin and evolution of language. The jungle world simulation can be used as a starting point for more advanced simulations in two ways.

First, in nature, spoken language is formed with sequential patterns of utterances. Messages may span multiple time steps rather than contained in a single step. Also, they may start at any step. Such sequential features are essential for simulating the evolution of more complex and structured languages.

Second, languages in the real world are usually structured based on syntax. The emergence of grammatical components and structures such as nouns and verbs, subjects and objects, phrases and sentences is an important aspect of language evolution. A possible approach in the jungle world is to establish social roles in the simulation, and create tasks around them. Grammatical structure might then emerge in order to communicate such role-based information (Bickerton 1990).

Integrating sequential and/or structural features into the jungle world framework will make simulations more realistic and informative.

Conclusion

This paper presents a framework for situated simulation of language evolution. It introduces an artificial environment, the jungle world, which can be used to simulate the evolution and acquisition of multitask languages. The paper also proposes a method: Evolutionary Reinforcement Learning with Potentiation and Memory (ERL-POM) for simulation of language evolution in this environment.

Experimental results indicate that languages can be evolved in the artificial environment if communication is necessary for some or all of the tasks. Languages can be used to coordinate efforts in multiple tasks where communication is required. When communication is not necessary for all tasks, languages can be leveraged to overcome local optima and discover better policies. Experimental results also show that memory and potentiation are necessary for such emergence. Extending the simulation to sequential and structured communication is a most interesting direction of future work.

Acknowledgement

This research was supported in part by NSF grants DBI-0939454 and IIS-0915038, and in part by NIH grant R01-GM105042.

Reference

Batali, J. (1998), Computational simulations of the emergence of grammar. In J. R. Hurford, M. Studdert-Kennedy, & C. Knight (Eds.), *Approaches to the evolution of language*, page 405–426. Cambridge, UK: Cambridge University Press.

Bickerton, D. (1990). *Species and Language*. Univ. of Chicago Press, Chicago.

Brighton, H. (2002). Compositional syntax from cultural transmission. *Artificial Life*, 8, page 25–54.

Cangelosi, A., Parisi, D. (2002), *Simulating the evolution of language*, 2002 edition, page 5 – 8, Springer.

Chomsky, N. (1972). *Language and mind*. New York: Harcourt Brace Jovanovich.

De Boer, B., & Vogt, P. (1999). Emergence of speech sounds in changing populations. *In Advances in artificial life*, page 664-673. Springer Berlin Heidelberg.

De Greeff, J., & Nolfi, S. (2010). Evolution of implicit and explicit communication in mobile robots. In *Evolution of Communication and Language in Embodied Agents* (pp. 179-214). Springer Berlin Heidelberg.

Kaplan, F. (2000). Semiotic schemata: Selection units for linguistic cultural evolution. In M. Bedau, J. McCaskill, N. Packard, & S. Rasmussen (Eds.), Artificial Life VII: *Proceedings of the Seventh Artificial Life Conference*, page 372–381. Cambridge, MA: MIT Press.

Kvasnicka, V. and Pospichal, J. (1999), An Emergence of Coordinated Communication in Populations of Agents. *Artificial Life* 5(4), page 319--342.

MacLennan, B., Burghardt, G. (1993), Synthetic ethology and evolution of cooperative communication, *Adaptive Behavior*, Vol 2, page 167-187.

Mirolli, M., Parisi, D. (2010), Producer biases and kin selection in the evolution of communication: how the phylogenetic and the adaptive problems of communication can be solved. In *Evolution of communication and language in embodied agents*, page 135–159. Springer Verlag.

Mitri, S., Floreano, D., & Keller, L. (2010). Evolutionary conditions for the emergence of communication. In *Evolution of Communication and Language in Embodied Agents*, page 123-134. Springer Berlin Heidelberg.

Nolfi, S., Mirolli, M. (2010), *Evolution of communication and language in embodied agents*. Berlin: Springer.

Quinn, M. (2001). Evolving communication without dedicated communication channels. In J. Kelemen & P. Sosí k (Eds.), *Advances in artificial life: The Sixth European Conference* (ECAL 2001), page 357–366. Berlin: Springer.

Rawal, A., Boughman, J., Miikkulainen, R. (2014), Evolution of communication in mate selection. In *Proceedings of the fourteenth international conference on the synthesis and simulation of living systems* (ALIFE 14), Cambridge, MA: MIT Press.

Smith, A. D. M. (2001). Establishing communication systems without explicit meaning transmission. In J. Kelemen & P. Sosí k (Eds.), *Advances in artificial life: The Sixth European Conference* (ECAL 2001), page 381–390. Berlin: Springer.

Smith, K. (2002). The cultural evolution of communication in a population of neural networks. *Connection Science*, 14(1), page 65–84.

Steels, L., & Oudeyer, P.-Y. (2000). The cultural evolution of syntactic constraints in phonology. In M. Bedau, J. McCaskill, N. Packard, & S. Rasmussen (Eds.), *Artificial life VII: Proceedings of the Seventh Artificial Life Conference*, page. 382–391. Cambridge, MA: MIT Press.

Wagner, K., Reggia, J., Uriagereka, J., Wilkinson, G. S. (2003), Progress in the simulation of emergent communication and language, *Adaptive Behavior*, Vol 11, page 37-69.