A competence-performance based model to develop a syntactic language for artificial agents

Jack Mario Mingo[a,*], Ricardo Aler[b]

[a]*Department of Computer Science, Universidad Autónoma de Madrid*
[b]*Department of Computer Science, Universidad Carlos III de Madrid*

Abstract

The hypothesis of language use is an attractive theory in order to explain how natural languages evolve and develop in social populations. In this paper we present a model partially based on the idea of language games, so that a group of artificial agents are able to produce and share a symbolic language with syntactic structure. Grammatical structure is induced by grammatical evolution of stochastic regular grammars with learning capabilities, while language development is refined by means of language games where the agents apply on-line probabilistic reinforcement learning. Within this framework, the model uses the concepts of competence and performance in language, as they have been proposed in some linguistic theories. The first experiments in this article have been organized around the linguistic description of visual scenes with the possibility of changing the referential situations. A second and more complicated experimental setting is also analyzed, where linguistic descriptions are enforced to keep word order constraints.

Keywords: Stochastic Grammars, Grammatical Evolution, Reinforcement Learning, Dynamics of Artificial Languages, Language Games, Multi-Agents Systems, Competence and Performance in Language.

---

*Corresponding author
Email addresses: `mario.mingo@uam.es` (Jack Mario Mingo), `aler@inf.uc3m.es` (Ricardo Aler)

1. Introduction

Language is one of the fundamental cognitive skills needed for the development of advanced multi-agent or multi-robot systems, as it allows communication and cooperation among individuals in the group. This necessity has recently motivated research where computational models are applied for developing artificial languages with increasing complexity. Computational simulations, and even experiments with real robots, have also been used for explaining properties of natural languages [1, 2]. Nevertheless, the aim of the present article is not to use artificial system to aid in the understanding of natural languages, but rather to propose an evolutionary and learning process so that a team of agents can construct a shared artificial syntactical language.

The initial approaches for communication within teams of agents have focused on creating a basic vocabulary or lexicon that is shared by all members of the group. This vocabulary is generally a simple mapping between symbols and meanings. Although this problem seems to be apparently simple, it is not trivial because of the problem known as *Symbol Grounding Problem*: how to effectively connect a symbol with its meaning (see [3] for definition and details). Humans do not seem to be limited by this issue, but artificial organisms have serious difficulties. While some works address this problem [4, 5, 6, 7, 8, 9, 10], other proposals bypass it and focus on the higher level, while the aspects related to the physical association between symbols and meanings are assumed to have been solved or are not considered [11, 12, 13, 14].

Although a shared vocabulary is essential and it has to be the first step, syntactic competence is vital for an agent to efficiently describe reality in a symbolic way. In this work, we approach the syntactic alignment of agents in a team by means of language games applying on-line reinforcement learning algorithms and grammatical evolution to allow the agents to evolve and adapt their language to the current linguistic situation. *syntactic alignment* refers here to the process that allows a team of agents to develop a common syntactic language, by pairwise interactions or language games, without being told by a centralized source.

The concept of *language games* is partly inspired by the ideas of Wittgenstein [15] and De Saussure [16] about the public and conventional dimensions of linguistic meaning and it has been applied in most of Steel's works (see [17] for a recent review) and related authors such as [18, 19]. The importance of syntactic competence is currently so relevant that other approaches based on different ideas have been proposed in [20, 21, 22, 23, 24, 25, 26, 27, 28, 29].

Most of the cited works, except [8, 9, 14], are focused on explaining properties of natural languages by means of computational simulations as we already mentioned. On the other hand, the above three exceptions are more oriented to practical applications (and our work follows the same line). However, they do not address syntax in depth. A recent work proposing a more complicated language with syntactic properties can be found in [30]. In Maravall et all's model, a team of agents reach syntactic alignment by learning the probabilities associated to the production rules in a stochastic regular grammar. Results show how different configurations of teams agree on a shared language with syntactic structure. It also compares when a human teacher is included in the group or when the agents are on their own. The Maravall et all's model is effective for team sizes ranging from 5 to 60 and it shows how a grammar is adjusted by tuning the probabilities of its production rules. However, the production rules themselves do not change, and they are directly encoded in the agents' controller from the beginning. As the grammar is defined in advance, it cannot adapt to changing linguist situations and it cannot explain how the agents could evolve this kind of grammatical knowledge. In this work, we follow this approach based on reinforcement learning and language games, but we study how the agents can evolve the grammar itself. Evolving the grammar allows the agent to look for the grammar that best describes the current linguistic situation which is a very important skill if the agents need to change their linguistics descriptions and "talk" about different contexts. To solve this problem we propose a model that takes into account the concepts of *Competence* and *Performance* as they were initially defined by Chomsky [31]. Competence is the knowledge that a speaker has about his/her language and performance stands for the specific use of the language in a specific context. Competence is developed by means of an evolutionary algorithm in our model and it is essentially a searching process in the space of possible grammars that describe the current linguistic situation. On the other hand, performance is achieved by means of a reinforcement learning mechanism, which rewards the most used structures.

The rest of the paper is organized as follows. Section two describes two tasks used in the computational experiments. Section three studies the process of producing and evolving the symbolic artificial language by means of grammatical evolution. Section four analyses how the language can be shared by means of on-line reinforcement learning algorithms. Section five summarizes the whole model. Section six enhances the initial model to keep syntactic order constraints. In sections seven both setting and the experimental results for the tasks are presented and discussed. A section about conclusions closes the work.

2. Describing Dynamic Linguistic Situations in Two Different Tasks

The first task to be solved by the agents in the team consists of evolving a symbolic language to describe two types of linguistic situations. The agents are situated in an environment that they can perceive with their specialized sensors. As in this paper we are mainly interested in the syntactic alignment of the agent team, we assume that agents are able to segment the images they watch and they are able to build intermediate or internal representations of them. Figures 1 and 2 show the type of visual scenes that the agents could watch.

Figure 1: Object Properties: blue book, yellow pencil and green ball

Figures 1 and 2 represent two linguistic situations which are made of several scenes. Figure 2 about spatial relations is essentially the same that it was

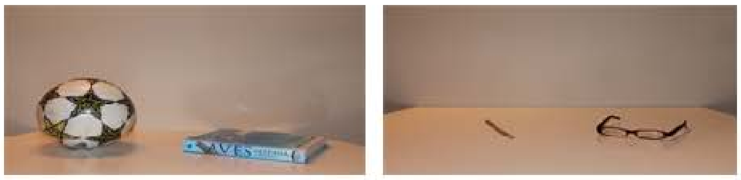

Figure 2: Spatial Relations: book on the right of the ball and pencil on the left of the glasses

used in [30] and we use it here in order to show how the proposed model can evolve and adapt itself to new linguistic situations without a predefined grammar and without human intervention such as the cited work did. In the easier context (Figure 1) the agents must agree on a language about simple objects and their colors while a slightly more complicated situation is set in the second case (Figure 2) where the agents must agree on a language about simple objects and their spatial relations. To simplify, we suppose that all agents watch the referential or shared attention scene from the same point of view. Perspective is very important in a spatial language and by adopting the same point of view, we follow what Levinson calls a relative frame of reference [32], that is, a coordinate system that "use the viewer's own bodily coordinates". This question is specially important in the case of spatial relationship such as Figure 2 shows. As another precondition, alignment language games should be used so that the agent team shares a vocabulary for the objects that can be located in the environment (*book, pencil, glasses, ball*) as well as a common lexicon for the color properties (*blue, yellow, green*) and the possible spatial relationships (*right, left*). This previous process is needed in order to efficiently map symbols and meanings for the basic vocabularies. Nevertheless, in this paper we do not describe the concrete processes behind these lexical alignments as our aim is to focus on syntactic alignment. This way, we assume that agents have names or words to refer to any item in the linguistic situation but they also need to compose complex (syntactic) sentences to express the situation in the most effective way.

For the type of situation defined in Figure 1, the agents' language is formed by sentences like "*the book* is *blue*", which can be formalized as strings representing syntactic structures such as Pa or aP where **P** is a color property and **a** is an object. In a similar way, for the type of situation defined in Figure 2, sentences would be like "the *book* is on the *right* of the *ball*", which can be formalized as strings representing syntactic structures aRb, Rab or abR where **a** and **b** are object names and **R** is a spatial relationship. As we are interested in using a language formalism that allows the agents to learn their language by means of an interactive process, we use stochastic grammars with learning capabilities because this kind of device has shown its effectiveness in a similar context [30]. However, as the linguistic situations are dynamic, we do not design in advance the set of grammatical rules and we propose a framework where the grammars themselves are evolved in order to adapt to the current linguistic situation. Both grammatical evolution and reinforcement mechanism are defined in subsequent sections.

Once the first task has been solved, we move the focus toward a more complex linguistic situation where the agents must agree upon scene descriptions using sentences like "the *blue book* is on the *right* of the *green ball*". These types of sentences combine previous syntactic knowledge. Specifically, each agent in the team has to apply the same word order in the first object's description and in the second one, such as natural languages constrain word order. Therefore, the strings representing syntactic structures in this new task will take the form of $PaRQb$, $aPRbQ$, $RPaQb$, $RaPbQ$, $PaQbR$ or $aPbQR$, where **a** and **b** are object names, **R** is a spatial relationship and **P** and **Q** are color properties. An advantage of syntactic constraints is a significant reduction in the number of possible combinations of words, as we can see in the previous example, but the drawback is that a more complicated evolutionary process will be required.

3. Evolving Grammatical Structures

As explained in the Introduction section, the model that we propose is based on two strategies: *evolution* and *learning*. Evolution allows the agents to produce, evolve and adapt their grammatical structures (*competence*) while learning drives the process of achieving linguistic consensus (*performance*). In this section we will define in depth the evolutionary process as a competence developing process.

Evolutionary algorithms can be an effective way to study artificial language emergent phenomena because they allow us to look for different syntactic structures as long as we can represent these structures in some way. *Grammatical Evolution by Grammatical Evolution* ($(\mathrm{GE})^2$) [33] is an evolutionary algorithm that can be used for this purpose. This algorithm is an extension of the *Grammatical Evolution* (GE) algorithm [34], but applied to the evolution of the grammar itself. $(\mathrm{GE})^2$ will be explained briefly in the next section in the context of this work.

3.1. Grammatical Evolution (GE) and Grammatical Evolution by Grammatical Evolution ($(GE)^2$)

GE was proposed by Collins, Ryan and O'Neill [34] and it is an evolutionary algorithm which uses variable-length linear genomes. Its initial application was to evolve computer programs represented in any language (defined by a grammar). But its application field is more general, and it can be used to evolve whatever structure that can be described by means of a grammar. In fact, GE's principles can be applied to evolve the grammar itself and this point of view was adopted in a new algorithm known as $(\mathrm{GE})^2$ [33]. In order to evolve the grammar, $(\mathrm{GE})^2$ uses a meta-grammar which specifies the structure that candidate grammars can adopt. By evolving the grammar the algorithm can fit better to dynamic environments and this feature is specially interesting in the changing linguistic situations we treat here. In summary, in $(\mathrm{GE})^2$ there are two different types of grammars:

- The *meta-grammar* describes the rules to build grammars. This grammar can be seen as playing the role of *Universal Grammar* proposed by some linguists.
- The *solution grammar* describes the rules to build solutions (sentences,

in our case). This grammar would represent the grammar which is traditionally used in a standard GE.

This way, $(GE)^2$ tries to solve a prolem by first generating a solution grammar from the meta-grammar, which in turn is used to generate the final solution. From this point of view the algorithm applies a double evolutionary process (see [33] to study the original algorithm in depth).

$(GE)^2$ is the evolutionary algorithm adopted in this work because it allows us to analyze how a suitable grammar can be developed in an initial stage, which can be finally used subsequently in a social context. In this case, the meta-grammar defines the rules to build stochastic regular grammars which in turn will allow the model to build the sentences in the language. However, sentences are generated from those solution grammars by means of a reinforcement learning process , instead of applying the second evolutionary process as the original $(GE)^2$ actually does (this will be explained later). Meta-grammars allow the model to evolve different solution grammars in a self-organized process that would not be possible if a fixed solution grammar was previously designed by a human designer. Besides, they can adapt to changes in the linguistic situations.

3.2. How to Generate Syntactic Structures

In order to generate a solution grammar from a meta-grammar (see production rules in Appendix A) the transcription and translation processes are applied as it is usual in a standard GE, [34]. In GE there is a clear distinction between the genotype and the phenotype corresponding to the individual so it needs a mapping process to transform the genotype into the phenotype. This is precisely what the transcription and translation processes do. Both processes are summarized below:

1. *Transcription.* In this stage, the original binary string that represents the individual's genotype is transformed into an integer string. We will show this process through an example. Consider a binary string which

is composed of *two-bits codons*: 01-00-11-10-00-01-01-10-00-11. After applying the transcription process this binary string is transformed into the equivalent integer string: 1-0-3-2-0-1-1-2-0-3.

2. *Translation.* Starting from the integer string and the meta-grammar's rules, a solution grammar can be built. In the previous example, the derivation tree associated to this process is showed in Figure 3. Grey nodes represent terminal symbols. Numbers displayed near the nodes represent the rule number to apply to the non-terminal symbol (white nodes). These numbers are taken from the integer string and they appear from left to right as the integer string is traversed. In order to determine which rule is going to be applied the following equation is used: $Rule = CIV\%MNR$; where *CIV* stands for *codon integer value* and *MNR* stands for the *maximum number of rules* corresponding to the non-terminal that it must be expanded at this time. Meta-grammar's rules are defined in the Appendix A.

The translation process is correct if it is possible to build a string composed exclusively of terminal symbols. As we can see in Figure 3, the terminal-only string is in its turn a solution grammar. In the example, the generated solution grammar would include the following rules (in BFN notation):

```
<sentence> ::= <object> <objectprop>
<object> ::= anObject
<objectprop> ::= aColor
```

This solution grammar defines a grammatical structure which can generate sentences with an object and a color. It is appropriate to describe a linguistic situation as the one we showed in Figure 1. It is worth noting that solution grammars are generic, in the sense that terms such as *anObject* are used, instead of any specific object. Later, an specialized module in the agent will replace these generic terms (*anObject*, *aColor*, ...) by the specific objects or colors related to the current visual scene. Replacement is applied during the language games as we will explain later. To some extent, the generic terms

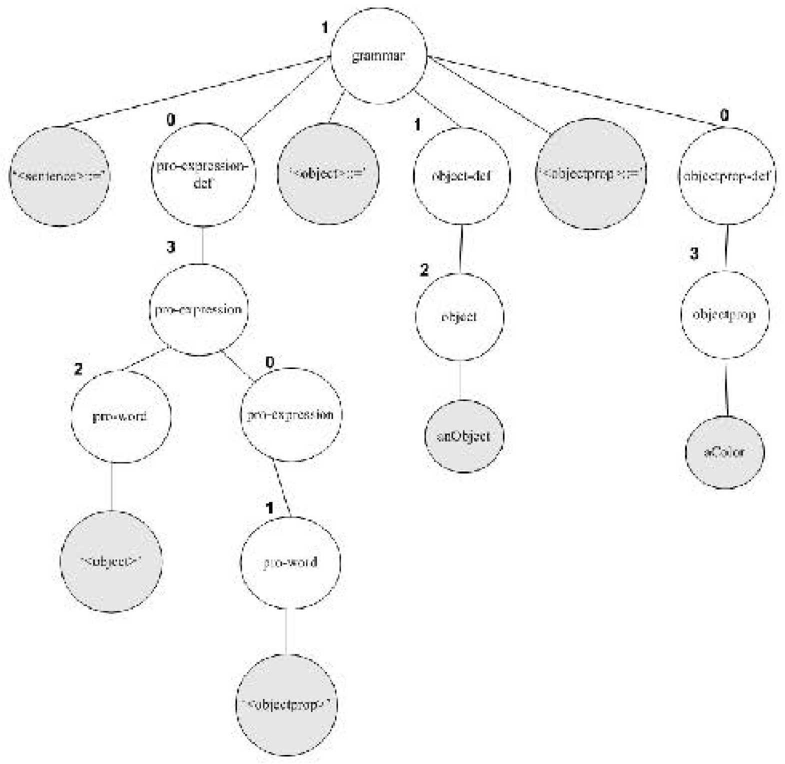

Figure 3: Example: derivation tree to generate a solution grammar (translation)

for the terminal symbols *anObject* and *aColor* are equivalent to lexical categories, widely used in linguistics. On the other hand, the non-terminal symbols <*sentence*>, <*object*> and <*objectprop*> represent grammatical categories, a concept commonly applied the literature about phrase structure grammars as well. This way, by using grammars as formalism to define languages we can easily include some important properties about natural languages such as hierarchy, recursion or compositionality.

3.3. The Evolutionary Process

The evolutionary process we propose here is simply the application of the (GE)^2 algorithm with the particularities that we described in the previous section. Another important difference is that we do not apply the second evolutionary process to the solution grammars that have been generated from the meta-grammar as in the original algorithm. Instead, we will apply a reinforcement learning algorithm to generate the final sentences as we will show later. The reason for using reinforcement learning is that we consider syntactic alignment as a social event where agents interact with each other by means of rewarded language games. The remaining items are implemented as it is usual in evolutionary algorithms, so a population of individuals is initially created and the individuals are evaluated for generations until an optimal solution is found (success) or a maximum number of generations is reached (failure). To evaluate an individual, a mapping process from its genotype (binary string) to its phenotype (solution grammar) is needed. Transcription and translation make this mapping possible. After each generation, a group of genetic operators such as crossover, mutation and duplication are applied to the individuals in the population with specific probabilities as we will show in the results section. This way, new populations are created and subsequently evaluated. Regarding the individuals' evaluation we define a fitness measure which is based on the generated solution grammar. This capacity is evaluated according to the concept of *current communicative intention*.

The *communicative intention* measures how well a solution grammar could generate sentences which are applicable to the current linguistic situation. In the previous example, the solution grammar could generate sentences with *an object* and *a color*, so this grammar could be useful if the agents are trying to agree a scene such as that in Figure 1. In order to compute a numeric value as fitness, we adapt some concepts that were previously defined in [35] and subsequently applied to the GE's algorithm in [36]. In both works, grammars are measured by means of a matrix representation of the grammar. Matrix representation helps to determine which strings are reachable starting from the start symbol of the grammar. In this work we define a matrix representation that we call *Communicative Intention Matrix*. This matrix is computed for each valid generated solution grammar since invalid generated solution grammars have no matrix representation and its fitness is directly zero (worst fitness) which ensures that they will be eventually discarded along the evolutionary process.

The *Communicative Intention Matrix* has as many rows as production rules and as many columns as terminal symbols plus an additional column which represents the length of the sentence that can be generated applying the production rule corresponding to the row. If we apply this representation to the solution grammar in the previous example we get the matrix that Table 1 shows.

Table 1: Example: Communicative Intention Matrix

Grammatical Category	Sentence Length	aColor	anObject
<object_0 >	0.0	0.0	0.0
<objectprop_0 >	0.0	0.0	0.0
<sentence_0 >	2.0	1.0	1.0

Numeric suffixes in the production rules stand for the non-terminal's production rule number. In the example there is a single rule associated to each non terminal symbol. The *aColor* and *anObject* columns are the terminal symbols in the example's solution grammar. Numeric values under these columns show the number of corresponding items in a sentence generated from the non-terminal symbol. This way, as *"sentence"* is the start symbol in all solution grammars (see meta-grammar's rules in Appendix A), a sentence starting from *"sentence"* is made of a *aColor* and a *anObject*. This is why we can see a numeric value of 1 under those columns. The length of this sentence is 2 and it stands for the number of words. Non-terminal symbols *<object_0 >* and *<objectprop_0 >* are not useful because any sentence can start from these non terminal symbols, so they have a zero as default value.

As we can see in the example, the *Communicative Intention Matrix* defines the solution grammar's skill in order to describe the communicative intention related to the current linguistic situation. In this case, if agents are trying to share a symbolic language about object colors this solution grammar would include at least a suitable production rule. The fitness value is finally computed by using Eq. 1:

$$Fitness = SPR \tag{1}$$

Where *SPR* is the number of suitable production rules according with the current linguistic situation. Ideally, this value would be the maximum number of rules which allow to combine words in every order. In the simple task about object colors there are only two possible combinations of words as we described when the task was defined. They are "Pa" or "aP" with **a** referring to an object and **P** stands for a color property. Therefore, the maximum fitness for this task is 2. It is worth noting to remember that we are studying how an artificial language evolves and is shared by a team of agents, so we cannot constraint more the language by limiting the number of possible expressions. This is why we define a fitness value with the possibility of generating whatever combination, although we can limit the fitness to smaller values. On the other hand, the fitness value is also dynamic because it depends on the current linguistic situation and the linguistic context can be changed at any time. It can be argued that by using a maximum fitness value, the evolutionary algorithm probably could be replaced for an algorithm such as breath-first or depth-first search, but in this case we have into account the communicative intention and this knowledge is very useful in order to guide the search process and avoid to explore useless alternatives in the space of possible solution grammars. Therefore, the evolutionary algorithm is not so blind and it looks for grammars that can potentially derive expressions with the highest communicative power. As we commented above the intention is represented by means of the communicative intention matrix.

4. Stochastic Learning-Grammars

In order to allow the agents to develop a suitable language for the description of the type of visual scenes displayed in Figures 1 and 2, which are composed of objects, colors or spatial relations, we propose to use an stochastic version of the solution grammars that we described in previous sections. In a stochastic solution grammar the probabilities of the production rules can be learnt by the agents through reinforcement as they engage in language games. The idea of learning the probabilities associated to the production rules in a stochastic regular grammar has been recently presented in [30]. However, in the cited work the grammar is defined in advance and it is therefore static, so it cannot adapt to different linguistic situations. In the model that we propose here grammars are not hand-designed but evolved by the evolutionary process we described above.

A stochastic solution grammar includes additionally a probability for each production rule (i.e. the probability that the production rule will be used). To simplify, we will show a simple example in a two-agent team. After the evolutionary process and the stochastic transformation, we suppose that an agent called *Agent 1* has a stochastic solution grammar such as:

```
<sentence> ::= <object> <objectprop> p0
        | <objectprop> <object>        p1
<object> ::= anObject
<objectprop> ::= aColor
```

On the other hand, agent *Agent 2* produces the following stochastic solution grammar:

```
<sentence> ::= <object> <object>      p0
        | <object>                     p1
        | <objectprop> <object>        p2
        | <object> <objectprop>        p3
<object> ::= anObject
<objectprop> ::= aColor
```

Both agents have suitable grammars if we suppose that they want to share a language for describing the linguistic situation that Figure 1 proposes. As the problem in the example is simple, the grammars are very similar. In fact, grammatical similarity is biased by the fitness definition that we adopt, and it is also a consequence of having a common meta-grammar or universal grammar, but this is the hypothesis that is defended by nativists in linguistics. The larger the number of suitable production rules there are in the grammar, the more likely is that many of them will be similar. Contrariwise, if the number of suitable production rules is only one, it will be more difficult for the reinforcement learning process to succeed because grammars will be less similar. In natural languages, it seem obvious that the more similar are the grammatical system of two speakers, the more likely for speakers to understand each other. Nevertheless, in the example, *Agent 2* has also evolved some irrelevant rules for the task about object's properties, such as the first two rules for <*sentence*>. The probabilities of the rewriting rules p_0, p_1, p_2 and p_3 (or p_n in general) are initialized arbitrarily but reinforcement learning modifies them and eventually allows the agents to converge to a common set of rewriting rules. To this end, we apply the so called Linear Reward-Inaction algorithm, L_{RI} [37] similarly to [30].

4.1. The Syntactic Language Games

For a team of N agents, each of them with its own stochastic grammar (which was previously evolved), the way to reach a syntactic alignment by means of reinforcement learning is based on language games. Specifically, a sequence of language games is performed until the team converges to an optimal communication system where all the agents use similar stochastic grammars. In each language game round, all the possible communicative acts are performed among the agents and each agent utters its own sentence according to its evolved private grammar. A reward or penalty signal is sent to the learning algorithm for the updating of the probabilities of the production rules. As the private grammar only defines the generic syntactic structures, the agent replaces the lexical categories such as *aColor* and *anObject* for the specific terms related to the current visual scene presented to the agent. We suppose that all the agents are endowed with the specialized module to make these changes. In this work, we use the *Communicative Efficiency (CE)* term (see Eq. 2) as it was defined in [30]:

$$CE(k) = (NSD/ND)100 \tag{2}$$

where NSD stands for *number of successful dialogs* and ND stands for *number of dialogs*. Communicative Efficiency helps to decide whether the syntactic alignment was possible or not.

Unlike the evolutionary process, the language games development process is a social event where all the agents interact with each other in order to agree on a shared language. We think about this process as *performance* because it refers to the actual use of the language, while the evolution is a *competence* because it refers to the grammatical knowledge development. Of course, both terms are simplifications and adaptations of the equivalent concepts about natural languages and they are not completely synonyms. For example, competence in the model is developed as a private event within each agent, while babies usually interact with adults even though they cannot talk. In any case, let us remember that our purpose is not to model exactly human languages, but to propose a model so that a team of artificial agents agree on a syntactical language. Although the evolution of grammar is a private event, all the agents participate in a shared scene where all of them watch the same situation. Besides, each agent evolves its own grammar but it has into account its own communicative intentions, so we need to suppose that agents share similar intentions. We do not have room to explain the role of intentions in artificial or natural organisms but we consider that it is a key factor in communication just as Tomasello ([38]) (among others) note.

5. The Whole Model

Once the evolutionary and reinforcement processes have been explained with detail, we consider useful to describe the whole model in order to provide a complete view of our work. Essentially, the model performs in two stages. First, an evolutionary algorithm is executed. In this evolutionary process each agent in the team tries to develop a private suitable solution grammar. A suitable grammar is a grammar which can express the agent's communicative intention. Communicative intention is determined according to the linguistic situation that agents can perceive in a shared scene. A communicative intention matrix helps to measure this intention. A solution grammar is evolved starting from the meta-grammar's rules, which acts as a type of Universal Grammar for the agents. Each agent manages a population of chromosomes (binary strings) that are transformed into solution grammars and the agent's aim in this stage is to find the best solution grammar. Evolution is executed until all the agents find a suitable solution grammar or a maximum number of evolutionary trials is reached. In the later case, the evolutionary algorithm fails. If the evolutionary process succeeds, a reinforcement learning process is started, where the whole team plays language games. A syntactic alignment can be reached if all the agents agree to use the same sentences to describe the linguistic situation. If the reinforcement learning process succeeds, the problem is solved. If it fails, a new evolutionary process is started again and the process is repeated. If all the evolutionary and reinforcement trials fail the syntactic alignment is not found. Figure 4 summarizes the whole process.

```
for k=1,2, …. Max evolutionary trials do
    Execute an evolutionary process on each agent
    If evolutionary process succeeds
        Execute a reinforcement learning process
        If reinforcement learning succeeds
            Syntactical consensus found
            break
        end if
    end if
end for
```

Figure 4: Pseudocode for the model's main-loop

An alternative explanation of the model from a linguistic point of view is as follow. In a first stage the agents evolve a grammar, that is, they develop a *competence* for the language according to their communicative intentions. This grammar contains potentially the production rules which can describe the referential scene, but they have to be tested in a social event because the evolutionary process is a private event for each agent. Then, a second stage explores this social interaction by means of language games and the language is finally shared through a reinforcement process where the most frequently used sentences are rewarded and finally adopted by the team. Therefore, this stage represents the *performance* of the language, that is, the actual use in a social or cultural environment. Competence in this way constraints the performance because agents need syntactic rules that allow them to build sentences. The more suitable production rules there are in each agent's grammar, the more likely is consensus in the performance stage. This is why the maximum fitness in equation 1 rewards the greedy solution. If the agent has little or no grammatical knowledge, it cannot express sentences. That is, it will not be able to perform in actual communication. As the model executes for a number of evolution-learning cycles, the agents have several opportunities to evolve a suitable solution grammar, even though the grammar is evolved in a private way and there is no feedback from learning to evolution in an explicit way. To some extent, the model trusts in the agent's intention as the force that guides the evolution to find the right grammar.

6. Imposing Order Constraints

The previous model presents two potential problems once it has to scale to more complicated situations. On one hand, evolutionary processes usually need a long time and resources to find solutions, specially if the number of states in the space search increases. This situation is likely in a referential scene combining colors, objects and spatial relations. In this case, there is a significant number of possible combinations of words associated to each item. On the other hand, the evolutionary process in the model does not impose constraints about the order of the words in a sentence. Obviously, natural languages impose syntactic limitations to certain structures. For example, if a combination of words *object-color* or *color-object* has been adopted for the first object, it must be kept to refer to the second one. This way, the number of combinations of words in a sentence is reduced. The most obvious way to solve this problem consists in adding specific production rules to the meta-grammar. However, we are studying here the phenomenon of emergence of artificial languages and we do not want to impose these syntactic constraints. An alternative solution is to include some kind of validation in order to check the syntactic constraints that we have described. Fortunately, grammars are powerful devices and they can include semantic rules in the same production rules that describe the syntactic properties. Details about attribute grammars or similar devices which allow to combine syntactic and semantic rules cannot be reviewed here but we can find comprehensive studies in [39] or [40]. For the aim of the present work, only a few issues need to be explained, as we will do next.

According to [39], a *translation scheme* is a context-free grammar with attributes associated with the non-terminal symbols. An attribute has a name and a value. This value is computed by means of a semantic rule. Semantic rules or actions are included in the right side of the production rules and the order in which an action is executed is determined by its position in the production. In this work, a semantic rule will be represented enclosed within square brackets in the grammar's production rules and it can be a conditional or an assignment sentence. In the latter case, a value will be assigned to the attribute. Conditional sentences will check the attribute's values and will return a true or false value. A false value means that some condition is wrong in the process of building the derivation tree. As we explained above, the model builds a derivation tree when it is generating a solution grammar during the *translation process*, so that the semantic rules avoid wrong solution grammars. Thus, invalid solutions are discarded even before they are evaluated. An example will be used to clarify this process. Figure 5 shows a derivation tree similar to the one of Figure 3, but the new derivation tree includes semantic rules, which appear as dark grey nodes.

As a curiosity, the derivation tree represented by Figure 5 would be built starting from a solution with an integer string such as 2-0-1-2-1-2-2-1-0-0-1-2-

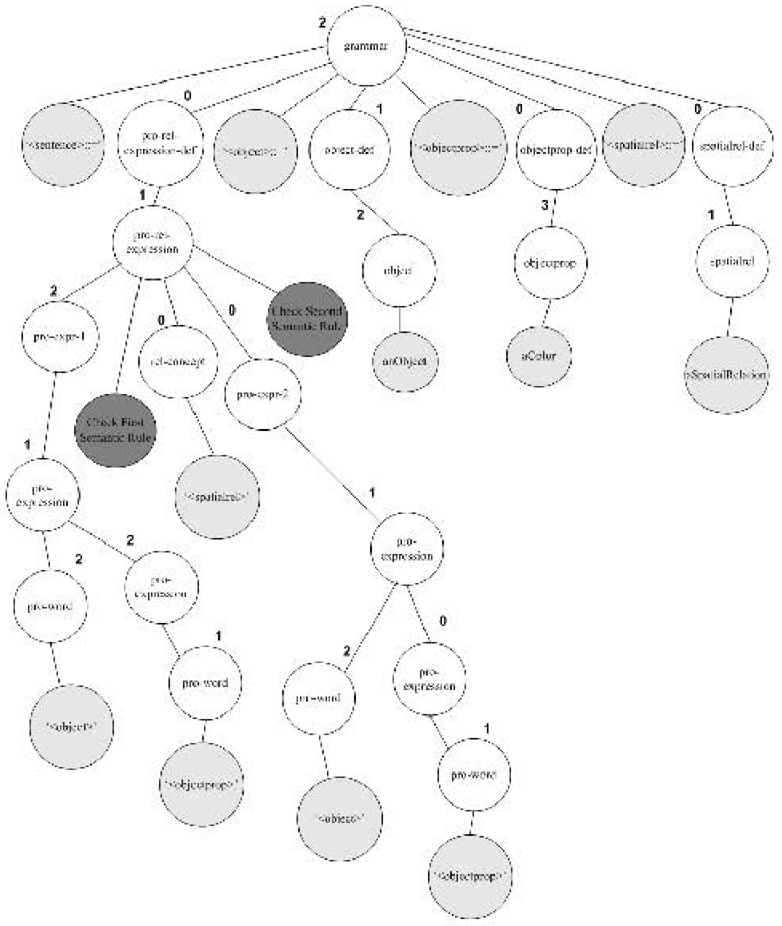

Figure 5: Derivation tree to generate a solution grammar with order constraints

0-1-1-2-0-3-0-1 during the translation process. We only need to explain what semantic rules really does. *Check First Semantic Rules* and *Check Second Semantic Rules* functions simply checks if the expressions contain the number of right words and the order is also equivalent. Both functions return false when the tests fail. In the example that is shown in Figure 5, there are no semantic errors, so the solution grammar is valid from the syntactic and semantic points of view. However if leave nodes $' <object>'$ and $' <objectprop>'$ under the node *<pro-expression-2>* were swapped or they represented the same symbol, the solution grammar would be syntactically valid but semantically invalid and this solution would be rejected. The most important consequence of semantic checking is that only those solution grammars with the same word-order are generated. Essentially, with the mechanism of semantic rules and the importance of the communicative intention matrix each agent is endowed with skills to evolve its own grammatical competence with some guarantees and the process is not blind at all.

7. Experimental Results

7.1. Suppositions and Setting

It is mandatory to comment here some important issues about the experiments and the focus of the model in this work. We are interested here in analyzing how a shared language can emerge in a population of artificial agents but we have in mind to translate the model to more complex device such as robots. Therefore, the model has to do with the symbolic or higher procedures, that is, it provides a framework to develop and use a language with a syntactical structure similar to the natural languages. Nevertheless, higher procedures are supported by other kind of processes such as perceptive and motor among others. We do not address these processes here and we suppose that agents are endowed with sensors and actuators that allow them to watch the scene and utter the sentences that they finally produce thanks to the model. Besides, other process or modules that are essentials to create intermediate representations about the perceived scenes are not treated here. All these suppositions allow us to concentrate efforts in the process of evolving the symbolic language.

Therefore, the experimental setting is a simple schema where agents are "notified" at the beginning of each referential scene and the previous hypothesis are included as initial knowledge. Then, the model helps to share a language with syntactic structure. A more realistic stage implies to include the model in a general architecture, for example, a hybrid architecture (see [41] for a comprehensive study about robot controllers and architectures) and specially to solve the Symbol Grounding Problem. However, we think that the model can solve part of the whole problem by contributing to implement some symbolic processes.

7.2. First Task

The experimental work in the initial task focus on medium-sized agent teams. We have experimented with two different linguistic situations such as the ones described in Figures 1 and 2. We suppose that all agents are located in the same place so their perspective are the same. Each linguistic situation includes a few specific objects, colors (in the case of Figure 1) and spatial relationships (in the case of Figure 2). More specifically, the scenes contain four simple objects (*book, pencil, glasses* and *ball*), three colors (*blue, yellow* and *green*) and two spatial relations (*left* and *right*). It is worth noting that the introduction of more objects, colors and spatial relations does not imply any significant difficulty for the agents' syntactic alignment since lexical categories are used, instead of specific objects or items.

We conducted a series of 50 experiments for the linguistic situation about objects and colors and a series of 20 experiments for the linguistic situation about objects and spatial relations. Different team sizes were tested, from 5 to 30 agents, with steps of 5 agents. Sentences are uttered by agents according to the probabilities of their learned stochastic grammars. In the first round, all the agents randomly choose a rule. During each language game round all the possible communicative acts between agent-pairs are performed and the production rules that allow the agent to make up the sentence are rewarded or punished depending on the success or failure in the corresponding communicative act. Success means that both agents share the same sentence to describe the linguistic situation. Syntactic alignment is attained when the whole team uses the same sentences to describe the scene. As the model can adapt itself to dynamic environments, the linguistic situations can be presented to the agents in any order, because we do not need to change anything while the system is executing. The agents can adapt their syntactic structures to each linguistic situation as the meta-grammar's production rule and the evolutionary algorithm are adaptive enough. In a typical execution we can set a scene representing a linguistic situation about object's properties such as the Figure 1 shows, then we start the system. Once the process of alignment ends we can set a new linguistic situation and repeat the process without redesign nor evolution nor the reinforcement algorithm. Alternatively we can start with a linguistic situation about spatial relations and move to another scene about object's properties. We think this is a key contribution of this model because it solves the problem of a previous design. With regard to the evolutionary algorithm, we can see in Table 2 some typical parameters with the values that we use in the experiments.

Table 2: Evolutionary Parameters

Population size	200
Maximum generations	Between 100 and 500 (depending on the number of agents in the team)
Crossover probability	0.8
Mutation probability	0.05
Duplication probability	0.05
Fitness value	2 in the color context and 3 in the spatial relationship context

Five initial parameters are established after some preliminary tests. The last one depends on the current linguistic situation and it defines the maximum number of rules in the evolved solution grammar. It is important not to confuse the terms *team* and *population*. A *team* is a collection of artificial agents who want to reach a shared language. A *population* is the term used in the evolutionary computation field to refer to the set of individuals or chromosomes (solution grammars in this case). Each agent manages a population of possible solution grammars.

As the model is split in two stages, they will be analyzed separately: first the evolutionary process that generates grammars, then, the reinforcement learning process.

7.2.1. Evolutionary Process Results

The evolutionary process is executed until the whole team evolves their suitable solution grammars or the maximum number of evolutionary trials is reached. We set this maximum value to 10 in the experiments. In order to measure how the evolutionary process performs, we use the standard success probability (the proportion of evolutionary runs where success is reached, out of the 20 or 50 total runs). Figure 6 shows results for the linguistic situation about objects and colors grouped by team-size.

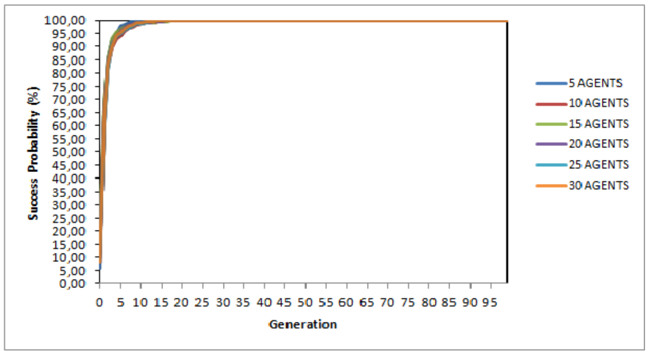

Figure 6: Success probability for the task about object-properties

Success probability is measured in terms of the whole team. Therefore, it is an average value for the team. As we can see in the figure, the proposed model found a solution in all the experiments (50 executions). Besides, solutions are found early in the evolutionary process because the model only needs a few generations (less than 30) to develop suitable solution grammars. The team size does not seem a significant limitation in the process and the type of suitable solution grammars that all agents evolve include at least two rules (fitness value) which allow them to build sentences with full sense according to the problem. Evolved grammars are similar to the following one:

```
<sentence> ::= .....
        | <objectprop> <object>
        | ....
        | <object> <objectprop>
        | ....
<object> ::= anObject
<objectprop> ::= aColor
```

As we can see, the model does not limit the number of production rules to evolve, but it looks for grammars that include as many valid production rules as it is specified in the fitness value. Valid production rules are those that potentially can derive correct expressions to describe the communicative intention. However, an evolved grammar can include other production rules. In any case, the type of evolved suitable grammars allows the agents to come into the reinforcement learning process with similar rules, so it is possible to reach syntactic alignment during language games. In fact, as we will show in the next section this is what it finally happens.

Spatial relations among objects is a slightly more complex problem because agents need to agree about sentences with two objects and a spatial relation. As we explained, there are three possibilities of ordering the words in the sentence. In this case, the evolutionary process searches for suitable solution grammars with at least the three following rules:

```
<sentence> ::= .....
        | <object> <spatialrel> <object>
        | ....
        | <spatialrel> <object> <object>
        | ....
        | <object> <object> <spatialrel>
<object> ::= anObject
<spatialRel> ::= aSpatialRelation
```

Figure 7 shows the success probability group by team size in this case.

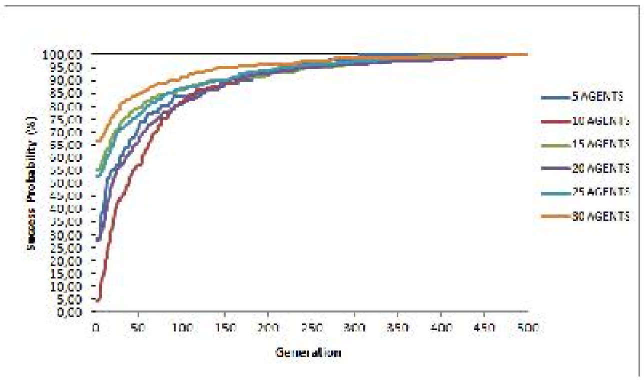

Figure 7: Success probability for the task about spatial relations among objects

In this task, even though the Figure 7 does not show it clearly, the model failed in a single experiment with 30 agents in the team. Failure happened because a single agent in the team was not able to build a suitable solution grammar with the evolutionary parameters that we set in the experiments. This agent failed after 10 evolutionary iterations but we think this kind of failure could be easily solved if the number of iterations were increased. In any case, as we can see in the figure, the model's performance is optimum because it is also able to find a solution with a small number of generations. It reaches around 95% of success with all team sizes in less than 250 generations.

To summarize, Figures 6 and 7 display how the evolutionary algorithm is able to develop suitable solution grammars for each agent in the team using few evolutionary generations. Most importantly, the model can adapt to new linguistic situations because the grammar is not fixed, but rather generated (restricted by a meta-grammar). A static pre-defined grammar such as the one proposed in [30], cannot adapt dynamically without redefining its production rules every time the linguistic situation changes.

7.2.2. Syntactic Alignment through Reinforcement Learning

Once the evolutionary stage has been completed, all the agents in the team have developed a suitable solution grammar and they are ready to play language games among them in order to find the syntactic alignment. This is where the reinforcement learning stage starts. In the evolutionary stage each agent in the team executes its own evolutionary algorithm. It has no interaction with other agents, but given that they share a common communicative intention and they watch the same referential scene, their solution grammars should be similar enough. In the reinforcement learning stage, agents interact with each other via language games. In order to measure how the reinforcement learning process is performing, we use the concept of *Communicative Efficiency* (CE) that we defined in Eq. 2. The higher the Communicative Efficiency, the higher the consensus within the team. 100% means complete syntactic alignment.

In Figure 8 we show the learning curves considering all the experiments and team sizes for the object properties task. The horizontal axis shows the rounds while the vertical axis shows the CE value (percentage). The maximum number of rounds depends on the number of agents and it ranges from 200 to 1000 for this task.

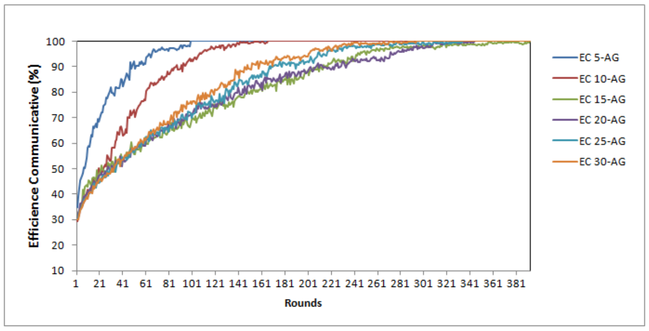

Figure 8: Communicative efficiency for the task about object-properties

As we can see in the figure a syntactic alignment is attained in all experiments for this task. Success in reinforcement learning is conditioned by the solution grammars evolved by agents in the previous stage, but results show that reinforcement learning produces a fast convergence, specially with small teams (5 and 10 agents where the syntactic alignment appears in less than 170 rounds).

Figure 9 shows a similar picture for the task about spatial relations among objects. In this case, only 20 experiments were run due to experiments being more computationally expensive.

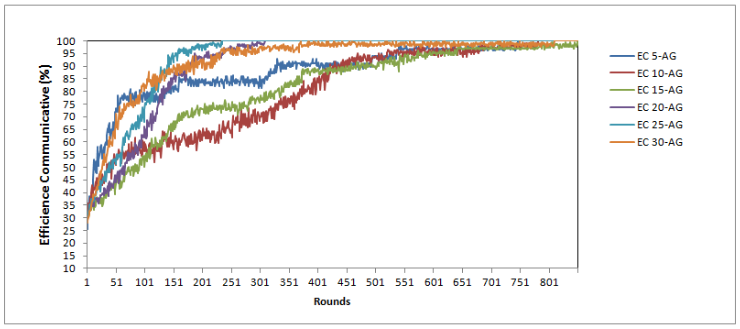

Figure 9: Communicative efficiency for the task about spatial relations among objects

Syntactic alignment is always reached, for all the cases where a suitable grammar was generated by all agents (there was such a failure for one of the experiments with 30 agents where no suitable grammar was generated by one of the agents). Results show a good performance in general terms because the proposed model can reach the final syntactic alignment in a relatively small number of language games rounds. As a curiosity, it can be observed that, once past 10 agents, adding more agents seems to make the learning process faster (communicative efficiency grows quicker). This is also true to some extent for Figure 8, past 15 agents. A similar behavior was also observed in [30] and it could be consequence of the larger number of interactions for large team sizes during the language games. In summary, both Figures 8 and 9 confirm that CE rises as rounds go by and this shows that reinforcement learning works and complements well the previous evolutionary process.

7.3. Second Task

In a second task the agents must agree sentences to describe a linguistic situation where they must combine words related to objects and colors and specify a spatial relation between two objects. As we commented in section two there at least 6 different combinations of words in this case, so the the maximum fitness value is 6 according to equation 1.

Under these conditions the evolutionary process is slower. To make it faster, the optimum fitness value could be decreased to 3, 4 or 5. However, if the optimum fitness is smaller than the maximum fitness, the evolutionary process will evolve solution grammars with fewer production rules and it will be less likely that the grammars are similar enough to reach consensus in the reinforcement learning stage. The computational cost is higher now, therefore only 20 experiments have been run, with teams of 5, 10 and 15 agents. For the 5 agents group we set the same evolutionary parameters as in Table 2, although **maximum generations** was fixed directly to 500. Optimum fitness was set to 3 for the experiments with 5 agents in order to reduce execution time. However, after some failed tests with this value we decided to work with the maximum fitness (6) in the case of 10 and 15 agents. Evolutionary parameters were also changed for these groups and they were finally fixed according to Table 3. The main differences between Tables 2 and 3 are in the crossover and mutation probabilities. As we did in the first task, we will first show results of the evolutionary process, then the reinforcement learning process.

Table 3: Evolutionary Parameters in Experiments with 10 and 15 agents

Population size	200
Maximum generations	500
Crossover probability	0.6
Mutation probability	0.1
Duplication probability	0.05
Fitness value	3 or 6

7.3.1. Evolutionary Results

We set the maximum number of evolutionary trials to 10 in all the experiments. Figure 10 shows results about success probability broken down by team size.

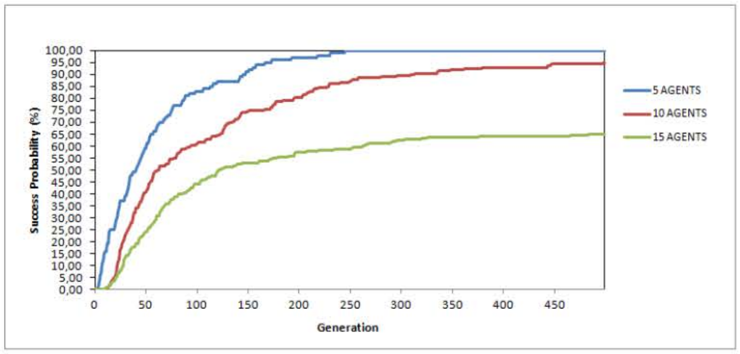

Figure 10: Success probability for the task with ordering constraints

As we can see in the Figure 10 the model evolved a suitable solution grammar for the group with 5 agents in all the experiments (100%) and it only needed less than 245 generations. On the other hand, it only failed in an experiment (95%) for the group with 10 agents. Nevertheless, it failed in seven experiments (65%) for the biggest group with 15 agents. In one of the experiments with 5 agents the evolved solution grammars were similar to the following one:

```
<sentence> ::= .....
| <object> <objectprop> <object> <objectprop> <spatialrel>
| ....
| <object> <objectprop> <spatialrel> <object> <objectprop>
| ....
| <spatialrel> <object> <objectprop> <object> <objectprop>
| ....
<object> ::= anObject
<objectprop> ::= <color>
<color> ::= aColor
<spatialrel> ::= aSpatialRelation
```

This type of solution grammar only includes 3 of the 6 combinations of words which are possible in this task. However, the important issue here is that all the team can play language games with the same set of production rules or a similar one. At minimum, to succeed in the reinforcement learning process it is mandatory that all evolved solution grammars (one for each agent) include at least a similar production rule. The more production rules in each grammar the more probabilities they will have at least a similar production rule. Of course, the reinforcement learning is responsible for reaching the final agreement but it depends on the quality of the evolved solution grammars. According to the experimental results with 10 and 15 agents (which used the maximum fitness value) we can argue that the team size could be determinant in order to evolve suitable grammars when more complex syntactic structures are needed.

7.3.2. Reinforcement Learning Results

Figure 11 shows the learning curve for this task. The maximum number of rounds depends on the number of agents and it ranges from 1000 to 25000 for this task.

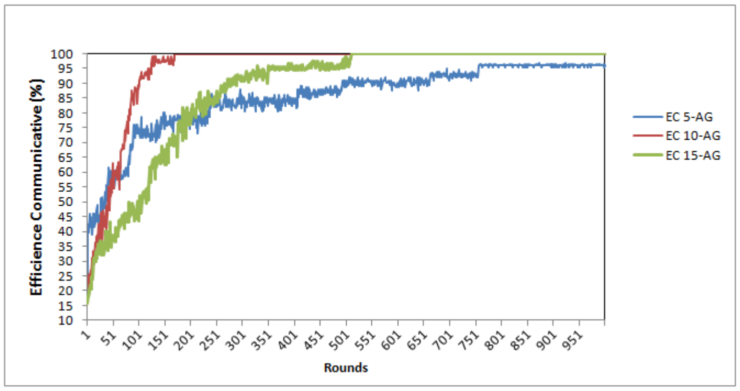

Figure 11: Communicative efficiency for the task with ordering constraints

It must be noticed that the results shown in Figure 11 do not consider the experiments that failed in the evolutionary process just as we did in the first task related to changing linguistic situations. This way, if we only take into account the reinforcement learning process the results are again successful because a 100% CE-value has been attained in groups with 10 and 15 agents and the group with 5 agents reached around 96%. In the last case, a single experiment failed during the learning stage even though its evolutionary process was successful. Despite this apparently worse performance in the smallest group, it should not be forgotten that the fitness value was smaller in this experiment than the maximum fitness that we use to evaluate higher groups. Apart from these details, the learning curve is essentially similar to the one we showed in Figures 8 and 9 and it reveals that the communicative efficiency also increases as the number of rounds increases.

We think it is important to return to the issue of failed experiments in order to analyze how language is developed and shared in this model. Success in reinforcement learning is conditioned to the suitable solution grammars that agents evolved in the previous stage. Therefore, syntactic alignment can fail for two reasons:

1. Evolution fails and some agent cannot evolve a suitable solution grammar to express its communicative intention. This is the reason why an experiment failed in the group of 10 agents and seven experiments failed in the group with 15 agents. The consequence of failed evolution is the incompetence to describe grammatically what the agent is watching.
2. Reinforcement learning fails and the team cannot agree sentences to describe the linguistic situation. Reinforcement learning is based on the previously evolved solution grammars. If all the agents evolve similar grammars the reinforcement learning process would only fail because the number of rounds was not enough for convergence. However, if there is an agent with a completely different grammar, the reinforcement learning process can also fail because it will be impossible for this agent to generate sentences which coincide with the sentences that the rest of the team utter through their own solution grammars. This is the reason why a experiment failed in the group with 5 agents. If we analyze the results we appreciate how the evolutionary process is not only more computationally expensive but it fails more frequently than the learning process. Comparing results it seems that the competence in the language is more difficult to reach than the performance.

Delving into differences among solution grammars we can appreciate the effect of a well-defined fitness measure in the evolutionary process. If we use the maximum value as fitness value to stop the evolution, there will be no difference among grammars because all of them will have similar production rules. Otherwise, it might be that production rules were completely different, so the syntactic alignment would be impossible. This problem did not happen in the task about changing linguistic situations that we described initially since we used there the maximum fitness: 2 for the context related to colors and 3 for the situation concerning spatial relations. However, we use 3 as optimum fitness value instead of 6 (maximum fitness value) in the second task for the group with 5 agents. As a consequence, in an experiment the model failed in the first evolutionary/reinforcement trial because the agents called *Agent_0, Agent_1* and *Agent_2* evolved similar solution grammars but agents called *Agent_3* and *Agent_4* evolved other different solution grammars. In this failed trial, three first agents would have reached their own syntactic alignment and the two last agents would have reached a different one if the model had allowed them to continue, but we assume that consensus is only accepted when the complete team agrees. Other trials failed in a similar way. Failed experiments show us how grammatical similarity is needed in order to share a language and it appears in the model as consequence of a selective pressure to look for individuals (solution grammars) with the most expressive power, according to the current communicative intention. Finally, we show in Appendix B an example of the final syntactic consensus in one of the valid experiments with 5 agents.

8. Conclusions

We propose in this work a model combining evolution and learning in order to study how a symbolic artificial language can self-emerge in a team of agents. The model considers some theories about natural language, such as the concepts of competence and performance in the language and the possibility of certain kind of innate grammatical knowledge expressed as a Universal Grammar. Competence is simulated as a private event for each agent in the group by means of an evolutionary process. The aim in this stage is to find the grammar that is suitable for the current linguistic situation. Evolution provides grammatical knowledge for the language but a shared language implies a social interaction. This public event is simulated in the learning component of the model, where a reinforcement learning algorithm uses a stochastic version of the evolved solution grammar for each agent, in order to converge to a common language with a syntactic structure. Syntactic alignment in the team is induced by means of language games which allow the agents to interact or communicate with each other. Therefore, the social interaction implies performance in the language.

Once the model's foundations have been established, a set of experiments was designed in order to test the model in an initial changing linguistic situation, where the agents must firstly agree sentences about object properties and secondly about spatial relations among the objects. This order is not important and it can be swapped. The main aim in the experiments was to prove if the model could adapt to the change in the linguistic situation without human intervention and results showed, not only how the model could find the syntactic alignment in both contexts, but how the model could adapt to the change without external help. Adaptation is possible thanks to the concept of meta-grammar which acts as a kind of Universal Grammar as we mentioned above. Experimental results also showed how both the evolutionary process and reinforcement learning process can find solutions in a few generations and rounds. A related work has previously shown the potential of the reinforcement learning in the emergence of artificial languages, but it needs that the grammar is defined in advance. It cannot adapt to new linguistic situations without modifying some parameter or feature. The model proposed here shows how these problems can be solved.

In a second task the agents had to maintain word order constraints in the sentences they generated. The evolutionary process is more time consuming in this case, specially if the number of agents in the team is large. Nevertheless, the model performed relatively well in small groups (5 and 10 agents). Besides, in a bigger team with 15 agents, the learning process performed equally fine but some problems were found in the evolutionary process. In any case, the model shows how semantic constraints can be included in the language, so the number of sentences that agents can express is potentially bigger.

Appendix A. Meta-grammar Definition

We use the usual BNF notation to specify the grammatical production rules and we adopt here the formalism proposed in [33]. In short, according to this convention, non terminal symbols between quotation marks stand for no terminal symbols in the solution grammar level which will be created starting from the meta-grammar's rules. Semantic rules are showed closed between brackets.

```
<grammar> ::= '<sentence>::=' <rel-expression-def> /
              '<object>::=' <object-def> /
              '<spatialrel>::=' <spatialrel-def>
            | '<sentence>::=' <pro-expression-def> /
              '<object>::=' <object-def> /
              '<objectprop>::=' <objectprop-def>
            | '<sentence>::=' <pro-rel-expression-def> /
              '<object>::=' <object-def> /
              '<objectprop>::=' <objectprop-def> /
              '<spatialrel>::=' <spatialrel-def>
<rel-expression-def> ::= <rel-expression>
            | <rel-expression> '|' <rel-expression-def>
<pro-expression-def> ::= <pro-expression>
            | <pro-expression> '|' <pro-expression-def>
<pro-rel-expression-def> ::= <pro-rel-expression>
            | <pro-rel-expression> '|' <pro-rel-expression-def>
<object-def> ::= <object>
<spatialrel-def> ::= <spatialrel>
<objectprop-def> ::= <objectprop>
<rel-expression> ::= <rel-word>
                   | <rel-word> <rel-expression>
<pro-expr-1> ::= <pro-expression>
<pro-expr-2> ::= <pro-expression>
<pro-expression> ::= <pro-word>
                   | <pro-word> <pro-expression>
<pro-rel-expression> ::= <rel-concept>
   <pro-expr-1>
   [if NOT(checkProExpr(<pro-expr-1>))
       then {"error"}]
   <pro-expr-2>
   [if NOT(checkSimilarityProExpr(<pro-expr-1>,<pro-expr-2>))
```

```
        then {"error"}]
   | <pro-expr-1>
     [if NOT(checkProExpr(<pro-expr-1>))
        then {"error"}]
     <rel-concept>
     <pro-expr-2>
     [if NOT(checkSimilarityProExpr(<pro-expr-1>,<pro-expr-2>))
        then {"error"}]
   | <pro-expr-1>
     [if NOT(checkProExpr(<pro-expr-1>))
        then {"error"}]
     <pro-expr-2>
     [if NOT(checkSimilarityProExpr(<pro-expr-1>,<pro-expr-2>))
        then {"error"}]
     <rel-concept>
<rel-concept> ::= '<spatialrel>'
<rel-word> ::= '<object>'
             | '<spatialrel>'
<pro-word> ::= '<object>'
             | '<objectprop>'
<object> ::= anObject
<spatialrel> :: =aSpatialRelation
<objectprop> ::= aColor
```

Where terminal symbols in this meta-grammar such as $' <sentence>::='$, $' <object>::='$, $' <spatialrel>::='$, $' <objectprop>::='$, $' <spatialrel>'$, $' <object>'$ and $' <objectprop>'$ will represent non terminals symbols in a solution grammar and the symbol $'|'$ stands for the alternative symbol as it is usual in BNF notation. Semantic functions such as *checkProExpr* and *checkSimilarityProExpr* implement order constraints.

Appendix B. Syntactic Consensus in 5-agents team for a task imposing order constraints

Syntactic consensus attained with the model in one of the experiments for a small team of 5 agents in the problem about order constraints.

```
SYNTACTIC STRUCTURES GENERATED AFTER THE GRAMMATICAL EVOLUTION
   Agent_0: anObject aColor aSpatialRelation anObject aColor
   Agent_1: anObject aColor aSpatialRelation anObject aColor
   Agent_2: anObject aColor aSpatialRelation anObject aColor
```

```
  Agent_3: anObject aColor aSpatialRelation anObject aColor
  Agent_4: anObject aColor aSpatialRelation anObject aColor

SYNTACTIC ALIGNMENT (SENTENCES) AFTER THE REINFORCEMENT LEARNING
  Agent_0
    Scene 0: book blue right ball green
    Scene 1: pencil yellow left glasses red
  Agent_1
    Scene 0: book blue right ball green
    Scene 1: pencil yellow left glasses red
  Agent_2
    Scene 0: book blue right ball green
    Scene 1: pencil yellow left glasses red
  Agent_3
    Scene 0: book blue right ball green
    Scene 1: pencil yellow left glasses red
  Agent_4
    Scene 0: book blue right ball green
    Scene 1: pencil yellow left glasses red
```

References

[1] P. Vogt. Language Evolution and robotics: Issues on symbol grounding and language acquisition. In: A. Loula, R. Gudwin, J. Queiroz (Eds.) Artificial Cognitive Systems. Herschey, PA. Idea Group, 176-209, 2006.

[2] H. Jaeger, A. Baronchelli, T. Briscoe, M. H. Christiansen, T. Griffiths, G. Jager, S. Kirby, N. L. Komarova, P. J. Richerson, L. Steels, J. Triesch. What Can Mathematical, Computational and Robotic Models Tell Us about the Origins of Syntax?. In: D. Bickerton and E. Szathmry (Eds.) Biological Foundations and Origin of Syntax. The MIT Press, 2009.

[3] S. Harnard. The symbol grounding problem. Physica, D 42, 335-346, 1990.

[4] L. Steels. Perceptually grounded meaning creation. In M. Tokoro (Ed.s). Proceedings of the International Conference on Multi-Agent Systems, Menlo Park CA: AAAIPress, 1996.

[5] L. Steels. The Spontaneous Self-organization of an Adaptive Language. In:

Muggleton, S. (ed.), Machine Intelligence 15. Oxford University Press, Oxford, 1996.

[6] L. Steels. Constructing and sharing perceptual distinctions. In Van Someren, M. and G. Widmed (eds.) Proceeding of the European conference on Machine Learning. Prague, April 1997. Springer-Verlag, Berlin, 1997.

[7] P. Vogt. Lexicon Grounding on Mobile Robot. Ph. D. Thesis. Vrije Universiteit Brussel. 2000.

[8] D. Jung, A. Zelinsky. Grounded Symbolic Communication between Heterogeneous Cooperating Robots. Autonomous Robots, vol. 8, 269-292, 2000.

[9] R. Schulz, A. Glover, M. J. Milford, G. Wyeth, J. Wiles. Lingodroids: Studies in Spatial Cognition and Language. IEEE International Conference on Robotics and Automation, May 9-13, Shanghai, China, 2011.

[10] M. Spranger. The Co-Evolution of Basic Spatial Terms and Categories. In Luc Steels (Ed.), Experiments in Cultural Language Evolution, 111-141. Amsterdam: John Benjamins. 2012.

[11] G. Werner, M. Dyer. Evolution of Communication in Artificial Organisms. In: Langton, C., et.al. (eds.). Artificial Life II. Addison-Wesley Pub. Co. Redwood City, Ca., 659-687, 1991.

[12] H. Yanko, L. Stein. An Adaptive Communication Protocol for Cooperating Mobile Robots. In: Meyer, J-A, H-L. Roitblat and S. Wilson. From Animals to Animats 2. Proceedings of the Second Interntational Conference on Simulation of Adaptive Behavior. The MIT Press, Cambridge Ma, 478-485, 1993.

[13] T. Hashimoto, T. Ikegami. Evolution of Symbolic Grammar Systems. In: Moran, F, A. Moreno, J. Merelo and P. Chacon (Eds.). Advances in Artificial Life. Third European Conference on Artificial Life. Granada, Spain, Springer-Verlag, Berlin, 812-823, 1995.

[14] D. Maravall, J. de Lope, R. Domínguez. Self-Emergence of Lexicon Consensus in a Population of Autonomous Agents by means of Binary-strings Evolution Strategies. Hybrid Artificial Intelligence Systems, LNCS 6077, E. Corchado, M. Graa, A. Manhaes (eds.) Springer-Verlag, 77-84, 2010.

[15] L. Wittgenstein. Philosophical Investigations. New York. Macmillan. 1953.

[16] F. de Saussure, Cours de Linguistic Général, Payot, Paris, 1916, English ed., McGraw-Hill. Ibidem Course on General Linguistic, New York, 1969.

[17] L. Steels (Ed.) Experiments in Cultural Language Evolution. John Benjamins, 2012.

[18] J. de Beule, J. V. Looeveren, W. H. Zuidema. Grounding Formal Syntax in an Almost Real World. Presented at the Belgium-Netherlands Artificial Intelligence Conference, 2002.

[19] J. V. Looveren, Design and Performance of Pre-Grammatical Language Games. PhD Thesis. Vrije Universiteit Brussel. March 2005.

[20] J. Batali. Computational Simulations of the Emergence of Grammar. In Hurford, J. R. and Studdert-Kennedy, M and Knight C., editors. Approaches to the Evolution of Language: Social and Cognitive Bases, 405-426, Cambridge University Press, 1998.

[21] J. Batali. The Negotiation and Acquisition of Recursive Grammars as a Result of Competition among Exemplars. In Linguistic Evolution through Language Acquisition: Formal and Computational Models, Ted Briscoe (editor), 111-172, 1999.

[22] T. Briscoe. Grammatical Acquisition: Inductive Bias and Coevolution of Language and the Language Acquisition Device, 2000.

[23] S. Kirby. Spontaneous Evolution of Linguistic Structure: An Iterated Learning Model of the Emergence of Regularity and Irregularity. IEEE Transactions on Evolutionary Computation, vol. 5, No. 2, 102-110, 2001.

[24] D. Roy. Learning Visually Grounded Words and Syntax of Natural Spoken Language. Evolution of Communication 4(1), 33-56, 2001.

[25] S. Kirby. Learning, Bottlenecks and the Evolution of Recursive Syntax. In T. Briscoe editor, Linguistic Evolution through Language Acquisition: Formal and Computational Models, 173-204, Cambridge University Press, 2002.

[26] D. Roy. Learning Visually-Grounded Words and Syntax for a Scene Description Task. Computer Speech and Language 16(3), 353-385, 2002.

[27] K. Smith, J. R. Hurford. Language Evolution in Populations: Extending the Iterated Learning Model. Advances in Artificial Live. Lecture Notes in Computer Science, vol. 2801, 507-516, 2003.

[28] M. R. McClain. Semantic Based Learning of Syntax in an Autonomous Robot. PhD Thesis. University of Illinois at Urbana-Champaign, 2006.

[29] M. Spranger, L. Steels. Emergent Functional Grammar for Space. In Luc Steels (Ed.), Experiments in Cultural Language Evolution, 207-232. Amsterdam: John Benjamins. 2012.

[30] D. Maravall, J. M. Mingo, J. de Lope. Alignment in Vision-based Syntactic Language Games for Teams of Robots using Stochastic Regular Grammars and Reinforcement Learning: The fully Autonomous case and the Human supervised case. Robotics and Autonomous Systems, Volume 63, Part 2, 180-186, January 2015.

[31] N. Chomsky. Aspects of the Theory of Syntax, MIT Press, Cambridge, MA, 1965.

[32] S. C. Levinson, D. P. Wilson. The Background to the Study of the Language of Space. In S. C. Levinson and D. P. Wilson editors. Grammars of Space. Explorations in Cognitive Diversity. Cambridge University Press, 2006.

[33] M. O'Neill and C. Ryan. Grammatical Evolution by Grammatical Evolution: The Evolution of Grammar and Genetic Code. In Maarten Keijzer, Una-May O'Reilly, Simon M. Lucas, Ernesto Costa, and Terence Soule, editors, Genetic Programming 7th European Conference, EuroGP 2004, Proceedings, volume 3003 of LNCS, pages 138-149, Coimbra, Portugal, 5-7, April 2004. Springer-Verlag. ISBN 3-540-21346-5.

[34] J. J. Collins, C. Ryan, M. O'Neill. Grammatical Evolution: Evolving Programs for an Arbitrary Language. Lecture Notes in Computer Science 1391, Proceedings of the First European Workshop on Genetic Programming. Springer-Verlag, 83-95, 1998.

[35] C. S. Wetherell. Probabilistic Language: A Review and Some Open Questions. ACM Computing Surveys (CSUR), 12(4):361-379, 1980.

[36] E. A. P. Hemberg. An Exploration of Grammars in Grammatical Evolution. PhD Thesis. University College Dublin. September, 2010.

[37] K. Narendra, M. A. L. Thathachar. Learning Automata- A Survey, IEEE Trans. on Systems, Man, and Cybernetics, 4 (4), 323-334, 1974.

[38] M. Tomasello. The Cultural Origins of Human Cognition. Harvard University Press, Cambridge, MA, 1999

[39] A. V. Aho, R. Sethi, J. D. Ullman. Compilers: Principles, Techniques and tools. Addison-Wesley. 1986.

[40] J. N. Shutt. Recursive Adaptable Grammars. Master's Thesis. Worcester Polytechnic Institute, Worcester, M. A. August 10, 1993, Emended December 16, 2003.

[41] R. Murphy. Introduction to AI Robotics. MIT Press, Cambridge. 2000.