Nothing1

Nothing2

Contents

List of Figures

Acknowledgments

Chapter 1 is a version of T. Teal, D. Albro, E. Stabler and C.E. Taylor. Compression and Adaptation. In *Proceedings of the Fifth European Conference on Artificial Life*, Springer-Verlag, Germany, 1999.

This work was supported by grant #5BR9720410

I would like to thank members of the UCLA Cognitive Science LIS Research Group for helpful discussions.

I would also like to thank Edward Stabler, who guided this research, and Daniel Albro who made significant contributions to this project.

Charles E. Taylor supervised the research that forms the basis of this thesis, and I thank him for invaluable discussions and support.

Part I

Compression and Adaptation

1 Introduction

Why are some systems more adaptable than others? A core feature of nearly all successful adaptive systems is the ability to distill experience into schemas, models or theories and then employ those abstracted structures in new circumstances. Information about the environment is not simply recorded as a look-up table, with generalization to new situations happening only at look-up time. Rather, it is plausible that salient features of situations are noted, and then departures from expectations are noted. This is the essence of compression.

Gell-Mann [6] has argued that a compressed form "is usually approximate, sometimes wrong, but it may be adaptive if it can make useful predictions including interpolation and extrapolation and sometimes generalize to situa-

tions very different from those previously encountered. In the presence of new information from the environment, the compressed schema unfolds to give predictions or behavior or both." We would like to know more about the role of compression in adaptation. For example, can we identify features of compression that make some forms more or less likely to be successful? What sorts of compression are best? How much is desirable?

This perspective on learning is not new, especially for cultural evolution. It is evident, for example, that there is frequently a practical necessity to simplify things so that we can understand them. Hence the ubiquitous use of simplified models in science. Which models are themselves best, is also a matter of simplicity. William of Ockham, among others, has observed that "it is futile to do with more what can be done with less" [23]. In another domain, Chomsky [4] has placed the desirability of compressed, "minimal" grammar at the heart of his theory of human language. Nonetheless, there have been few attempts to explore, in a systematic way, how compression alone may affect adaptation.

In this paper we first state some heuristics that we conjecture to be generally, if not always, true. We then introduce some elementary definitions and principles about data compression and formal languages. We have chosen to work with

formal languages because it is possible to discuss them concretely and because such systems quite clearly show changes in their ability to generalize [18, 9, 14, 13]. We next discuss how compression can occur as an agent learns a language by hearing examples of it. We also describe some experiments where agents learn the languages with compressed grammars, and where the languages evolve as a result. Finally, we discuss some features of compression and evolution in our system in the light of our conjectures. We emphasize that while our discussion will be directed mostly to cultural evolution - scientific theory and language, we believe these heuristics apply to other adaptive complex systems, such as organic evolution, immune systems, and neural networks.

2 Conjectures about compression and adaptation

Conjecture 1 ***Compression aids in generalization.***

From a series of observations like "Crow A is black" and "Crow B is black" we compress a look-up table of crows and their colors to the generalization that

"All crows are black." The generalization is clearly smaller, more "compressed" than a list of many instances. The precise characterization of the circumstances in which such generalization is appropriate, the problem of induction, is a long-standing philosophical problem.

The history of science is a history of finding generalizations that allow a succinct statement of the facts. Following the invention of the spectroscope, the spectral emissions from various elements, including hydrogen, were cataloged. About 1885 J. J. Balmer discovered formulae that would describe the frequencies for hydrogen emissions both compactly and accurately, though they were simply formulas without a model behind them. In 1913, Niels Bohr published a model for the atom that would compactly describe emissions from hydrogen, and several other atoms, in very compressed and desirable form.

While the history of science can be regarded as a series of successively better compressions, it should also be recognized that the resulting compressions may make predictions that are only approximate or even wrong. Although models can give us insight into systems, the actual model used greatly affects the predictions that can be made and the types of behaviors that can be explained. Many scientific theories, no matter how well they might compress a set of

observations, are subsequently proven wrong.

Conjecture 2 ***Compression occurs more easily in a "smooth", as opposed to "rugged", string space.***

Related or connected sets of observations form a better basis for generalization than do similar or unrelated ones. We would like to be concrete about the meaning of "related or connected" in this context. Kauffman [11] has explored the use of adaptive landscapes in a variety of contexts, and we build on his example by exploring a string space of languages, below.

Unrelated observations, like Lord Morton's mare for Darwin's theory of inheritance, can make theory formation quite difficult [17]. As a result, it is thought generally better to focus initial scientific study on simple and well-defined systems, where the smoother space is more easily explored, as Mendel did.

Smoothness and ruggedness are, to some extent, a property of the substitution operators. In molecular evolution for example, adding or deleting tandem repeats of 2 or 4 nucleotides is often easier than adding or deleting a single nucleotide [28].

Conjecture 3 ***Constraints from compression make it likely that natural languages come to have smooth string spaces.***

Language learners are regarded as systems that aim to identify rule systems that describe the (infinite) language of the community on the basis of finite evidence. This can only be successful in certain circumstances, whether one assumes that success is perfect identification in the limit (the "Gold" paradigm) [7], or that success is feasible convergence to arbitrarily good approximate identification [12, 27]. Since humans clearly learn to speak natural languages that can generate an infinite number of sentences, and do so largely from examples and without conscious awareness of grammatical rules, most linguists believe that, roughly speaking, when a young person is confronted with a new sentence they will "try out" candidate grammars and then, from those grammars that can accommodate it, choose the one that is simplest [3, 8, 24].

Of course, human language learners hear ungrammatical sentences and sentence fragments, but this "noise" apparently does not make communication with the community impossible. We postulate that sufficiently rare complications are typically not incorporated; they will be cataloged as exceptions or simply ig-

nored. As a result, there will be a natural selection for languages that are progressively smoother. This is perhaps analogous to the way that biological systems sometimes "solve" other NP-complete problems. They seem to employ only those parts of the problem space that can be solved simply, even though the general problem might be insoluble [20].

3 Data compression

Data can be compressed only when it contains some regularity that can be exploited. [22]. When there is a regularity, listing it repeatedly makes for an eliminable redundancy.

It is important to recognize that some schemes might provide very efficient coding of data, but would involve lengthy and complicated specifications of the decoder. The decoder could, for example, simply contain a list of any finite set of strings to be encoded. A better measure of compression would recognize both costs. The *minimum description length algorithm (MDL)* we use accomplishes this. In the formal language framework the sum of the *grammar-encoding-length* (the cost of the rule set) and the *data-encoding-length* (the cost of the coding

of the data) is minimized [15, 21].

4 Formal Languages

A (formal) language is a set of strings of symbols drawn from some alphabet. Here we shall be concerned only with *regular* languages, languages that can be accepted by a *finite automaton*. Such automata can be described by a quintuple, $(Q, \Sigma, \delta, q_0, F)$, where Q is a set of states, Σ is an input alphabet, $q_0 \in Q$ is the initial state, $\delta \subseteq Q \times \Sigma \times Q$ is a transition function, and $F \subseteq Q$ is the set of final states [10]. An automaton is said to be *deterministic* if the transition δ is a function $\delta : Q \times \Sigma \rightarrow Q$ with respect to its first two arguments. Associated with each finite state automaton is a *transition diagram*, with an arc connecting states q_1 and q_2 labeled with vocabulary element a if, and only if, $(q_1, a, q_2) \in \delta$. The automaton accepts a string s if, and only if, the string labels a path from the initial state to a final state.

For purposes of illustration, consider languages `1-s` and `1-r`. Language `1-s` consists of the strings `aaaa, aaab, aaba, abaa` and `abba`. Language `1-r` consists of `aaab, abaa, aaba, abbb,` and `bbab`. Their respective "prefix tree"

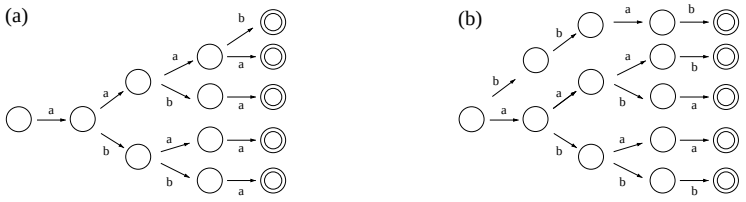

Figure 1: Transition state diagrams for uncompressed grammars: (a) language 1-s (b) language 1-r.

transition diagrams are shown in Figure 1.

Since each string can be specified by a path through a deterministic automaton, we calculate one simple approximation of the data-encoding-length in bits given by the formula:

$$\sum_{i=1}^{m} \sum_{j=1}^{|s_i|} \log_2 z_{i,j},$$

where m is the number of sentences in the sequence of strings encoded, $|s_i|$ is the length of the i'th string s_i, and $z_{i,j}$ is the number of ways to leave the state reached on the j'th symbol of sentence s_i. (A more succinct encoding is obviously possible when the probabilities of the transitions are not uniform. The simple approximation suffices for purposes of this preliminary investigation.)

To specify the automaton itself, we must specify all the triples $(q_1, a, q_2) \in \delta$ and we must also specify the final states, so we calculate the grammar encoding

length:

$$|\delta|[2(\log_2|Q|) + \log_2|\Sigma|] + |F|[\log_2|Q|],$$

where $|\delta|$ is the number of triples in δ and $|F|$ is the number of final states in F. The MDL of a language is defined as the sum of its grammar-encoding-length and its data-encoding-length. For language `1-s` the MDL is 119.3 and for `1-r` it is 168.0.

We refer to language `1-s` as "smooth" and language `1-r` as "rugged." The reasons for this are evident from Figure 2, which shows how the strings in each set are related by symbol substitution. (More generally, one might use deletion and duplication operators, as well as symbol-substitutions. Duplication and deletion are known to be important for the evolution of DNA and for natural language, but these operators add considerable complication for some of the comparisons we are interested in, so for this paper we shall restrict our attention to operators that are simple symbol substitutions.)

Each string in `1-s` is one symbol-substitution away from its nearest neighbor. The symbol in only one position is changed from one string to the next, and the strings are very similar or regular, while in `1-r` no string is closer than

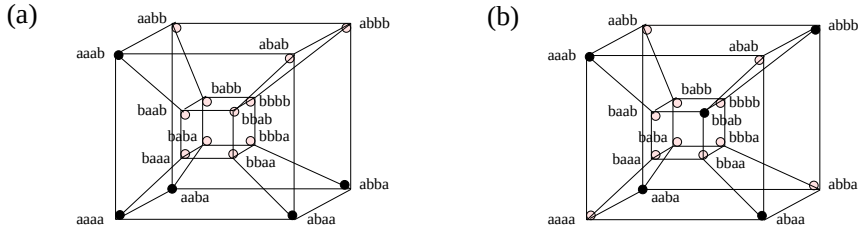

Figure 2: Hypercube representation of the landscape for strings of length 4. Nodes connected by lines are one symbol-substitution away from each other. Black dots are included in the language. Gray dots are not included. (a) Representation of language `1-s` (b) Representation of language `1-r`.

two substitutions away, making the strings not very related. Further, in the hypercube representation, there is a hyperplane in `1-s` that provides regularity which might be utilized for compression, while such regularity is not evident in `1-r`. This occurs in much the same way that the equation of a line offers a compressed representation of a set of data points in linear regression. These two grammars also differ in their MDL measures: the MDL of `1-s` is only 71% that of `1-r`.

5 Grammar compression

Our model of language learning will be that of a child, who when listening to an adult speaking a language s/he does not yet understand, tries out different

candidate grammars that might accept that language. The minimalist theory supposes that those grammars found suitable are then examined further, and the grammar with the shortest MDL is preferred and tentatively accepted as describing the parent's language.

In our study, an agent is exposed to all the legal sentences in the language – say of `1-s` or `1-r`. It constructs an automaton with associated grammar as shown in Figure 6, above. The agent then compresses the original grammar by attempting to combine states and transitions in such a way that the outcome is still a deterministic automaton and then combines them iff the MDL of the new automaton is smaller than that of the starting one. Given a machine $A = (Q, \Sigma, \delta, q_0, F)$, the result of merging states $q_i, q_j \in Q$ is the machine A' which has these two states replaced by a new state q_{ij} as follows:

$$A' = ((Q - q_i, q_j) \cup \{q_{ij}\}, \Sigma, \delta', q_0', F')$$

where: $q_0' = q_{ij}$ if q_0 is either q_i or q_j, $F' = (F - q_i, q_j) \cup \{q_{ij}\}$ if either q_i or $q_j \in F$, and δ' is the result of replacing all instances of both q_i and q_j by q_{ij} in all the triples (q_n, a, q_m) that define δ. This is applied recursively in a

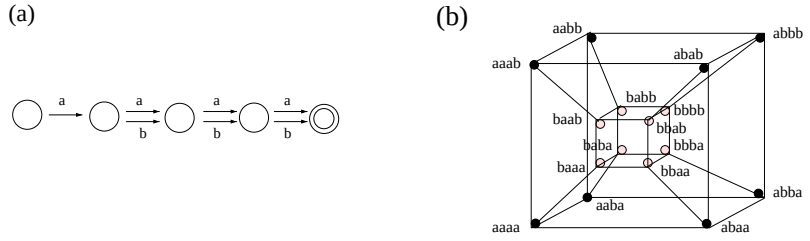

Figure 3: (a)Transition diagram of the compressed grammar for language `1-s`. (b) Hypercube representation of the sentences included in the compressed grammar.

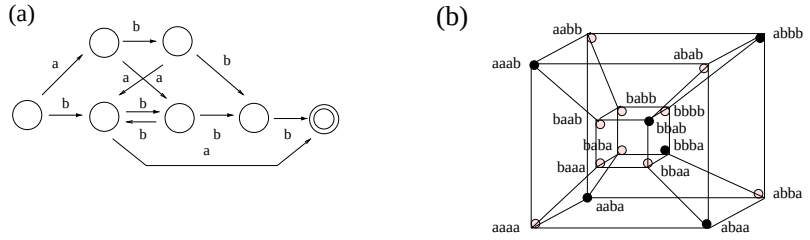

Figure 4: (a)Transition diagram of the compressed grammar for language `1-r`. (b) Hypercube representation of the sentences included in the compressed grammar.

hill-climbing manner.

The compressed transition diagrams and hypercubes for languages `1-s` and `1-r` are shown in Figures 9 and 10.

The compressed `1-s` is much smaller than that of the original ($MDL = 56.8$ vs. 119.3), due to changes both in the number of states (11 to 5) and in the number of transitions (12 to 7). All of the accepted strings in this example are of

the correct length, but the compressed version generalizes to include the entire outer cube of sentences in the hypercube. In "real life" such a generalization may or may not be desirable.

The compressed version of language `1-r` is also substantially smaller than the original (MDL = 92.6 vs. 168.0). Again this resulted both from a decrease in the number of states (15 to 7) and in the number of transitions (15 to 11). Compression did not lead to new planes or observable regularities, except that the one new sentence of length 4 which was added, `bbba`, was also a distance 2 from its nearest neighbor. Here, however, generalization was such that sentence lengths other than 4 were allowable, ranging from lengths of 2 to infinity, so long as certain regularities, such as embedded tandem repeats of `bb`, are observed.

6 Compression of languages

Compression results from exploiting regularities in the data. We expect that languages with smooth string spaces contain more regularity to be exploited than languages with rugged string spaces. Therefore, we expect more compression from smooth languages. Is this the case?

We created 40 languages with each of several degrees of smoothness. Each consisted of 10 strings, with lengths of 6 symbols drawn from the alphabet $\{a, b\}$. Smoothness was varied as follows: *distance-1, distance-2, distance-3* and *random*. All strings were 1, 2, 3 or random symbol-substitutions, respectively, from their nearest neighbor, drawn successively from a random starting string.

These languages were presented to agents who created the finite state automata grammars for them, then compressed the grammars using the algorithm described above. Compression was measured by MDL of original and compressed grammars. Error was measured by presenting the agents with all possible strings of length 6, then counting the number that were accepted but were not strings in the original language. Recall that the compression algorithm will always accept the original language, so error can also be viewed as the amount of generalization (to strings of length 6) that the grammar achieves; it says nothing about error in recognizing strings of other lengths.

As would be expected, smoother languages could be encoded more economically than the rugged ones. The amount of compression that was achieved is measured by the compressed MDL which was: 126.6 for distance-1, 199.6 for distance-2, 174.8 for distance-3 and 181.4 for the random language.

It is evident that compressed MDL was much less (i.e. more compression was possible) for the distance-1 languages than for the distance-2 ones. This was expected. But the grammars for distance-3 and random languages suddenly became more compressible.

One possibility for the higher compression of distance-3 and random might be that they were unable to extract regularity and simply accepted more strings into the language. The mean numbers of errors was consistent with this explanation: 7.18 for distance-2, 29.37 for distance-3 and 27.42 for random languages. However, the mean number of errors for distance-1 was also large, 23.48, so at best the explanation is still unclear. It is possible that this will change if the agents are presented with more examples, so that the cost of encoding the data is significantly higher for distance-3 and random landscapes. We are attempting to understand this better.

7 Evolution of language

We explored the role of compression in the evolution of language in a simple way. Each language initially began with 10 strings of length 6, drawn from the

alphabet $\{a, b\}$ as above. There were 10 replicates of distance-1 languages and 10 of distance-2. The parent in generation n then produced 10 sentences, at random from its language, and presented these to generation $n+1$, who would generate the appropriate grammar. The generation $n+1$ would then compress that grammar and, with the compressed grammar, produce 10 more sentences for generation $n+2$ and so on. This was continued for 10 generations.

There were clear changes observed with all languages. Foremost among the changes were the numbers of strings in the language. While all began with an average 7.7 strings without duplicates, after 10 generations this had changed to an average of 11.8 for distance-1 and to 6.1 for distance-2. This is statistically significant at the .05 level by a paired t-test. Transmission from generation 0 to 10 was lossy, with the average number of strings that were included in the original set of examples lost by generation 10 being 2.3 for distance-1 and 5.3 for distance-2.

The mean smallest distance between strings stayed about the same in both distance-1 (from 1.0 to 1.18) and distance-2 (from 2.0 to 1.72). We had expected that smoothness would increase with time, but this apparently was not the case since there were not significant changes. An increase must be expected from

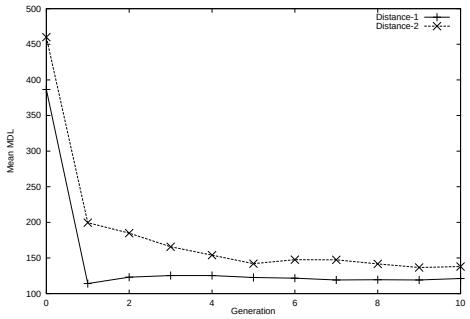

Figure 5: Mean Minimum Description Lengths for evolving languages of symbol substitution distances 1 and 2 for generations 0 to 10.

the distance-1 languages, because they began at the lowest value possible, and there is nowhere else to go but up. The mean distance did increase slightly. For the distance-2 languages, however, the mean smallest distance decreased only slightly.

Somewhat paradoxically, the mean MDL for both grammars decreased, in spite of their lack of change in ruggedness. Figure 5 illustrates the change in MDL which occurred, showing a large difference in the early generations, with only modest change later.

It is clear that while the complexity of the grammar was decreased in both, the distance-2 grammars remained more complex, though they admitted a much smaller number of sentences.

8 Discussion

We have examined several conjectures about compression in adaptive complex systems. We employed agents that could learn and evolve one class of formal languages, using one compression algorithm. This learning and evolution was based on syntax alone, without any reference to semantics, which is likely to be important [13]. Our results, while so limited, do seem illuminating.

Conjecture 1: Compression aids in generalization This was examined with systems that learned and compressed languages of varied ruggedness. In all cases we observed a compression of grammar, and in nearly all cases the compressed grammars generalized to admit new strings into the language.

It should be recognized that such generalization may be desirable, because it reduces the complexity of rules, but it can also admit mistakes. To date we have made only a cursory study of the tradeoff between MDL and error in our system, but there clearly are important issues that will warrant further study.

Conjecture 2:Compression occurs more easily in a "smooth", as opposed to "rugged", string space We explored compression in systems where strings were 1-, 2-, or 3- symbol substitutions apart from their nearest neighbor or were ran-

domly placed in the sentence space. We observed that even the uncompressed grammars were smaller for the smooth than for the rugged languages. Compression, as measured by compressed MDL, was clearly greater in in languages where sentences were 2 symbol substitutions apart than if they were only 1 substitution apart. We interpret this to mean that patterns can more easily be identified and exploited by the compression algorithm in the smoother language. However, where smoothness is still less, in the distance-3 and random languages, the compression is also greater than for distance-2 languages. The reason(s) for this remain unclear, but there is a correlation between compression ratio, error rate, and number of examples presented that is important here and needs further exploration.

Conjecture 3: Constraints from compression make it likely that natural languages come to have smooth string spaces

For the purposes of this paper we accept as true the theory that when learning language we (a) compress grammar with simple rules and (b), all else being equal, apply these compressed rules preferentially to new situations. The agents in our study automatically created grammars which admitted all legal sentences. These grammars eliminated some redundancy, but retained logical equivalence.

They were already *programmed*, quite literally, to conform to this theory. They were also programmed to compress those grammars, when so directed.

Here it is important to distinguish carefully between the smoothness of a language and the complexity of the grammar needed to describe it. While related, they are not the same. Both distance-1 and distance-2 languages evolved into languages with sentences that were, on average, separated by about the same number of symbol substitutions as when they started. That is to say, they did not become smoother in the sense of becoming more connected or to consist of strings lying together on the same hyperplane. At the same time, the grammars describing them came to have smaller MDLs. That is, both languages came to be described by simpler grammars. When we generated languages for compression, we observed that the smoother languages did have, on average, smaller MDLs, but not invariably so. Clearly there is some subtlety. The suggestion here is that grammatical complexity does, indeed, become simpler, but that this is not the same as saying that the languages become smooth, as smoothness is used in this paper.

In his study of evolving bit strings, albeit with semantic content, Kirby [13] observed two phase transitions in grammatical structure. The first of these

occurred when the languages became suddenly more expressive, with a concomitant increase in grammatical complexity. The second phase change occurred after a high degree of expressivity was achieved, then the grammatical complexity suddenly started to become much less. That study, and ours, are in agreement that even in such simple cases as evolving bit strings there are unlikely to be simple rules about changes in smoothness of languages, or grammatical complexity – though in the long run both studies did result in simpler grammars after sufficient time.

In summary, we observed broad agreement with the conjectures made prior to the start of the study. It is evident, however, that even in our simple system there is significant subtlety that must be recognized in the inductive tradeoff of generalization through compression versus error of overgeneralization.

Part II

The Effects of Acquisition on Language Evolution

1 Introduction

A key feature of nearly all successful complex adaptive systems is the ability to distill information about the environment into schemas and then use these models to make predictions or adapt to new situations. Instead of simply recording information in a look-up table, compact representations are created by identifying and using structure that exists in the data [22]. This compression not only allows large amounts of information to be stored more efficiently, but also enables the system to generalize.

Learning, as the process of finding a hypothesis that explains some observed data, can also be seen in this context as a type of compression [8]. There are

many ways to choose candidate hypotheses for testing. One would think each should lead to different types of generalizations. As these are passed on, each generation learning through its own experience, the different ways to generalize and accept hypotheses might lead to different sorts of generalizations about the world. The relative success of one method or chosen candidate hypothesis versus another will depend on what sort of structure there is in the data, which can be generalized. We are interested in how learning alone can change generalizations or understanding over time. This type of behavior is observed in many contexts, from data compression algorithms to communication and cultural evolution.

Language learning provides an especially good paradigm for exploring this issue. As we are learning language we do not memorize every sentence we hear, and then only repeat them verbatim. Instead we acquire some model of the language as we learn it - compressing the data that we hear into a schema, or grammar. Using this grammar, we are able to generalize what we have heard and adapt our language to new situations and compose sentences we have never heard before. Since the set of sentences in the human language is infinite, we do not hear every possible sentence, so the grammar learned might be different from that of the "teacher's". Consequently we speak a slightly modified language from our

parents. Across generations the language will therefore change. The learner in this situation is employing a type of lossy compression, where information is being lost, added or changed from generation to generation. The changes in information are transmission errors. As language is transmitted from generation to generation, these errors build up, so that language is changed over time, or evolves, simply as a result of the language acquisition system - a type of neutral evolution.

Language evolution has often been studied from the perspective of artificial life and evolution [1, 2, 13, 18, 25, 29]. In most studies it has been assumed that genetic evolution is the main driving force for language change or that natural selection acts on the learners of language [1, 29]. Other studies use semantics as the key for language evolution [14, 25]. We wish to explore the possibility that language evolution can be approximated by a model of cultural evolution, where there is no genetic transmission of information or schemas, only learning.

In this paper, we first give some background on data compression and formal languages. We explore the issue of language evolution using formal language, because it is a complex adaptive system in which it is possible to discuss things concretely, and changes in such systems can be seen clearly. We next describe

the model and summarize previous results. We then extend these results to explain how: (1) tradeoffs exist between error and compression; (2) initial conditions affect the language learned; and (3) different types of languages evolve differently. We discuss these results and finally the insights into compression and evolution given by this model.

2 Data Compression

Data can be compressed only when it contains some regularity that can be exploited. Consider, for example, the sequence of ordered pairs: (10,30), (15, 40), (20, 50), (25, 60). This can be represented by 8 ASCII characters of two letters each, for a total of 8 numbers $\times$ 2 characters/number $\times$ 7 bits/character = 112 bits. Since the sequence contains only numbers, not characters, then one need not use ASCII characters, but can represent them in binary form, requiring only 8 numbers $\times$ 6 bits/number = 48 bits, for a compression ratio of 0.43. Note, however, that the points lie on a straight line, $y = mx + b$, where $b = 10$ and $m = 2$. If we code b and m with 4 bits each, then we need code only the first coordinate of each pair, for a total of 4 numbers $\times$ 6 bits/number

+ 8 bits for a coefficient = 32 bits.

Compression in this example, as in all cases, requires a *model* of its regularity, e.g. in this case, there is a linear relationship. It is the model that aids in generalization [22]. From this linear model, for example, it is easy to interpolate or to extrapolate, where a look-up table would not suffice. Of course it may also be true that the *true* relationship among the variables is not linear – e.g. it might plateau just after the highest data point, so that extrapolation is not warranted. Also, the relationship might be just *approximately* linear, if not exactly so. For some purposes this *lossy* compression might be fine, though for other purposes (e.g. bank balances) the numbers should be exact, and the corresponding compression should be *lossless*.

It is important to recognize that some schemes might provide very efficient coding of data, but would involve lengthy and complicated computation to encode and decode it. A better measure of compression would recognize both costs, and the *minimum description length algorithm (MDL)* accomplishes this. As developed by Rissanen and Ristad [21], it states that the best model is the one which permits the shortest encoding of the observed data together with the model itself. The performance of a model is therefore measured by both

the length, in bits, of the description of the theory - *grammar-encoding-length* - and the length, in bits, of the data when encoded with the help of the theory - *data-encoding-length*.

When this combined code length is minimized a balance has been struck between the correctness of the model and its complexity [8]. The MDL measure has been used extensively in the fields of statistics and machine learning, and its theoretical properties have been well investigated [15, 21]. It has also been used to successfully model language acquisition [8, 15, 21].

3 Formal Language and FSAs

A formal language is a set of strings of symbols from some one alphabet, where a string is a finite sequence of symbols juxtaposed and an alphabet is a finite set of symbols. For example, `a, b` and `c` are symbols is the alphabet `(a,b,c)` and `abac` is a string [10]. A language consists of a set of these strings. The language can be a random set of strings such as `abcc, bacbc, bbaca` or it can contain some regularity such as the language *a[ab][ab]* which is the set of all strings drawn from the alphabet `(a,b)` of length three that start with an *a* -

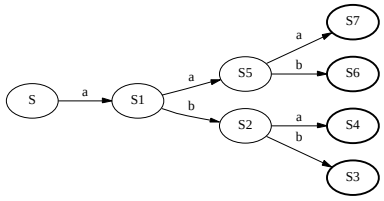

Figure 6: An example of a deterministic finite state automaton. S is the initial or start state and states with bold circles are the end or stop states.

aaa, aab, aba, abb.

These languages can be represented as directed graphs, or deterministic finite state automata (FSA) with vertices labeled by symbols connecting states. A deterministic finite state automata for the language *a[ab][ab]* above is the graph in Figure 6. You can see that each string or sentence in the language is a path along the nodes or states labeled with `S`'s. The description of this FSA is then the *grammar* for the language.

Deterministic finite state automata can be described mathematically by a quintuple, $(Q, \Sigma, \delta, q_0, F)$, where Q is a set of states, Σ is an input alphabet, $q_0 \in Q$ is the initial state, $\delta \subseteq Q \times \Sigma \times Q$ is a transition function, and $F \subseteq Q$ is the set of final states [10]. We will use this notation in the mathematical formulation of the MDL algorithm below. For the above example we can see that $Q =$ *(S, S1, S2, S3, S4, S5, S6, S7)*, $\Sigma =$ `(a,b)`, $q_0 = S$, $F =$ *states outlined in bold*

and δ is the set of all the transitions or arrows between states. The automaton accepts a string s if, and only if, the string labels a path from the initial state to a final state. The FSA in Figure 6 would accept the string *aba*, but not the string *baa* because there is no way to go from state S to state *S1* along a vertice labelled b, so there is no path that can be followed for the string to go from the initial state to the final state. Since an FSA is the defining grammar for a language, any string that can be accepted by the FSA is said to be a part of that language.

4 Language Acquisition

Language learners are regarded as systems that aim to identify rule systems that describe the (infinite) language of the community on the basis of finite evidence. This can only be successful in certain circumstances, whether one assumes that success is perfect identification in the limit (the "Gold" paradigm) [7], or that success is feasible convergence to arbitrarily good approximate identification [12, 27]. Since humans clearly learn to speak natural languages that can generate an infinite number of sentences, and do so largely from examples and

without conscious awareness of grammatical rules, most linguists believe that, roughly speaking, when a young person is confronted with a new sentence s/he will "try out" candidate grammars and then, from those accepting grammars, choose the one that is simplest [3, 8, 24].

Language acquisition can therefore be seen as a problem of unsupervised learning [18, 5]. Learners are given examples of what is in the language, then develop a grammar but receive no feedback when they speak as to whether they are right or wrong. It may be argued that teachers correct our speech, telling us to say "he ran" instead of "he runned", but this is not very common, especially when much of our "practicing" is done with peers in the same generation. Evidence for significant amounts of feedback is shaky, and there is little evidence that children even rely on it [16]. It has also been proposed that a learner may have access to side information not explicitly contained in the input. The theory that a type of inherent grammar exists is widely debated by linguists. Some researchers assume that there is some type of inherent knowledge of the structure of the language, such as Steven Pinker's "language instinct" or Chomsky's "universal grammar". While it has been shown that semantic clues are important for learning, it is better not to assume that side information is available to the

learner [5]. For one thing, the evidence as to what side information is available to children is tenuous, and significant learning may need to take place before side information becomes available. It is not clear how much extralinguistic information such as this is necessary for learning language, and there is the risk of attributing results we don't understand to the "magic" of this extralinguistic information. Also, as we wish to use this model to show how information can change over time, due to the encoding alone, for the purpose of this study we make the fewest assumptions and restrict our study to learning from only the available data.

5 The Nature of the Input

Viewing language learning as a mapping from a set of observed data to a grammar, it is evident that the input will greatly influence the types of grammars that the learners generate and the types of output that result. It is therefore important to vary the types of input to the model, in a way that can be computationally represented and discussed.

In our study the input is sets of strings of length six drawn from the alphabet

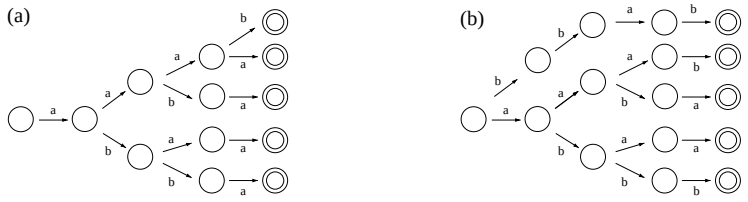

Figure 7: Transition state diagrams for uncompressed grammars: (a) Language 1 (b) Language 2.

a,b, and the input sets are varied based on a measure of smoothness - string edit distance. Each string in a language is a given symbol-substitution away from its nearest neighbor. Strings are also random, with each position in the string generated randomly.

Take for example Language 1 consisting of the strings aaaa, aaab, aaba, abaa and abba and Language 2 which is the set aaab, abaa, aaba, abbb and bbab. They have the finite state automata representation in Figures 7 a and b.

Language 1 is a smooth language, because the symbol in only one position is changed from one string to the next, and the strings are very similar or regular. It is generated by first obtaining a random string of a's and b's, in this case aaaa. This string is then subjected to one mutation or string edit at a random location in the string. Here the fourth position is chosen and the a is switched to b giving aaab. This process continues until a set of the desired length is

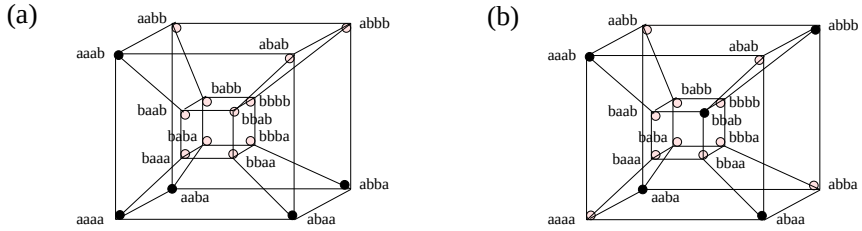

Figure 8: Hypercube representation of the landscape for strings of length 4. Nodes connected by lines are one symbol-substitution away from each other. Black dots are included in the language. Gray dots are not included. (a) Representation of language 1 (b) Representation of language 2.

generated. This language is considered smooth because a smooth path can be mapped out in string space, as can be seen by the hypercube representation in Figure 8a. Language 2, however, is more rugged. It is generated with a string edit distance of 2, so that at each step, 2 positions are randomly chosen to be edited. This does not allow for a very obvious path through string space, and as seen in Figure 8b, there is less regularity in the input.

In previous work we have looked at the grammars generated by languages with string edit distances 1, 2, 3 and random [26]. We found that compression occurs more easily in smooth string spaces than rugged ones. This affects the type of grammar that is generated and therefore the ability of the learner to generalize. It also affects the types of sentences the learner can produce once the grammar is formed. For example, after compression Language 1 becomes

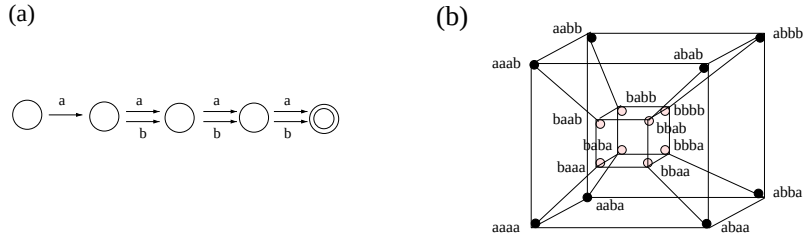

Figure 9: (a)Transition diagram of the compressed grammar for language 1. (b) Hypercube representation of the sentences included in the compressed grammar.

the set aaaa, aaab, aaba, abaa, aabb, abab, abba, abbb and Language 2 becomes aaab, aaba, abaa, bbab, bbba, abbb. As seen in Figure 9a, the representation of the grammar, or its finite state automata, becomes very compressed. Language 1 easily generalizes to the outer cube of the hypercube in this representation. However, for Language 2 there is still little structure in the final language as can be seen in Figure 10. The FSA is much less compressed because there is not enough regularity available to be used to develop a more compressed representation. Using a model of language acquisition, we expect these different types of languages to evolve differently over time.

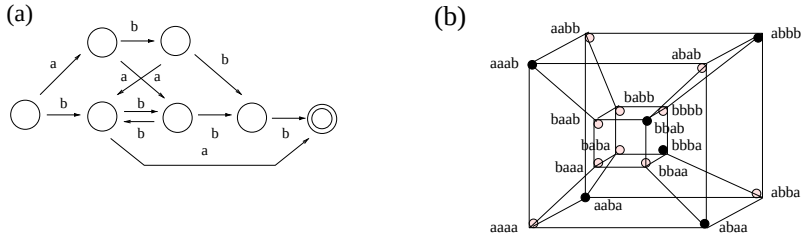

Figure 10: (a)Transition diagram of the compressed grammar for language 2. (b) Hypercube representation of the sentences included in the compressed grammar.

6 The Model

Our model of language learning will be that of a child, who when listening to an adult speaking a language s/he does not yet understand, tries out different candidate grammars that might accept that language. The minimalist theory supposes that those grammars which are deterministic and accept all the input sentences, are then examined further, and the grammar with the shortest MDL is preferred and tentatively accepted as describing the parent's language.

To calculate the MDL we take the sum of the *data-encoding-length* (Δ) (the cost of the coding of the data) and the *grammar-encoding-length* (Γ) (the cost of the rule set).

Since each string can be specified by a path through a deterministic automaton,

we find Δ by counting the number of choices that have to be made as each sentence's path is traced out. This is calculated in bits by the formula:

$$\Delta = \sum_{i=1}^{m} \sum_{j=1}^{|s_i|} \log_2 z_{i,j},$$

where m is the number of sentences in the sequence of strings encoded, $|s_i|$ is the length of the i'th string s_i, and $z_{i,j}$ is the number of ways to leave the state reached on the j'th symbol of sentence s_i. (A more succinct encoding is obviously possible when the probabilities of the transitions are not uniform. The minimalist approximation suffices for purposes of this preliminary investigation.)

Γ in this model is the number of bits required to encode the FSA. To specify the automaton itself, we must specify all the triples $(q_1, a, q_2) \in \delta$ and we must also specify the final states, so we calculate the grammar-encoding-length:

$$\Gamma = |\delta|[2(\log_2 |Q|) + \log_2 |\Sigma|] + |F|[\log_2 |Q|],$$

where $|\delta|$ is the number of triples in δ and $|F|$ is the number of final states in

F.

Using this algorithm to determine the appropriateness of its grammar, the agent learns the language presented in the following way.

An agent is exposed to all the legal sentences in the language – say of Language 1 or Language 2. It constructs an automaton with an associated grammar which accepts all of the sentences from the initial input and nothing else. The agent then compresses the original grammar by attempting to combine states and transitions in such a way that the outcome is still a deterministic automaton and then combines them if and only if the MDL of the new automaton is smaller than that of the starting one. This enables the best grammar to be found at one step according to *Algorithm 1*, outlined below.

A hill-climbing search is then performed using *Algorithm 2*, until the FSA with the smallest MDL is found.

Take for example a language of 10 examples with string edit distance 1 - `(aaabab aaaaab aaaabb abaabb aaaabb abaabb bbaabb bbbabb bbaabb bbaaba).` This generates the prefix tree FSA in Figure 11 where $\Gamma = 334.7$ $\Delta = 36.0$ and $MDL = 370.7$. The program runs until it cannot find an FSA with a smaller

Algorithm 1 Best Grammar Derivable In One Step (*current-grammar*)

```
1: best-encoding ← ∞
2: best-grammar ← ∅
3: for all possible pairings (s1, s2) of states from current-grammar do
4:    if s1 and s2 can be combined into a single state without making a non-
      deterministic FSM then
5:       grammar ← CombineStatesInGrammar(current-grammar, s1, s2)
6:       Calculate the encoding length for this grammar via the equations given.
7:       if encoding(grammar) < best-encoding then
8:          best-encoding ← encoding(grammar)
9:          best-grammar ← grammar
10:      end if
11:   end if
12: end for
```

Algorithm 2 Hill-Climbing Search (*current-grammar*)

```
1: current-length ← encoding length of current-grammar
2: loop
3:    grammar ← best grammar derivable in one step from current-grammar
      (Algorithm 1)
4:    grammar-length ← encoding length of grammar
5:    if grammar-length < current-length then
6:       current-grammar ← grammar
7:       current-length ← grammar-length
8:    else
9:       return current-grammar
10:   end if
11: end loop
```

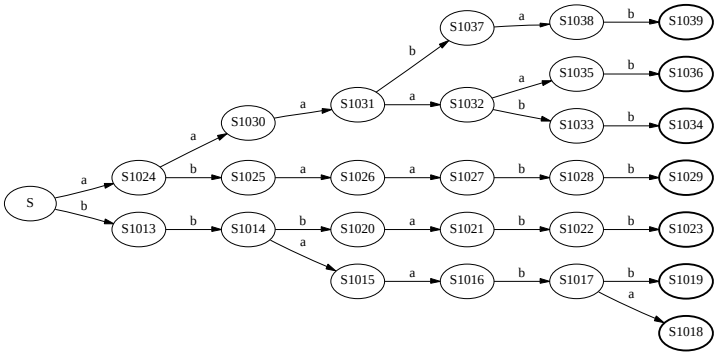

Figure 11: FSA for an example language of string edit distance 1. This is the prefix-FSA and is uncompressed.

Figure 12: Example FSA of string edit distance 1 from Figure 13 after the language has been learned.

MDL. This gives the grammar in Figure 13 with $\Gamma = 82.2$, $\Delta = 60.0$ and $MDL = 142.2$. The cost tradeoffs between the data-encoding-length and the grammar-encoding-length has been optimized. A language of 10 examples with string edit distance 2: `(abaaaa ababba ababba baabba aaabaa aabbba ababba abaaaa ababba aaaaba)` compresses to the grammar in Figure 13 with $\Gamma = 122.2$, $\Delta = 32.0$ and the $MDL = 154.2$. Both of these grammars accept more sentences than the original input, generalizing the language from the original subset that it heard. The string edit distance 2 language generalizes much less however.

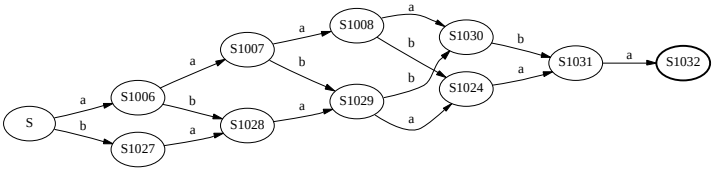

Figure 13: A compressed FSA for an example language of string edit distance 2.

In order to investigate the effects of this learning on language evolution, we must allow this learner to then teach a new generation, so this process is continued for several generations. The first generation is run as described above. After this first generation learner has developed a grammar, it has a set of sentences of string length 6 that its grammar can accept or speak. A set of these sentences is chosen randomly with replacement and used as the input for the learner of the next generation. This agent then uses the same model for learning.

To investigate the role of learning in language evolution, we take this to be one generation and iterate the process. We generate the first set of input as we did above, and find the FSA with the smallest MDL. Next we find all the strings of length 6 that are accepted by that compressed FSA. We then choose n of these randomly and present them as input to the learner for the next run. This process is iterated for 10 generations.

7 Experiments

In previous work, we investigated the question of how compression affects generalization, and how easy or difficult it is for languages to be generalized [26]. In our experiments we found that evolution of language led to a more compressed representation, or grammar, of the language over time, although the languages do not become any less complex. Here we extend these results to study important details: how in this system (1) tradeoffs exist between error and compression; (2) the effect of the initial conditions on the language found is significant; and (3) different types of languages evolve differently.

7.1 Tradeoffs exist between compression and error

In previous work, we saw that compression aids in generalization [26]. However with this compression comes a loss of information. This change in information can be seen in differences between the sentences in the original input language and the sentences accepted by the final compressed grammar. Since all sentences in the original input must be accepted by the compressed grammar, this can be seen as the number of sentences that were accepted by the final that

were not in the original input. We define this to be the *error*. We expect that when the MDL is high or there is a lot of compression, this error will be high. In an attempt to find a concise representation of the language, overgeneralization can occur so that too many sentences are admitted as correct sentences. The MDL algorithm accounts for much of this tradeoff, however, by using both the data-encoding-length (minimizing error) and the grammar-encoding-length (minimizing the model size) in the calculation to find the optimal encoding. We therefore expect that some of the trade-offs will be accounted for in the generation of the compressed model, and the trade-offs we would see between error and compression in the final language would be correlated with the trade-offs made between the data-encoding-length and the grammar-encoding-length in the generation of the grammar.

To investigate this tradeoff, we did 20 runs of one generation for string edit distances 1, 2, 3, 4 and random as described above. The error was calculated as the number of sentences that were accepted by the final grammar, that were not included in the original language. Compression was measured by the MDL and the grammar-encoding-length (Γ) and data-encoding-length (Δ) were measured separately as well.

We found that this trade-off between error and compression in the final grammar does exist, and that it is highly correlated with the data-encoding-lengths and grammar-encoding-lengths of the grammars. In Figure 14 we see that when the MDL is low, the error is high and vice versa. This is especially obvious when looking at the difference between the languages generated as string edit distances 1, 3 and random and the ones with even numbered string edit distances (2 and 4). These results are highly correlated with the measures of data-encoding-length and grammar-encoding-length in the MDL. The data-encoding-length is positively correlated with error (r^2=0.82), and the grammar-encoding-length is positively correlated with compression (r^2=0.98) as expected. When data-encoding-length is high, the sentences are being less compressed into a model, so there is less over-generalization and therefore less error. In a model of language-learning and in many other situations, it is acceptable to have a certain amount of error, as long as it can make useful predictions and is able to generalize to new situations [6]. In other instances however, it may be necessary that there be no errors, and it is important to realize that in any model of the environment these tradeoffs exist.

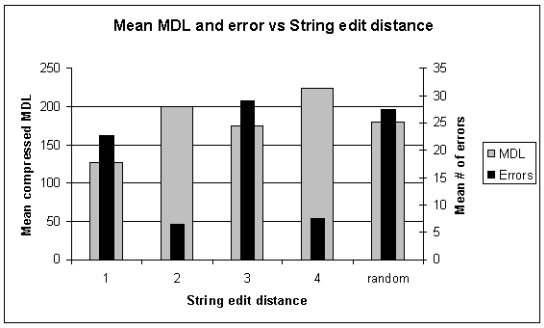

Figure 14: Mean compressed MDL measure versus mean number of errors after compression for string edit distances 1, 2, 3, 4 and random.

7.1.1 How tradeoffs are chosen

While we did expect the tradeoffs that we saw, we did not expect the even numbered string edit distances to be optimized in favor of the data-encoding-length the way that it was. We had expected there to be a steady increase in the MDL for increasing ruggedness in the languages.

The even numbered string edit distances are different from expected however, and the MDL is higher and the error lower than the string-edit-distances before and after them, as can be seen in the above Figure 14. In these sets the compressed data-encoding-length is also consistently lower. We therefore expect that there is some redundancy in the data that is not being accounted for by the string edit distance. The only other type of redundancy possible is in the

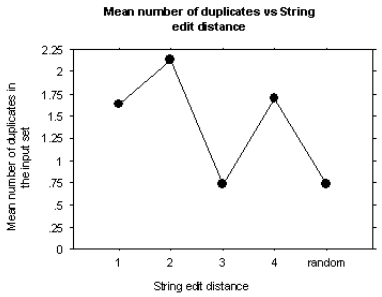

Figure 15: There are more duplicates for even string edit distances.

number of duplicates. When a set is being generated, it is possible for the same location to be changed twice, forming a duplicate sentence to one already in the set. We expect that if there is a high number of duplications, leading to a lot of regularity in the data, it will initially be more optimal to minimize the data-encoding-length. Then, since in the model the MDL cannot go down, or states cannot uncombine to ultimately find a better MDL, the data-encoding-length is optimized at the expense of the grammar-encoding-length. The data-encoding-length has to go up in order to decrease the grammar-encoding-length and this is not possible in a hill climbing search. Instead the grammar is getting stuck at a local optimum.

To see if it is increased regularity that is leading to a forced optimization of the data-encoding-length in a hill climbing search, we looked at the mean number of

duplicates for each string edit distance and the number of steps that were being taken to find the final grammar. We found that the mean number of duplicates for the even numbered string edit distances was much higher than for the other string edit distances (Figure 15), confirming this as the source of redundancy. We also found that for even numbered string edit distances less steps were being taken to find the final grammar, with a mean of 45.2 steps for string edit distance 1, 40.3 steps for distance 2, 49.8 for distance 3, 43.8 for distance 4 and 53.0 for random sets. While this may just mean that the optimum is being reached more efficiently, it is most likely that a local optimum is being found. (We attempted to overcome this bias by using a depth-first search. However, it was found to be not computationally feasible.) Finally, by optimizing the data encoding length there should be less error, as these have been seen to be correlated above. This is consistent with the low number of errors seen for the even numbered string edit distances.

So, while we did not see the smooth increase in MDL that we expected, it was not because our expectations about the model were wrong; it was that our understanding of the regularity in the input was not complete. This demonstrated the important effect of input and language type on learning, as has been shown

in other studies [19].

7.2 Effects of initial conditions

A benefit of the formalization of the language acquisition system is that we can study the effects of varying initial conditions which would be difficult to do with human language learners [18]. These variations of inputs are not different types of languages, but only different ways of presenting the same languages. We expect certain variations on the input to be more important than others. We expect that the type of language being learned is usually the most important factor for generating the final grammar. However, when changes in the initial conditions affect the weighting of the grammar-encoding-length and the grammar-encoding-length in the MDL, the final grammar could be affected significantly. We are interested in these effects on language acquisition and change.

7.2.1 Input size

In the calculation of the MDL the input size is an important factor. The data-encoding-length increases linearly with the number of unique examples in the input. In the grammar-encoding-length the number of states as well as the number of transitions, will increase, at least in the initial encoding of the language in the prefix FSA. We expect that new trade-offs will be made with more numbers of examples, causing the final MDLs to be different.

To look at this effect of input size, we did 8 runs for 10 examples, 4 for 20, 4 for 40 and 2 for 80. (We made fewer runs for the higher example set sizes because of computational costs.) In these experiments we found that the input size did have an effect on the MDL of the grammar found and the number of sentences that were accepted by that grammar. From Figure 16 it can be seen that there is less difference between the MDL of the different string edit distances with larger example sets. The pattern of the MDL measure has changed for these larger input sizes. The effect of language type on the final language learned is less significant. For the higher input sizes, the data-encoding-length is being optimized almost to the exclusion of the grammar-encoding-length as

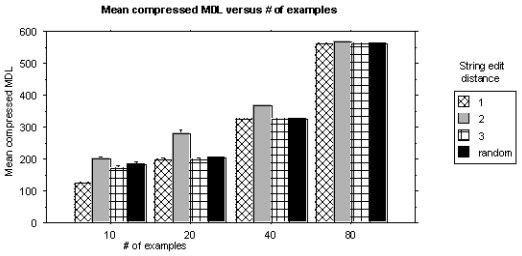

Figure 16: Mean compressed MDL measure versus number of examples in the input set for string edit distances 1, 2, 3 and random.

can be seen in Figure 17. With more data, it simply costs more to encode it. Also a larger proportion of the strings of length 6 are being presented to the learner. With 80 examples, each learner, regardless of the string edit distance of the input language, is hearing almost all the strings of length 6. All of these learners are therefore hearing essentially the same language. Therefore, the data-encoding-length of almost all these languages will be the same, and with its increased weighting, the MDLs are all similar. With 40 examples, the learners are still hearing many of the same examples, until at 20 and 10 the languages being presented are more unique and influenced by their smoothness in string space. The amount of input the learner receives influences the grammar that it is able to generate for the language. This is an important feature of the MDL, and is something that is almost certainly true in human language learning.

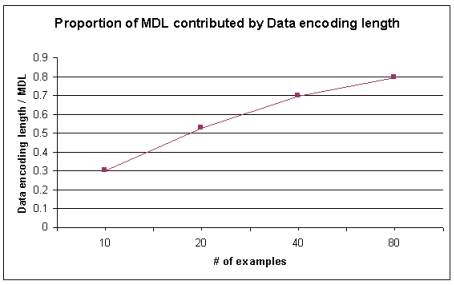

Figure 17: The proportion of the MDL that is calculated by Δ increases for larger input sizes.

7.2.2 Alphabet Size

Also important in the initial conditions is alphabet size. The sentences may be generated the same way, but changing the number of symbols in that language also affects the way it is compressed. We do not expect that the alphabet size should have a very large effect however, since this does not significantly change either the data-encoding-length nor the grammar-encoding-length. We think that it is important to look at an odd-numbered alphabet size however, to rule out the possibility that it is the effect of the even-numbered alphabet size interacting with the even numbered string edit distances which produces the grammars with the low MDLs.

To investigate this question we did 10 runs of string edit distances 1, 2, 3, 4

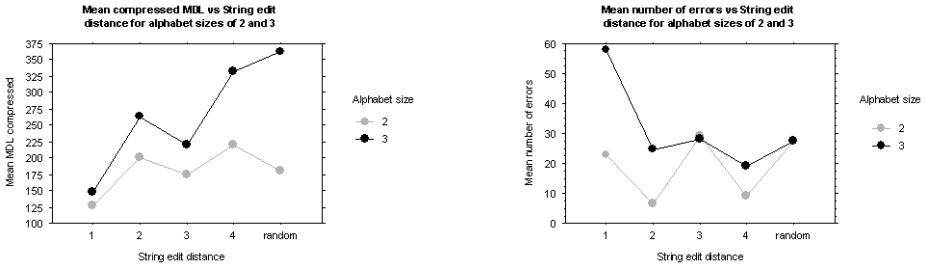

Figure 18: Patterns for the MDL measure and number of errors is the same for both alphabet sizes.

and random for the alphabet `(a,b)` and `(a,b,c)`.

We found that while the MDL was consistently higher for the 3 letter alphabet, as expected, the patterns for the MDL measure and number of errors for both alphabet sizes were similar as can be seen in Figure 18. This shows that the type of language used as input is more important in generating a grammar than alphabet size. This also demonstrates that it is not an interaction between the alphabet size and string edit distances that causes low MDL for even numbered string edit distances.

7.3 Different languages evolve differently over time

We have seen that in one generation a very important factor in what type of grammar is generated and the type of language that is spoken by the learner, is

the type or smoothness of the language used as input. We also know that compression or learning aids in generalization, so that after each generation more strings are accepted than were originally given as input. Therefore language could potentially change over time using just this learning mechanism. We expect that the different types of languages will evolve along different language trajectories over time. We expect the grammar to become simpler or more compressed, but this simplicity is not always reflected in the language that the learner finally speaks, or the sentences accepted by the learner's grammar, as we have seen in previous results [26]. To investigate this we look at the MDL, number of errors and accepted sentences of the final grammar and language.

We simulate evolution in our model as described above, first generating a target language as the input set. For each run a target language was generated from an input set of 10 examples. The set of examples was of string length 1, 2, 3, 4 or random and we did 10 of each. Once the target language was generated using the model for compression described above, 20 sentences from that language were selected and given to the first generation agent as input. Given this subset of the target language, the first generation agent developed its own model of the language and formed a new language, which it would then use to produce an

input set of 20 for the next generation. This was continued for 10 generations.

As expected, we found that MDL decreases over time, with the greatest decrease occurring in the first generation when the learner compresses the language from the prefix-tree FSA. This can be seen in Figure 19. It can also be seen that languages of different string edit distances have different fluctuations in their MDL measure. Grammars produced from languages of string edit distance 1, reach their optimal MDL almost immediately and do not fluctuate through the generations. The MDL of grammars produced by more complex or rugged languages however, such as string edit distances 2, 3, and 4 often continue to fluctuate for a few generations. Sets of random sentences produce some grammars that have varying MDLs and some that become fixed quickly. This is to be expected since languages with some regularity may be generated randomly.

Also, as language evolves, the number of sentences accepted follows a pattern similar to that of the MDL with the number of sentences accepted constant in the smooth languages, and fluctuating in the more rugged languages, as can be seen in Figure 20. As the grammars are able to generalize, there are also more errors or differences between the current language of the generation and the

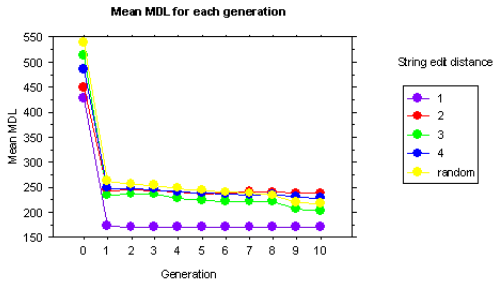

Figure 19: Mean MDL for each generation.

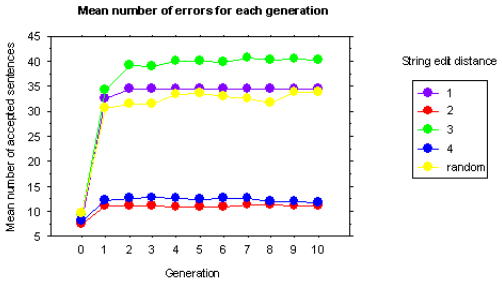

Figure 20: Mean number of sentences that were accepted in the langauage at each generation.

original generation. There are even fairly large differences between one generation's language and the next. This can be seen by looking at the differences between the target language and the final language and the differences between generations in the Figures 21 and 22 respectively.

These results demonstrate that in simple, regular languages there is not much change over time. The best grammar to describe that language has been found,

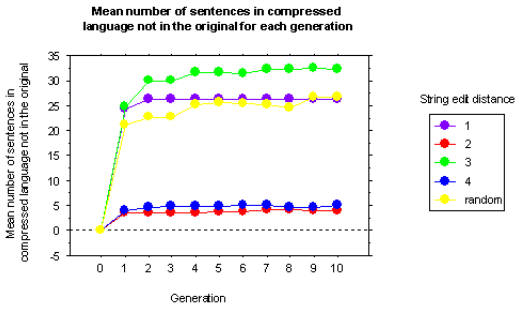

Figure 21: Mean number of sentences that were accepted in the langauage at each generation which weren't in the original language.

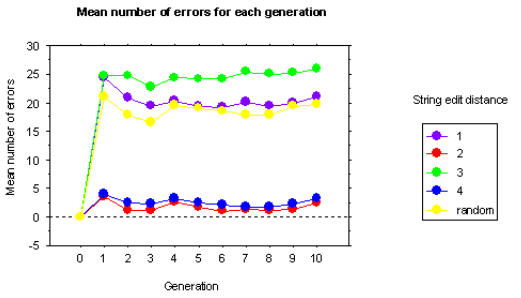

Figure 22: Mean number of errors between the input and the compressed grammar for each generation.

so no matter what examples are presented to it, the same language is found in every generation. In more complex language systems however, there is some fluctuation in the languages that are produced by the agents in each generation. There is not enough structure to be able to find a completely accurate model, so at each generation, the agent's perception of what the language is, is slightly different. The model of the language is very dependent upon the input received, and there are errors transmitted from one generation to the next. For example, an agent in generation 3 can generate a grammar that produces an incorrect language. That agent then teaches the next generation with that incorrect model, and that error becomes a part of the language. Due to this mode of learning and transmission, different languages follow different trajectories over time. We see this in human language, where for example, in King Alfred's writing, German and English were very similar, until now the languages are quite different.

8 Conclusions

We have shown that changes in language can occur as a result of transmission error alone. We have also shown that this is more true of more complex languages, which human language certainly is. This could also explain why we do not see much change in other animal's communication patterns. Bird calls and dolphin song may be simple enough that they are not subject to these same cultural transmission forces. Also, animals may have a different encoding scheme so they don't compress their language, but do organize it in a look-up table or in a way that simply minimizes the data-encoding-length and does not require tradeoffs.

The language trajectories could also be important in collective learning of a language by agents or robots. Different types of environments might provide different information that would influence the language the robots learn. Agents learning in different environments might not be able to communicate once their language has evolved for several generations and has followed a distinct communication trajectory. This conjecture has many possible applications in not only language learning, but any type of collective information sharing. In the

future, we hope to explore some of these possibilities, especially with robots in which the constraints of the acquisition device is particularly important.

These results can be applied to many systems that make use of lossy compression. It is important to realize the tradeoffs between error and compression in these systems, and how the compression algorithm used can affect information over time.

Part III

References

[1] T. Arita and Y. Koyama. Evolution of linguistic diversity in a simple communication sustem. In H. Kitano C. Adami, R. Belew and C.E. Taylor, editors, *Sixth International Conference on Artificial Life*, Cambridge, MA, 1998. MIT Press.

[2] J. Batali. Innate biases and critical periods: Combining evolution and learning is the acquisition of syntax. In R. Brooks and P. Maes, editors, *Fourth International Conference on Artificial Life*, Cambridge, MA, 1994. MIT Press.

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

[4] N. Chomsky. *The Minimalist Program.* MIT Press, Cambridge, MA, 1995.

[5] Carl G. de Marcken. Unsupervised language acquisition. *PhD Thesis - Department of Electrical Engineering and Computer Science*, 1996.

[6] M. Gell-Mann. Talk at Santa Fe Institute, January 9 1990.

[7] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[8] P. Grünwald. A minimum description length approach to grammar inference. In G. Scheler S. Wermter, E. Riloff, editor, *Symbolic, Connectionist and Statistical Approaches to Learning for Natural Language Processing*, LNCS #1040. Springer-Verlag, Berlin, 1996.

[9] T. Hashimoto. Usage-based structuralization of relationships between words. In P. Husbands and I. Harvey, editors, *Fourth European Conference on Artificial Life*, pages 483 – 492, Cambridge, MA, 1997. MIT Press.

[10] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata theory, Languages and Computation.* Addison Wesley, Reading, MA, 1979.

[11] S.A. Kauffman. *The Origins of Order: Self-Organization in Evolution.* Oxford University Press, New York, 1993.

[12] M.J. Kearns and U.V. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, Cambridge, MA, 1994.

[13] S. Kirby. Syntax without natural selection: How compositionality emerges from vocabulary in a population of learners. In Chris Knight, editor, *Approaches to the Evolution of Language*. 1999.

[14] S. Kirby and J. Hurford. Learning, culture and evolution in the origin of linguistic constraints. In P. Husbands and I. Harvey, editors, *Fourth European Conference on Artificial Life*, pages 493 – 502, Cambridge, MA, 1997. MIT Press.

[15] M. Li and P. Vitányi. Minimum description length induction, bayesianism and kolmogorov complexity. In *1998 IEEE International Symposium on Information Theory*, Cambridge, MA, 1998. MIT Press.

[16] E.V.M. Lievan. Crosslinguistic and crosscultural aspects of language addressed to children. In C. Gallaway and B.J. Richards, editors, *Input and Interaction in Language Acquisition*. Cambridge University Press, New York, NY, 1994.

[17] J.A. Moore. *Science as a Way of Knowing*. Harvard University Press, Cambridge, MA, 1993.

[18] P. Niyogi and R.C. Berwick. The logical problem of language change. Technical Report A. I. Memo No. 1516, MIT Artificial Intelligence Laboratory, July 1995.

[19] P. Niyogi and R.C. Berwick. Evolutionary consequences of language learning. *Linguistics and Philosophy*, 1997.

[20] C.H. Papadimtriou and M. Sideri. On the evolution of easy instances. *Unpublished manuscript*, 1998.

[21] J. Rissanen and E. Ristad. Language acquisition in the {MDL} framework. In E. Ristad, editor, *Language Computations*. American Mathematical Society, Philadelphia, PA, 1994.

[22] K. Sayood. *Introduction to Data Compression*. Morgan Kauffman, San Francisco, CA, 1996.

[23] L. B. Slobodkin. *Simplicity and Complexity in Games of the Intellect*. Harvard University Press, Cambridge, MA, 1992.

[24] E. Stabler. *Computational Minimalism: Acquiring and Parsing Languages with Movement*. Basil Blackwell, Oxford, 1999.

[25] L. Steels. Self-organising vocabularies. In Langton and Shimohara, editors, *Fifth International Conference on Artificial Life*, Cambridge, MA, 1996. MIT Press.

[26] T. Teal, D. Albro, E. Stabler, and C.E. Taylor. Compression and adaptation. In *Fifth European Conference on Artificial Life*, Heidelberg, Germany, 1999. Springer-Verlag.

[27] V.N. Vapnik. *Statistical Learning Theory.* Wiley, NY, 1998.

[28] J. Weber and C. Wong. Mutation of human short tandem repeats. *Human Molecular Genetics*, 2(8):1123–1128, 1993.

[29] G.M. Werner and M.G. Dyer. Evolution of communication in artificial organisms. In J.D. Farmer C. Langton, C.E. Taylor and S. Rassmussen, editors, *Second International Conference on Artificial Life*, Redwood City, CA, 1990. Addison Wesley.