PREFINAL DRAFT: Kirby, S. (1999). Learning, bottlenecks and infinity: a working model of the evolution of syntactic communication. In Dautenhahn, K. and Nehaniv, C., editors, Proceedings of the AISB'99 Symposium on Imitation in Animals and Artifacts.

Learning, Bottlenecks and Infinity: a working model of the evolution of syntactic communication

Simon Kirby
Language Evolution and Computation Research Unit,
Department of Linguistics, University of Edinburgh
simon@ling.ed.ac.uk

Abstract

Human language is unique in having a learned, arbitrary mapping between meanings and signals that is compositional and recursive. This paper presents a new approach to understanding its origins and evolution. Rather than turning to natural selection for an explanation, it is argued that general properties of the transmission of learned behaviour are sufficient to explain the particular properties of language. A computational model of linguistic transmission is described in which complex structured languages spontaneously emerge in populations of learners, even though the populations have no language initially, and are not subject to any equivalent of biological change. These results are claimed to be general and are explained in terms of properties of mappings. Essentially, as mappings are passed down through generations of imitators, syntactic ones are intrinsically better at surviving through the learning "bottleneck".

1 Introduction

Why does human language have certain properties and not others? This is the central question for linguistic explanation. Of particular interest to linguists are the properties that human language has that appear to make it unique among communication systems in the natural world. One such property is syntax: the mapping between meanings and signals in human languages is uniquely compositional and recursive. That is, the meaning of a signal is composed from the meanings of parts of that signal, and furthermore, arbitrarily complex meanings can be expressed by embedding signals inside signals.

A common approach in linguistics to the explanation of these basic properties of syntax is to appeal to innately given properties of our language faculty. The Chomskyan approach (e.g. Chomsky, 1986), for example, holds that our innate language acquisition device constrains directly what types of language we may learn. In this perspective, the question that started this paper can be answered by hypothesizing those properties as innately given. Another approach broadly termed "functionalism" holds that much of the constraints on variation amongst languages can be explained by appealing to the communicative functions of language (e.g. Comrie, 1981; Hawkins, 1988; Croft, 1990, and references therein). Put simply, language is the way it is because it is shaped by the way it is used (see Newmeyer, 1999; Kirby, 1999a, for reviews of the distinction between functional and innatist approaches). A dominant approach in evolutionary linguistics (e.g. Pinker and Bloom, 1990) combines functionalism and innateness by arguing that any innate language acquisition device would have been shaped by natural selection in our evolutionary past, and that the selective pressures would have been related to communication.

One way of looking at this paper is as an attempt at an alternative (though not incompatible) explanation for the origins of compositionality and recursion in human language; one which does not appeal to a strongly constraining innate language acquisition device, nor an explicit mechanism whereby the communicative aspects of language influence its form. Instead, the approach put forward here looks to general properties of the way in which linguistic information is transmitted over time for an explanation of its structure.

Another way of seeing this paper, however, is as a demonstration of the value of looking at the properties of behaviour that is repeatedly imitated[1] in a population. In this light, human language can be seen as a case-study of how repeated learning and use affects, over time, the structure of the behaviour being learned. Thinking about imitated behaviour in this way, we can begin to see suggestive parallels between information transmission via learning in a social/cultural context, and information transmission via reproduction in a biological context (see figure 1).

2 The learning bottleneck

The transmission of linguistic behaviour is a special case of the system diagrammed in figure 1. A language in its internal form is a mapping between meanings and signals

[1] I use the term "imitation" here as it has been used in this conference. See Oliphant (1997) for some discussion of why it may not be the best term to use in this context.

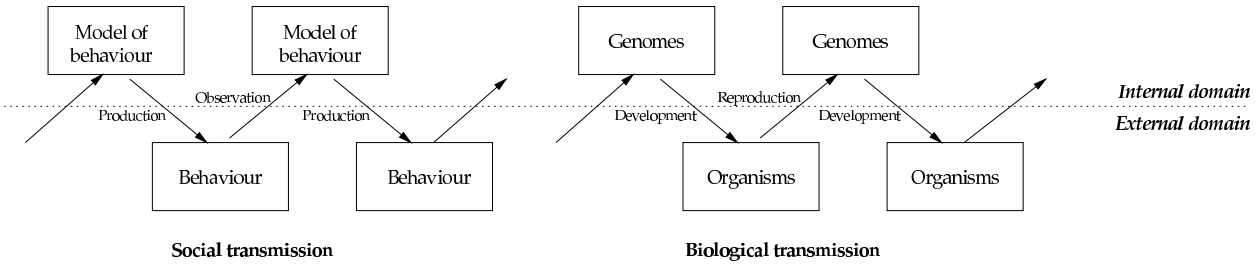

Figure 1: Both social transmission (via observation and imitation) and biological transmission, involve transformation of information between an internal and external domain. This transformation acts as a bottleneck on the flow of information through the respective systems, and may have an impact on the emergent structure of this information.

(typically strings of phonemes). That is, an individual in possession of a language will have some internal mental representation of that language that specifies how meanings are paired with strings. Languages also exists in an external form, however, as actual instances of signals being paired with meanings. The way in which a particular language (in both its forms) persists over time is by repeated transformation from the internal to external domains via actual use, and back into the internal domain via observation and learning (i.e. imitation).

The transformation between the internal and external domains of language — I-language and E-language in Chomsky's terms (see Chomsky, 1964; Andersen, 1973; Chomsky, 1986; Hurford, 1987; Kirby, 1999a; Hurford, 1999, for discussion) — act as a bottleneck on information flowing through the system. Just as the bottleneck on transmission of genetic information in biological systems eventually has implications for the structure of organisms that emerge, we should expect that the equivalent bottleneck in the linguistic system to have a role to play in the explanation of parts of linguistic structure. So, a particular piece of genetic information may not persist because the phenotype that it expresses may not survive to reproduce. In a similar way, a particular feature of language (that is, a particular part of the representation of the meanings-to-strings mapping) may not persist because the utterances it gives rise to may not reconstruct it through learning.

3 Simulating linguistic transmission

In order to test the hypothesis that, in the case of language, the learning bottleneck determines in part the eventual structure of what is being learned, the model of linguistic transmission has been implemented computationally. This type of simulation based approach has recently been adopted by several researchers in the evolutionary linguistics literature (e.g. Steels, 1997; Kirby and Hurford, 1997; Kirby, 1998; Hurford, 1998; Batali, 1998; Briscoe, 1998; Niyogi and Berwick, 1999) as it offers a "third way" between verbal theorising on the one hand and mathematical analytic approaches on the other.

The simulation consists of:

- a population of computational agents,
- a predefined meaning space. That is, a set of concepts which the agents may wish to express,
- a predefined signal space. That is, a set of concatenations of symbols available to the agents.

Each agent's behaviour is determined by a grammar internal to that agent which it has learnt solely through observing the behaviour of other agents in the population. The population model is generational in that agents "die" and are replaced with new agents which have no initial grammar. The agents are not prejudiced or rewarded according to their behaviour in any way. In other words, there is no natural selection in this model, and since each agent is identical at "birth", the only information that flows through the simulation from one generation to the next is in the form of utterances.

For the experiments described in this paper, a very simple population model was used: one in which there is, at any point in time, only a single speaker and a single hearer. The simulation cycle is outlined below:

c.1 Set up the initial population to have two agents: a speaker and a hearer, both of whom have no grammar.

c.2 **Repeat** some pre-specified number of times:

  c.2a Pick a meaning m at random from the meaning space,

  c.2b **If** the speaker can produce a string s for m using its grammar, **then** let the hearer *learn* from the pair $< s, m >$, **else** let the speaker *invent* a string s and let both hearer and speaker learn from the pair $< s, m >$.

c.3 Remove the speaker, make the hearer the new speaker and introduce a new hearer with no grammar.

c.4 **Go to c.2**.

Clearly the critical details for that determine the behaviour of this simulation are: the meaning space, the signal space, the grammatical representation, the learning algorithm, and the invention algorithm.

3.1 The meaning space

The meanings in the simulation are simple propositions made up of a predicate and two arguments. Typical propositions include, for example:

loves(mary,john)
admires(gavin,heather)
etc.

Certain predicates can take propositions as their second argument leading to propositions such as:

says(peter,loves(mary,john))
believes(heather,says(peter,loves(mary,john)))
etc.

In the simulations reported here, the number of different semantic atoms (predicates/arguments) can be varied, to test how it affects the evolving language. Because embedding of propositions is possible, the range of possible meanings is potentially infinite.

3.2 The signal space

The signals in the simulation are linear concatenations of symbols chosen from the 26 lower-case letters of the alphabet. The shortest signal would contain only one letter, whilst there is potentially no upper bound on signal length. There is no pre-defined equivalent of the "space" character. An example utterance (i.e. signal-meaning pair) of an agent that knew a language like English could be:

< `marylovesjohn`, loves(mary,john) >

3.3 The grammatical representation

Now that we have a specific meaning space and signal space in mind, it is useful to understand what makes a mapping between these two spaces "syntactic" in the way I am using the term here. I have mentioned two unique properties of the mapping that one finds in human languages: compositionality and recursion. For an agent with a language like English, a signal for a meaning is constructed by generating strings for subparts of that meaning and concatenating them in a particular order. So, for example, the meaning

knows(john,says(heather,loves(mary,peter)))

is mapped onto a string by finding the strings that correspond to john, knows and says(heather, loves(mary, peter)) and concatenating them in that order. This is what makes the language compositional. The language is also recursive because the construction of the string for says(heather, loves(mary, peter)) is carried out using the same procedure.

A non-compositional language, on the other hand, maps between meanings and strings in a quite different way. In such a language, the string corresponding to loves(mary, peter) might have no relation whatsoever to the string corresponding to loves(mary, john). In fact, there are degrees of compositionality, even with these quite simple meanings and signal spaces.[2]

The agents' internal representation of the mapping between meanings and signals must be able to express the different degrees of compositionality and recursion that are possible. For these simulations, a simplified form of definite clause grammar formalism was used. See figure 2 for examples of how different types of language can be expressed in this representation scheme.[3]

3.4 The learning algorithm

The learning algorithm will not be described in detail here. More complete coverage can be found in Kirby (1999b) and Kirby (1999c). The algorithm works incrementally, processing each string-meaning pair as the agent hears it. The induction takes place in two steps:

Incorporation The string-meaning pair is made into a single grammatical rule and this is added to the learner's grammar.

Generalisation The algorithm tries to integrate the new rule into the rest of the grammar by looking for possible generalisations. These generalisations are essentially subsumptions over pairs of rules. In other words, the algorithm takes a pair of rules from the grammar, and tries to find a more general rule to replace them with (within a set of heuristic constraints). This process of finding generalisation over pairs of rules continues until no new ones can be found, after which the algorithm halts and the agent is free to process the next string-meaning pair.

Rather than go into details of the induction algorithm, an idea of how learning works can be demonstrated with a few examples. Consider an agent that has no grammar and hears the utterance:

< `heatherlovesjohn`, loves(heather,john) >

The agent will incorporate this as the following rule:

S/loves(heather,john) → `heatherlovesjohn`

[2] If a more complex and fine-grained meaning representation were to be used, it would be clear that real human languages are actually only partly compositional too. This is particularly obvious if we look at morphology. The string `loves` can be thought of as compositionally derived from the strings for the meaning love and the meaning present-tense. However, the string `is` cannot be composed from parts of its meaning be+present-tense. Kirby (1998) discusses a possible explanation for these data in terms of a similar model to the one given here.

[3] Other types of representation are, of course, possible (see Batali, 1999, for a radical alternative) but this one was chosen partly for its familiarity to linguists.

Non-compositional	Partly compositional	Compositional and recursive
S/loves(heather,john) $\rightarrow$ `heatherlovesjohn` S/loves(john,heather) $\rightarrow$ `johnlovesheather`	S/loves(x, y) $\rightarrow$ N/x `loves` N/y N/heather $\rightarrow$ `heather` N/john $\rightarrow$ `john`	$S/p(x, y) \rightarrow N/x\ V_1/p\ N/y$ $S/p(x, q) \rightarrow N/x\ V_2/p\ S/q$ V_1/loves $\rightarrow$ `loves` V_2/knows $\rightarrow$ `knows` N/heather $\rightarrow$ `heather` N/john $\rightarrow$ `john`

Figure 2: Three types of language expressed in the formalism used by the simulation. The material after the slash on category labels is the semantic representation of that category. Semantic information is passed between rules using variables (in italics here).

Now, imagine that the agent hears a second utterance:

$<$ `heatherlovespeter`, loves(heather,peter) $>$

The learner incorporates this utterance, and now has a grammar with two rules:

S/loves(heather,john) $\rightarrow$ `heatherlovesjohn`
S/loves(heather,peter) $\rightarrow$ `heatherlovespeter`

There is now a way in which these two S rules can be generalised in such a way that they can be replaced with a single S rule:

S/loves(heather,x) $\rightarrow$ `heatherloves` A/x

This new rule refers to an (arbitrarily named) category A. In order that this rule may generate at least the string-meaning pairs that the old rules did, the inducer must add two A rules:

A/john $\rightarrow$ `john`
A/peter $\rightarrow$ `peter`

This type of subsumption, where the differences between two rules are extracted out into a separate set of rules, in itself is not particularly useful for learning, because the grammar as a whole will never become more general. However, this type of subsumption is paired up with another which can "merge" category names. Consider the state of the agent described above after hearing the following two utterances:

$<$ `marylovesjohn`, loves(mary,john) $>$
$<$ `marylovesheather`, loves(mary,heather) $>$

The grammar of the agent after incorporating these utterances and generalising the rules would be:

S/loves(heather,x) $\rightarrow$ `heatherloves` A/x
S/loves(mary,x) $\rightarrow$ `maryloves` B/x
A/john $\rightarrow$ `john`
A/peter $\rightarrow$ `peter`
B/john $\rightarrow$ `john`
B/heather $\rightarrow$ `heather`

Now, there are two "john" rules which are identical except for their category name. A subsumption of these two rules can be made simply by rewriting all the Bs in the grammar with As (or vice versa). If this is done with the grammar above, then this means that the two S rules can now be subsumed by one by extracting out "mary" and "heather". After another merging of category names, the grammar becomes:

S/loves(x, y) $\rightarrow$ A/x `loves` A/y
A/john $\rightarrow$ `john`
A/peter $\rightarrow$ `peter`
A/heather $\rightarrow$ `heather`
A/mary $\rightarrow$ `mary`

This grammar shows that the learner has generalised beyond the data given: the learner was given only 4 utterances, but can now produce 16 distinct ones.

3.5 The invention algorithm

So far, we have seen what the agents' meaning space and signal space looks like, and how they learn the grammars that allow them to map from one to the other. However, since the simulation starts with a speaker-agent with no language, and no language is provided from "outside" of the simulation, what has been described so far will not produce any utterances at all.

The agents must, therefore, have some way of inventing new strings for meaning which they cannot currently produce using their grammars. If the agent wishes to produce a string for a particular meaning, and that agent has no grammar at all, then a completely random string of symbols is produced. In the simulations reported below, the random strings vary between one and three symbols in length.

Although, this completely random invention strategy seems sensible where an agent has no grammar at all, or has a non-compositional grammar, it seems less likely where an agent is already in possession of a syntactic language. This latter situation is akin to a speaker of English needing to produce a sentence that refers to a completely novel object. It seems very implausible that the speaker will decide to invent a new "word" that stands in for the whole sentence. Instead, it seems likely that the speaker of a compositional language will understand that she can invent a new word for only that part of the sentence that relates to the novel meaning.

To simulate this, the invention algorithm used by the agents never introduces any new structure into an utter-

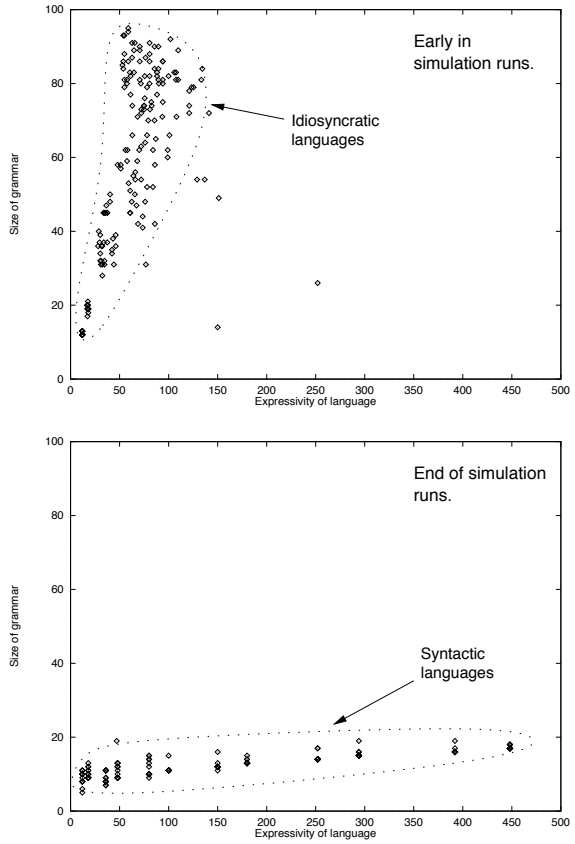

Figure 3: A scatter plot comparing early languages with those that emerge towards the end of the runs. Each point represents one run. The runs varied with respect to the size of the space possible meanings about which the agents produced utterances.

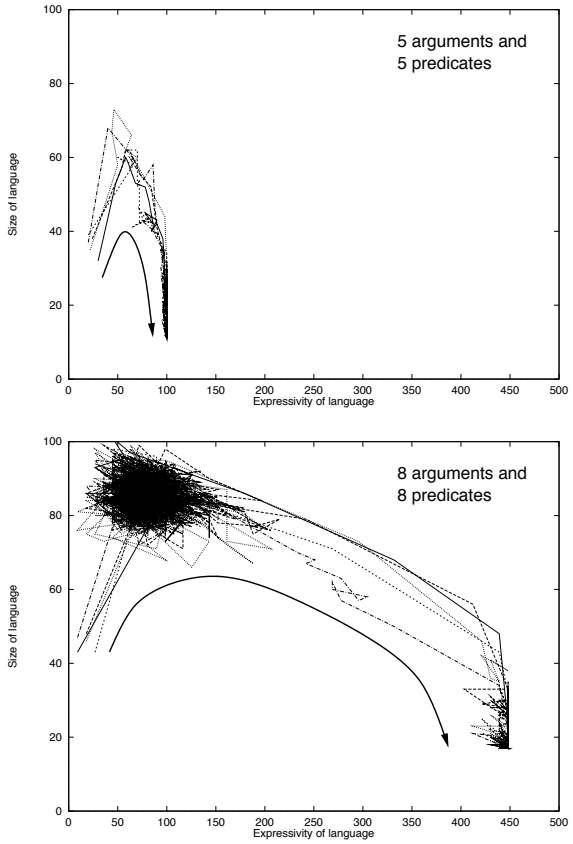

Figure 4: The movement over time of the languages in various runs of the simulation. The arrows show the overall direction of movement in expressivity/size space. The languages in the simulation start as vocabularies, grow rapidly but eventually become syntactic. For a larger meaning space (and a fixed bottleneck) the time it takes to achieve syntax increases.

ance, but similarly always to preserver any structure that already exists in the language. Again, for reasons of conciseness, details of the algorithm are not given here but can be found elsewhere (Kirby, 1998, 1999b).

4 Experimental results

This section describes several results of running the simulation described above. The simulator was designed to output three sets of data for each generation in a run: the actual grammar of the speaker, the size of grammar of the speaker (in number of rules), and the proportion of the meanings that the speaker expressed without recourse to invention. The last statistic allows us to estimate the expressive power of the speaker's grammar.

4.1 The emergence of degree-0 compositionality

For the first set of experimental results, only degree-0 meanings were used. In other words, no predicates such as believes or says were included in the meaning space of the agents. The size of the meaning space is varied from run to run by altering the number of distinct atomic predicates and arguments there could be. Each speaker in every run attempts to produce 50 randomly chosen degree-0 meanings in its lifetime. I will refer to this value as the *bottleneck size*, reflecting the fact that the languages in the simulations must repeatedly squeeze through a bottleneck of 50 samples to persist over time.

A variety of simulation runs were performed with the size of the meaning space varying from 12 possible meanings to 448 possible meanings.[4] The results of each simulation can be plotted on a graph of expressivity of language against size of grammar (this can be thought of as a plot of external size against internal size of a language). The expressivity measure is calculated by multiplying the proportion of meanings that a speaker produced without

[4]For implementational reasons, reflexive meanings were pruned from the meaning space. In other words, a proposition such as loves(john,john) is not allowed. The largest meaning space was made up of 8 possible atomic predicates and 8 possible atomic arguments.

invention by the size of meaning space. This gives us an estimate of the total number of meanings that a speaker's grammar is able of expressing. Two scatter plots of this type are in figure 3. Each point on the top plot is the situation in a simulation run after only 5 generations, whereas the bottom plot shows the result after 5000 generations (after which the languages are typically very stable).

Another way to visualise these results is to observe the movement of particular languages over time as they are transmitted from generation to generation. Figure 4 shows two sets of simulation runs, one with a fairly small meaning space, and one with a larger one. Notice that the behaviour of the languages in these runs is very similar: they start with low expressivity and medium size, rapidly increase in size with an approximately linear increase in expressivity, before eventually changing direction in the space with a rapid increase in expressivity and reduction in size. The final expressivity is determined exactly by the size of the meaning space.

On the first set of graphs I have distinguished between two types of language. Examples of these types are given below. In the simulation results, predicate atoms are labelled as ac0, ac1 etc. (standing for "action") whereas argument atoms are labelled as ob0, ob1 etc. (standing for "object"):

Idiosyncratic Early in the simulations, the languages are vocabulary-like, in that they tend to have no predictable correspondence between meanings and strings. In other words, they are non-compositional. For the majority of meanings, a corresponding string is simply listed in the grammar (although even early on there may be other rather non-productive rules). Here is a small subset of the rules in a grammar from a simulation with 8 possible predicates and 8 possible actions. The complete grammar had 43 rules and covered only 2% of the possible meaning space.

S/ac5(ob6,ob3) → ttm
S/ac4(ob2,ob4) → eue
S/ac6(ob5,ob4) → nx
S/ac3(ob1,ob3) → eib
S/ac4(ob6,ob5) → ve
S/ac2(ob1,ob6) → n
S/ac1(ob0,ob5) → ec
S/ac1(ob6,ob7) → mv
S/ac0(ob3,ob4) → xi
S/ac5(ob4,ob5) → h
S/ac4(ob3,ob4) → if
S/ac1(ob0,ob7) → j
S/ac3(ob1,ob0) → o
and 30 others...

Syntactic At the end of the simulation runs, the languages are able to express all the meanings in the meaning space of that particular run with a relatively small grammar. This is possible because syntax has emerged. The final languages exhibit complete compositionality as well as an emergent noun/verb distinction. The grammar below is an example of the end result of a run with 5 predicates and 5 arguments (the categories A and B are arbitrarily chosen by the inducer, and appear to correspond to noun and verb.)

$S/p(x,y) \rightarrow B/p\ A/y$ i A/x
A/ob3 → z
A/ob4 → qu
A/ob1 → f
A/ob0 → vco
B/ac3 → rr
B/ac2 → l
B/ac4 → b
B/ac1 → hta
A/ob2 → p
B/ac0 → qg

What these results show is that, for a large range of initial conditions for the simulation, syntax inevitably emerges.

4.2 The emergence of recursion

The semantic space in the simulations shown so far has been strictly finite. Only combinations of atomic predicates and two atomic arguments have been allowed. The simulation has also been run with a potentially infinite meaning space, using predicates like believes which take a propositional second argument. For these runs, there are always five possible atomic arguments, five possible "normal" predicates, and five possible "embedding" predicates. Each generation, the speakers produce 50 random utterances with degree-0 semantics (as in the previous simulations), followed by 50 random utterances with degree-1 semantics (one embedding), and finally, 50 random utterances with degree-2 semantics (two embeddings).

Unfortunately, it is impossible to plot the results of these runs in the same way as those of the previous simulations, because the expressivity of a language cannot be calculated as simply a number of meanings covered. However, the results of different runs is remarkably consistent,[5] and the behaviour of the system can be easily understood by looking at an example language as it changes over time. (The "subordinating" predicates are given the names su0, su1, etc. in the output of the simulation.)

Idiosyncratic The initial grammars are very similar to those in the previous degree-0 simulation runs. In other words, they too appear to be simple idiosyncratic vocabulary lists for a subset of the meanings in the space. One difference, of course, is that there are words for the more complex degree-1 and degree-2 meanings. Here is a small subset of the language we are tracking early in the run:

S/ac3(ob1,ob4) → de
S/ac2(ob0,ob1) → ak
S/ac2(ob0,ob3) → t
S/ac3(ob0,ob1) → sdx
S/ac4(ob0,ob3) → g

[5] See Kirby (1999c) for a rather different way of visualising the results of this type of simulation run.

S/ac0(ob2,ob4) → vnu
S/ac0(ob1,ob3) → gj
S/ac0(ob3,ob4) → nui
S/su4(ob2,ac0(ob1,ob2)) → oeb
S/su4(ob4,ac4(ob2,ob1)) → ew
S/su1(ob3,ac2(ob2,ob4)) → vri
S/su4(ob3,ac2(ob4,ob0)) → y
S/su4(ob4,ac2(ob3,ob0)) → pff
S/su1(ob0,ac3(ob1,ob3)) → fi
S/su2(ob0,su3(ob2,ac0(ob1,ob2))) → jt
S/su1(ob4,su2(ob0,ac2(ob0,ob3))) → vz
S/su2(ob0,su2(ob3,ac4(ob3,ob1))) → z
S/su1(ob0,su1(ob2,ac1(ob0,ob3))) → gb
S/su0(ob4,su4(ob3,ac0(ob0,ob1))) → r
S/su4(ob1,su3(ob2,ac2(ob3,ob4))) → cr
S/su3(ob1,su1(ob3,ac3(ob1,ob4))) → sz
S/su2(ob0,su2(ob1,ac0(ob1,ob2))) → ixh
and 94 others...

Degree-0 compositionality After 100 generations, this language has changed, again in a way similar to the previous runs. The proportion of degree-0 meanings that are produced without invention has climbed rapidly, so that now the speakers can express every degree-0 meaning using this language. The listing below gives a small subset of the language, showing how a compositional encoding for degree-0 meanings has emerged. There are three major categories: D is a verbal category, and A and C appear to be case-marked nominals, with A acting like a nominative, and C like an accusative. These categories occasionally appear in partly compositional rules for more complex meanings, but generally, the degree-1 and degree-2 part of the meaning space is still expressed idiosyncratically, and therefore with poor coverage.

$S/p(x,y)$ → gj C/y z A/x D/p
A/ob2 → dl
C/ob1 → ovp
A/ob1 → tej
C/ob2 → x
D/ac3 → xe
D/ac0 → m
A/ob3 → qp
A/ob0 → h
C/ob0 → y
D/ac2 → c
C/ob4 → i
D/ac1 → b
C/ob3 → h
S/su1(x,su4(ob1,ac4(ob0,ob3))) → C/x jwyjtejdbznuy
S/su1(ob4,su0(ob1,ac4(ob4,ob2))) → htejyjndbznuy
S/su3(x,su0(ob0,ac0(ob0,ob4))) → qzjw A/x ya
and 68 others...

Syntax and recursion At the end of the simulation run (here the run lasted for 1000 generations) the language covers the entire meaning space. That is, the speakers can produce strings for any degree-0,1 or 2 meaning without recourse to invention. Furthermore, the speakers could produce strings for an infinite range of meanings with any depth of embedding. This is possible due to the appearance of recursion in the syntax, as shown below. In this language the nominal system has simplified to one form that is used both for accusative and nominative, and a new verbal category has emerged for predicates that take a propositional second argument. The second S rule in this grammar is the recursive one, as its last right-hand side category is also S.

$S/p(x,y)$ → gj A/y f A/x B/p
$S/p(x,q)$ → i A/x D/p S/q
A/ob3 → qp
A/ob2 → dl
B/ac2 → c
B/ac0 → m
A/ob1 → tej
A/ob4 → n
B/ac4 → e
A/ob0 → h
B/ac1 → b
B/ac3 → wp
D/su4 → m
D/su1 → u
D/su2 → g
D/su0 → p
D/su3 → ipr

Once again, we have seen a movement of languages in the simulation from an initial random, idiosyncratic, vocabulary-like stage, to one in which all the meanings can be expressed using a highly structured syntactic system.

5 Linguistic transmission favours syntactic mappings

The simulation results in the previous section show that compositional, recursive language emerge in a population which initially has no language, even where there is no selection pressure on individuals to communicate well, or indeed any biological evolution at all. Purely through the process of being repeatedly mapped from an internal form as a grammar to an external form as utterances and back again, language evolves. Syntactic structure appears to emerge inevitably when a mapping between two structured domains must be passed on over time through a learning bottleneck. Why might this be? What are the properties of syntactic mappings and learning bottlenecks that make this inevitable?

Figure 5 is a schematic representation of a possible mapping between two spaces. Let us assume, for the purposes of this explanation, that the structure in the two spaces being mapped onto each other is spatial. That is, two points in a space are more similar if they are close together in the representation of that space than if they are further apart. The mapping in this diagram therefore does not preserve structure from one space to the other. In other words, there is a random relation between a point in one space and its corresponding point in the other space.

Now, imagine that this mapping must be learned. In the diagram, some of the pairings are shown in bold — if these where the only ones a learner was exposed to, would that learner be able to reconstruct the whole mapping? Not easily: the only way a random mapping could

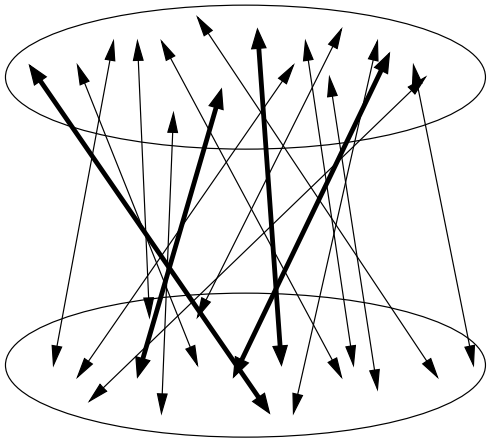

Figure 5: A non-structure preserving mapping between two spaces with spatial structure. The bold lines indicate an imaginary subsample of the mapping that might be evidence for a learner. This mapping could only be learnt by a learner with a very specific prior bias.

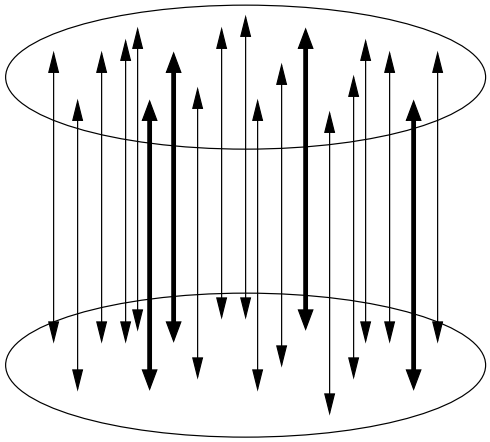

Figure 6: A mapping in which structure is preserved. The bold lines indicate an imaginary subsample of the mapping that might be evidence for a learner. This mapping is more likely to be successfully learnt by a learner with a more general prior bias.

be reliably learnt from a subset of pairings would be if the learner had a very informative and domain specific prior bias to learn that particular mapping. Whilst this is possible if the spaces are finite, it is in principle impossible where they are potentially unbounded.

Figure 6 on the other hand, shows a mapping in which structure in one space is preserved in the other. Given the sample in bold, it seems that a learner has a higher chance of reconstructing the mapping. A learner that is biased to construct concise models, for example, would learn this mapping more easily than that in the first figure. Importantly, this bias is more likely to be domain general than one that explicitly codes for a particular idiosyncratic mapping. Furthermore a model can be constructed that would map the spaces even if they were potentially infinite in extent.

The first type of mapping (figure 5) is very like the vocabulary-like systems described in the previous section, where points in the meaning space where arbitrarily paired with points in the signal space. To put it more precisely, similarity between two point in either space is no guarantee of similarity between the points that they map onto in the other space.[6] The second type of mapping is much more like a syntactic system, where strings have a *non*-arbitrary relation with the meanings they correspond to. Similarity between two strings in these systems is a very good indicator of similarity between their corresponding meanings.

In the second set of simulations, as in real language, both the meaning space and the signal space are potentially infinite in extent. This means that it is in principle impossible for a learner to acquire a mapping of the first type. We can conclude, then, that where a learner is exposed to a sub-sampling of the string-meaning pairings in a language — in other words, where there is a learning bottleneck — idiosyncratic, vocabulary-like languages are unlikely to be learned successfully. The initial, random languages in the simulations are unstable over time as long as the bottleneck is tight enough that they cannot fit through intact. This is not a feature of syntactically structured languages, however. Structure in the mapping improves the survivability of that mapping from one generation to the next.

What we are left with is a very general story about the (cultural/social/historical) evolution of mappings. Structure-preserving mappings are more successful survivors through the learning bottleneck. This fact, coupled with random invention of pairings in languages that have incomplete coverage of the meaning space, and the unboundedness of the meaning and signal spaces, leads inevitably to the emergence of syntax.

[6] For an example of this type of mapping consider: Edinburgh is more like Glasgow than it is like Erinsborough (the fictional setting for the Australian soap *Neighbours*), and yet the string *Edinburgh* is more like *Erinsborough* than it is like *Glasgow*.

Acknowledgements

This work benefited greatly from conversation with and comments from Jim Hurford, Mike Oliphant, Mark Ellison and Ted Briscoe and was supported by ESRC grant R000237551.

A shorter presentation of some of the simulation results appears as Kirby (1999c), and the learning algorithm is discussed in more detail in Kirby (1999b).

References

H. Andersen. Abductive and deductive change. *Language*, 40:765–793, 1973.

John Batali. Computational simulations of the emergence of grammar. In James Hurford, Chris Knight, and Michael Studdert-Kennedy, editors, *Approaches to the Evolution of Language: Social and Cognitive Bases*, pages 405–426, Cambridge, 1998. Cambridge University Press.

John Batali. The negotiation and acquisition of recursive grammars as a result of competition among exemplars. In E.J. Briscoe, editor, *Linguistic evolution through language acquisition: formal and computational models*. Cambridge University Press, Cambridge, 1999. Forthcoming.

E. J. Briscoe. Language as a complex adaptive system: co-evolution of language and of the language acquisition device. In P. Coppen, H. van Halteren, and L. Teunissen, editors, *8th Meeting of Comp. Linguistics in the Netherlands*, pages 3–40, Amsterdam, 1998. Rodopi.

Noam Chomsky. *Current Issues in Linguistic Theory*. Mouton, 1964.

Noam Chomsky. *Knowledge of Language*. Praeger, 1986.

Bernard Comrie. *Language Universals and Linguistic Typology*. Basil Blackwell, 1981.

William Croft. *Typology and universals*. Cambridge University Press, Cambridge, 1990.

John A. Hawkins. Explaining language universals. In John A. Hawkins, editor, *Explaining Language Universals*. Basil Blackwell, 1988.

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

James Hurford. Social transmission favours linguistic generalisation. In Chris Knight, James Hurford, and Michael Studdert-Kennedy, editors, *The Emergence of Language*, 1998. To appear.

James Hurford. Expression/induction models of language evolution: dimensions and issues. In E.J. Briscoe, editor, *Linguistic evolution through language acquisition: formal and computational models*. Cambridge University Press, Cambridge, 1999. Forthcoming.

Simon Kirby. Syntax without natural selection: How compositionality emerges from vocabulary in a population of learners. In Chris Knight, James Hurford, and Michael Studdert-Kennedy, editors, *The Emergence of Language*, 1998. To appear.

Simon Kirby. *Function, Selection and Innateness: the Emergence of Language Universals*. Oxford University Press, Oxford, 1999a.

Simon Kirby. Learning, bottlenecks, and the evolution of recursive syntax. In E. J. Briscoe, editor, *Linguistic evolution through language acquisition: formal and computational models*. Cambridge University Press, Cambridge, 1999b. Forthcoming.

Simon Kirby. Syntax out of learning: the cultural evolution of structured communication in a population of induction algorithms. In *European Conference on Artificial Life '99*, 1999c. Under review.

Simon Kirby and James Hurford. Learning, culture and evolution in the origin of linguistic constraints. In *Fourth European Conference on Artificial Life*, pages 493–502. MIT Press, 1997.

Frederick J. Newmeyer. *Language Form and Language Function*. MIT Press, Cambridge, MA, 1999.

Partha Niyogi and Robert Berwick. The logical problem of language change. *Journal of Complex Systems*, 1999. In press.

Michael Oliphant. *Formal Approaches to Innate and Learned Communication: Laying the Foundation for Language*. PhD thesis, UCSD, 1997.

Steven Pinker and Paul Bloom. Natural language and natural selection. *Behavioral and Brain Sciences*, 13:707–784, 1990.

Luc Steels. The synthetic modelling of language origins. *Evolution of Communication*, 1(1), 1997.