Language evolution without natural selection: From vocabulary to syntax in a population of learners

Simon Kirby
simon@ling.ed.ac.uk

1 Introduction

How can we explain the origins of our uniquely human compositional system of communication? Recently there has been a resurgence of interest in this and other questions surrounding the evolution of human language and the origins of syntax in particular (Bickerton 1990; Pinker & Bloom 1990; Newmeyer 1991; Hurford *et al.* 1998).[1] Much of this is due to an explicit attempt to relate models of our innate linguistic endowment with neo-Darwinian evolutionary theory. These are essentially functional stories, arguing that the central features of human language are genetically encoded and have emerged over evolutionary time in response to natural selection pressures.

In this paper I put forward a new approach to understanding the origins of some of the key ingredients in a syntactic system. I show using a computational model that compositional syntax is an inevitable outcome of the dynamics of observationally learned communication systems. In the model described, a population of simple learning mechanisms train each other to produce utterances. The "language" in the population develops from a simple idiosyncratic vocabulary with limited expressive power and little coordination among members of the population, to one with nouns and verbs, word order expressing meaning distinctions, full compositionality, all the meaning space covered and complete coordination. All this happens without any selection of learners — indeed without any biological change — or any notion of function being built into the system.

This approach does not deny the possibility that much of our linguistic ability is genetically coded and may be explained in terms of natural selection, but it does highlight the fact that biological evolution is by no means the only powerful adaptive system at work in the origins of human language.

In the following section, the biological approach to explanation is outlined, and reasons given for why we might wish to look for an, at least partial, alternative. Section 3 sets out the computational approach, showing how the recent flurry of activity in the simulation of populations and techniques for modelling learning can be combined to approach an adequate model of this kind of evolution. After presenting the results of this model, the paper suggests understanding the behaviour of the system in terms of the replicator dynamics of

[1]This research was carried out at the Collegium Budapest Institute for Advanced Study, and the Language Evolution and Computation Research Unit in the Department of Linguistics at Edinburgh, funded by ESRC grant R000236551. Central to the success of the work described here have been the discussions and technical assistance of both Mike Oliphant and Jim Hurford. The paper has also benefited from the input of Bob Berwick, Ted Briscoe, and the audiences at the Collegium Budapest workshop on the Evolution of Syntax, the MIT AI lab colloquium series, and the Edinburgh AI department Evolutionary Computation talk series. Of course, any errors that remain are solely due to the fact that you are reading a draft copy of this paper. Please address any comments to `simon@ling.ed.ac.uk`.

generalisations. Section 6 shows how the computational model relates to linguistic theory, including universals, frequency effects, and creolisation. The final sections examine the generality of the results and outline an evolutionary theory of learning that might be applied to other domains.

2 The origins of syntax

In their influential paper, Pinker & Bloom (1990) argue that an analysis of the design features of human language, and of syntax in particular, leads to the conclusion that the best way of understanding their origins is as biological adaptations. The central questions that should be asked in their view are:

> "Do the cognitive mechanisms underlying language show signs of design for some function in the same way the anatomical structures of the eye show signs of design for the purpose of vision? What are the engineering demands on a system that must carry out such a function? And are the mechanisms of language tailored to meet those demands?" (Pinker & Bloom 1990:712)

The design features that they are interested in include: major and minor lexical categories, major phrasal categories, phrase structure rules, linear order rules, case affixes, verb affixes, auxiliaries, anaphoric elements, complementation, control and wh-movement. They claim that these features of grammars – which from their perspective form part of the innate endowment of humans and directly constrain the class of learnable languages – these features work together to make "communication of propositional structures" possible. For example, the existence of linear order, phrase structure and major lexical categories together will allow a language user to "distinguish among the argument positions that an entity assumes with respect to a predicate" (p. 713), suggesting that their presence in human languages requires a biological/adaptationist explanation.

Of course, there have been many authors (see, e.g. Hurford 1998 for a recent review) who have argued that it is useful to look at syntax as a product of natural selection — Newmeyer (1991); Newmeyer (1992), for example, looks in detail at the features of the "Principles and Parameters" model of syntax and gives them an evolutionary explanation. The reasons for this are clear, as Pinker & Bloom (1990:707) point out: "Evolutionary theory offers clear criteria for when a trait should be attributed to natural selection: complex design for some function, and the absence of alternative processes capable of explaining such complexity. Human language meets these criteria."

In this paper, I will not attempt to deny the logic of this argument, nor come up with an alternative explanation for all the design features that Pinker and Bloom list. The main message I wish to put across is that, for at least some features of syntax, there are in fact "alternative processes capable of explaining such complexity", and that some of the qualitative evolution of human language proceeded without natural selection. The kind of evolution we will be looking at is not biological, but relies on a notion of *languages* as complex adaptive systems just as the Pinker and Bloom explanation relies on the notion of the *language faculty* as a complex adaptive system (Kirby 1998; Kirby 1997b; Christiansen 1994; Deacon 1997; Kirby 1997a; Briscoe 1997; Gell-Mann 1992).

The particular feature of syntax that will be explored in this light — and one which subsumes many of Pinker and Bloom's list — is *compositionality*. Cann (1993:4) gives the following definition of the principle of compositionality, a universal of human language:

> "The meaning of an expression is a monotonic function of the meaning of its parts and the way they are put together."

The *Pergamon Encyclopedia* gives two possible definitions of compositional:

> "1. Esp. in ... phrase structure grammars, hierarchical; larger units being composed of smaller units, as a sentence of clauses, a word of morphemes, etc. 2. ... of the meaning of a phrase, etc., composed of the meaning of its constituent parts."(Asher 1994:5104)

These definitions make clear that, although compositionality is often taken to be a property of semantics, it is actually a property of the system that links forms and meanings. I will define such a system as *syntactic* if it behaves compositionally.

3 A computational approach

If we are to fully understand the ways in which a learned, culturally transmitted, system such as language can evolve we need some sophisticated population models of learners. Simple theorising about the likely behaviour of complex adaptive systems is not good enough. As Niyogi & Berwick (1997) point out, our intuitions about the evolution of even simple dynamical systems are often wrong. Recently many researchers have responded to these problems in tackling the origins of human language by taking a *computational* perspective (for example, Hurford 1989; Hurford 1991; MacLennan 1991; Batali 1994; Oliphant 1996; Cangelosi & Parisi 1996; Steels 1996; Kirby & Hurford 1997; Briscoe 1997).

This paper follows on from this line of work, and also borrows from language learning algorithms developed in computational linguistics (namely, Stolcke 1994) in order to see if a significant portion of the evolution of syntax can proceed without biological change. In some ways, this work is a logical extension of the work of Batali (1997) who simulates a population of recurrent neural networks (Elman 1990). We will return to his work in a later section.

3.1 Features of a desirable model

In order for it to be a successful model of the cultural adaptation of language, we need the computational simulation to have a set of key features. These set out our minimum requirements. In general, we wish to make the model as simple as possible initially, and see if the complex behaviour that we are looking for emerges without extra assumptions.

1. Individuals that *observationally learn*. In other words, all the knowledge in the population is learned by individuals observing other's behaviour. In this way, as Steels's (1997) suggests in his review of the literature, individuals have "limited rationality" in that they do not have direct access to each other's internal states, nor to any structural information that cannot be directly observed from language use.

2. A spatially organised population of individuals. This requirement fulfils Steels's (1997) "distributed systems constraint" – that in these kinds of simulations, no single individual in the population should have a complete view of the behaviour of all the other individuals.[2]

3. A gradual turnover of members of the population over time. By ensuring that members of the population are not "immortal" we can see that there is true historical/cultural transmission of knowledge through the system.

4. No selection of individuals. In order to show that biological evolution is not a factor in the results of the simulation, the "death" of members of the population should be completely random and not related in any way to their success at communication.

5. Initial non-linguistic population. Those that make up the initial population should have no communication system at all. This means that any biases that emerge in later states of the simulation are purely a product of the agents and the population model.

3.2 Components of the model

The remainder of this section describes a simple model that fulfils the requirements listed above. Of course, this should be considered only one of a class of potential models. A later section considers how general the results achieved using this specific set up might be.

Some of the details of the model that might be required for replication are left to the appendix, but the central features of the system are explored here.

3.2.1 Semantics

The semantic space in the simulation sets out what the agents will talk about. In other words, the space defines the set of meanings that may potentially be paired with signals in particular utterances. The meanings used in this paper are made up of an *agent*, *patient*, and *predicate* part. Essentially, every meaning that the individuals will be prompted to express will be about "who does what to whom".

The meanings are each a triple of attribute-value pairs, one attribute for agent, patient and predicate. The values of the first two attributes vary over five "objects", and the value of the predicate attribute varies over five "actions". An extra constraint that is introduced to make the induction simpler (see appendix) is that in no one meaning can agent and patient be the same object. Therefore there are 100 possible meanings in this semantic space. For example, $< Agent = \text{John}, Patient = \text{Mary}, Predicate = \text{Loves} >$, is a possible meaning in the system that the individuals might want to express.

This is a very simple semantic space, of course. For example, the attribute-value notation does not allow for recursive representations. The last section of this paper will return to this issue and suggest what impact introducing a richer semantics might have on the system.

[2]It should be pointed out that this simulation fulfils only two of Steels' three constraints fully. The system that evolves is not completely open, since new meanings do not evolve, only new signals.

3.2.2 Utterances

The utterances that the individuals in the simulation will use are strings of symbols taken from the set $< a, b, c, d, e >$. These can be thought of as the basic phonetic gestures available to the individuals although, unlike in real speech, there are no phonotactic constraints on combinations of gestures. There is no limit placed on the length of an utterance, and the shortest possible utterance is one symbol long. Random utterances, on the other hand, (which are introduced as noise or as completely novel innovations, as we shall see later) vary between six and ten symbols in length in the simulations reported here.

3.2.3 Grammars

The knowledge of each individual in the simulation is simply a grammar that maps meanings onto utterances and vice versa. The grammar formalism chosen is a probabilistic attribute grammar (PAG) after Stolcke (1994). This is a context free grammar enriched with statistical information and a way of introducing attribute-value pairs as a semantic part of each rule. A more formal description of this type of grammar is given in the appendix, but some simple examples are given below.

An important point to note about this representational space is that it does not bias the simulation to natural language-like grammars. The space of possible PAG's is huge; the vast majority of PAG grammars are very unnatural.

Example One Imagine a language learner who heard the meaning *John loves Mary* expressed as "ab" twenty times and "ba" ten times. The simplest grammar that that learner could end up with would look like:

```
1 ==> a [30]

2 ==> b [30]

S --> 1 2  [20]
   Agent=John
   Patient=Mary
   Predicate=Loves

S --> 2 1  [10]
   Agent=John
   Patient=Mary
   Predicate=Loves
```

The `S` symbol is the "start" symbol for the grammar. Every utterance produced by the individuals is an expansion of an `S` rule. The numbers `1` and `2` are arbitrary category names – in the simulations reported here, all the category names (other than `S`) are arbitrary numbers. The symbols `-->` and `==>` can be thought of as meaning "is made up of", the latter being reserved for terminals (i.e. categories that spell out phonetic gestures). So, an utterance can either be made up of a `1` followed by a `2`, or a `2` followed by a `1`. Similarly, a `1` is made up of an `a` symbol, and a `2` is made up of a `b` symbol. The number in square brackets is the rule statistic, reflecting the amount of evidence given to that particular rule. Finally, the equations under each `S` rule in this example specify the semantics for that particular rule.

Example Two Now, imagine a different language in which *John loves Mary* is expressed as "abccc" and *John loves Tunde* is expressed as "abddd". A simple grammar for this language (ignoring the statistics on the rules for simplicity), could look like:

```
1 ==> a

2 ==> b

3 ==> c

4 ==> d

5 --> 3 3 3
   0=Mary

5 --> 4 4 4
   0=Tunde

S --> 1 2 5
   Agent=John
   Patient=(2).0
   Predicate=Loves
```

Firstly, notice that there is only one `S` rule, but it can be used to express two meanings. This is because there are two intermediate words `ccc` and `ddd` which themselves mean *Mary* and *Tunde* respectively. The whole utterance is made up of a `1` (which is an `a`) and then a `2` (which is a `b`), followed by a `5` which can either expand to `3 3 3` *or* `4 4 4` (which, in turn expand to `ccc` or `ddd`).

The semantics of the whole utterance is *composed* by taking the meaning of the `0` feature of the `5` word and assigning that to the `Patient` feature of the whole utterance. The `0` feature is simply an arbitrary feature name — a placeholder for the semantic information, if you like. (If we were to give these features and categories names instead of numbers, we might call the category `5` "noun" and the feature `0` "person".) The line `Patient=(2).0` in the semantics is what assigns `Patient` the value of the `0` feature. The number in brackets tells us which of the categories on the right hand side of the rule contains the semantic information we need. The convention used here is that this number is an index into the right hand side categories, starting from zero. We can think of the right hand side of the `S` rule in this example as having implicit indices: $1_{(0)}\ 2_{(1)}\ 5_{(2)}$. This allows us to always be able to explicitly refer to one particular right hand side category.

3.2.4 Producer

Given a grammar, each individual must have some way of producing utterances for a particular meaning. This is actually trivial to do using standard computational linguistics techniques. This simulation uses a top down, left-to-right generation algorithm. The production is *deterministic* because, given two possible expansions of a category, the producer always searches the one that has the higher statistic in the grammar. In other words, rules that have been given more weight statistically are preferred. This means that if the grammar has two ways to express one meaning, then only one of those ways will ever be used.

3.2.5 Inventor

Since one of our requirements for a model is that the individuals in the initial population have no knowledge of language, some way is needed of new forms being produced, otherwise no-one would ever be able to say anything. This is where the invention algorithm comes into play. It is based on the assumption that occasionally individuals, even though they have no normal way in which to express a certain meaning, will nonetheless produce some invented string of symbol.

There are different ways in which this might be done. The simplest approach is to produce a completely random string of symbols. Another possibility, suggested by James Hurford, is to break down the meaning that is to be expressed into its atomic components, and then try to "synthesise" a symbolic representation of the sum of those components, perhaps by checking a lexicon for any matches to these atomic meanings. So, for example, if an individual was trying to express $< Agent = \text{Zoltan}, Patient = \text{Mike}, Predicate = \text{Knows} >$, then Hurford's technique would check to see if there was a way to say "Zoltan", "Mike" and "Knows" in isolation, and put together an utterance by combining these parts.

Unfortunately, the latter approach actually builds in the central features of syntax as we have defined it, namely compositionality. It *forces* utterances to be composed synthetically of pieces that correspond to subparts of the semantic representation. Moreover, Morford & Oberst (1997) report evidence from post-critical period adults which, they argue, suggests that language evolution did not proceed by the synthesis of small components into larger syntactic units.

Given this, it would seem sensible to opt for a random invention technique. However, this is rather unrealistic for some cases. For example, imagine that you as an English speaker do not know the word for a new object that you have never seen before. It seems implausible that, if you needed to express a meaning that mentioned this object somewhere in it, you would utter a completely random string of phonetic gestures *for the whole sentence*.

What we need is an algorithm that does not introduce new structure into inventions, but equally does not throw away structure that is already part of the knowledge of the speaker. Just such an algorithm is described in the appendix. This algorithm will generate random strings where the speaker has no grammatical structure, but for meanings that can be partially expressed with a particular grammar will only randomise those parts of the string that are *known by the speaker* not to correspond to expressible meaning.

3.2.6 Inducer

The most important part of the model is the grammar inducer. This allows the individuals to build grammars on the basis of example meaning-form pairs provided to them by their spatial neighbours in the population. What was required to make this simulation work was a computationally cheap way of determining the most appropriate PAG which is consistent with the input that the individual experiences. There are several ways which might be appropriate suggested by the literature in computational language learning (see, for example, Charniak 1993; Briscoe 1997 for review), but they require some form of constrained search through a space of possible grammars. This search process was deemed too costly for the resources available, so a different strategy was required.

Stolcke's (1994) grammar inducer is particularly interesting to us since it works with PAGs. Essentially, the induction algorithm works in two stages for each utterance received, *incorporation* and *merging*:

Incorporation On receiving a meaning-form pair, the algorithm immediately builds a grammatical model for that pair which makes no generalising assumptions about it.[3] This is best illustrated with an example.

Input meaning: $< Agent = \text{Mike}, Patient = \text{Zoltan}, Predicate = \text{Knows} >$

Input form: *abab*

Incorporated model:

```
1 ==> a [1]
2 ==> b [1]
3 ==> a [1]
4 ==> b [1]
S --> 1 2 3 4  [1]
   Agent=Mike
   Patient=Zoltan
   Predicate=Knows
```

Merging Having built a grammatical model of a single utterance, the algorithm seeks to merge this model with the existing model for any previous utterances. Stolcke achieves this with a set of *operators* that make global changes to the grammar. An example of such an operator is "category merge". This simply takes two categories in the grammar and makes them the same. Another operator is "chunk", which takes a string of categories and replaces them in the grammar with one category, introducing a new rule into the grammar. There are four such operators needed in order to successfully merge two PAGs. They are described briefly in the appendix.

In order for Stolcke's system to be a successful inducer, the merging operators cannot simply be applied at random. The metric that is used to direct the search through the possible operator applications is provided by Bayesian learning theory, which formalises a trade-off between model complexity and data fit. Informally, the best grammar is one that accounts for the data given in the most concise fashion — essentially, the shortest grammar that still assigns the given data set a high probability.

As noted above, this kind of search for a good sequence of operator applications is too computationally costly for the simulations reported here. Instead, a set of heuristics have been implemented which approximate this search for *minimal* grammars. Each heuristic (described in the appendix) takes a pair of rules and compares them, looking for operators that may be applied that will make the rules more similar. The heuristics have been constructed in such a way that at each step they should not introduce large numbers of new grammatical strings. For example, the left-hand side categories on two rules are not merged unless the right-hand sides of the rules are already equivalent, and they have identical semantics. If, when these operators are applied, these two rules become identical, then one of the rules is deleted from the grammar.[4] This means that the heuristics will tend to make the grammar shorter, hence approximating the induction process described by Stolcke.

[3] In the simulations reported here, induction only takes place if the hearer *cannot* already parse the input form. If he can, no induction takes place, but the rule statistics are incremented once for each rule each time it is used in the parsing operation.

[4] Furthermore, the statistic on the surviving rule is increased.

Of course, the use of heuristics is a cheap and dirty way of doing induction. Discovering successful heuristics is a laborious task and is not well motivated theoretically. Moreover, they necessarily build in constraints on the space of possible PAGs that the inducer may explore. In the simulations reported in this paper, for example, the grammars cannot become recursive. This is not a serious concern for us here, since the semantics of PAGs is not recursive anyway. However, work is currently underway to extend the simulation to allow for recursive semantics and an induction algorithm with search will probably be required.

A second, and more important, limitation on the induction algorithm without search is that it requires some help in *segmenting* the input data. This problem is analogous to finding the words in a string of text in an unknown language without spaces. This is not an insoluble computational problem, of course, but it generally requires some search through the possible segmentations. One solution to the difficulty of doing this without search is to provide the learner with bracketed examples. In these simulations, we do not go this far. Instead, individuals when speaking produce a "pause symbol", that the inducer can detect, whenever a new subtree is entered into in the generation process. As we shall see, in the initial stages of the simulation, these pauses have no effect, because all the grammars are completely flat. However, later on, when words are discovered by the population, they greatly simplify the job of the inducer.

3.3 The simulation loop

Given a computational model of an individual – made up of an inducer, a producer, and an inventor – we need to set out the ways in which a population of individuals interacts. The population in the simulations reported here is made up of ten individuals at any one point in time, organised in a ring. In other words, each member of the population has two neighbours. This is a simple way of modelling the spatial organisation that was mentioned in section 3.1.

Figure 1 and 2 show how this population is updated over time. The most important features of these two flowcharts are:

- Each individual learns only from utterances (form-meaning pairs) produced by its neighbours.
- The makeup of the population changes over time.
- Individuals are replaced entirely at random.
- The probability that one individual will hear all forms for all the possible meanings is vanishingly small.[5]

4 Results

This section looks in some detail at one particular run of the simulation described above. The behaviour of the simulation is consistent from run to run, so a careful analysis of one case is worthwhile. Any points of variability across runs are pointed out when they arise.

[5]There are 100 different possible meanings, and a maximum of 100 utterances heard by each individual. Even if an individual is lucky enough to hear 100 utterances in its lifetime, the chances that these will cover the entire meaning space are $\frac{100!}{100^{100}}$.

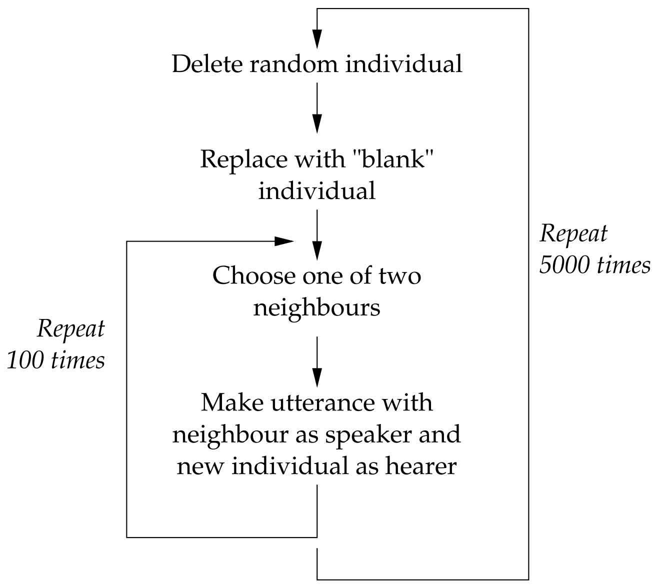

Figure 1: The main loop used in the simulations.

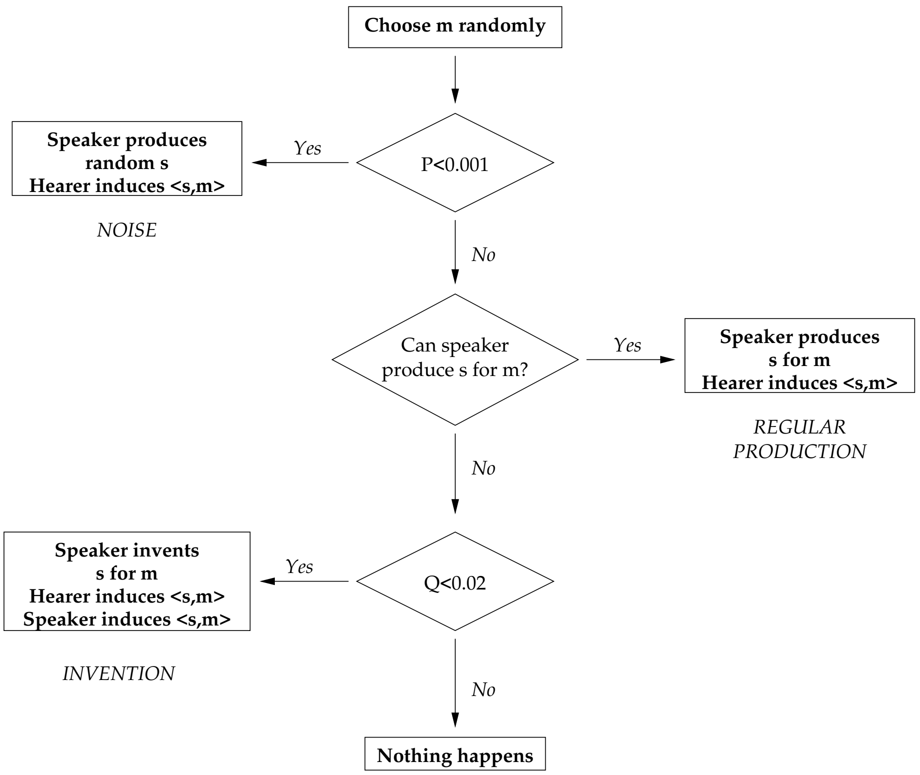

Figure 2: The flowchart for each utterance. “P” and “Q” are random numbers chosen between zero and one. They simulate the effect of noise and occasional innovation respectively. “m” is a meaning, and “s” is a string of symbols.

The initial population is made up of ten individuals, all of which have no knowledge of language — that is, they have zero grammars. The simulation loop described in figures 1 and 2 is then initialised and left to run until the behaviour of the population stabilises (after several thousand generations). Periodically, various measures of the population's behaviour and knowledge are taken:

1. **Meanings** The number of meanings that an individual can express (without invention).
2. **Size** The number of rules in an individual's grammar.
3. **Coverage** The average statistic on all the top level rules in the grammar. (This gives some measure of the generality of the grammar, since coverage is maximum when one top level rule is used for all utterances.)
4. **Grammars** The actual grammars of the individuals in the simulation can be directly inspected.

A graph of the population average of the first three of these measures over a run of half a million cycles through the simulation is given in figure 3.

The graph has been partitioned into three stages between which the population appears to make "phase transitions" into radically different types of behaviour. In particular, the relationships between the three measures graphed and also the structure of the grammars changes radically at these points. These stages are present in every run of the simulation, although the timing of the transitions is variable.

4.1 Stage I

In the first few generations[6] of the simulation run nothing much happens. No individual in the population has any grammar, so they have no way of producing utterances. Each time an individual is asked to produce a string for a particular randomly chosen meaning, they consult their grammar and discover they have no way of producing a string so they say nothing. Consequently the new individuals have no exemplars for acquisition and also end up with zero-grammars. Recall, however, that there are occasional random *invention* and *noise* events. Whenever one of these occurs, the new agent has something to internalise: a pairing of a randomly constructed string of symbols, and a randomly chosen meaning. Then, if this individual is later called upon to produce an utterance with that meaning, that same string of symbols will again appear in the input of a new learner.

This process of random invention and re-use leads to the situation that is stable throughout the first emergent stage in the simulation. The population can express only a small percentage of the meanings, using a small grammar which, nonetheless, has more rules than expressed meanings. A typical grammar drawn from a random individual at this stage in the simulation is shown in figure 4. We can understand this grammar as a simple vocabulary list for 11 of the possible 100 meanings. The first five rules in the grammar simply assign the five possible "phonetic" gestures (the letters *a* to *e*) arbitrary category labels. The remaining

[6] I will use the term *generation* for convenience here to refer to one circuit of the outer loop in figure 1. A complete turnover of the population would in fact take at least 10 generations.

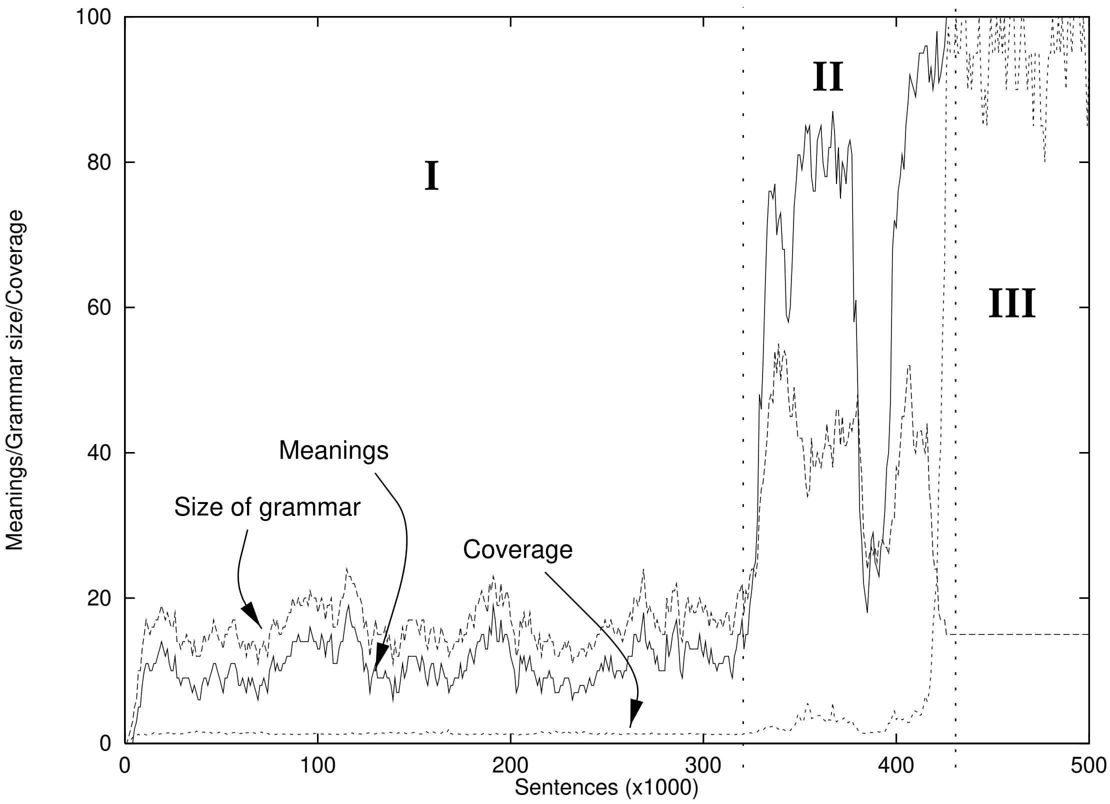

Figure 3: The population average of size, meanings and coverage over 500,000 sentences, where a "sentence" is an instruction to a speaker to produce a random meaning. Each new agent is exposed to 100 such sentences (although some may be silent if the speaker does not know how to express a particular meaning). The graph is divided into three stages signifying major "phase changes" in the grammars of the population.

```
6 ==> e [39]

2 ==> a [34]

3 ==> d [27]

5 ==> b [26]

15 ==> c [21]

S --> 6 15 3 15 6 2 2 5 3 3 2  [3]
   Agent=John
   Patient=Zoltan
   Predicate=Hates

S --> 2 3 2 5 6 6 5  [2]
   Agent=Mary
   Patient=Zoltan
   Predicate=Finds

S --> 6  [2]
   Agent=Zoltan
   Patient=Mike
   Predicate=Hates

S --> 2 3 2 5 2 5 15 15 15 6 15 2 3 15 5 15 6  [1]
   Agent=Mary
   Patient=Tunde
   Predicate=Hates

S --> 3 3 2 3 5 5 5 5 2 5 6 6 3 2 6 6 6  [1]
   Agent=Mary
   Patient=John
   Predicate=Hates

S --> 2 3 3 3 5 3 15 15 6 2 2  [1]
   Agent=Mike
   Patient=Mary
   Predicate=Hates

S --> 2 5 2 15 3 3 3 5 6  [1]
   Agent=Mary
   Patient=Tunde
   Predicate=Loves

S --> 2 15 6 2 5 5 15 6 6 6 2 5 6 6 2  [1]
   Agent=John
   Patient=Mary
   Predicate=Finds

S --> 15 6 6 2 6 5 6 6 5 15 6 15 2 5 3 6 6  [1]
   Agent=Mary
   Patient=Zoltan
   Predicate=Hates

S --> 15 6 2 3 2 6 6 2 5 5 6  [1]
   Agent=John
   Patient=Zoltan
   Predicate=Finds

S --> 3  [1]
   Agent=Zoltan
   Patient=John
   Predicate=Hates
```

Figure 4: A typical grammar from **stage I**. It is essentially a simple, idiosyncratic vocabulary with one word for each complex meaning, and limited expressive power.

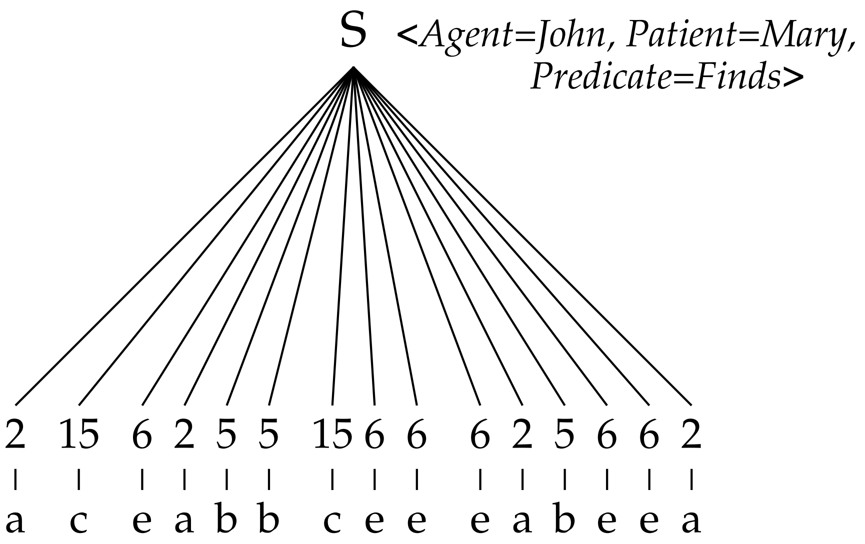

Figure 5: A **stage I** phrase structure tree showing the utterance "aceabbceeeabeea" meaning *John finds Mary*. Note the complete flatness of the structure.

11 rules act like a dictionary mapping meanings such as *John finds Mary*[7] directly onto strings of categories (such as *2 15 6 2 5 ...*) and therefore onto a string of symbols. The structure of an utterance for this individual is expressed as a tree in figure 5.

The language at this stage in the simulation is clearly fairly impoverished since only a small portion of the meanings can be expressed. Furthermore, each utterance seems to be completely idiosyncratic. For example, *John hates Zoltan* is expressed as *ecdceaabdda*, whereas *Zoltan hates John* is expressed as *d*.

4.2 Stage II

The second stage in the simulation results is marked by a sudden change in the population measures. The number of meanings covered increases dramatically as does the size of the grammar. More importantly, the number of meanings becomes greater than the number of rules in the grammar. It is clear from this that the language is no longer simply behaving as a list of unanalysed vocabulary items for complex meanings as it was in **stage I**.

In fact the grammars at this stage are far more complex and byzantine than the earlier ones. Figure 6 shows one such grammar. The details of what is going on in the language of this individual are hard to figure out. There are, however, a few points that should be noted. Firstly, there are now "intermediate" syntactic categories. There are categories for each of the basic symbols *a* to *e* as before, and also the top level category *S* — but there are also categories that rewrite to other categories. Importantly, some of these intermediate categories, or *words*, have a semantics of their own.

There are only a few of these words for atomic meanings, and they are not consistently given one particular syntactic class. So, although the category *107* expands to *ce*, *ed* and *aa*,

[7]Here I am using the English sentence *John finds Mary* as a short-hand for the feature-attribute structure $< Agent = \text{John}, Patient = \text{Mary}, Predicate = \text{Finds} >$.

```
2 ==> d [173]

17 ==> a [127]

4 ==> e [117]

149 --> 4  [52]
   11=Mary

9 ==> c [47]

5 ==> b [45]

148 --> 17  [36]

290 --> 5  [29]

107 --> 9 4  [25]
   0=John

107 --> 4 2  [19]
   0=Mary

173 --> 17 17  [16]
   12=Tunde

147 --> 2 2  [12]

107 --> 17 17 2  [7]
   0=Tunde

S --> 290 107 148 148 2  [5]
   Agent=(1).0
   Patient=Tunde
   Predicate=Finds

S --> 290 107 148 149  [5]
   Agent=(1).0
   Patient=Mike
   Predicate=Finds

S --> 149 107 107  [4]
   Agent=(1).0
   Patient=(2).0
   Predicate=Hates

S --> 2 107 2 149  [4]
   Agent=(1).0
   Patient=Mike
   Predicate=Knows

S --> 149 107 2 2 2  [4]
   Agent=(1).0
   Patient=Zoltan
   Predicate=Knows

S --> 2 9 9 148 148 9 148 148 9 2 107 148  [3]
   Agent=Zoltan
   Patient=(10).0
   Predicate=Hates

S --> 149 148 149 107  [2]
   Agent=Mike
   Patient=(3).0
   Predicate=Hates

S --> 149 2 149 2 149 2 2 290
 2 2 107 149  [2]
   Agent=Zoltan
   Patient=(10).0
   Predicate=Finds

S --> 290 290 2 173 149 149 17 107 290  [2]
   Agent=Mike
   Patient=(7).0
   Predicate=Likes

S --> 2 148 148 2 148 2 2 148 149 148 148  [2]
   Agent=Zoltan
   Patient=Mike
   Predicate=Knows

S --> 9 107 147 2  [2]
   Agent=(1).0
   Patient=Tunde
   Predicate=Loves

S --> 2 2 149 290 2 149 2 107 149 290 2  [2]
   Agent=Tunde
   Patient=(7).0
   Predicate=Knows

S --> 148 107 147 2  [2]
   Agent=(1).0
   Patient=Tunde
   Predicate=Likes

S --> 290 17 290 17 290 173 9 173 173 2 2 90  [2]
   Agent=Zoltan
   Patient=Tunde
   Predicate=Loves

S --> 9 107 149 290 9  [1]
   Agent=(1).0
   Patient=Tunde
   Predicate=Loves

S --> 148 147 2 107  [1]
   Agent=Tunde
   Patient=(3).0
   Predicate=Likes

S --> 290 149 2 107  [1]
   Agent=(1).11
   Patient=(3).0
   Predicate=Finds

S --> 2 107 147 2  [1]
   Agent=(1).0
   Patient=Tunde
   Predicate=Knows

S --> 2 148 149 2 149  [1]
   Agent=Mike
   Patient=John
   Predicate=Loves

S --> 149 149 2 148 149  [1]
   Agent=Mary
   Patient=Mike
   Predicate=Hates

S --> 2 2 148 148 2 290 2  [1]
   Agent=Tunde
   Patient=Zoltan
   Predicate=Knows

S --> 290 17 149 173 2  [1]
   Agent=Mike
   Patient=(3).12
   Predicate=Finds

S --> 290 147 2 149 2  [1]
   Agent=Tunde
   Patient=(3).11
   Predicate=Finds

S --> 2 2 2 173 2 173 2 2 107 17  [1]
   Agent=Zoltan
   Patient=(8).0
   Predicate=Hates

S --> 149 149 9 147 2  [1]
   Agent=Mary
   Patient=Zoltan
   Predicate=Loves

S --> 2 17 149 173 2  [1]
   Agent=Mike
   Patient=(3).12
   Predicate=Knows

S --> 149 147 290 290  [1]
   Agent=Mike
   Patient=Zoltan
   Predicate=Likes

S --> 2 149 149 149 149 147 149 147 149 2  [1]
   Agent=Zoltan
   Patient=Mike
   Predicate=Hates

S --> 290 107 2 149  [1]
   Agent=(1).0
   Patient=Mike
   Predicate=Finds

S --> 2 149 2 173 2  [1]
   Agent=(1).11
   Patient=(3).12
   Predicate=Knows

S --> 149 173 2 2 149  [1]
   Agent=(1).12
   Patient=Mike
   Predicate=Hates

S --> 147 107 5 5  [1]
   Agent=(1).0
   Patient=Zoltan
   Predicate=Finds

S --> 2 173 2 17 2 2 149 2 173  [1]
   Agent=Zoltan
   Patient=(6).11
   Predicate=Knows

S --> 149 17 149 2 2 2  [1]
   Agent=Mike
   Patient=Zoltan
   Predicate=Knows

S --> 2 173 2 17 149  [1]
   Agent=(1).12
   Patient=Mike
   Predicate=Knows

S --> 2 173 2 107  [1]
   Agent=(1).12
   Patient=(3).0
   Predicate=Knows

S --> 2 149 2 17 149  [1]
   Agent=Mary
   Patient=Mike
   Predicate=Knows

S --> 5 17 5 17 5 392 9 392 149 2 5  [1]
   Agent=Zoltan
   Patient=(8).11
   Predicate=Loves

392 --> 17 17  [1]

S --> 17 17 4 147 2  [1]
   Agent=Mike
   Patient=Tunde
   Predicate=Likes

S --> 2 2 173 2 5 9  [1]
   Agent=(2).12
   Patient=Zoltan
   Predicate=Loves

S --> 2 17 4 107  [1]
   Agent=Mike
   Patient=(3).0
   Predicate=Knows

S --> 9 173 2 107  [1]
   Agent=(1).12
   Patient=(3).0
   Predicate=Loves
```

Figure 6: A typical grammar from **stage II**. It has some words for atomic meanings and some compositionality, but no regular word classes or word order.

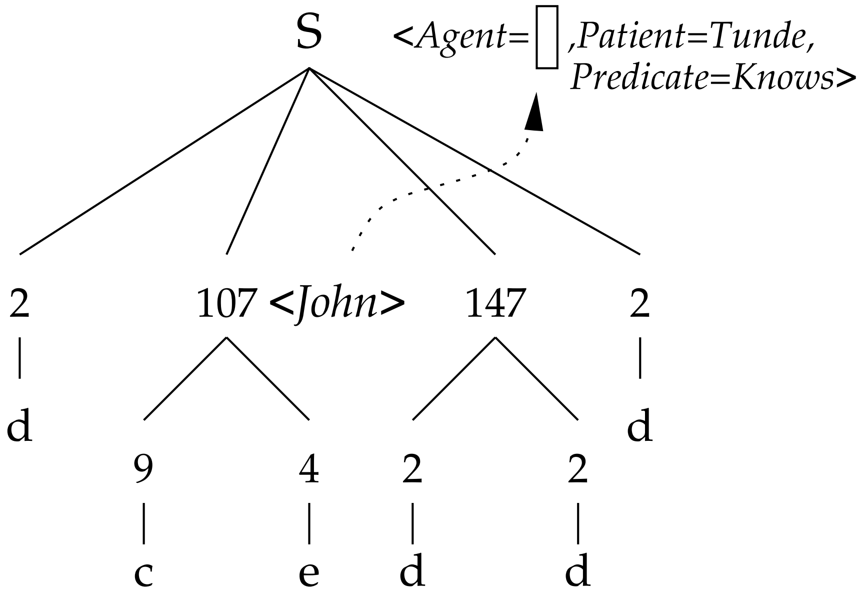

Figure 7: A **stage II** tree showing "dceddd" meaning *John knows Tunde*.

meaning *John*, *Mary* and *Tunde* respectively, there is another category (*149*) that means *Mary* and also another one (*107*) for *Tunde*. There is no word for *Mike* or *Zoltan*, nor are there any words for any of the actions. These few words are utilised by some of the top level rules to hierarchically *compose* the meaning of the whole. So, *dceddd* means *John knows Tunde* because the *ce* near the start of the string means *John*. The tree for this sentence is shown in figure 7.

In summary, the languages of this stage in the simulation have some words for atomic meanings, and some compositionality, but no regular word classes or word order.

4.3 Stage III

After a second "phase transition", the population switches into a third and final stage. The simulation has been tested on very long runs, and no significant changes occur in the population's behaviour after this point. The transition is marked by a sudden increase of the number of meanings that can be produced to the maximum value; a drop in the size of the grammars; and a large increase in the coverage measure. This last change suggests that the average generality of the top level rules in the grammar has increased enormously.

If we look at the grammar in figure 8 we can see a stark contrast with the grammars of **stage II**. Most strikingly, there is only one top level *S* rule that is used to express all the meanings in the language. This rule makes no mention of actual objects or events in its semantics. The semantics at the top level is entirely composed from the semantics of the categories on the right hand side of the rule, which is part of the reason that this rule can be so general as to cover every utterance in the language. The *S* rule is also very concise, consisting of only three categories on its right hand side, a *66* and two *62*s.

Turning our attention to these lower level categories, we can see that every atomic meaning (i.e. all the objects and all the actions) has one (and only one) word. These words are all only two symbols long, which, given the constraint on the symbol inventory that the

```
2 ==> c [205]

3 ==> e [167]

10 ==> a [120]

4 ==> d [108]

S --> 66 62 62  [100]
   Agent=(2).0
   Patient=(1).0
   Predicate=(0).4

62 --> 3 2  [49]
   0=Mary

62 --> 2 10  [40]
   0=Zoltan

62 --> 10 4  [39]
   0=Mike

62 --> 2 2  [38]
   0=Tunde

62 --> 4 3  [34]
   0=John

66 --> 10 3  [24]
   4=Likes

66 --> 2 3  [22]
   4=Loves

66 --> 3 3  [19]
   4=Knows

66 --> 4 10  [18]
   4=Finds

66 --> 2 4  [18]
   4=Hates
```

Figure 8: A typical **stage III** grammar, with highly regularised syntax, nouns and verbs, fixed word order, and full compositionality.

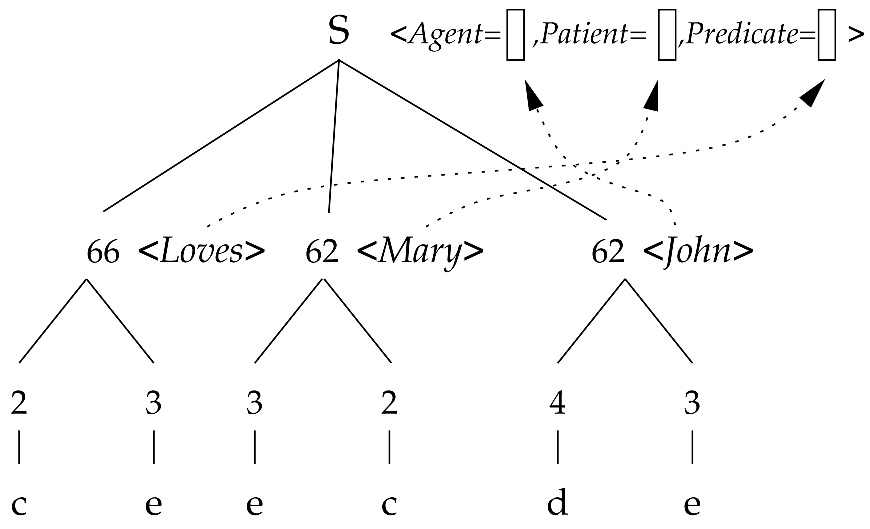

Figure 9: A **stage III** tree showing "ceecde" meaning *John loves Mary*.

individuals have, turns out to be the shortest that the words can be and still cover all ten atomic meanings. Interestingly, although the languages of the previous stages used all five symbols *a* to *e*, this language has dropped the *b* symbol. The four symbols that are left are the minimum number required to encode ten meanings with words that are two symbols long.

Furthermore, all the objects (*Mary, Zoltan, Mike, Tunde* and *John*) belong to the same syntactic category (*62*) and all the actions (*Likes, Loves, Knows, Finds* and *Hates*) belong to a second category (*66*). In other words, this language encodes a classic noun/verb distinction syntactically.

The language is a VOS language in that the verb is the first word in the sentence, and the semantic roles of the two following nouns is determined by word order such that the first noun is the patient and the second is the agent. Figure 9 shows the tree structure for *John loves Mary* in this language. The emergent ordering differs from run to run, but the general pattern is the same: a noun/verb distinction encoded in the lexicon with the agent/patient distinction encoded by word order.

Although the result of this run is full compositionality, in that the sentence rule does not add any atomic semantic content, this is not always the case. Occasionally, one of the atomic meanings does not become *lexicalised* as a noun or a verb, and an idiosyncratic sentence rule is used to express meanings that include the missing word. We will return to the reasons for this later.

4.4 Summary of the results

What we have seen in this run, and in every run of the simulation that has been attempted, is the emergence of simple, yet language-like, syntax from randomness in a population that is *not* constrained to learn *only* a syntactic language.

The communication system of the population quickly emerges from nothing as an impoverished, idiosyncratic vocabulary of one-word utterances, nothing more than an inventory of calls expressing unanalysed meanings. This system is passed on only "culturally" through observational learning by new individuals, and there is nothing else inherited by later generations from earlier ones.

Eventually, after many generations, the system that is used to express meanings balloons in complexity. Utterances are no longer unanalysed strings of symbols. They are made up of common chunks of several symbols. Some of these chunks even have meanings of their own, although they are not regularly used to signify these meanings in a larger context. The language of the population now goes through radical and unpredictable changes over time as the range of meanings that are readily expressible changes wildly. The language appears to be brittle in some way and liable to break and lose its expressive power suddenly.

At some point, all this changes, and the population converges on a simple system, a syntactic system. Now, every sentence is made up of nouns and verbs (drawn from a concise lexicon lacking synonymy and homonymy) in a fixed order which encodes meaning distinctions compositionally, and every possible meaning can be expressed.

From this point forward, the language is stable. The perturbations in the coverage measure in figure 3 are due to the occasional "noisy" example that is learned by a new agent as an idiosyncratic, idiomatic form. These forms do not persist, however,[8] and the language always returns to its previous state.

5 Why does this model work?

The individuals in the simulation simply observe each others' behaviour and learn from it, occasionally inventing, at random, new behaviours of their own. From this apparent randomness, organisation emerges. Given that so little is built into the simulation, why is syntax inevitable?

To answer this question, we need to understand how the languages of the individuals in the population change over time. Language exists in two forms both in the simulation, and in reality (Chomsky 1986; Hurford 1987; Kirby 1998):

I-language This is (internal) language as represented in the brains of the population. It is the language user's knowledge of language. In the simulation, the I-language of an individual is completely described by its PAG.

E-language This is the (external) language that exists as utterances in the arena of use (Hurford 1987). In the simulation, we can describe E-language by listing the form-meaning pairs of an individual.

These two types of language are clearly made of very different stuff, but influence each other in profound ways. E-language can clearly be seen as a product of the I-language of speakers. However, the I-language of language learners is a product of the E-language that they have access to (see figure 10). Two different I-languages can produce the same E-language. For example, in the simulation two I-languages can differ in their choice of intermediate category numbers, but still produce/parse exactly the same set of strings. Similarly, two different E-language experiences can still lead to the same I-language in a learner. Nevertheless, meaningful change does occur. To put it another way, a *particular* I-language or E-language can fail to persist from one point in time to another because the processes that map from one form to the other and back again are not necessarily preservative.

[8] Although, see the section on frequency, below, for an interesting counter-example.

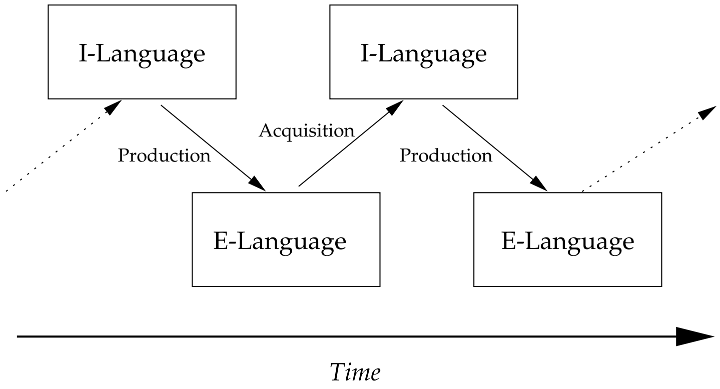

Figure 10: The cycle of language acquisition and use, which maps I-language objects to E-language objects and vice versa. These transformations act as bottlenecks for the information flowing through the system. For a particular feature of language to survive over time, it must be faithfully preserved through these mappings.

We can divide up I-language into units — *replicators* — that may or may not persist through time. There are many ways in which we can do this: for example, we could see the entire I-language as a replicator, or a particular symbol as a replicator. As long as we can identify the units' success or failure to persist over time, it is useful to think of them in this way. Clearly which types of I-language persist over time is related to the success of the replicators that make up those languages. In other words, the languages which are more easily transmitted from generation to generation will persist.

In the model described, what size of replicators should we look for, and what determines their success? Here it is useful to introduce a notion of competition into the picture. It turns out that within a population, certain replicators actually compete for survival. That is, the success of one must be measured relative to the success of others in the population at that time. These competing replicators are those rules which potentially express the same meaning. If there are two ways of saying *John loves Mary*, then on a particular exposure to this meaning, the learner can obviously only hear one of them. Therefore, on one exposure, only one of the rules (or, more properly, set of rules) that can be used to express *John loves Mary* has a chance of being induced by the learner.

At face value, it would seem that the two competing rules (or rule-sets) will have an equal chance of being the one chosen for producing the meaning, so the replicative success of all rules in a language should be equal. This would be true *if each rule only ever expressed one meaning*. However, if one rule can be used to express more meanings than another, then, all other things being equal, that rule will have a greater chance of being expressed in the E-language input to the learner. In this case, the more general rule is the better replicator.

For a more concrete example, consider a situation where, in the population of I-languages, there are two competing rules. One is a rule that expresses *John loves Mary* as an unanalysed string of symbols — essentially as one word. The other rule expresses *John loves Mary* as a string of symbols, but can also be used to express any meaning where someone loves Mary. So, the latter rule can also be used to express *Zoltan loves Mary* and so on. Further imagine that both rules have an equal chance of being used to express *John loves Mary*. The more general rule is still a better replicator, because for any randomly chosen set of meanings, we can expect it to be used more often than the idiosyncratic rule. Its chances of survival to the next generation are far more secure than the idiosyncratic rule. Furthermore, even if both rules are acquired by a particular hearer, the statistic associated with the general rule is likely to be higher. This means that, when it comes to the point when the hearer needs to say *John loves Mary*, the more general rule will probably be employed.

Of course, the more general rule will not be learned as easily as the idiosyncratic rule. In the simulations described above, an idiosyncratic pairing of one meaning to one form takes only one exposure to learn, but the most general rule takes several. However, the idiosyncratic rule only covers one meaning, whereas the most general rule covers 100. It is clear, therefore, that the probability of a acquiring a particular rule given a random sample of meanings increases with the generality of that rule. The success of I-languages which contain general rules seems secure.

The picture that emerges, then, is of the language of the population acting as an adaptive system (Gell-Mann 1992; Kirby 1998; Kirby 1997a) in its own right. Initially, the rules are minimally general, each pairing one string with one meaning. At some point, a chance invention or random noise will lead a learner to "go beyond the data" in making a generalisation that the previous generation had not made. This generalisation will then compete with the idiosyncratic rule(s) for the same meaning(s). Given that generalisations are better replicators, the idiosyncratic rules will be pushed out over time. The competition will then be replayed amongst generalisations, always with the more general rules surviving.

The inevitable end state of this process is a language with a syntax that supports compositionally derived semantics in a highly regular fashion. The grammar for such a language appears to be the shortest (in terms of numbers of rules) that can express the entire meaning space. The shorter the grammar, the higher the generality of each of the rules — the shortest grammar that can still do the job of expressing meanings is therefore the one made up of optimal replicators.

6 Linguistic adaptation as a theory for the origin of language

The previous sections have demonstrated an idealised model of a population of learners and shown that compositional syntactic systems are attractors in the space of possible languages. It has been argued that languages are behaving as adaptive systems through a process of competitive replication of generalisations. This section assesses the importance of this result for linguistic theory.

6.1 Constraints on variation

As mentioned in the introduction to this paper, the theory put forward here is an alternative to the standard approach which places central importance on an innate, constraining language acquisition device. At this point it is useful to unpack exactly what the differences are between these two perspectives on the origins of syntax. In particular, the relationship between the model of the acquirer and constraints on cross-linguistic variation are quite different.

Traditionally, the Chomskyan language acquisition device (LAD) directly constrains what makes a possible human language by limiting directly what can or cannot be acquired. This limit is said to closely map the observed constraints on variation (Hoekstra & Kooij

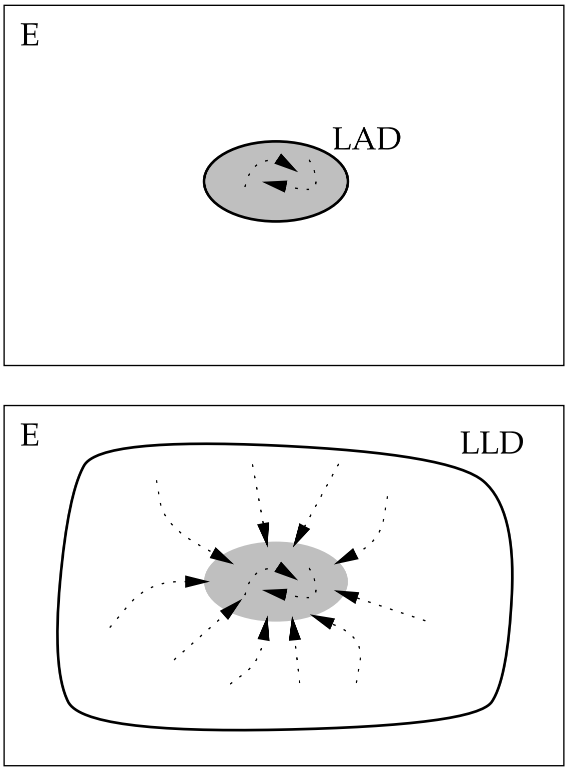

Figure 11: Two venn diagrams showing the different approaches to explaining observed constraints on cross-linguistic variation. E is the set of all logically possible languages, the gray area signifies the set of occurring human languages. In the top diagram, the Chomskyan language acquisition device constrains the learner directly and nothing else is required to explain the limits on variation. In the bottom diagram, the language *learning* device is less constraining, and the particular characteristics of human languages are the end result of a historical evolution of languages in populations (represented by arrows).

1988). Part of the generative research program involves accounting for variation between languages explicitly within the model of the language acquirer. In fact, Universal Grammar (UG) and the LAD are often treated as synonymous within this tradition. It is not generally considered that the dynamics of language acquisition and use impose further constraints within the boundaries imposed by the structure of the LAD (although see (Niyogi & Berwick 1995; Clark 1996) for interesting exceptions).

Figure 11 contrasts this view with that proposed in this paper. The language learning device clearly does impose constraints directly in a similar fashion — there are certain types of language that the learner simply cannot acquire — however these constraints are far less severe than those imposed by the LAD. As can be seen in the first two stages of the simulation, very un-language like systems can be acquired by this learner. The constraints on variation are not built into the learner, but are instead emergent properties of the social dynamics of learned communication systems and the structure of the semantic space that the individuals wish to express.

The theory presented here gives us a neat explanation of why human languages use syntactic structure to compositionally derive semantics, have words with major syntactic categories such as noun and verb, and use structural relations (such as word order) to encode meaning distinctions. However, it does not seem to allow us to understand more specific universals. For example, particular constituent orders are far more frequent than others across the languages of the world (Hawkins 1983; Dryer 1992).

Perhaps the best explanation for these types of universal should look at the effect of parsing and generation in the transmission of replicators (see Kirby 1998; Kirby 1997a for details). On the other hand, at least some of these word order constraints may eventually be explained in terms of linguistic adaptation without appealing to processing (see, Christiansen 1994; Christiansen & Devlin 1997 for some suggestions along these lines). X-bar theory — a sub part of UG which constrains the structure of syntactic trees cross categorially (Jackendoff 1977) — has been implicated in various word order universals. Daniel Nettle suggests that X-bar is just the sort of over-arching generalisation that the theory put forward in this paper predicts. It can be characterised as a pair of phrase structure rules:

$$XP \rightarrow Spec\ X' \text{ or } XP \rightarrow X'\ Spec$$
$$X' \rightarrow X\ YP \text{ or } X' \rightarrow YP\ X$$

These rules are like standard context free rules except that X and Y are variables that can range over the lexical categories in the language.

This use of variables in phrase structure rules is not possible with the PAG formalism, so this result is not possible in the simulation. Nevertheless, if the language learning device were able to make a generalisation such as that expressed by X-bar, we would expect it to thrive as a replicator. More generally, we should expect languages to behave in such a way that their word orders can be expressed in the most compact way, since this will reflect the behaviour of the optimal, most general, replicator. Dryer (1992) shows with a large-scale cross-linguistic survey, that this is indeed the case; languages tend to order their non-branching nodes on the same side of their branching nodes across the phrasal categories of the language.

6.2 Frequency effects

If the emergence of compositionality is due to the differential success of competing replicators, we can make an interesting prediction about the effect of frequency of meanings on the evolution of the languages in the simulation. Recall that the success of a replicator is related to the probability it will show up in the input E-language of a learner. This suggests that if a particular meaning is expressed particularly frequently by speakers, any rule that contributes to the production of a string for that meaning will be a good replicator.

We have argued that the emergence of compositionality in the languages of the population is due to compositional rules forcing out idiosyncratic rules, because idiosyncratic rules are employed in a relatively small portion of E-language. However, if one meaning is particularly frequent, and if the theory of replicator dynamics is correct, then we should find that an idiosyncratic form for a frequent meaning should survive longer than one for an infrequent meaning.

To test this, the simulation was run again, but the maximum number of utterances produced was doubled to 200, and approximately half of all the utterances had the meaning *John loves Mary*. In other words, at each point in time, there was a 50% chance that a speaker would be asked to produce a string for this one meaning. This makes *John loves Mary* far more common than the other 99 possible meanings.

The results of such runs are consistent with the hypothesis that it is replicator dynamics that is driving the evolution of language in the simulation. In **stage I** of the simulation, all complex meanings are expressed non-compositionally as one word sentences, and the utterance for *John loves Mary* is no exception. The difference at this stage is that *every* individual has a word for this meaning, whereas the other meanings are only sparsely covered.

In **stage II** completely non-compositional rules are rare and do not survive from generation to generation. However, every individual with few exceptions still has a non-compositional rule for *John loves Mary*. In other words they have a rule of the form:

```
S --> <some string of categories>
   Agent=John
   Patient=Mary
   Predicate=Loves
```

Finally, even in **stage III**, where typically all idiosyncratic rules are eliminated, and the grammar becomes completely compositional, an idiosyncratic form for *John loves Mary* can survive, although in some runs it is eventually eliminated.

These results strongly suggest that the correct analysis of the behaviour of the system is in terms of the relative success of rules as replicators. It is also possible that in natural language there are some examples of frequency affecting the compositionality of forms. For example, in morphology, suppletive forms tend to correlate with highly frequent meanings. So, for example, the past tense form of the frequent verb, *go* is the non-compositional *went* not *goed*. The ordinal versions of the English numbers after *three* are compositional — *third*, *fourth*, *fifth* etc. — but the more frequent *first* and *second* are not.

6.3 Creolisation

The discussion in this paper so far points to a *historical* emergence of compositional syntax in language as opposed to a biological one. It has been argued here that this central feature of human language is not a biological adaptation but a response of adaptation by languages themselves to the pressures imposed by the mapping from I-language to E-language and back again.

The distinction between the two views can be highlighted by a thought experiment. Imagine that some virus spreads through an isolated community on some remote island that removes people's ability to communicate with each other, and that this illness is virulent in the population for several generations. Children born into this community would be deprived completely of linguistic input, and eventually the island would be populated solely by people who had never been exposed to any language. After the virus has passed what happens to the population?

If we were to take the innatist stance, we would say that the damage would be short-lived. Even the generations of children born into silence preserve information about the syntactic nature of human language intact in their genomes which they pass on to the next generation. However, if we consider syntax to be a historically emergent property, then the imaginary situation is just like the initial stages of the population in the simulation — it will take many hundreds of generations to repair the damage.

A thought experiment such as this is useful in setting out the differences between two approaches, but it also seems to have historical parallels that could cause serious problems for this theory of the origin of syntax. Bickerton (1981) describes cases where a pidgin language is formed as adults from varied linguistic backgrounds are forced together (for example, as plantation slaves). This type communication system, Bickerton claims, is best described as a *protolanguage* rather than a full-blown language, and at least in its earliest incarnation has no syntactic structure at all. In certain cases, according to Bickerton, the pidgin spoken by the adult community has been transformed in one generation by the child population into a creole language which has a syntactic system as rich as any normally occurring human language.

At least with respect to the syntactic structure of language, this seems to be similar to the thought experiment outlined above. If children in a population of pidgin speakers can innovate syntax in one generation (and see Senghas 1995 for some excellent evidence that this is indeed what is happening) then this suggests that syntactic structures must be innate rather than an emergent property of language transmission over time. To put it another way, if the individuals in our simulation were put in an environment without syntax, we might expect them to take hundreds of generations to achieve a syntactic language again, just as it does at the very start of the simulation. If this is true, then it weakens the case for syntax being a historically emergent property.

To test this, we need some way of simulating a pidgin environment for the first generation of learners in the simulation. A protolanguage for the semantics we use would have vocabulary items for the atomic meanings such as *John* and *loves*, but not encode meaning distinctions such as agent vs. patient using structural cues. A simple way of doing this is to set up a vocabulary of atomic meanings, and each time a protolanguage utterance for a complex meaning is needed, randomly order the words for each of the atomic components of that meaning. So, if *John* is *ba*, *knows* is *ee*, and *Zoltan* is *cb*, then the utterance meaning *John knows Zoltan* could be *baeecb*, *bacbee*, *eebacb*, *eecbba*, *cbbaee* or *cbeeba*. Critically, the set of possible pidgin utterances for *Zoltan knows John* will be exactly the same.

No claims are made here that this is how a real pidgin would encode these meanings. Instead, this protolanguage has been set up to model, in the simplest way possible, a system that has a vocabulary but no structural cues to meaning. With the simple semantics of the simulation, the expressive power of the pidgin is impoverished because there is no way of expressing the distinction between agent and patient.[9]

A simulation was run that was identical to the one described in section 4 except that the initial population was exposed to the artificial protolanguage described above. The results are radically different. Recall that the simulation with no initial language took hundreds of generation to achieve a language that reliably encoded the agent/patient distinction. With an initial “pidgin” input, on the other hand, a language emerges that manages this before even one complete turnover of the members of the population. The only *S* rule in this language is shown here:

```
S --> 59 44 59  [100]
   Agent=(2).3
   Patient=(0).3
```

[9]This simulated pidgin looks a bit like a free word order language. The critical difference is that a free word order language for this semantics would have some other means of coding the difference between agent and patient, by using some kind of case marking for example.

```
Predicate=(1).0
```

(`44` is the category for verbs and `59` is the category for nouns.)

Clearly, this is an extremely simplified model of what is going on in creolisation. In particular, given the limited semantic space available to the simulation, it is very difficult to realistically model protolanguage. However, the results suggest that the relatively unconstrained learners in the population and the limited structure inherent in the pidgin are enough together to get the population rapidly to innovate a syntactic system that is not in the input data. We should be very careful, therefore, before we use the evidence from creoles to posit a highly constraining innate language faculty.

6.4 How general is this result?

It has been argued that syntax as a compositional mapping between forms and meanings is an emergent property of observationally learned communication systems. However, this argument uses the results from a highly idealised simulation of language learning and use, which raises the question of whether we should generalise from the results of the model to the real system. The inevitable concern is that the behaviour of the simulation derives from, for example, some small detail of the heuristics in the learning algorithm, and cannot be extended to other systems.

One approach to these concerns is the explanation, in section 5 of this paper, of the behaviour of the simulation in terms of replicator dynamics. This explanation is couched in terms that are general enough to cover both the behaviour of the simulation model and of languages. To the extent to which it is correct to think of generalisations as replicators, this makes the fine details of the simulation irrelevant. In other words, it should be expected that any similar simulation will give rise to similar results.

In fact, the simulation of Batali (1997), mentioned earlier, can be thought of in these terms, although Batali does not do so explicitly. The simulation has a somewhat simpler semantic space than the one used here. Batali's meanings are made up of predicates and one "referent" which are coded as bit vectors.[10] With this meaning space there can be no emergent word order that codes for the difference between agent and patient, obviously, but it still allows for a language which distinguishes between a word for the predicate and a word for the referent.

The most important difference between Batali's simulation and ours is that it uses a completely different model of learning. The individuals in the population are recurrent neural networks (e.g. Elman 1990), trained by back propagation of error to predict the meaning vectors that correspond to a string of symbols. In such a learner, the I-language is represented not as an explicit context-free grammar, but as a pattern of connection weights within the network. It is clear that such a learner and will have very different biases to the one employed in this paper. Certainly, there is no sense in which the network learning algorithm is searching for minimal grammars directly.

In spite of these differences, Batali's results can be interpreted as a move towards compositional languages very similar to that found in the simulations I have reported. Initially, the population communicates using idiosyncratic pairings of meanings with symbol sequences. At the end of a run, however, the sequences used by the individuals mainly appear to be

[10]The vectors do not explicitly encode the features predicate and referent, but the encoding is sparse so that the predicate/referent distinction is implicit in the meaning space.

composed of a "verbal" root plus a "pronominal" suffix. For example, in one run, every utterance whose meaning included the predicate *tired* started with the letters *cd* and most utterances whose referent was *you* ended with the letter *c*. So, the meaning *you are tired* would be expressed *cdc*. Although the I-language of the networks cannot be studied directly, their E-language behaviour appears to be compositional.

The fact that the results of Batali's simulation and ours are of a similar type is strong evidence that the explanation for emergence of compositionality in language is a general one. It is likely that Batali's result can be understood in terms of the dynamics of replicating generalisations. One problem with an analysis of his simulation in this way is that in Batali's population model agents are immortal. This means that the pressure on generalisations cannot be to survive from one generation to another (there are no generations). However, the persistence of any feature of the language over time relies on it being produced and re-learned again and again by the population. This is because the network architecture imposes a finite information holding capacity — forcing form-meaning pairings into competition.

7 Conclusion

The goal of this paper has been to understand the role of truly linguistic evolution (as opposed to biological evolution) in the emergence of syntax. Human language is almost unique in the natural world as being a phenomenon that persists over generations via observational learning. As such it has its own interesting and complex dynamical properties, which the simulation described attempts to explore. Although the extent to which these properties have played a role in the origins of universals of human language is not well understood, they should not be ignored when looking at human evolution.

There are three main ways in which the results of studies such as this one should impact theories of the biological evolution of language. Firstly, it means that potentially we have less to explain. We need not pin down every constraint on UG as an innately coded response to pressures from natural selection. Instead our model of the language faculty can be less constrained, and possibly less domain specific.

Secondly, if syntax is not a response to pressures from natural selection, we might ask the question "why are there no non-human syntactic communication systems?". In fact, one might rephrase the question as Oliphant (1997:108) does, and ask "why are there so few learned communication systems".

> "Most approaches to the evolution of language point to the evolution of syntax as the primary barrier differentiating the communicative abilities of man and other species ... another, more basic, bottleneck exists ... It seems that the ability to learn by observing others is rare, if present at all, in other animals ... While most attention is generally focused on syntax, I argue that the ability to learn observationally may be an equally, if not more important evolutionary milestone."(Oliphant 1997:118-119)

Thirdly, if languages are adaptive systems which undergo qualitative evolution themselves, then this will profoundly alter the selection pressures on the (proto-)human language faculty. In other words a complete picture of the evolution of language may be a co-evolutionary one. Certain classes of languages — well adapted for their own survival — may emerge through linguistic evolution, and the natural selection pressure on the evolving language faculty will be to learn that class of languages more efficiently. This kind of "genetic assimilation" of patterns in the environment of learning is controversial amongst biologists (see Maynard Smith 1987 for a positive introduction), but there is a growing body of literature that suggests taking a co-evolutionary perspective is useful (e.g. Deacon 1997; Kirby & Hurford 1997; Hurford & Kirby 1997; Briscoe 1997).

Throughout, this paper has concentrated on one particular feature of syntax, namely compositionality. In section 6.1, it was suggested that some more specific features of UG might also be explained by looking at replicator dynamics. It is important to realise, however, that there is a long way to go before a simulation like the one described here can possibly hope to demonstrate the emergence of all the constraints that syntacticians are familiar with. (Bickerton 1997) drives this point home:

> "... in language evolution circles generally, the belief seems widespread that once you've gotten as far as 'John loves Mary' ... syntax is off and running and nothing stands in the way of it gradually expanding to embrace all of the many complexities found in contemporary languages.
>
> Every serious syntactician knows that nothing could be further from the truth."

These are sobering words, but this model represents only the first step towards a demonstration of the emergence of a syntactic communication system from something qualitatively simpler. Future research will have to proceed on two fronts: by making the simulation model more powerful, particularly through the expansion of the semantic space to include recursive representations; and by applying ideas of replicator dynamics to different features of Universal Grammar. Only once we understand the limits of linguistic evolution *without* natural selection can we really begin to understand the biological evolution of our language faculty.

References

ASHER, R. E. (ed.) 1994. *The Encyclopedia of Language and Linguistics*. Pergamon Press.

BATALI, JOHN. 1994. Innate biases and critical periods: Combining evolution and learning in the acquisition of syntax. In *Artificial Life IV*, ed. by Rodney Brooks & Pattie Maes, 160–171. MIT Press.

——. 1997. Computational simulations of the emergence of grammar. In *Evolution of Language: Social and Cognitive Bases for the Emergence of Phonology and Syntax*, ed. by James Hurford, Chris Knight, & Michael Studdert-Kennedy. Cambridge University Press. In press.

BICKERTON, DEREK. 1981. *Roots of Language*. Karoma Publishers.

——. 1990. *Language and Species*. University of Chicago Press.

——. 1997. Catastrophic evolution: the case for a single step from protolanguage to full human language. In *Evolution of Language: Social and Cognitive Bases for the Emergence of Phonology and Syntax*, ed. by James Hurford, Chris Knight, & Michael Studdert-Kennedy. Cambridge University Press. In press.

BRISCOE, TED, 1997. Language acquisition: the bioprogram hypothesis and the Baldwin Effect. MS, Computer Laboratory, University of Cambridge.

CANGELOSI, ANGELO, & DOMENICO PARISI. 1996. The emergence of a language in an evolving population of neural networks. Technical Report NSAL–96004, National Research Council, Rome.

CANN, RONNIE. 1993. *Formal Semantics: An introduction*. Cambridge: Cambridge University Press.

CHARNIAK, EUGENE. 1993. *Statistical language learning*. Cambridge, MA: MIT Press.

CHOMSKY, NOAM. 1986. *Knowledge of Language*. Praeger.

CHRISTIANSEN, MORTEN, 1994. *Infinite Languages, Finite Minds: Connectionism, Learning and Linguistic Structure*. University of Edinburgh dissertation.

——, & JOSEPH DEVLIN. 1997. Recursive inconsistencies are hard to learn: A connectionist perspective on universal word order correlations. In *Proceedings of the 19th Annual Cognitive Science Society Conference*, 113–118. Mahwah, NJ: Lawrence Erlbaum Associates.

CLARK, ROBERT, 1996. Internal and external factors affecting language change: A computational model. Master's thesis, University of Edinburgh.

DEACON, TERRENCE W. 1997. *The Symbolic Species: The co-evolution of language and the brain*. New York: Norton.

DRYER, MATTHEW. 1992. The Greenbergian word order correlations. *Language* 68.81–138.

ELMAN, JEFFREY. 1990. Finding structure in time. *Cognitive Science* 14.179–211.

GELL-MANN, MURRAY. 1992. Complexity and complex adaptive systems. In *The Evolution of Human Languages*, ed. by J.A. Hawkins & M. Gell-Mann. Addison-Wesley.

HAWKINS, JOHN A. 1983. *Word Order Universals*. Academic Press.

HOEKSTRA, TEUN, & JAN G. KOOIJ. 1988. The innateness hypothesis. In *Explaining Language Universals*, ed. by John A. Hawkins. Blackwell.

HURFORD, JAMES. 1987. *Language and Number: the Emergence of a Cognitive System*. Cambridge, MA: Basil Blackwell.

——. 1989. Biological evolution of the Saussurean sign as a component of the language acquisition device. *Lingua* 77.187–222.

——. 1991. The evolution of the critical period for language acquisition. *Cognition* 40.159–201.

——. 1998. The evolution of language and languages. In *The Evolution of Culture*, ed. by Chris Knight Robin Dunbar & Camilla Power. Edinburgh University Press. In press.

——, & SIMON KIRBY. 1997. Co-evolution of language-size and the critical period. In *New Perspectives on the Critical Period Hypothesis and Second Language Acquisition*, ed. by David Birdsong. Lawrence Erlbaum. In press.

——, CHRIS KNIGHT, & MICHAEL STUDDERT-KENNEDY (eds.) 1998. *Evolution of Language: Social and Cognitive Bases for the Emergence of Phonology and Syntax*, Cambridge. Cambridge University Press.

JACKENDOFF, R. 1977. *X-Syntax: A Study of Phrase Structure*. MIT Press.

KIRBY, SIMON. 1997a. Competing motivations and emergence: explaining implicational hierarchies. *Language Typology* 1.5–32.

——. 1997b. Fitness and the selective adaptation of language. In *Evolution of Language: Social and cognitive bases for the emergence of phonology and syntax*, ed. by J. Hurford, C. Knight, & M. Studdert-Kennedy. Cambridge University Press. In press.

——. 1998. *Function, Selection and Innateness: The Emergence of Language Universals*. Oxford: Oxford University Press. In press.

——, & JAMES HURFORD. 1997. Learning, culture and evolution in the origin of linguistic constraints. In *Fourth European Conference on Artificial Life*, 493–502. MIT Press.

MACLENNAN, BRUCE. 1991. Synthetic ethology: an approach to the study of communication. In *Artificial Life II*, ed. by C.G. Langton, C. Taylor, J.D. Farmer, & S. Ramussen, 631–657. Addison-Wesley.

MAYNARD SMITH, JOHN. 1987. Natural selection: when learning guides selection. *Nature* 329.761–762.

MORFORD, JILL P., & ACHIM OBERST, 1997. Describing the path from intermediate forms to modern language. Unpublished manuscript, University of New Mexico.

NEWMEYER, FREDERICK J. 1991. Functional explanation in linguistics and the origins of language. *Language and Communication* 11.3–28.

—— 1992. Iconicity and generative grammar. *Language* 68.756–796.

NIYOGI, PARTHA, & ROBERT BERWICK. 1995. The logical problem of language change. Technical Report AI Memo 1516 / CBCL Paper 115, MIT AI Laboratory and Center for Biological and Computational Learning, Department of Brain and Cognitive Sciences.

——, & ——, 1997. Populations of learners: the case of Portugese. Unpublished manuscript, MIT.

OLIPHANT, MICHAEL. 1996. The dilemma of saussurean communication. *BioSystems* 37.31–38.

——, 1997. *Formal Approaches to Innate and Learned Communication: Laying the Foundation for Language*. UCSD dissertation.

PINKER, STEVEN, & PAUL BLOOM. 1990. Natural language and natural selection. *Behavioral and Brain Sciences* 13.707–784.

SENGHAS, ANN, 1995. *Children's contribution to the birth of Nicaraguan sign language*. Department of Brain and Cognitive Sciences, MIT dissertation.

STEELS, LUC. 1996. Emergent adaptive lexicons. In *Proceedings of the Simulation of Adaptive Behavior Conference*, ed. by P. Maes. Cambridge, Ma: The MIT Press.

——. 1997. The synthetic modelling of language origins. *Evolution of Communication* 1.

STOLCKE, ANDREAS, 1994. *Bayesian Learning of Probabilistic Language Models*. University of California at Berkeley dissertation.

A Technical details of the simulation

A.1 A formal definition of PAGs

The Probabilistic Attribute Grammar (PAG) formalism used in this simulation is a variant of the one discussed in detail in Stolcke (1994) (the definitions here are modified only slightly from Stolcke's pages 76–77 and 105–106). As mentioned in the text, they are an extension of a standard stochastic context free grammar (SCFG) framework.

The SCFGs are context free grammars with an integer count attached to each rule. Such a grammar consists of:

1. a set of nonterminal symbols $\mathcal{N}$,
2. a set of terminal symbols Σ,
3. a start nonterminal S, a member of $\mathcal{N}$,
4. a set of productions $\mathcal{R}$, of the form
$$X \rightarrow \lambda$$
where $X \in \mathcal{N}$ and $\lambda \in (\mathcal{N} \cup \Sigma)^*$,
5. production counts $C(r)$ for all $r \in \mathcal{R}$.

This SCFG is enriched with a feature/value semantics:

1. Each nonterminal $X \in \mathcal{N}$, in a SCFG derivation has an associated finite set of features from a set $\mathcal{F}$. A feature $f \in \mathcal{F}$ is a function that assigns an element from the set of feature values $\mathcal{V}$ to the nonterminals it is defined upon. The fact that feature f on nonterminal X has value v is denoted, $X.f = v$.[11]
2. An extended context-free production specifies the feature values attached to the left hand side nonterminal in terms of either constant values of the features of right hand side nonterminals. Thus, a production
$$X \rightarrow Y^{(0)} Y^{(1)} \dots Y^{(k)}$$
can have feature specifications
$$X.f = v$$
for some $f \in \mathcal{F}$ and $v \in \mathcal{V}$, or
$$X.f = (i).g$$
for some i, $0 \leq i \leq k$, and $g \in \mathcal{F}$. The latter specification determines the value of $X.f$ is whatever the value of feature g on $Y^{(i)}$ is.

A.2 Model merging operators

The model merging operators are global operations over the grammar which modify the grammar in some way. Again, they are taken from Stolcke (1994).

[11] In the output of the simulation, this is simply shown as `f=v`.

A.2.1 MergeCat

This operator takes two category names as arguments, $X_1, X_2 \in \mathcal{N}$. The operator has the effect of rewriting all occurrences of X_1 in the grammar as X_2.[12] If this causes two rules, $r_1, r_2 \in \mathcal{R}$, to become identical then r_2 is deleted from the grammar and the count on r_1 is increased so that $C(r_1) = C(r_1) + C(r_2)$.

A.2.2 MergeFeat

This takes two feature names, $f_1, f_2 \in \mathcal{F}$. The operator rewrites all occurrences of f_1 in the grammar by f_2. In the simulations described here, there can be no two feature equations on a rule assigning a value to one feature, so if there is a rule $r \in \mathcal{R}$ with n feature equations $X.f = \ldots$, then the rule is split into n rules $r_1, r_2 \ldots r_n$. Each new rule is identical to the old rule except that it has only one of the feature equations $X.f = \ldots$. The counts are divided so that $C(r_i) = C(r)/n, 1 \leq i \leq n$.[13]

A.2.3 Chunk

This takes a list of categories, $X_1, X_2 \ldots X_n \in \mathcal{N}$. The operator constructs a new rule r, $X \rightarrow X_1 X_2 \ldots X_n$ and replaces all occurrences of $X_1 X_2 \ldots X_n$ in rules $r_1, r_2 \ldots r_k$ in the grammar with X.[14] In order that the semantics of the affected rules work in exactly the same way after the operation as before, new feature equations are added to the new rule (each with a new feature name) that preserve the information that is passed up to the old rules. The new rule inherits the statistics on the old rules, so that $C(r) = \sum_{i=1}^{k} C(r_i)$.

A.2.4 AttribFeat

This operator takes a category $X \in \mathcal{N}$ and a value $v \in \mathcal{V}$. The operator has the effect of "attributing" the value v of a semantic feature $f \in \mathcal{F}$ in a rule to the left-hand side category X. It does this by replacing all occurrences of $Y.f = v$, in rules, $Y \rightarrow \lambda, X \in \lambda$, with $Y.f = (i).f_1$, where f_1 is a newly constructed feature name, and i is the position of X in λ. For each such rewrite, every rule that has X on its *right*-hand side has $X.f_1 = v$ added to its feature equation list.[15] The statistics on the grammar are not changed by this operation.

A.3 Model merging heuristics

Each of the low level operators listed in the previous section has an associated set of heuristics which decide when, and with what arguments, they should be used. As discussed in the

[12] In the trace in the next section, the categories are given numerical values. The inducer always renames the category with the higher value.

[13] This is a poor approximation, however empirical studies show that "splitting" is very rare.

[14] For the simulations reported here, $r_1, r_2 \ldots r_k$ must all have the left-hand side category S, $X_1, X_2 \ldots X_n$ must all expand to terminal symbols, and none of $X_1, X_2 \ldots X_n$ should be the "pause symbol" mentioned in the text. These constraints greatly aid the discovery of chunks in the input without search, and if they are combined with a constraint that stops categories that expand to terminals from being merged with those that do not, then the grammar cannot become recursive. This is justified here since the semantics of PAGs is not recursive.

[15] For the simulations reported here, feature attribution is constrained so that features are not attributes to terminals and features are not attributed to rules which already have a non-null semantics. If these constraints are relaxed, the heuristics described in the next section tend to result in semantically over-complex grammars. Stolcke (1994) shows that constraints such as these are not required where induction with search is possible.

text, the heuristics have to be used instead of a search for the best grammar (as in Stolcke's (1994) inducer) in order for the simulation to be computationally feasible. Future work will avoid the use of heuristics as their properties are less well understood. Each heuristic takes a pair of rules from the grammar as arguments and, if possible, calls an operator in order to make those two rules more similar or alter them in such a way that a different heuristic may be able to make those rules more similar. All heuristics are applied to all pairs of rules exhaustively.

A.3.1 MergeCat heuristic

1. Given $r_1, r_2 \in \mathcal{R}$ and $X_1, X_2 \in \mathcal{N}$, if MergeCat(X_1, X_2) would make r_1 equivalent to r_2 (except for their statistics), then apply MergeCat(X_1, X_2).

A.3.2 MergeFeat heuristics

1. Given $r_1, r_2 \in \mathcal{R}$ with the same number of categories on their right-hand sides, and $f_1, f_2 \in \mathcal{F}$, if MergeFeat(f_1, f_2) would make r_1 equivalent to r_2 (except for their statistics and the names of the categories), then apply MergeFeat(f_1, f_2).

2. Given $r_1, r_2 \in \mathcal{R}$ with identical right-hand sides, and $f_1, f_2 \in \mathcal{F}$, if MergeFeat(f_1, f_2) would make the right-hand side of the feature equations in the two rules identical, then apply MergeFeat(f_1, f_2).

A.3.3 Chunk heuristics

1. Given $r_1, r_2 \in \mathcal{R}$, where $r_1 = X_1 \to \lambda_1$, and $r_2 = X_2 \to \lambda_2$, if λ_1 is a proper substring of λ_2, then apply Chunk(λ_1).

2. Given $r_1, r_2 \in \mathcal{R}$, where $r_1 = X \to \lambda_1$, and $r_2 = X \to \lambda_2$, if the feature equation lists on each rule differ by one value $v \in \mathcal{V}$, and λ_1 and λ_2 differ by two substrings λ_3, λ_4, then apply Chunk(λ_3) and Chunk(λ_4).[16]

A.3.4 AttribFeat heuristics

1. Given $r_1, r_2 \in \mathcal{R}$, whose right-hand sides are identical, if the only difference in the feature equations of the two rules is that r_1 does not have a feature specification $\ldots . f = v$ but r_2 does, then apply AttribFeat(X, v), where X is the left-hand side category of r_1.

2. Given $r_1, r_2 \in \mathcal{R}$ whose syntax differ only by a pair of right-hand side categories $X_1, X_2 \in \mathcal{N}$ respectively, if their semantics differ only by a pair of values $v_1, v_2 \in \mathcal{V}$ respectively, then apply AttribFeat(X_1, v_1) and AttribFeat(X_2, v_2).

[16]For the simulations reported here, this heuristic does not apply if both λ_3 and λ_4 are only one category long because this tends to make the grammar longer rather than shorter. Also, it was found that the chunks should not take up too much of the right-hand side for a similar reason. It is hard to see a principled way of constraining chunking without search, so as a rough rule-of-thumb, a chunk is only made if it is shorter of equal than the right-hand side of the rule divided by the number of values in the rule's semantics. This stops rules' right-hand sides becoming too short before feature attribution has taken place.

3. Given $r_1, r_2 \in \mathcal{R}$ whose syntax differs by at most one right-hand side category, if the semantics of the two rules are the same except that r_1 has a feature equation $X.f_1 = v$ where r_2 has the feature equation $X.f_1 = (i).f_2$ and i points to the syntactic category difference (if there is one), then apply AttribFeat(Y, v), where Y is the i^{th} right-hand side category of r_1.

A.4 The invention algorithm

The invention algorithm produces a string of symbols s for a meaning m given a grammar that *cannot* generate a string for that meaning. The goal of invention is to produce a string that is consistent with the structure of the language embodied by the grammar even though that grammar cannot itself produce the string.

The first step of the process is to find the nearest meaning m' to the target meaning such that a string s' can be generated for m' by the grammar. In practice this is approximated by removing a random feature from the semantics and attempting to generate with this new "underspecified" meaning.

Since m' is missing a feature f which m has, and as a consequence a string can be generated, then we know that the value v for the feature f in m is causing the problem for generation. We will assume that there is another meaning m'' which will generate s' and assigns the feature f a value $v' \neq v$. Some part of the string s' is thus contributing the "wrong" meaning $f = v'$. It is this part of the string that we would like to replace with a random innovation. To do this, all we need to do is to find the point in the tree for the string s' where f is assigned the value v'. Every node under this point in the tree needs to be randomised. A simple way of doing this is to replace each symbol in the string under this node with a randomly chosen symbol.[17]

This simple algorithm ensures that innovation will never introduce more structure into an utterance than is already implicit in the grammar of an individual. If the grammar specifies a vocabulary of flat one-word strings for each complex meaning, then an innovative string will always be completely random, because the point in the tree at which the value v' is introduced is the topmost node. Conversely, if the grammar is completely compositional, then most of the string will be preserved, since the value v' will be introduced low in the tree above the word corresponding to the incorrect meaning.

[17] To allow the lengths of strings to change, we can occasionally delete symbols or add extra ones.

B Trace of induction

This section gives a trace of what is going on in one individual given a tiny subset of English, namely the four sentences "John loves Mary", "Mary hates Pete", "Pete loves Mary" and "Pete hates John" in this order. These sentences are given to the inducer with an appropriate semantics similar to that used in this paper (i.e. Agent/Patient/Predicate feature triples). The semantics representations are given as English sentences in the listing for clarity. The strings of letters are given without spaces or capitals. The grammar is listed after incorporating each new sentence, and after all the model merging is complete. Each low-level operator call is listed.

Notice that after hearing all four sentences, the grammar is compositional and has a noun class and a verb class. Interestingly, the "es" suffix on "loves" and "hates" is not incorporated into the lexicon, but is left as non-semantic material in the top level *S* rule.

Incorporate `johnlovesmary` meaning *John loves Mary*

```
2 ==> j [1]

3 ==> o [1]

4 ==> h [1]

5 ==> n [1]

6 ==> l [1]

7 ==> o [1]

8 ==> v [1]

9 ==> e [1]

10 ==> s [1]

11 ==> m [1]

12 ==> a [1]

13 ==> r [1]

14 ==> y [1]

S --> 2 3 4 5 6 7 8 9 10 11 12
13 14  [1]
   Agent=John
   Patient=Mary
   Predicate=Loves

-----------------------------

MergeCat(3,7)

-----------------------------

3 ==> o [2]

2 ==> j [1]

4 ==> h [1]

5 ==> n [1]

6 ==> l [1]

8 ==> v [1]

9 ==> e [1]

10 ==> s [1]

11 ==> m [1]

12 ==> a [1]

13 ==> r [1]

14 ==> y [1]

S --> 2 3 4 5 6 3 8 9 10 11 12
13 14  [1]
   Agent=John
   Patient=Mary
   Predicate=Loves
```

Incorporate `johnlovespete` meaning *John loves Pete*

```
3 ==> o [2]

2 ==> j [1]

4 ==> h [1]

5 ==> n [1]

6 ==> l [1]

8 ==> v [1]

9 ==> e [1]

10 ==> s [1]

11 ==> m [1]

12 ==> a [1]

13 ==> r [1]

14 ==> y [1]

S --> 2 3 4 5 6 3 8 9 10 11 12
13 14  [1]
   Agent=John
   Patient=Mary
   Predicate=Loves

15 ==> j [1]

16 ==> o [1]

17 ==> h [1]

18 ==> n [1]

19 ==> l [1]

20 ==> o [1]

21 ==> v [1]

22 ==> e [1]

23 ==> s [1]
```

```
24 ==> p [1]

25 ==> e [1]

26 ==> t [1]

27 ==> e [1]

S --> 15 16 17 18 19 20 21 22
23 24 25 26 27  [1]
   Agent=John
   Patient=Pete
   Predicate=Loves

-----------------------------

MergeCat(15,2)
MergeCat(16,3)
MergeCat(3,20)
MergeCat(17,4)
MergeCat(18,5)
MergeCat(19,6)
MergeCat(21,8)
MergeCat(22,9)
MergeCat(9,25)
MergeCat(9,27)
MergeCat(23,10)
Chunk(24 9 26 9 )
Chunk(11 12 13 14 )
AttribFeat(29,Mary)
AttribFeat(28,Pete)
MergeFeat(0,1)
MergeCat(29,28)

-----------------------------

3 ==> o [4]

9 ==> e [4]

2 ==> j [2]

4 ==> h [2]

5 ==> n [2]

6 ==> l [2]

8 ==> v [2]

10 ==> s [2]

S --> 2 3 4 5 6 3 8 9 10
28  [2]
   Agent=John
   Patient=(9).0
   Predicate=Loves

11 ==> m [1]

12 ==> a [1]

13 ==> r [1]

14 ==> y [1]

24 ==> p [1]

26 ==> t [1]

28 --> 24 9 26 9  [1]
   0=Pete

28 --> 11 12 13 14  [1]
   0=Mary
```

Incorporate **`petelovesmary`** meaning *Pete loves Mary*

```
3 ==> o [4]

9 ==> e [4]

2 ==> j [2]

4 ==> h [2]

5 ==> n [2]

6 ==> l [2]

8 ==> v [2]

10 ==> s [2]

S --> 2 3 4 5 6 3 8 9 10
28  [2]
   Agent=John
   Patient=(9).0
   Predicate=Loves

11 ==> m [1]

12 ==> a [1]

13 ==> r [1]

14 ==> y [1]

24 ==> p [1]

26 ==> t [1]

28 --> 24 9 26 9  [1]
   0=Pete

28 --> 11 12 13 14  [1]
   0=Mary

30 ==> p [1]

31 ==> e [1]

32 ==> t [1]

33 ==> e [1]

34 ==> l [1]

35 ==> o [1]

36 ==> v [1]

37 ==> e [1]

38 ==> s [1]

39 ==> m [1]

40 ==> a [1]

41 ==> r [1]

42 ==> y [1]

S --> 30 31 32 33 34 35 36 37
38 39 40 41 42  [1]
   Agent=Pete
   Patient=Mary
   Predicate=Loves

-----------------------------

MergeCat(30,24)
MergeCat(31,9)
MergeCat(9,33)
MergeCat(9,37)
MergeCat(32,26)
MergeCat(34,6)
MergeCat(35,3)
MergeCat(36,8)
MergeCat(38,10)
MergeCat(39,11)
MergeCat(40,12)
MergeCat(41,13)
MergeCat(42,14)
Chunk(24 9 26 9 )
Chunk(11 12 13 14 )
AttribFeat(43,Pete)
MergeFeat(2,0)
MergeCat(43,28)
AttribFeat(44,Mary)
MergeFeat(3,0)
MergeCat(44,28)
Chunk(2 3 4 5 )
AttribFeat(45,John)
MergeFeat(4,0)
MergeCat(45,28)

-----------------------------

9 ==> e [7]

3 ==> o [5]

6 ==> l [3]

8 ==> v [3]

10 ==> s [3]
```

```
S --> 28 6 3 8 9 10 28  [3]
   Agent=(0).0
   Patient=(6).0
   Predicate=Loves

2 ==> j [2]

4 ==> h [2]

5 ==> n [2]

24 ==> p [2]

26 ==> t [2]

11 ==> m [2]

12 ==> a [2]

13 ==> r [2]

14 ==> y [2]

28 --> 24 9 26 9  [2]
   0=Pete

28 --> 11 12 13 14  [2]
   0=Mary

28 --> 2 3 4 5  [2]
   0=John
```

Incorporate **`johnhatesmary`** meaning *John hates Mary*

```
9 ==> e [7]

3 ==> o [5]

6 ==> l [3]

8 ==> v [3]

10 ==> s [3]

S --> 28 6 3 8 9 10 28  [3]
   Agent=(0).0
   Patient=(6).0
   Predicate=Loves

2 ==> j [2]

4 ==> h [2]

5 ==> n [2]

24 ==> p [2]

26 ==> t [2]

11 ==> m [2]

12 ==> a [2]

13 ==> r [2]

14 ==> y [2]

28 --> 24 9 26 9  [2]
   0=Pete

28 --> 11 12 13 14  [2]
   0=Mary

28 --> 2 3 4 5  [2]
   0=John

46 ==> j [1]

47 ==> o [1]

48 ==> h [1]

49 ==> n [1]

50 ==> h [1]

51 ==> a [1]

52 ==> t [1]

53 ==> e [1]

54 ==> s [1]

55 ==> m [1]

56 ==> a [1]

57 ==> r [1]

58 ==> y [1]

S --> 46 47 48 49 50 51 52 53
54 55 56 57 58  [1]
   Agent=John
   Patient=Mary
   Predicate=Hates

-----------------------------

MergeCat(46,2)
MergeCat(47,3)
MergeCat(48,4)
MergeCat(4,50)
MergeCat(49,5)
MergeCat(51,12)
MergeCat(12,56)
MergeCat(52,26)
MergeCat(53,9)
MergeCat(54,10)
MergeCat(55,11)
MergeCat(57,13)
MergeCat(58,14)
Chunk(11 12 13 14 )
Chunk(2 3 4 5 )
AttribFeat(59,Mary)
MergeFeat(5,0)
MergeCat(59,28)
AttribFeat(60,John)
MergeFeat(6,0)
MergeCat(60,28)
Chunk(4 12 26 )
Chunk(6 3 8 )
AttribFeat(62,Loves)
AttribFeat(61,Hates)
MergeFeat(7,8)
MergeCat(62,61)

-----------------------------

9 ==> e [8]

3 ==> o [6]

4 ==> h [4]

10 ==> s [4]

12 ==> a [4]

S --> 28 61 9 10 28  [4]
   Agent=(0).0
   Patient=(4).0
   Predicate=(1).7

6 ==> l [3]

8 ==> v [3]

2 ==> j [3]

5 ==> n [3]

26 ==> t [3]

11 ==> m [3]

13 ==> r [3]

14 ==> y [3]

28 --> 11 12 13 14  [3]
   0=Mary

28 --> 2 3 4 5  [3]
   0=John

61 --> 6 3 8  [3]
   7=Loves

24 ==> p [2]

28 --> 24 9 26 9  [2]
   0=Pete

61 --> 4 12 26  [1]
   7=Hates
```