Propositional Logic Syntax Acquisition*

Josefina Sierra-Santibáñez

Departamento de Lenguajes y Sistemas Informáticos
Universidad Politécnica de Barcelona, Spain
jsierra@lsi.upc.edu

Abstract. This paper addresses the problem of the acquisition of the syntax of propositional logic. An approach based on general purpose cognitive capacities such as invention, adoption, parsing, generation and induction is proposed. Self-organisation principles are used to show how a shared set of preferred lexical entries and grammatical constructions, i.e., a *language*, can emerge in a population of autonomous agents which do not have any initial linguistic knowledge.

Experiments in which a population of autonomous agents constructs a language that allows communicating the formulas of a propositional language are presented. This language although simple has interesting properties found in natural languages, such as compositionality and recursion.

1 Introduction

Recent work in linguistics and artificial intelligence [1,2,3,4,5,6,7,8] has described interesting experiments showing the emergence of compositional and recursive syntax in populations of agents without initial linguistic knowledge. This paper combines general purpose cognitive capacities (e.g., invention, adoption, parsing, generation and induction) and self-organisation principles in order to address the problem of the acquisition of the syntax of propositional logic.

The important role of logic in knowledge representation and reasoning [9] is well known in artificial intelligence. Much of the knowledge used by artificial intelligent agents today is represented in logic, and linguists use it as well for representing the meanings of words and sentences. This paper differs from previous approaches in using the syntax of logic as the subject of learning. Some could argue that it is not necessary to learn such a syntax, because it is built in the internal knowledge representation formalism used by the agents. We'd argue on the contrary that logical connectives and logical constructions are a fundamental part of natural language, and that it is necessary to understand how an agent can both conceptualise and communicate them to other agents.

The research presented in this paper assumes previous work on the conceptualisation of logical connectives [10,11]. In [12] a grounded approach to the acquisition of logical categories (connectives) based on the discrimination of a "subset

* This work is partially funded by the DGICYT TIN2005-08832-C03-03 project (MOISES-BAR).

P. Vogt et al. (Eds.): EELC 2006, LNAI 4211, pp. 128–142, 2006.

of objects" from the rest of the objects in a given context is described. The "subset of objects" is characterized by a logical formula constructed from perceptually grounded categories. This formula is satisfied by the objects in the subset and not satisfied by the rest of the objects in the context. In this paper we only focus on the problem of the acquisition of the syntax of propositional logic, because it is a necessary step to solve the complete problem of the acquisition of a grounded logical language (encompassing the acquisition of both the syntax and the semantics of propositional logic) and to our knowledge it has not been addressed before.

The emergence of recursive communication systems in populations of autonomous agents has been studied by other authors[1]. The research presented in [6] differs from the work described in the present paper by focusing on learning exemplars rather than grammar rules. These exemplars have costs, as our grammar rules do, and their costs are reinforced and discouraged using self-organization principles as well. The main challenge for the agents in the experiments described in [6] is to construct a communication system that is capable of naming atomic formulas and, more importantly, marking the identity relations among the arguments of the different atomic formulas that constitute the meaning of a given string of characters. This task is quite different from the learning task proposed in the present paper which focusses on categorizing propositional sentences and connectives, and marking the scope of each connective using the order of the constituents of a string of characters.

The most important difference between our work and that presented in [7] is that the latter one focusses on language transmission over generations. Rather than studying the emergence of recursive communication systems in a single population of agents, as we do, it shows that the bottleneck established by language transmission over several generations favors the propagation of compositional and recursive rules because of their compactness and generality. In the experiments described in [7] the population consists of a single agent of a generation that acts as a teacher and another agent of the following generation that acts as a learner. There is no negotiation process involved, because the learned never has the opportunity to act as a speaker in a single iteration. We consider however populations of three agents which can act both as speakers and hearers during the simulations. Having more than two agents ensures that the interaction histories of the agents are different from each other, in such a way that they have to negotiate in order to reach agreements on how to name and order the constituents of a sentence.

The induction mechanisms used in the present paper are based on the rules for chunking and simplification in [7], although we extend them so that they can be applied to grammar rules which have costs and usage counters attached to them. In particular we use the approach proposed in [8] for adding costs to the grammar rules, and computing the costs of sentences and meanings from the costs of the rules used for generating such sentences or meanings.

Finally the meaning space used in [7] (a restricted form of atomic formulas of second order logic) is different as well from the meaning space considered in the

[1] We review the work of the authors mentioned in this introduction in section 5.

present paper (arbitrary formulas from a propositional logic language), although both of them require the use of recursion.

The rest of the paper is organised as follows. First we present the formalism used for representing the grammars constructed by the agents. Then we describe in some detail the language games played by the agents, focusing on the main cognitive processes they use for constructing a shared lexicon and grammar: invention, adoption, induction and self-organisation. Next we report the results of some experiments in which a population of autonomous agents constructs a shared language that allows communicating propositional logic formulas. Finally we summarize some related work and the contributions of the paper.

2 Grammatical Formalism

We use a restricted form of definite-clause grammar in which non-terminals have three arguments attached to them. The first argument conveys semantic information. The second is a *score* in the interval [0, 1] that estimates the usefulness of that association in previous communication. The third argument is a counter that records the number of times the association has been used in previous language games.

Many grammars can be used to express the same meaning. The following holistic grammar can be used to express the propositional formula $right \wedge light$[2].

$$s([and, right, light]), 0.01) \rightarrow andrightlight \tag{1}$$

This grammar consists of a single rule which states that 'andrightlight' is a valid sentence meaning $right \wedge light$.

The same formula can be expressed using the following compositional, recursive grammar: s is the start symbol, $c1$ and $c2$ are the names of two syntactic categories associated with unary and binary connectives, respectively. Like in Prolog, variables start with a capital letter and constants with a lower case letter.

$$s(\text{light}, 0.70) \rightarrow \text{light} \tag{2}$$
$$s(\text{right}, 0.25) \rightarrow \text{right} \tag{3}$$
$$s(\text{up}, 0.60) \rightarrow \text{up} \tag{4}$$
$$c1(\text{not}, 0.80) \rightarrow \text{not} \tag{5}$$
$$s([P, Q], S) \rightarrow c1(P, S1), s(Q, S2), \{S\ is\ S1{*}S2{*}0.10\} \tag{6}$$
$$c2(\text{or}, 0.30) \rightarrow \text{or} \tag{7}$$
$$c2(\text{and}, 0.50) \rightarrow \text{and} \tag{8}$$
$$c2(\text{if}, 0.90) \rightarrow \text{if} \tag{9}$$
$$c2(\text{iff}, 0.60) \rightarrow \text{iff} \tag{10}$$
$$s([P, Q, R], S) \rightarrow c2(P, S1), s(Q, S2), s(R, S3), \{S\ is\ S1 * S2 * S3 * 0.01\} \tag{11}$$

[2] Notice that we use Prolog grammar rules for describing the grammars. The semantic argument of the rules uses Lisp like (prefix) notation for representing propositional formulas (e.g., the Prolog list $[and, [not, right], light]$ is equivalent to $\neg right \wedge light$). The third argument (the use counter) of non-terminals is not shown in the examples.

This grammar breaks down the sentence 'andrightlight' into subparts with independent meanings. The whole sentence is constructed concatenating these subparts. The meaning of the sentence is composed combining the meanings of the subparts using the variables P, Q and R.

The *score of a lexical rule* is the value of the second argument of the left hand side of the rule (e.g., the score of rule 8 is 0.50). The *score of a grammatical rule* is the last number of the arithmetic expression that appears on the right hand side of the rule[3](e.g., the score of rule 11 is 0.01). The score of a sentence generated using a grammatical rule is computed using the arithmetic expression on the right hand side of that rule (e.g., the score of sentence *andrightlight* is 0.50*0.25*0.70*0.01=0.00875).

3 Language Games

Syntax acquisition is seen as a collective process by which a population of autonomous agents constructs a *grammar* that allows them to communicate some set of meanings. In order to reach such an agreement the agents interact with each other playing language games. In the experiments described in this paper a particular type of *language game* called *the guessing game* [13,14] is played by two agents, a *speaker* and a *hearer*:

1. The speaker chooses a formula from a given propositional language, generates a sentence that expresses it and communicates that sentence to the hearer.
2. The hearer tries to interpret the sentence generated by the speaker. If it can parse the sentence using its lexicon and grammar, it extracts a meaning which can be equal or not to the formula intended by the speaker.
3. The speaker communicates the meaning it had in mind to the hearer and both agents adjust their grammars in order to become successful in future language games.

In a typical experiment hundreds of language games are played by pairs of agents randomly chosen from a population. The goal of the experiment is to observe the evolution of: (1) the communicative success[4]; (2) the internal grammars constructed by the individual agents; and (3) the external language used by the population.

3.1 Invention

In the first step of a language game the speaker tries to generate a sentence that expresses a propositional logic formula.

[3] The Prolog operator *"is"* allows evaluating the arithmetic expression at its right hand side.

[4] The *communicative success* is the average of successful language games in the last ten language games played by the agents. A language game is *successful* if the hearer can parse the sentence generated by the speaker, and the meaning interpreted by hearer is equal to the meaning intended by the speaker.

The agents in the population start with an empty lexicon and grammar. It is not surprising thus that they cannot generate sentences for some meanings at the early stages of a simulation run. In order to allow language to get off the ground, the agents are allowed to invent new words for those meanings they cannot express using their lexicons and grammars[5].

The invention algorithm is a recursive procedure that invents a sentence E for a meaning M. If M is atomic (not a list), it generates a new word E. If M is a list of elements (i.e., a unary or binary connective followed by one or two formulas, respectively), it tries to generate an expression for each of the elements in M using the agent's grammar. If it cannot generate an expression for an element of M using the agent's grammar, it invents an expression for that element calling itself recursively on that element. Once it has generated an expression for each element in M, it concatenates these expressions randomly in order to construct a sentence E for the whole meaning M.

As the agents play language games they learn associations between expressions and meanings, and induce linguistic knowledge from such associations in the form of grammatical rules and lexical entries. Once the agents can generate sentences for expressing a particular meaning using their own grammars, they select the sentence with the highest score out of the set of sentences they can generate for expressing that meaning, and communicate that sentence to the hearer. The algorithm used for calculating the score of a sentence from the scores of the grammatical rules applied in its generation is explained in detail later.

3.2 Adoption

The hearer tries to interpret the sentence generated by the speaker. If it can parse the sentence using its lexicon and grammar, it extracts a meaning which can be equal or not to the formula intended by the speaker.

As we have explained earlier the agents start with no linguistic knowledge at all. Therefore they cannot parse the sentences generated by the speakers at the early stages of a simulation run. When this happens the speaker communicates the formula it had in mind to the hearer, and the hearer adopts an association between that formula and the sentence used by the speaker.

It is also possible that the grammars and lexicons of speaker and hearer are not consistent, because each agent constructs its own grammar from the linguistic interactions in which it participates, and it is very unlikely that speaker and hearer share the same history of linguistic interactions unless the population consists only of these two agents. When this happens the hearer may be able to parse the sentence generated by the speaker, but its interpretation of that sentence may be different from the meaning the speaker had in mind. In this case, the strategy used to coordinate the grammars of speaker and hearer is to decrement the score of the rules used by speaker and hearer in the processes of generation and parsing, respectively, and allow the hearer to adopt an association between the sentence and the meaning used by the speaker.

[5] New words are sequences from 1 to 3 letters randomly chosen from the alphabet.

The adoption algorithm used in this paper is very simple. Given a sentence E and a meaning M, the agent checks whether it can parse E and interpret it as meaning M. This may happen when the hearer can parse the sentence used by the speaker, but it obtains a different meaning from the one intended by the speaker. In a language game the hearer always chooses the interpretation with the highest score out of the set of all the interpretations it that can obtain for a given sentence. So it is possible that the hearer knows the grammatical rules used by the speaker, but the scores of these rules are not higher than the scores of the rules it used for interpretation. If the hearer can interpret sentence E as meaning M, the hearer does not take any action. Otherwise it adopts the association used by the speaker by adding a new holistic rule of the form $s(M, 0.01) \rightarrow E$ to its grammar. The induction algorithm, used to generalise and simplify the agents' grammars, compares this rule with other rules already present in the grammar and replaces it with more general rules whenever it is possible.

3.3 Induction

In addition to invent and adopt associations between sentences and meanings, the agents use some *induction rules* [7] to extract generalizations from the grammar rules they have learnt so far [15]. The induction rules are applied whenever the agents invent or adopt a new association, to avoid redundancy and increase generality in their grammars.

Simplification: *Let* $r1$ *and* $r2$ *be a pair of grammar rules such that the left hand side semantics of* $r1$ *contains a subterm* $m1$, $r2$ *is of the form* $n(m1, S) \rightarrow e1$, *and* $e1$ *is a substring of the terminals of* $r1$. *Then simplification can be applied to* $r1$ *replacing it with a new rule that is identical to* $r1$ *except that* $m1$ *is replaced with a new variable* X *in the left hand side semantics, and* $e1$ *is replaced with* $n(X, S)$ *on the right hand side. The second argument of the left hand side of* $r1$ *is replaced with a new variable* SR. *If the score of* $r1$ *was a constant value* $c1$, *an expression of the form* $\{SR \textit{ is } S * 0.01\}$ *is added to the right hand side of* $r1$. *If the score of* $r1$ *was a variable, then the arithmetic expression* $\{SR \textit{ is } S1 * c1\}$ *in the right hand side of* $r1$ *is replaced by* $\{SR \textit{ is } S * S1 * 0.01\}$.

Suppose an agent's grammar contains rules 2, 3 and 4, which it has invented or adopted in previous language games. It plays a language game with another agent, and it invents or adopts the following rule.

$$s([and, light, right], 0.01) \rightarrow andlightright. \tag{12}$$

It could apply simplification to rule 12 (using rule 3) replacing it with rule 13.

$$s([and, light, R], S) \rightarrow andlight, s(R, SR), \{S \textit{ is } SR * 0.01\} \tag{13}$$

Rule 13 could be simplified again using rule 2, replacing it with 14.

$$s([and, Q, R], S) \rightarrow and, s(Q, SQ), s(R, SR), \{S \textit{ is } SQ * SR * 0.01\} \tag{14}$$

Suppose the agent plays another language game in which it invents or adopts a holistic rule for expressing the formula $[or, up, light]$ and applies simplification in a similar way. Then the agent's grammar would contain the following rules that are compositional and recursive, but which do not use syntactic categories for unary or binary connectives.

$$s([and, Q, R], S) \rightarrow and, s(Q, SQ), s(R, SR), \{S\ is\ SQ * SR * 0.01\} \quad (15)$$

$$s([or, Q, R], S) \rightarrow or, s(Q, SQ), s(R, SR), \{S\ is\ SQ * SR * 0.01\} \quad (16)$$

Chunk I. *Let $r1$ and $r2$ be a pair of grammar rules with the same left hand side category symbol. If the left hand side semantics of the two rules differ in only one position, and there exist two strings of terminals that, if removed, would make the right hand sides of the two rules the same, then chunking can be applied.*

Let $m1$ and $m2$ be the differences in the left hand side semantics of the two rules, and $e1$ and $e2$ the strings of terminals that, if removed, would make the right hand sides of the rules the same. A new category n is created and the following two new rules are added to the grammar.

$$n(m1, 0.01) \rightarrow e1 \qquad\qquad n(m2, 0.01) \rightarrow e2$$

Rules $r1$ and $r2$ are replaced by a new rule that is identical to $r1$ (or $r2$) except that $e1$ (or $e2$) is replaced with $n(X, S)$ on the right hand side, and $m1$ (or $m2$) is replaced with a new variable X in the left hand side semantics. The second argument of the left hand side of $r1$ is replaced with a new variable SR. If the score of $r1$ was a constant value $c1$, an expression of the form $\{SR\ is\ S * 0.01\}$ is added to the right hand side of $r1$. If the score of $r1$ was a variable, then the arithmetic expression $\{SR\ is\ S1 * c1\}$ in the right hand side of $r1$ is replaced by $\{SR\ is\ S * S1 * 0.01\}$.

For example the agent of previous examples, which has rules 15 and 16 for conjunctive and disjunctive formulas in its grammar, could apply chunking to these rules and create a new syntactic category for binary connectives as follows.

$$s([P, Q, R], S) \rightarrow c2(P, S1), s(Q, S2), s(R, S3), \{S\ is\ S1 * S2 * S3 * 0.01\} \quad (17)$$

$$c2(\text{and}, 0.01) \rightarrow \text{and} \quad (18)$$

$$c2(\text{or}, 0.01) \rightarrow \text{or} \quad (19)$$

Rules 15 and 16 would be replaced with rule 17, which generalises them because it can be applied to arbitrary formulas constructed using binary connectives, and rules 18 and 19, which state that *and* and *or* belong to $c2$ (the syntactic category of binary connectives), would be added to the grammar.

Chunk II. *If the left hand side semantics of two grammar rules $r1$ and $r2$ can be unified applying substitution $X/m1$ to $r1$ and there exists a string of terminals $e1$ in $r2$ that corresponds to a nonterminal $c(X, S)$ in $r1$, then chunking can be applied to $r2$ as follows. Rule $r2$ is deleted from the grammar and a new rule of the following form $c(m1, 0.01) \rightarrow e1$ is added to it.*

Suppose the agent of previous examples adopts or invents the following rule[6].

$$s([\text{iff}, up, right], 0.01) \rightarrow \text{iffupright}. \tag{20}$$

Simplification of rule 20 with rules 4 and 3 leads to replace rule 20 with 21.

$$s([\text{iff}, Q, R], S) \rightarrow \text{iff}, s(Q, SQ), s(R, SR), \{S\ is\ SQ * SR * 0.01\} \tag{21}$$

Then chunking could be applied to 21 and 17, replacing rule 21 with 22.

$$c2(\text{iff}, 0.01) \rightarrow \text{iff} \tag{22}$$

3.4 Self-organisation

The agent in the previous examples has been very lucky, but things are not always that easy. Different agents can invent different words for referring to the same propositional constants or connectives, and the invention process uses a random order to concatenate the expressions associated with the components of a given meaning. This has important consequences, because the simplification rule takes into account the order in which the expressions associated with the meaning components appear in the terminals of a rule. Imagine an agent invented/adopted the following holistic rules for expressing [and,light,right] and [if,light,right].

$$s([and, light, right], 0.01) \rightarrow \textit{andlightright}$$
$$s([\textit{if}, light, right], 0.01) \rightarrow \textit{ifrightlight}$$

The result of simplifying these rules using rules 2 and 3 would be the following pair of rules which cannot be used for constructing a syntactic category for binary connectives, because they do not satisfy the preconditions of chunking.

$$S([and, X, Y], SC) \rightarrow and, s(X, SX), s(Y, SY), \{SC\ is\ SX * SY * 0.56\}$$
$$S([\text{if}, X, Y], SC) \rightarrow \text{if}, s(Y, SY), s(X, SX), \{SC\ is\ SX * SY * 0.56\}$$

The agents must therefore reach agreements on how to name propositional constants and connectives, and on how to order the expressions associated with the different components of non-atomic meanings. Self-organisation principles help to coordinate the agents' grammars in such a way that they prefer to use the rules that are used more often by other agents [16,6,3]. The set of rules preferred by most agents for naming atomic meanings and for ordering the expressions associated with the components of non-atomic meanings constitutes the *external language* spread over the population.

The goal of the self-organisation process is that the agents in the population be able to construct a shared external language and that they prefer using the rules in that language over the rest of the rules in their individual grammars.

[6] Notice that the scores of all rules created using invention, adoption or induction are initialised to 0.01. The use counters (not shown in the examples) are initialised to 0.

Coordination takes place at the third stage of a language game, when the speaker communicates the meaning it had in mind to the hearer. Depending on the outcome of the language game speaker and hearer take different actions. We have talked about some of them already, such as invention or adoption, but they can also adjust the scores of the rules in their grammars to become more successful in future games.

First we consider the case in which the speaker can generate a sentence for the meaning using the rules in its grammar. If the speaker can generate several sentences for expressing that meaning, it chooses the sentence with the highest score, the rest are called *competing sentences*.

The *score of a sentence* (or a *meaning*) is computed at generation (parsing) multiplying the scores of the rules involved [8]. Consider the generation of a sentence for expressing the meaning $[and, right, light]$ using the following rules.

$$s(\text{light}, 0.70) \rightarrow \text{light} \quad (23)$$
$$s(\text{right}, 0.25) \rightarrow \text{right} \quad (24)$$
$$c2(\text{and}, 0.50) \rightarrow \text{and} \quad (25)$$
$$s([P, Q, R], S) \rightarrow c2(P, S1), s(Q, S2), s(R, S3), \{S\ is\ S1{\cdot}S2{\cdot}S3{\cdot}0.01\} \quad (26)$$

The score S of the sentence *andrightligth*, generated by rule 26, is computed multiplying the score of that rule (0.01) by the scores of the rules 25, 24 and 23 which generate the substrings of that sentence. The *score of a lexical rule* is the value of the second argument of the left hand side of the rule (e.g., the score of rule 25 is 0.50). The *score of a grammatical rule* is the last number of the arithmetic expression that appears on the right hand side of the rule[7](e.g., the score of rule 26 is 0.01). The score of a sentence generated using a grammatical rule is computed using the arithmetic expression on the right hand side of that rule (e.g., the score of sentence *andrightlight* is 0.50*0.25*0.70*0.01=0.00875).

Suppose the hearer can interpret the sentence communicated by the speaker. If the hearer can obtain several interpretations for that sentence, the meaning with the highest score is selected, the rest are called *competing meanings*.

If the meaning interpreted by the hearer is the same as the meaning the speaker had in mind, the game succeeds and both agents adjust the scores of the rules in their grammars. The speaker increases the scores of the rules it used for generating the sentence communicated to the hearer and decreases the scores of the rules it used for generating competing sentences. The hearer increases the scores of the rules it used for obtaining the meaning the speaker had in mind and decreases the scores of the rules it used for obtaining competing meanings. This way the rules that have been used successfully get reinforced, and the rules that have been used for generating competing sentences or competing meanings are inhibited to avoid ambiguity in future language games.

The rules used for *updating the scores of grammar rules* are the same as those proposed in [13]. The rule's original score S is replaced with the result of

[7] The Prolog operator *"is"* allows evaluating the arithmetic expression at its right hand side.

evaluating expression 27 if the score is *increased*, and with the result of evaluating expression 28 if the score is *decreased*. The constant μ is a leaning parameter which is set to 0.1.

$$maximum(1,\ S + \mu) \tag{27}$$

$$minimum(0,\ S - \mu) \tag{28}$$

If the meaning interpreted by the hearer it is not equal to the meaning the speaker had in mind, the game fails, and speaker and hearer decrease the scores of the rules they used for generating and interpreting the sentence, respectively. This way the rules that have been used without success are inhibited.

If the speaker can generate a sentence for the meaning it has in mind, but the hearer cannot interpret that sentence, the hearer adopts a holistic rule associating the meaning and the sentence used by the speaker. This holistic rule can be simplified and chunked later using the rest of the rules in the hearer's grammar.

In order to simplify the agents's grammars and avoid possible sources of ambiguity a **mechanism for purging rules** that have not been useful in past language games is introduced. Every ten language games the rules which have been used more than thirty times and have scores lower than 0.01 are removed from the agents' grammars.

4 Experiments

We present the results of some experiments in which three agents construct a shared language that allows communicating the formulas of a logical language $L = \{a, b, c, l, r, u\}$ with six propositional constants. The agents build different, but *compatible, compositional, recursive grammars* that allow them to communicate each other the infinite set of meanings that can be represented in L.

First the agents play 600 language games in which they try to communicate propositional constants. Then they play 1200 language games in which they try to communicate propositional constants and logical formulas constructed using unary and binary connectives (i.e., $\neg, \wedge, \vee, \rightarrow$ and $\leftrightarrow$).

Tables 1 and 2 describe the individual lexicons and grammars built by the agents at the end of a particular simulation run. The grammars built by the agents, although different, are compatible enough to allow total communicative success. That is, the agents always generate sentences that are understood by the other agents.

The grammars of all the agents have recursive rules for expressing formulas constructed using unary and binary connectives. The expression C associated with the connective is always placed at the start of a sentence by the induction algorithm. Let 1 and 2 be the expressions associated with the first and second arguments, respectively, of a formula constructed using a binary connective. The order in which 1 and 2 are concatenated determines thus the form of the sentence. We call *C12-constructions* to those rules that construct a sentence concatenating C, 1 and 2 in C12 order, and *C21-constructions* to those rules that concatenate

Table 1. The lexicons of all the agents are identical, i.e., all the agents prefer the same words for referring to the propositional constants of the language L = { a, b, c, l, r, u}

Lexicon for Propositional Constants		
Lexicon a1	**Lexicon a2**	**Lexicon a3**
s(a1,a,1) → e	s(a2,a,1) → e	s(a3,a,1) → e
s(a1,b,1) → uo	s(a2,b,1) → uo	s(a3,b,1) → uo
s(a1,c,1) → bt	s(a2,c,1) → bt	s(a3,c,1) → bt
s(a1,l,1) → u	s(a2,l,1) → u	s(a3,l,1) → u
s(a1,r,1) → ihg	s(a2,r,1) → ihg	s(a3,r,1) → ihg
s(a1,u,1) → y	s(a2,u,1) → y	s(a3,u,1) → y

them in C21 order. All agents prefer C12-constructions (third rules of a1 and a2, and second rule of a3) for expressing conjunctive and disjunctive formulas, and they prefer to C21-constructions (sixth rules of a1 and a2, and fifth rule of a3) for expressing implications and equivalences. Agents a1 and a2 have invented a syntactic category (c3) for unary connectives, because they probably had several words for expressing negation which were eliminated afterwards by the purging mechanism; a3 has a specific rule for formulas constructed using negation, which uses the word "ps" preferred by the others.

All agents have created syntactic categories (c2, c1) for *binary connectives used in C12-constructions* and they prefer the same words for the connectives *and* and *or* (scores 1). They have created syntactic categories for *binary connectives used in C21-constructions* and they prefer the same words for the connectives *if* and *iff* (scores 1). There are no alternative words for any connective. This is probably due to the fact that the purging mechanism has eliminated such words from the lexicons of the agents.

Figure 1 shows some preliminary results about the evolution of the communicative success, averaged over ten simulation runs with different initial random seeds, for a population of three agents[8].

The agents reach a communicative success of 98% in 250 language games and of 100% in 1000 language games. That is, after each agent has played, on average, 200 language games about propositional constants, and 333 language games about propositional constants and formulas constructed using logical connectives.

5 Related Work

Batali [6] studies the emergence of recursive communication systems as the result of a process of negotiation among the members of a population. The alternative explored in this research is that learners simply store all of their analyzed ob-

[8] The *communicative success* is the average of successful language games in the last ten language games played by the agents.

Table 2. Grammars constructed by the agents at the end of a simulation run

Grammars for Propositional Logic	
Gram a1	s(a1,[X,Y],R) → c3(X,P), s(Y,Q), {R is P*Q*1} c3(a1,not,1) → ps s(a1,[X,Y,Z],T) → c2(X,P), s(Y,Q), s(Z,R), {T is P*Q*R*1} c2(a1,and,1) → oyv c2(a1,or,1) → gs s(a1,[X,Y,Z],T) → c1(X,P), s(Z,Q), s(Y,R), {T is P*Q*R*1} c1(a1,if,1) → ogb c1(a1,iff,1) → qan
Gram a2	s(a2,[X,Y],R) → c3(X,P), s(Y,Q), {R is P*Q*1} c3(a2,not,1) → ps s(a2,[X,Y,Z],T) → c1(X,P), s(Y,Q), s(Z,R), {T is P*Q*R*1} c1(a2,and,1) → oyv c1(a2,or,1) → gs s(a2,[X,Y,Z],T) → c2(X,P), s(Z,Q), s(Y,R), {T is P*Q*R*1} c2(a2,if,1) → ogb c2(a2,iff,1) → qan
Gram a3	s(a3,[not,Y],R) → ps, s(Y,Q), {R is Q*1} s(a3,[X,Y,Z],T) → c1(X,P), s(Y,Q), s(Z,R), {T is P*Q*R*1} c1(a3,and,1) → oyv c1(a3,or,1) → gs s(a3,[X,Y,Z],T) → c2(X,P), s(Z,Q), s(Y,R), {T is P*Q*R*1} c2(a3,if,1) → ogb c2(a3,iff,1) → qan

servations as exemplars. No rules or principles are induced from them. Instead exemplars are used directly to convey meanings and to interpret signals.

The agents acquire their exemplars by recording observations of other agents expressing meanings. A learner finds the cheapest phrase with the observed string and meaning that can be created by combining or modifying phrases from its existing set of exemplars, creating new tokens and phrases if necessary.

As an agent continues to record learning observations, its exemplar set accumulates redundant and contradictory elements. In order to choose which of a

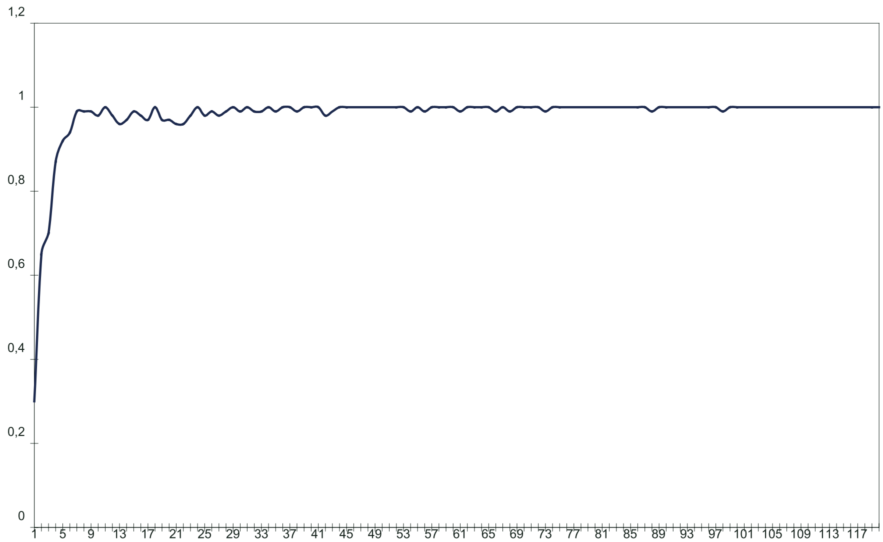

Fig. 1. Evolution of communicative success in experiments involving 3 agents and 1200 language games about propositional constants and formulas of $L = \{a, b, c, r, l, u\}$ constructed using unary and binary connectives (i.e., $\neg, \wedge, \vee, \rightarrow$ or $\leftrightarrow$)

set of alternative exemplars, or modified analyses based on them, will be used in a particular episode the cost of different solution phrases are compared, and a competition process among exemplars based on reinforcement and discouragement is established. An exemplar is reinforced when it is used in the phrase an agent constructs to record a learning observation, and it is discouraged when it is found to be inconsistent with a learning observation. Reinforcement and discouragement implement therefore a competition among groups of exemplars.

In the computational simulations described in [6] ten agents negotiate communication systems that enable them to accurately convey meanings consisting of sets of 2 to 7 atomic formulas (constructed from 22 unary and 10 binary predicates) which involve at most 3 different variables, after each agent has made fewer than 10000 learning observations. Each agent acquires several hundred exemplars, of which a few dozen are singleton tokens identical to those of other agents in the population.

The agents express meanings by combining their singleton tokens into complex phrases using the order of phrases, as well as the presence and position of empty tokens, to indicate configurations of predicate arguments. Empty tokens are also used to signal the boundaries of constituents, the presence of specific argument maps, and details of the structure of the phrases containing them.

Kirby [7] studies the emergence of basic structural properties of language, such as compositionality and recursion, as a result of the influence of learning on the complex dynamical process of language transmission over generations.

This paper describes computational simulations of language transmission over generations consisting of only two agents: an adult speaker and a new learner.

Each generation in a simulation goes through the following steps: 1.- The speaker is given a set of meanings, and produces a set of utterances for expressing them either using its knowledge of language or by some random process of invention. 2.- The learner takes this set of the utterance-meaning pairs and uses it as input for its induction learning algorithm. 3.- Finally a new generation is created where the old speaker is discarded, the learner becomes the new speaker, and a new individual is added to become a new learner. At the start of a simulation run neither the speaker nor the learner have any grammar at all.

The induction algorithm thus proceeds by taking an utterance, incorporating the simplest possible rule that generates that utterance directly, searching then through all pairs of rules in the grammar for possible subsumptions until no further generalisations can be found, and deleting finally any duplicate rules that are left over. The inducer uses *merging* and *chunking* to discover new rules that subsume pairs of rules that have been learnt through incorporation, and *simplification* for generalising some rules using other rules in the grammar.

The meaning space of the second experiment described in [7] consists of formulas constructed using 5 binary predicates, 5 objects and 5 embedding binary predicates. Reflexive expressions are not allowed (i.e., the arguments of each predicate must be different). Each speaker tries to produce 50 degree-0 meanings, then 50 degree-1 meanings, and finally 50 degree-2 meanings. The grammar of generation 115 in one of the simulation runs has syntactic categories for nouns, verbs, and verbs that have a subordinating function. It also has a grammar rule that allows expressing degree-0 sentences using VOS (verb, object, subject) order, and another recursive rule that allows expressing meanings of degree greater than 0. In the ten simulation runs performed the proportion of meanings of degrees 0, 1 and 2 expressed without invention in generation 1000 is 100%.

6 Conclusions

This paper has addressed the problem of the acquisition of the syntax of propositional logic. An approach based on general purpose cognitive capacities such as invention, adoption, parsing, generation and induction has been proposed. Self-organisation principles have been used to show how a shared set of preferred lexical entries and grammatical constructions, i.e., a *language*, can emerge in a population of autonomous agents which do not have any initial linguistic knowledge.

Experiments in which a population of autonomous agents comes up with a language that allows them to communicate about the formulas of a propositional language have been presented. This language although simple has interesting properties found in natural languages, such as compositionality and recursion.

Acknowledgements

The author would like to thank the anonymous reviewers for their valuable comments and suggestions on how to improve the structure of the paper.

References

1. Steels, L.: The origins of syntax in visually grounded robotic agents. Artificial Intelligence **103(1-2)** (1998) 133–156
2. Steels, L.: The emergence of grammar in communicating autonomous robotic agents. In: Proceedings of the European Conference on Artificial Intelligence. IOS Publishing, Amsterdam. (2000)
3. Steels, L.: Constructivist development of grounded construction grammars. In: Proc. Annual Meeting of Association for Computational Linguistics. (2004) 9–16
4. Steels, L., Wellens, P.: How grammar emerges to dampen combinatorial search in parsing. In: Proc. of Third International Symposium on the Emergence and Evolution of Linguistic Communication. (2006)
5. Hurford, J.: Social transmission favors linguistic generalization. In: The Evolutionary Emergence of Language: Social Function and the Origins of Linguistic Form, Cambridge University Press (2000) 324–352
6. Batali, J.: The negotiation and acquisition of recursive grammars as a result of competition among exemplars. In: Linguistic Evolution through Language Acquisition: Formal and Computational Models, Cambridge U.P. (2002) 111–172
7. Kirby, S.: Learning, bottlenecks and the evolution of recursive syntax. In: Linguistic Evolution through Language Acquisition: Formal and Computational Models, Cambridge University Press (2002) 96–109
8. Vogt, P.: The emergence of compositional structures in perceptually grounded language games. Artificial Intelligence **167(1-2)** (2005) 206–242
9. McCarthy, J.: Formalizing Common Sense. Papers by John McCarthy. Ablex. Edited by Vladimir Lifschitz (1990)
10. Sierra-Santibáñez, J.: Grounded models as a basis for intuitive reasoning. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence. (2001) 401–406
11. Sierra-Santibáñez, J.: Grounded models as a basis for intuitive and deductive reasoning: The acquisition of logical categories. In: Proceedings of the European Conference on Artificial Intelligence. (2002) 93–97
12. Sierra-Santibáñez, J.: Grounded models as a basis for intuitive reasoning: the origins of logical categories. In: Papers from AAAI–2001 Fall Symposium on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems. Technical Report FS-01-01, AAAI Press. (2001) 101–108
13. Steels, L.: The Talking Heads Experiment. Volume 1. Words and Meanings. Special Pre-edition for LABORATORIUM, Antwerpen (1999)
14. Steels, L., Kaplan, F., McIntyre, A., V Looveren, J.: Crucial factors in the origins of word-meaning. In: The Transition to Language, Oxford Univ Press (2002) 252–271
15. Steels, L.: Macro-operators for the emergence of construction grammars. SONY CSL (2004)
16. Steels, L.: The synthetic modeling of language origins. Evolution of Communication **1(1)** (1997) 1–35