Luc Steels* and Emília Garcia Casademont

Ambiguity and the origins of syntax

Abstract: The paper argues that syntax is motivated by the need to avoid combinatorial search in parsing and semantic ambiguity in interpretation. It reports on a case study for the emergence and sharing of first-order phrase structures in a population of agents playing language games. First-order phrase structures combine words into phrases but do not yet generalise to hierarchical or recursive phrases. To study why human languages exhibit phrase structure, a series of strategies for creating and sharing linguistic conventions are examined, starting from a lexical strategy without syntax and then studying the use of groups, n-grams and patterns. Each time we show in which way a strategy improves on the computational complexity of the previous one.

Keywords: origins of syntax, language games, language strategies, semiotic dynamics

DOI 10.1515/tlr-2014-0021

1 Introduction

The question why human languages exhibit intricate syntax and how this could have arisen in the evolution of our species is one of the most profound deep questions in linguistics. We assume here a selectionist approach which implies that syntactic structure is not simply an accidental property of language but is motivated by the challenge of collectively building a communication system that can be produced and comprehended by a human brain and a human sensory apparatus, and this implies in particular that only finite resources in terms of memory, processing power, processing time, and learning time should be required. Explaining the emergence of syntactic structure, and more specifically phrase structure, therefore requires (i) showing why syntactic categories (lexical and phrasal) are useful, (ii) why patterns, defined as sequences of slots filled by elements belonging to specific syntactic categories, are advantageous, and (iii) why ordering

***Corresponding author: Luc Steels:** ICREA. Institut de Biologia Evolutiva (CSIC-UPF) Barcelona.
E-mail: steels@ai.vub.ac.be

Emília Garcia Casademont: Institut de Biologia Evolutiva (CSIC-UPF) Barcelona.
E-mail: emilia-maria.garcia@upf.edu

relations between the slots of a pattern aid in comprehension and production. In this paper, we address these issues but, due to space limitations, we only report on the first stage in the emergence of phrase structure, namely first-order phrase structures that group lexical categories into phrases but not yet phrases into higher order phrases. Generalisation to fully recursive hierarchical structure is reported in a forthcoming paper.

There has already been a substantial body of earlier work on the emergence of syntactic structure. One widespread hypothesis (following from research in Iterated Learning) is that language learners use a learning algorithm (for example minimal description length learning) that seeks structure in the data, even if the data is not or only weakly structured (Smith et al. 2003). Once they have hypothesised structure by overgeneralisation, learners impose it on their own utterances as speakers so that the next generation of learners picks up this structure again and possibly imposes more structure of their own. In this approach, the introduction of phrase structure is exclusively in the hands of the learner and is motivated by overcoming the transmission bottleneck.

In contrast, we argue here that syntax arises from the need to *avoid combinatorial explosions in parsing and semantic ambiguity in interpretation*. So we seek a functionalist as opposed to structuralist explanation. Moreover, the way structure arises is not through a transmission bottleneck but by strategies for the stepwise invention, adoption and alignment of linguistic conventions in a population based on a *cultural selectionist dynamics* (Steels 1997, 2012a).

Cultural selection (more precisely *linguistic selection*) projects Darwin's original idea of natural selection to the cultural/linguistic level. It can be summarised as follows:

1. Language users employ a variety of *strategies* to build, optimise and maintain their language systems. Each strategy addresses a particular linguistic issue and includes an approach to learn, expand, or align linguistic conventions. Linguistic issues either concern the expression of certain aspects of meaning, for example quantification or the expression of time, or reducing the cognitive effort in articulation, speech recognition, parsing, production and/or semantic interpretation and conceptualisation.
2. Because every individual can make their own changes, Variation in the language of individuals in a speech population is unavoidable, both in terms of which strategies they employ and how they use a shared strategy to handle a concrete issue.
3. The choice which variants are retained and become dominant in the population is based on linguistic selection criteria: Those variants that allow speakers or hearers to have more communicative success with less effort will be

preferred by them. Hence they are maintained in their language, used more, and thus spread faster.

- 4. New strategies arise by the recruitment of generic mechanisms, such as ‘group items together which belong together’ or ‘use a general pattern for imposing order on the elements of a set’. Strategies only survive when they make a contribution to the expressive power and efficacy of the language.

The application of this selectionist framework critically depends on identifying the selectionist criteria that motivate the adoption of a language change. To study these criteria in the case of syntax, the paper takes a computer science approach. The computational analysis of a complex task starts normally from the simplest algorithm that already has some but not all of the desired functions or behaviors to handle the task and then this algorithm is progressively made more complex until it approaches the full complexity of the original challenge. At each step, it must be shown that the algorithm fulfills its function and that the computational complexity is manageable, more specifically, whether performance scales reasonably with an increase in the task parameters. This progressive complexification allows a systematic and careful study why each component of a complex system is needed and what its role is.

In this paper, we use this standard computer science methodology to study four strategies (i.e. agent-based collective algorithms): lexicalisation, grouping, n-grams, and patterns (see Figure 1). The lexicalisation strategy does not involve

Fig. 1: Overview of the strategies discussed in the paper. Starting from a lexical strategy without syntax, the paper examines the functioning and computational complexity of three syntax-related strategies: grouping, n-grams, and patterning. Only the patterning strategy gives rise to (first-order) phrase structure.

syntax, the grouping strategy groups words about the same referent together, the n-gram strategy imposes an ordering on the words in a group, and the pattern strategy introduces lexical categories and generic patterns. So each strategy integrates additional mechanisms to tackle an additional feature of constituent structure.

For each strategy, we introduce an effective procedure for the speaker and hearer that implements the strategy and show how a shared successful language arises. We also study how the resource use of a strategy (size of the search space and size of the inventory) scales with respect to the number of words in an utterance. For the first two strategies (lexicalisation and grouping) it is possible to do this in an analytic way, but for the more complex strategies (n-grams and patterns) we use a computer simulation of the strategy using a population of artificial agents playing language games and then analyse the performance. One of the fundamental components needed in all agent-based experiments is a formalism for the lexicon and the grammar. We use here Fluid Construction Grammar (FCG), which was explicitly designed for the purpose of evolutionary experiments (Steels 2011, 2012b).

2 The lexical strategy

We start from a language which does not have syntax, i.e. we assume that linguistic agents simply build utterances consisting of an unordered set of words. Given such an utterance, what is the computational complexity for the hearer to interpret it? To answer this question, we need a model of the interpretation process, which should be as simple as necessary. Let us use the well established framework of the predicate calculus to formulate this model.

We assume an ontology O consisting of a set of predicates in the form of attribute-value pairs, such as `color-green` where `color` is the attribute and `green` the value. To simplify the model, predicates are assumed to have only a single argument. Higher order phrase structure requires predicates with multiple arguments but this falls outside the scope of the present paper.

A *situation model* W_s is equal to a set of facts $W_s = \{f_1, \dots, f_n\}$, where each fact $f_i = p_j(o_k)$ is a proposition stating that a predicate p_j is true for an object o_k in the present situation s . The *object-description* of an object o_k in a situation model W_s is equal to the set of facts that are valid for the same object o_k .

We assume an indefinite number of attributes a and values v . We also assume, for the sake of the argument, that v is constant for all attributes and that all

attributes apply to all objects. Then the total number of possible distinct object descriptions in all possible situation models is equal to:

$$\sum_{n=1}^a \binom{a}{n} \cdot v^n = (v+1)^a - 1$$

(1)

We also assume that there are between 1 and 3 objects in each situation model and that an object description has between 1 and *a* facts, where *a* is the number of attributes.

In the computer simulations reported later, the set of attributes and values is entirely abstract, but in order to illustrate the present paper, we use a more intuitive ontology with 3 attributes that each have 2 values: `color` with values `red` and `green`, `shape` with `cube` and `sphere`, and `size` with `big` and `small`. For this simplified example, the set of possible object descriptions is 26. Some example object-descriptions are: {`color-green(o1)`}, {`color-green(o1)`, `shape-cube(o1)`}, {`color-red(o2)`, `shape-cube(o2)`}, etc. An example situation model involving two objects *o₁* and *o₂* is *W_{sl}* = {`color-green(o1)`, `size-big(o1)`, `shape-cube(o1)`, `color-red(o2)`, `shape-cube(o2)`}.

Next we assume that there is a set of words *w₁*, ..., *w_m* in the shared lexicon of all linguistic agents, where each word *w_d* introduces a predicate *p_e* and a variable argument ?*v_p*, as in *w_d* = *p_e*(?*v_p*).¹ For the sake of simplicity, we assume that each word introduces only a single predicate.

In the computer simulations reported later, the lexicon is randomly generated, with words like “vapola”, “xetufe”, etc. But to make the exposition easier to follow, we will use Spanish words for the object language and English words for the predicates, using the following mini-lexicon:

<i>word</i>	<i>meaning</i>	<i>word</i>	<i>meaning</i>	<i>word</i>	<i>meaning</i>
verde	color-green(?x)	pequeño	size-small(?y)	cubo	shape-cube(?v)
rojo	color-red(?z)	grande	size-big(?u)	esfera	shape-sphere(?w)

Now we can define a *lexical strategy*, by specifying the behavior of the speaker and the hearer in playing a language game in which the speaker expresses to the hearer all the facts in a shared situation model. We assume a population of agents and each agent can both be speaker and hearer.

1 Variables are denoted by symbols preceded by a question mark.

Lexical Strategy

Speaker: I look up in my lexicon the minimal set of words that expresses the facts in the situation model and transmit these words in any order to the hearer.

Hearer: I recover predicates $p_1(?v_1), \dots, p_n(?v_n)$ by looking up the words in my own lexicon. Then I try to find bindings for all the variables, such that for every word w_d in the utterance with meaning $p_e(?v_p)$, there is a fact in the world model $f_i = p_j(o_k)$ where $p_j = p_e$ and $?v_i$ is bound to o_k .

If the hearer cannot find a consistent binding-list, the utterance cannot be interpreted by the hearer and the game is a failure. If there is more than one possible binding-list the utterance is semantically ambiguous and the language game fails as well.

For example, to express the two object descriptions $\{\text{color-green}(o_1), \text{size-big}(o_1), \text{shape-cube}(o_1)\}$; and $\{\text{shape-cube}(o_2)\}$ according to the lexicon given in the above table, the speaker produces “verde cubo cubo grande”. The hearer recovers the predicates $\text{color-green}(?x)$, $\text{size-big}(?u)$, $\text{shape-cube}(?y)$, $\text{shape-cube}(?y_2)$. When matching this to the world model W_{s1} , the hearer finds the following consistent binding-list: $\{?x = o_1, ?u = o_1, ?y = o_1, ?y_2 = o_2\}$. A second possibility is $\{?x = o_1, ?u = o_2, ?y = o_1, ?y_2 = o_1\}$.

It often happens, as in this example, that several variables bind to the same variable. It is useful to represent this information in advance of attempting to match the predicates against the situation model, because that considerably reduces the computational complexity of the interpretation process. This can be done by replacing all the variables that should bind to the same object by a single one, as in $\{\text{color-green}(?x), \text{size-big}(?x), \text{shape-cube}(?x), \text{shape-cube}(?y)\}$. We call a list of predicates with equalised variables, a *predicate combination* and we call a set of predicate combinations that covers the whole utterance a *predicate combination hypothesis* or simply *hypothesis*.

It is now possible to define in more detail an effective procedure that the hearer can use to interpret the set of predicates obtained after lexicon look up. The procedure has two steps:

1. *Generate:* This process generates all possible hypotheses for the predicates provided by the words in an utterance. We call this the syntactic component.
2. *Test:* This process filters out those hypotheses for which there exists a consistent binding-list for the current situation model. We call this the semantic component.

For example, for the utterance “verde esfera grande” and the lexicon given earlier, the hearer’s generator would come up with the following set of hypotheses:

- i. color-green(?x), shape-sphere(?y), size-big(?z)
- ii. color-green(?x), shape-sphere(?x), size-big(?y)
- iii. color-green(?x) shape-sphere(?y), size-big(?y)
- iv. color-green(?x), shape-sphere(?y), size-big(?x)

If the world contains the facts: color-green(o_1), shape-sphere(o_1) and size-big(o_2), then only hypothesis (ii) would lead to a valid binding-list, namely $?x = o_1$ and $?y = o_2$.

The hypothesis generator progressively builds up a search tree of which the leafs are hypotheses. It starts with a single hypothesis and every time a new predicate combination is initialised a new hypothesis branches off. There are two possibilities:

1. **Init**: starts a new predicate combination and hence a new branch in the hypothesis tree.
2. **Extend**: links a word into an existing predicate combination by making the variables of the word's predicate equal with those of the other predicates in that combination.

When these steps are applied exhaustively, they produce all possible hypotheses which are then handed over to the **test** process that matches hypotheses against the situation model.

The set of all hypotheses is called $\mathcal{H}$ (for hypothesis set) and the set of all hypotheses for which a consistent set of bindings can be found in the world model is called $\mathcal{M}$ (for meaning set). As long as $\#(\mathcal{H}) > 1$ we say that there is syntactic ambiguity and as long as $\#(\mathcal{M}) > 1$ there is semantic ambiguity.

How does $\mathcal{H}$ scale in relation to the number of words in an utterance? As already shown in an earlier paper (Beuls and Steels 2013), the number of hypotheses H_n is equal to the number of partitions of the set D of words in an utterance of size n , where a partition of D is defined as a set of nonempty, pairwise disjoint subsets of D whose union is D . H_n is known as the Bell number and defined using the following equation (Bell 1938):

$$H_{n+1} = \sum_{k=0}^n \binom{n}{k} H_k \quad (2)$$

with $H_0 = H_1 = 1$. So H_n , the cardinality of $\mathcal{H}$, grows double exponentially with the number of words in the utterance (see Figure 4). It means that the sentence you are now reading (which contains 20 words) generates 51,724,158,235,372 partitions and hence possible hypotheses.

So this is the core of the problem. Without some way to limit the number of hypotheses, the lexical strategy will not be effective for utterances which contain more than a few words. We argue that this is the reason why syntax is used in human languages. Indeed, if information can be provided to the hearer to restrict the set of hypotheses or to consider as quickly as possible only those that are relevant, then the combinatorial explosion in the interpretation process can get drastically reduced and semantic ambiguity avoided.

3 The grouping strategy

A first step towards a reduction of the search space can be achieved if the syntactic component would somehow be able to group the elements which refer to the same object together. We call this the *grouping strategy*. Grouping is a first very minimal form of syntax.

For example, to express the two object descriptions {color-green(o_1), size-big(o_1), shape-cube(o_1)} and {shape-cube(o_2)} with the words “verde”, “cubo”, “grande” and “cubo”, the speaker could put “verde” “cubo” and “grande” together as in “verde cubo grande cubo”, instead of in some random order, such as “verde cubo cubo grande”. The hearer is still in doubt about the boundaries of word groups, but at least some combinations, for example between “verde” and “grande” are now excluded.

A hypothesis can be represented by a tree structure with units grouping the words together that form a single predicate combination. Without the grouping strategy, the lines in this structure potentially cross (as in Figure 2, left), but with the grouping strategy they no longer do so. Note that there is not yet any ordering of the words within a group. For example, “verde grande cubo cubo” is also a possibility for the same meaning.

Fig. 2: The grouping strategy puts the words together that are part of the same predicate combination. On the left is an utterance shown based on the pure lexicalisation strategy and on the right an utterance using the grouping strategy.

Two questions now need to be addressed: (i) What is an effective procedure for speaking and comprehending that implements such a strategy? and (ii) What is the gain in computational complexity compared to the lexical strategy?

Grouping Strategy

Speaker: I look up in my lexicon the minimal set of words that expresses the facts in the situation model, group all the words that express predicates about the same object, and transmit these groups in any order to the hearer.

Hearer: I build up the search tree of possible hypotheses (as in the lexical strategy) taking into account the word order constraint (see Figure 3):

- 1. *Init* starts a new predicate combination.
- 2. *Extend* links a new word into an existing predicate combination but only if the word is on the right boundary of the right most word in this combination.

The tree is developed in a depth-first fashion and intermediate nodes as well as leafs of the tree are matched against the situation model in order to eliminate nodes as quickly as possible.

As before, if the hearer cannot find a consistent binding-list the game is a failure and if there is more than one possible binding-list the utterance is semantically ambiguous and the language game fails as well.

Fig. 3: The grouping strategy progressively builds a tree where the leaf nodes are equal to hypotheses. When a word is encountered, it is both added to an existing hypothesis (extend), or a new combination is started, branching off from the hypothesis built so far (init).

Fig. 4: Diagram comparing the growth in the size of the possible hypotheses for the lexical strategy, which is equal to the Bell number, and for the grouping strategy. The x-axis plots n , the size of the words in an utterance, and the y-axis the number of possible hypotheses H_n on a logarithmic scale.

The computational complexity of the grouping strategy can be derived in a straightforward way from this algorithm. The number of possible hypotheses H_n is now 2^{n-1} with n the number of words in the utterance. The growth of hypotheses in relation to the utterance length is significantly less than the Bell number (see Figure 4) but it is still exponential and therefore does not really allow yet a significant scaling up of the size of the utterance. For example, for a sentence of 20 words we still have 524,288 possible combinations.

4 The n-gram strategy

The grouping strategy introduces the first syntactic device, namely grouping those words together that are about the same referent. But we have just seen (see Figure 4) that the gain is far from enough to explain how human languages have been able to scale up utterances beyond a few words. The next logical step towards more syntax is to introduce ordering among the words in a group, which implies that there has to be an inventory of constructions which recognises and imposes this ordering.

Ordering should help because the boundaries of a group become more delineated. For example, consider the utterance “pequeño verde esfera rojo grande cubo”. If the word sequences “pequeño verde esfera” and “rojo grande cubo” form known sequences, then the hearer can infer that there is a boundary between “esfera” and “rojo”. “verde esfera rojo” will never appear because the same attribute (color) cannot occur more than once in the same object description. “esfera rojo” (or “esfera rojo grande”) may appear unless a still more powerful strategy is used (as discussed in the next section). But nevertheless there is already some gain.

There is a second benefit. Stored sequences act like chunks. The building of a predicate combination can now happen in one step instead of several, so that the search space is shrinking. For example, “pequeño verde esfera rojo grande cubo” can be parsed in two steps rather than 5 if the hearer has constructions for the sequence “pequeño verde esfera” and “rojo grande cubo”.

A sequence of words in the inventory of an agent is called an *n-gram* in computational linguistics parlance (Manning and Schütze 1999), and we therefore call this strategy the *n-gram strategy*. N-grams are usually extracted from corpora with additional information about their frequency of occurrence. Instead of storing all possible n-grams for every utterance, agents store here only those n-grams for which the words were part of a predicate combination validated as a successful hypothesis in a situation model. This drastically reduces the inventory and keeps the set of n-grams always adapted to the situations encountered by the agents. Another difference with classical n-grams is that we use a construction grammar approach (Steels 2011) which means that not only the form (the words and how they are ordered) but also the meaning (the list of predicates with equalised variables) is stored, so that the n-gram construction can be used both in parsing and in producing.

The ordering of the words in a sequence is conventional. Hence agents need a way to negotiate which ordering to use for a particular set of words and consequently they have to store constructions for all permutations of a word sequence that are in use in the populations. These permutations are *competitors* of each other and the population should strive for a single choice in order to minimise the construction inventory and thus the cognitive effort needed to store, learn, and consult this inventory. Following a lot of earlier work on convention sharing, particularly in the context of the Naming Game, we adopt a lateral inhibition learning method for reaching a consensus (Steels 1998; de Vylder 2006). Agents associate with each n-gram a score reflecting how well entrenched the n-gram is in the population from the viewpoint of the agent.

An effective procedure for the n-gram strategy is the following:

N-gram Strategy

Speaker: I look for the constructions in my inventory that cover the largest subsets of the words that I need to convey after lexicon lookup.

⊙ If constructions are found covering the whole set of words: I pick the ones with the highest score and use them to impose an ordering on the words.

⊙ If constructions are missing for some of the words: I use the grouping strategy and then build new n-gram constructions with an initial score $\sigma = 0.5$. The semantic pole of the construction contains the predicates involved with the variables made equal. The syntactic pole contains the words involved in each predicate combination and how they were sequentially ordered.

Hearer: I look for those constructions in the inventory of n-grams that match with the sub-sequences of words in the input. The constructions with a larger scope are tried first and for those with the same scope the ones with a higher score are preferred.

⊙ If constructions are found: I use them to construct a hypothesis. If the hypothesis lead to a consistent binding-list when being matched with the situation model, I increase the score of each used construction c_i and decrease the score of its competitors c_j . The lateral inhibition learning rule is defined as follows, with $\gamma = 0.2$:

$$\sigma_{c_i} \leftarrow \sigma_{c_i} (1 - \gamma) + \gamma \quad (3)$$

$$\sigma_{c_j} \leftarrow \sigma_{c_j} (1 - \gamma) \quad (4)$$

⊙ If constructions are missing: I use the grouping constructions for the other words and then construct a new n-gram for each predicate combination, similar to the way a new n-gram construction is built by the speaker, with initial score $\sigma = 0.5$.

Figure 5 shows through a computer simulation that the lateral inhibition dynamics has the desired effect of bringing the population towards a shared set of unique n-gram constructions for each possible ordering of the words in the groups that are relevant to their world.

Figure 6 shows the time evolution of the total number of n-grams. We see the typical dynamics of lateral inhibition, also observed in the Naming Game (Steels 1998) where after an initial growth, variation is damped and the shared inventory settles (on 20 in this case). This happens very quickly after less than 2000 games (which is on average 400 per agent for a population of 10).

The maximum number of possible n-grams is equal to

$$\sum_{n=2}^a \binom{a}{n} \cdot v^n \cdot n! \quad (5)$$

Fig. 5: The x-axis plots the number of games played. At each time instant only 2 agents interact. The y-axis plots the running average of scores of all agents for each n-gram construction. The population size is 10 (but could of course be scaled up). A clear winner-take-all effect is observed as one ordering becomes dominant for each possible predicate combination.

Fig. 6: The x-axis plots the number of games played and the y-axis the average number of construction application results per agent with minimum and maximum for a series of 10 experiments. The top graph is for the n-gram strategy and the bottom for the pattern-strategy (discussed in the next section). Both lead quickly to convergence of a shared set of constructions for the situations have the agents communicate about. Clearly the pattern-strategy leads to fewer patterns and faster convergence.

Given that $a = 3$ and $v = 2$ in the present experiments, this is equal to 72. And the minimum is

$$\sum_{n=2}^a \binom{a}{n} \cdot v^n \quad (6)$$

which is 20 for the same parameter settings. Figure 6 shows that the agents do not make all possible n -grams because once they have already acquired an n -gram they do not make a new one themselves. The same figure also shows that agents settle on the optimum inventory size (namely 20), demonstrating that the proposed strategy is indeed effective and optimal.

Let us now study the computational complexity of the search process for the n -gram strategy in comparison to the grouping strategy. We use three performance measures:

1. Search space size S : This is the number of all nodes that are created by the generator. Each node is the result of the application of one construction. Many nodes are created which are not further explored, for example, because they occur already somewhere else in the search space, they could not be interpreted in the situation model or a valid solution was reached because they could be explored (assuming a depth-first search).
2. Explored search space size E : This is the number of all nodes that are effectively further expanded.
3. Depth solution path P : This is the number of nodes that were on the path towards the final solution.

Figure 7 is showing that the n -gram strategy is effective in minimising the number of constructions that agents have to apply before they find a solution. The diagrams compare the efficiency of the grouping strategy (left) with the n -gram strategy (right) for utterances of length $n = 6$. For each language game (x-axis), the number of construction applications, i.e. the size of the search space, is shown (y-axis). We see clearly that the n -gram strategy requires a smaller search space. The n -gram strategy uses the grouping strategy in the beginning but once the n -grams are there, it has consistently a much better performance.

Figure 8 compares the number of explored branches. Again we see that the n -gram strategy is more efficient compared to the grouping strategy.

Figure 9 compares the length of the branch that provides a successful interpretation of the utterance. The n -gram strategy improves slightly on the grouping strategy and is basically optimal.

Fig. 7: Search space size for a series of games using the grouping strategy (left) and the n-gram strategy (right). The utterance size is kept fixed to 6. The n-gram strategy is considerable more efficient.

Fig. 8: The number of explored branches for a series of language games using the grouping strategy (left) and the n-gram strategy (right) for utterances of size $n = 6$. The n-gram strategy is again more efficient once an n-gram inventory is in place.

Fig. 9: The number of explored branches for a series of language games using the grouping strategy (left) and the n-gram strategy (right) for utterances of size $n = 6$. The n-gram strategy is again more efficient once an n-gram inventory is in place.

5 The pattern strategy

The n-gram strategy leads to a clear improvement in terms of reducing the size of the search space and how it is traversed. On the other hand this benefit comes with a serious cost. The number of n-grams is equal to the number of possible object descriptions and (according to equation 1) this grows exponentially with the number of attributes. For example, given $v = 2$, we get 727 n-grams for $a = 6$, 6560 for $a = 8$, 59048 for $a = 10$, etc. So storing exact word combinations is not a viable strategy in the long run (even though the Google n-gram database released in 2009 stored already 4,600,926,713 n-grams compressed to 27.9 GigaByte of data).

Human languages introduce more general patterns that each can cover a large group of n-grams. The pattern is phrased in terms of categories (called lexical categories or parts of speech such as noun, adjective, adverb, etc.). We call this the *pattern strategy*. When it is applied effectively, the inventory should become restricted to manageable proportions while retaining the same advantages as the n-gram strategy, i.e. generating fewer hypotheses and shrinking the search space.

What is an effective procedure for implementing the pattern strategy? Note that agents start without any prior categories and without any patterns. So they have to create both of them and agree without any central control or telepathy

where one agent can inspect or influence the categories or patterns used by another one. There are several solutions possible, depending on how the category formation process is organised. Here we just give one example, based on coercion and reuse.

The lexicon of each agent now stores with every word w_i not only the meaning in the form of a predicate with its argument but also an associated set of categories $cat(w_i)$. A pattern p in the grammar of each agent is represented as a series of lexical categories: $p = [c_1, ..., c_n]$. A pattern $p = [c_1, ..., c_n]$ is said to *match completely* with a subsequence of words $w_1, ..., w_n$ in the input if and only if for all $1 \leq i \leq n$, $c_i \in cat(w_i)$. A pattern $p = [c_1, ..., c_n]$ is said to *match partially* with $w_1, ..., w_n$ if for all except one word, $c_i \in cat(w_i)$ for $1 \leq i \leq n$.

The pattern strategy can be defined in terms of three primitive actions: initialise, coerce and reuse:

Pattern strategy (construction of the inventory)

- *Initialise* Initially, no patterns exist and words have empty category sets. When a group has been constructed by the speaker or the hearer (using the grouping strategy discussed in Section 4), a new pattern construction is made by (i) creating a new category for each word in the group and adding this category to the category-set of the respective word in the agent's lexicon, and (ii) creating a new construction for this sequence of categories. Creating a new category means simply to create a symbol, such as *cat-51*. The meaning of the symbol is entirely determined by its role in the grammar, i.e. it defines possible positions in patterns.
 - *Coercion* is a way to minimise the number of patterns that are used. When a pattern is partially matching with a subsequence of words in the input, then the word whose category c_i is not matching with the one expected by the pattern can be coerced into filling that slot by adding the expected category c_i to the possible categories of the word. This is like coercing a noun (such as “google”) to be used as a verb (as in “she googled me on the Internet”). There is possibly more than one partially matching pattern, and in that case the pattern with the highest score is chosen.
 - *Reuse* is a way to minimise the number of categories. When a new pattern is created for a group of words (using *initialise*) and a word already belongs to some lexical categories, then the agent reuses one of the existing categories of this word in the new pattern. Instead of making a random choice, agents keep track of the frequency of use of the categories associated with a word and use the one with the highest frequency.
-

Here are examples showing the three primitives in action:

[Example 1] *Initialising new patterns*: Suppose no patterns exist yet and the words have no categories. Suppose also that the first utterance has to express {color-green(o_1), shape-cube(o_1), size-big(o_2), shape-sphere(o_2)}, so that the words “verde”, “cubo”, “grande” and “esfera” have to be expressed. After the grouping

strategy is applied, which already puts words together that are about the same object, the speaker builds the following utterance:

“verde cubo grande sfera”

There is no ordering among the words within a group. The hearer can parse these words with the grouping strategy and find the appropriate bindings. Both the speaker and the hearer now build a first pattern-construction. New categories are created for each of the words, so that the lexicon for one of the agents could be as follows:

<i>word</i>	<i>meaning</i>	<i>categories</i>
“verde”	green(?x)	{cat-1}
“cubo”	shape-cube(?y)	{cat-2}
“grande”	size-big(?u)	{cat-3}
“esfera”	shape-sphere(?v)	{cat-4}

Together with these categories two patterns are built: $p_1 = [\text{cat-1 cat-2}]$ and $p_2 = [\text{cat-3 cat-4}]$.

<i>construction</i>	<i>categories</i>
p_1	[cat-1 cat-2]
p_2	[cat-3 cat-4]

The other agent builds other internal categories (for example cat-5, cat-6, etc.) and functionally equivalent patterns, because the lexicon and grammar is local to each agent.

Both speaker and hearer store these patterns and when the same words appear in the utterance, they apply. The speaker will reproduce the same word order and the hearer will recognise the two patterns and make the variables of the predicates of the words filling the two slots in each pattern equal:

“verde	cubo	grande	esfera”
cat-1	cat-2	cat-3	cat-4
green(?x)	shape-cube(?x)	size-big(?u)	shape-sphere(?u)

[Example 2] *Coercion*: Suppose that the next utterance has to express {color-green(o_1), shape-sphere(o_1), color-green(o_2), shape-cube(o_2)}, so that the words “verde”, “esfera”, “verde” and “cubo” have to be expressed. None of the two patterns so far can match completely. However there are two partial matches. They can be turned into complete matches by coercing “cubo” to belong to cat-4 and “esfera” to cat-2, so that the lexicon looks as follows:

word	meaning	categories
“verde”	green(?x)	{cat-1}
“cubo”	shape-cube(?y)	{cat-2, cat-4}
“grande”	size-big(?u)	{cat-3}
“esfera”	shape-sphere(?v)	{cat-4, cat-2}

[Example 3] *Reuse*: Suppose that the next utterance has to express color-green(o_1), shape-cube(o_1), size-big(o_1), ... so that the words “verde”, “cubo” and “grande” now form a group. None of the patterns match completely. The pattern strategy used here is not sophisticated enough to extend patterns, so a new pattern is made. However, because all the words involved have already categories, these can be reused to make a new pattern p_3 , so that the grammar is now:

p_1	[cat-1 cat-2]
p_2	[cat-3 cat-4]
p_3	[cat-1 cat-2 cat-3]

In addition to these primitive operations, agents still need the lateral inhibition strategy, already used for n-grams, in order to negotiate which patterns become part of the shared language. For example, it is perfectly possible that some agents express color-green(o_1), shape-cube(o_1) and size-big(o_1) using “cubo grande verde” instead of “verde cubo grande”, because there is no global agency that determines how every agent has to speak.

Patterns are in competition with each other when they express exactly the same set of predicates. The same equations as used earlier (equations 3 and 4) govern the learning rule. When an agent encounters a new pattern, he will store it, even if it is different from the one already in his own grammar, with initial score $\sigma = 0.5$. When a pattern p_j was part of a successful solution (after competition against other patterns), the score of p_j is increased and its competitors decreased.

Experimental results show unequivocally that the semiotic dynamics generated by these primitive operations and the lateral inhibition learning rule lead to a shared grammar within a population of agents (see Figure 6). Unexpectedly, the pattern-strategy leads to fewer patterns compared to the n-gram strategy and to faster convergence.

Figure 10 shows that the categories of the individual agents progressively cluster together, based on the patterns in which they play a role, and despite the fact that they were not provided a priori. A multi-dimensional scaling plot (Cox and Cox 2001) is used. The dimensions are based on vectors for each category c_i where a vector consists of a series of values $c_i = [\dots v_{w_j} \dots]$ for each word w_j in the lexicon. $v_{w_j} = 1$ if w_j belongs to c_i and 0 otherwise. The categorial dimensions are then reduced to a 2-dimensional plot using standard algorithms. Figure 10 is based on a snapshot in the evolution of the pattern grammar. We see clearly that clusters of categories are emerging in the population.

The efficiency of the pattern strategy is examined for the same measures as used earlier to compare the grouping strategy and the n-gram strategy. Figure 11 (left) shows the size of the search space (to be compared with Figure 7) and Figure 11 (right) shows the number of explored patterns (to be compared with Figure 8). We see that the search space is bigger for the pattern strategy compared to the n-gram strategy because more patterns apply in each case, however the number

Fig. 10: A two-dimensional MDS plot of the different categories of the agents. Clusters emerge, showing that agents have developed similar lexical categories.

Fig. 11: Left: Search space size for a series of games using the pattern strategy. Right: number of explored branches.

of explored branches is equally effective. Figure 12 shows the length of the path that lead to a winning solution, i.e. an hypothesis that could be matched successfully against the situation model yielding a consistent binding-list. We see that the same performance is reached as with the n-gram strategy. The gain of the pattern-strategy lies in terms of its reduction of the construction inventory. This is already reduced to half for the (very low values of) the parameters used here (see Figure 6) and the gain can only increase with larger lexicons and ontologies.

6 Conclusions

This paper uses a computer science approach for explaining why human languages use phrase structure. This approach constructs and studies a series of strategies for building languages that progressively approach the characteristics of languages with phrase structure. At each step the computational complexity is examined and shown to improve for critical parameters, such as for scaling up the length of the utterance or reducing the size of the inventory. The present study is far from exhaustive. For example, there are alternative strategies used by natural languages for dealing with the combinatorial explosions that occur with purely lexical languages in the form of agreement systems. They also allow a reduction of syntactic and semantic ambiguity, and the same computer science methodology has been applied to examine why they have the properties we empirically observe (Beuls and Steels 2013).

Fig. 12: The length of the solution branch for the pattern strategy for a series of games.

There are also many refinements or alternative strategies possible for improving the strategies reported here. Here are just two examples:

1. It is possible to optimise the effectiveness of the n-gram and pattern strategy by using a border heuristic. This heuristic operates when updating the score of n-grams or patterns. When a successful hypothesis is found consisting of different predicate combinations (each based on applying a construction), then alternative constructions which cross the borders of these combinations can be inhibited. For example, given the word sequences “pequeño verde esfera” and “rojo grande cubo” then “esfera rojo” or “esfera rojo grande” can be inhibited in favor of “rojo esfera”, so that “esfera rojo” would not trigger in the future for the same word sequence. The effectiveness of this strategy is discussed in a forthcoming paper. Many other variations and heuristics can be explored as well.
2. The pattern strategy allows that a word belongs to many different categories so that it can appear in many different patterns, the same way “bike” can both be a noun (“she bought a new bike”) or a verb (“they bike home”). However this generates increased syntactic ambiguity. Many languages have therefore introduced morphological marking to differentiate the use of the same predicate and the same stem in different syntactic contexts. For example, “-ly” marks that an adjective is adverbially used as in “slow” versus

“slowly”; “-ment” turns a verb into a noun as in “govern-ment” or “pay-ment”. Marking categories morphologically reduces the syntactic search space because it restricts the number of possible patterns for a word in the input, while avoiding multiplication of words in the lexicon.

In any case, the framework introduced in this paper and the strategies we presented demonstrate clearly the main claim of the paper, namely that syntax is able to dampen syntactic and semantic ambiguity and hence lead to a language with greater communicative precision and less cognitive effort.

Acknowledgment: The authors jointly worked on the theory and design of the strategies. EGC focused on the implementation of the computational simulations and LS wrote the paper. Research was conducted at the Institut de Biologia Evolutiva in Barcelona (UPF-CSIC). The authors gratefully acknowledge funding from ICREA (for LS) and from the FP7 EU Project Insight (for EGC). They are indebted to researchers from the Sony Computer Science Laboratory in Paris (particularly Michael Spranger and Remi van Trijp) and the VUB Artificial Intelligence Laboratory in Brussels (particularly Katrien Beuls and Pieter Wellens) for technical suggestions during the implementation.

References

- Cox, Trevor F. & Michael A. A. Cox. 2001. *Multidimensional Scaling*. New York: Chapman and Hall.
- Bell, Eric Temple. 1938. The iterated exponential integrals. *The Annals of Mathematics* 39.
- de Vylder, Bart & Karl Tuyls. 2006. How to reach linguistic consensus: A proof of convergence for the Naming Game. *Journal of Theoretical Biology* 242. 818–831.
- Manning, Chris & Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Boston: MIT Press.
- Smith, Kenny, Simon Kirby & Henry Brighton. 2003. Iterated learning: a framework for the emergence of language. *Artificial Life* 9(4). 371–386.
- Steels, Luc. 1997. Synthesising the Origins of Language and Meaning Using Co-evolution, Self-organisation and Level formation. In Hurford, Jim, Chris Knight & Michael Studdert-Kennedy (eds.), *Approaches to the Evolution of Language: Social and Cognitive bases*. Edinburgh: Edinburgh University Press.
- Steels, Luc. 1998. The Origins of Ontologies and Communication Conventions in Multi-Agent Systems. *Journal of Agents and Multi-Agent Systems* 1(2). 169–194.
- Steels, Luc (ed.). 2011. *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.
- Steels, Luc (ed.). 2012. *Computational Issues in Fluid Construction Grammar* (LNAI 7249). Berlin: Springer-Verlag.

- Beuls, Katrien & Luc Steels. 2013. Agent-Based Models of Strategies for the Emergence and Evolution of Grammatical Agreement. *PLOS ONE* 8(3). <http://dx.plos.org/10.1371/journal.pone.0058960> (accessed 18 March 2013).
- Steels, Luc. 2012. Self-organization and selection in language evolution. In Steels, Luc (ed.), *Experiments in Cultural Language Evolution*. 1–37. Amsterdam: John Benjamins.
- Steels, Luc (ed.). 2012. *Experiments in Cultural Language Evolution*. Amsterdam: John Benjamins.